The Dissertation Committee for Jeremiah Fletcher Palmer
certifies that this is the approved version of the following dissertation:

Efficient Frequency Response Computation for Structures with Structural Damping

Committee:

Jeffrey K. Bennighof, Supervisor

Tan T. Bui

Eric P. Fahrenthold

Robert van de Geijn

Jayant Sirohi

Efficient Frequency Response Computation for Structures with Structural Damping

by

Jeremiah Fletcher Palmer, B.S., M.S.C.A.M.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2015

Dedicated to my wife, April, and my sons, Jacob and Jackson.

Acknowledgments

This dissertation would not have been possible without the support of many crucial individuals. The first is Dr. Bennighof, who has been my teacher and adviser for many years. I am grateful for his inspiration and encouragement concerning my research and dissertation topics and his efforts which helped me improve the quality of this dissertation. I have also benefited greatly from his insight into both academia and the automobile industry.

I would also like to express my gratitude to my other committee members, Dr. Bui, Dr. Fahrenthold, Dr. van de Geijn, and Dr. Sirohi for their time and advice through this process.

Next, I am especially appreciative of Mark Muller, who has been an excellent resource, helping me understand how to create efficient commercial software programs. I also would like to thank Tim Allison, Josh Haben, Chang-Wan Kim, Mintae Kim, Qin Qin Li, Garrett Moran, Jack Poulson, Eric Swenson, and David Vargas for research discussions we had while we worked together.

I am thankful for Mladen Chargin from CDH AG, who provided the industrial models used in this dissertation. I am grateful for the volume of experience he has in the automobile industry and his support on this research.

Finally, I would like thank my family. I appreciate my parents for always being very supportive of my academic pursuits. I am particularly grateful for my

wife, April, who has a seemingly unending amount of patience and love for me.

Jeremiah Palmer

The University of Texas at Austin

May 2015

Efficient Frequency Response Computation for Structures with Structural Damping

Jeremiah Fletcher Palmer, Ph.D.
The University of Texas at Austin, 2015

Supervisor: Jeffrey K. Bennighof

The modern procedure for analyzing the dynamics of a large, complex structure, such as an automobile, is to use the finite element method to discretize the structure with millions of degrees of freedom. For the steady-state response to a harmonic excitation, a frequency response problem (FRP) is derived for the finite element discretization. To ease computational cost, modal analysis is performed, creating a corresponding FRP in an approximating modal subspace with a substantial reduction in dimension.

Typically, more than one level of structural damping is present in a complex structure. This results in a fully populated modal damping matrix, so that the frequency-dependent coefficient matrix of the modal FRP is full. This problem is traditionally solved using a brute-force approach, which can be prohibitively expensive since it requires $O(n^3)$ operations for each of the hundreds of frequencies.

This dissertation presents two new approaches for solving modal FRPs of automobile structures that have any composition of structural damping. Each approach requires a single frequency-independent $O(n^3)$ operation which changes the

full coefficient matrix of the modal FRP into one with a simpler form. The first approach presents a new method which creates a low rank approximation of the modal structural damping matrix. The second approach is used when the modal structural damping matrix has high rank and relies on a new method for determining an accurate eigenvalue decomposition of a complex symmetric matrix. Computing responses using these two approaches then only requires $O(n^2)$ operations for every frequency.

Automobile companies perform analyses on computers with multi-core CPU processors and graphics processing units which can perform dense linear algebra operations with high efficiency. This dissertation shows how the two approaches are implemented to take advantage of these parallel technologies. The accuracy and performance of the two new approaches are presented and compared with the brute-force approach.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Analyzing the dynamics of a complex structure, such as an automobile, is a challenging task to accomplish efficiently. The current state of the art is to use the finite element (FE) method to discretize the structure with typically millions of degrees of freedom (N_{dof}). Through FE analysis, a system of ODEs is derived in terms of the discretization points' time-dependent displacement, velocity, and acceleration as well as the time-dependent forcing function and the specified initial conditions. This system of ODEs can be simplified to algebraic equations when the forcing function is harmonic in time, in which case the displacements, velocities, and accelerations are also harmonic in time. The problem of solving this system of algebraic equations for the response over a range of frequencies is called the frequency response problem (FRP). To ease computational cost, the FE FRP is projected onto an approximating modal subspace. The modal subspace is obtained by solving the generalized eigenvalue problem associated with the mass and stiffness matrices, and consequently, their modal counterparts are diagonal. When other matrices involved in the modal FRP are transformed into the modal subspace, they become full, except in certain special cases, but substantially reduced in dimension.

In the automobile industry, there is a continual desire to perform analyses on

models with more automobile components modeled, more degrees of freedom, and finer FE meshes, in order to obtain more accurate and useful automobile representations. A consequence of this trend is that as N_{dof} increases and the frequency ranges of interest grow, the dimension of the modal FRP increases. With a higher number of structural modes, it becomes important to account for the varying levels of structural damping of the materials which make up the model. When all of the model's structural damping is accurately represented, an additional matrix of data is included in the FRP, which causes the frequency-dependent coefficient matrix of the modal FRP to be full.

Typically, a modal FRP with a full coefficient matrix is solved using a brute-force approach which is straightforward, yet expensive. This dissertation presents two new approaches which solve modal FRPs with structural damping much more efficiently than the brute-force approach.

1.1 FE Frequency Response Problem

The FRP for a typical automobile structure has the form:

$$[-\omega^2\mathcal{M} + i\omega\mathcal{B} + (1 + i\gamma)\mathcal{K} + i\mathcal{K}_s]\mathcal{X}_s(\omega) = \mathcal{F}_s(\omega). \tag{1.1}$$

$\mathcal{M}$, $\mathcal{B}$, $\mathcal{K}$, and $\mathcal{K}_s$ represent the finite element mass, viscous damping, stiffness, and structural damping matrices, respectively. These matrices are all real, square with dimension N_{dof}, and symmetric, unless gyroscopic effects are present, in which case $\mathcal{B}$ is nonsymmetric. The matrix $\mathcal{F}_s(\omega) \in \mathbb{C}^{N_{dof} \times N_c}$ is the frequency-dependent force matrix with N_c representing the number of load cases. The scalar γ is the global

structural damping coefficient; ω is the excitation frequency in radians per second; and $i = \sqrt{-1}$. The matrix $\mathcal{X}_s(\omega) \in \mathbb{C}^{N_{dof} \times N_c}$ contains the solutions of the FRP for a particular ω, which are the nodal displacements. Let N_{freq} represent the number of frequencies in the frequency range of interest.

Acoustic fluid elements can be used to model the air in the interior of an automobile. The solution to the modal FRP involving the structure and the acoustic fluid can be used to determine the noise, vibration, and harshness (NVH) that a human would perceive. Then, an analyst can make modifications to the design to create a quieter vehicle. After FE discretization, the FRP for interaction between a structure and an acoustic fluid, with a pressure representation for the fluid, is

$$\begin{bmatrix} \mathcal{Z}_s(\omega) & i\omega\mathcal{A} \\ i\omega\mathcal{A}^T & -(-\omega^2\mathcal{E} + i\omega\mathcal{C} + \mathcal{H})/\rho \end{bmatrix} \begin{Bmatrix} \mathcal{X}_s(\omega) \\ \mathcal{X}_f(\omega) \end{Bmatrix} = \begin{Bmatrix} \mathcal{F}_s(\omega) \\ \mathcal{F}_f(\omega) \end{Bmatrix} \tag{1.2}$$

where $\mathcal{Z}_s$ is the sum of the matrices in the square brackets in equation (1.1) and $\mathcal{A}$ is the area (or coupling) matrix which converts fluid pressure to structural loading. The matrices $\mathcal{E}$, $\mathcal{C}$, and $\mathcal{H}$ represent the FE fluid "mass", "damping", and "stiffness" matrices, respectively; and ρ is the fluid mass density [19]. The matrix $\mathcal{X}_f$ contains acoustic fluid pressures.

1.2 Structural Damping

Structural, or material, damping is a type of damping which causes a loss of energy in materials but the loss of energy per oscillatory cycle is determined by the amplitude of the displacement rather than that of the velocity. It was demonstrated experimentally by Kimball and Lovell [28] that the energy loss per cycle is

independent of the frequency for many materials, such as metals, glass, and rubber, but approximately proportional to the amplitude of vibration squared:

$$\triangle E_{cyc} = \alpha X^2. \tag{1.3}$$

The way that this observation is usually taken into account in vibration analysis can be understood through the following thought experiment. Consider a single degree-of-freedom system consisting of a massless spring, which is excited by a force that is harmonic in time, so that its equation of motion is simply

$$kx(t) = f(t) = Ake^{i\omega t}. \tag{1.4}$$

The motion that satisfies this equation is proportional to $f(t)$: $x(t) = Ae^{i\omega t}$. The complex representation is often used in harmonic response analysis for convenience, with the understanding that the actual excitation or response is only the real or the imaginary part of its complex representation. The amount of energy dissipated per cycle is a real quantity and is equal to the work done by the excitation, so it can be determined by integrating the real part of the force over the real part of the displacement:

$$\triangle E_{cyc} = \int_{cyc} (\text{Re } f)(\text{Re } dx).$$

Since $dx = \dot{x}dt$, and one cycle is equal to 2π radians, the amount of energy dissipated

per oscillatory cycle is

$$\begin{aligned}\triangle E_{cyc} &= \int_0^{2\pi/\omega} (\text{Re } f)(\text{Re } \dot{x})dt \\ &= \int_0^{2\pi/\omega} (Ak\cos(\omega t))(-A\omega\sin(\omega t))dt \\ &= -kA^2\omega \int_0^{2\pi/\omega} \cos(\omega t)\sin(\omega t)dt \\ &= 0.\end{aligned}$$

Over the course of a cycle, there is no net energy dissipated for this system, because k is real-valued. However, if the spring constant is complex-valued, there is a nonzero dissipation of energy over each cycle, so the spring is hysteretic. Then, the following equation governs the motion of the spring:

$$k(1+i\gamma)x(t) = f(t) = Ake^{i\omega t}. \tag{1.5}$$

Here, k is multiplied by the dimensionless constant γ to obtain the imaginary part of the complex spring constant. This equation can be rearranged into the form

$$k\sqrt{1+\gamma^2}e^{i\phi}x(t) = Ake^{i\omega t}, \text{ where } e^{i\phi} = \cos\phi + i\sin\phi = \frac{1+i\gamma}{\sqrt{1+\gamma^2}}. \tag{1.6}$$

The motion $x(t)$ with the complex-valued spring constant is now not in phase with $f(t)$:

$$x(t) = \frac{A}{\sqrt{1+\gamma^2}}e^{i(\omega t-\phi)}. \tag{1.7}$$

As a result, the energy lost per oscillatory cycle for this representation is now not

equal to zero:

$$\begin{aligned}
\triangle E_{cyc} &= \int_0^{2\pi/\omega} (\text{Re } f)(\text{Re } \dot{x}) dt \\
&= \int_0^{2\pi/\omega} (Ak\cos(\omega t)) \left(\frac{-A\omega}{\sqrt{1+\gamma^2}} \sin(\omega t - \phi) \right) dt \\
&= \frac{-kA^2\omega}{\sqrt{1+\gamma^2}} \int_0^{2\pi/\omega} \cos(\omega t)\sin(\omega t - \phi) dt \\
&= \frac{-kA^2\omega}{\sqrt{1+\gamma^2}} \cdot \frac{-\pi \sin\phi}{\omega} \\
&= \frac{k\pi A^2}{\sqrt{1+\gamma^2}} \cdot \frac{\gamma}{\sqrt{1+\gamma^2}} \\
&= k\pi\gamma \left(\frac{A}{\sqrt{1+\gamma^2}} \right)^2 . \qquad (1.8)
\end{aligned}$$

The term in the parentheses in equation (1.8) is the magnitude X of the displacement from equation (1.7). Therefore, comparing equations (1.8) and (1.3), the value α must be equal to the product $k\pi\gamma$. This gives the constant γ, which is the structural damping coefficient, in terms of the value of the α, which is determined experimentally, and k, the spring stiffness, as $\gamma = \frac{\alpha}{\pi k}$. The system's structural damping in equation (1.5) is represented by $i\gamma k$.

For a multi-degree-of-freedom system composed of hysteretic springs or elastic members, all with the same structural damping coefficient γ, the equation of motion is

$$(1 + i\gamma)K\mathbf{x}(t) = \mathbf{f}(t) = \mathbf{F}e^{i\omega t} \qquad (1.9)$$

where K is the system's stiffness matrix. Using the definition of $e^{i\phi}$ from equation

(1.6), the displacement vector $\mathbf{x}(t)$ is

$$\mathbf{x}(t) = \frac{1}{\sqrt{1+\gamma^2}} K^{-1}\mathbf{F} e^{i(\omega t - \phi)} = \mathbf{X} e^{i(\omega t - \phi)}, \tag{1.10}$$

where the vector of amplitudes $\mathbf{X}$ is

$$\mathbf{X} = \frac{1}{\sqrt{1+\gamma^2}} K^{-1}\mathbf{F}.$$

Following steps similar to those for the scalar case, the energy dissipated per cycle becomes

$$\begin{aligned}
\triangle E_{cyc} &= \int_0^{2\pi/\omega} (\text{Re } \mathbf{f})^T (\text{Re } \dot{\mathbf{x}}) dt \\
&= \int_0^{2\pi/\omega} \left(\mathbf{F}^T \cos(\omega t)\right) \left(-\omega \mathbf{X} \sin(\omega t - \phi)\right) dt \\
&= -\omega \mathbf{F}^T \mathbf{X} \int_0^{2\pi/\omega} \cos(\omega t) \sin(\omega t - \phi) dt \\
&= \left(-\omega\sqrt{1+\gamma^2}\right) \mathbf{X}^T K \mathbf{X} \left(\frac{-\pi \sin\phi}{\omega}\right) \\
&= \mathbf{X}^T \left(\pi\gamma K\right) \mathbf{X}.
\end{aligned}$$

Experimentally, the energy dissipated per cycle by structural damping is found to be quadratic in the displacement amplitudes, as for the single-degree-of-freedom case:

$$\triangle E_{cyc} = \mathbf{X}^T \left[\alpha\right] \mathbf{X}$$

where $[\alpha]$ is a square matrix. By comparing the two equations, the matrix $[\alpha]$ is found to be proportional to the stiffness matrix K:

$$[\alpha] = \pi\gamma K.$$

When an automobile model comprises components made of multiple materials each with different structural damping values, the complete structural damping in the FE space is equal to $i\gamma\mathcal{K} + i\mathcal{K}_s$, where $\mathcal{K}$ is the FE stiffness matrix and γ is the structural damping value of the predominant material. The matrix $\mathcal{K}_s$ represents deviations of other materials' structural damping from γ.

1.3 Modal Frequency Response Problem

The coefficient matrix in equation (1.1) is very large, so it is projected onto a smaller, approximating subspace to reduce the problem's dimension. This is accomplished by finding an approximate solution to the generalized eigenvalue problem of the form:

$$\mathcal{K}\Phi_s = \mathcal{M}\Phi_s\Lambda. \tag{1.11}$$

For many years, analysts relied on the Lanzcos method to solve the eigenvalue problem. However, the computational time required to perform the Lanczos method for modern automobile models is usually very high. The Automated Multi-Level Substructuring (AMLS) method provides an approximate solution to the eigenvalue problem for the pertinent parts of the model much more efficiently. The commercial implementation of the AMLS method is now the standard software used by nearly every automobile company in the world to compute approximate modes for large FE models.

Let N_{ev} represent the number of modes of vibration found by approximating the solution to (1.11) using AMLS. Then, the modes of vibration are the columns of $\Phi_s \in \mathbb{R}^{N_{dof}\times N_{ev}}$. The diagonal matrix $\Lambda \in \mathbb{R}^{N_{ev}\times N_{ev}}$ contains the squares of the

natural frequencies of the system. After Φ_s is mass-normalized so that $\Phi_s^T \mathcal{M} \Phi_s = I$ and $\Phi_s^T \mathcal{K} \Phi_s = \Lambda$, let the nodal displacements be approximated as

$$\mathcal{X}_s(\omega) \approx \Phi_s X_s(\omega). \tag{1.12}$$

Premultiplying equation (1.1) by Φ_s^T with the substitution in (1.12), yields the modal FRP:

$$[-\omega^2 I + i\omega B + (1 + i\gamma)\Lambda + iK_s] X_s(\omega) = F_s(\omega) \tag{1.13}$$

where $B = \Phi_s^T \mathcal{B} \Phi_s$, $K_s = \Phi_s^T \mathcal{K}_s \Phi_s$, and $F_s(\omega) = \Phi_s^T \mathcal{F}_s(\omega)$. The dimension of the modal subspace, N_{ev}, is typically around 10,000 which is much smaller than the millions of degrees of freedom of the FE space. Therefore, the modal FRP in (1.13) can be solved much more economically than the FE FRP in (1.1). Unfortunately, the coefficient matrix in equation (1.13) is full because both B and K_s are full matrices which makes computing the response challenging.

If FE acoustic fluid elements are included in the analysis, the FE acoustic fluid matrices are also projected onto a smaller subspace using AMLS. The following eigenvalue problem is solved:

$$\mathcal{H} \Phi_f = \mathcal{E} \Phi_f \Lambda_f.$$

The number of fluid modes, N_f, is typically much smaller than the number of structure modes. Φ_f is normalized so that $\Phi_f^T \mathcal{E} \Phi_f = \rho I_f$, where I_f is an $N_f \times N_f$ identity matrix, and as a result, $\Phi_f^T \mathcal{H} \Phi_f = \rho \Lambda_f$. The structural displacements and fluid pressures are represented as

$$\begin{Bmatrix} \mathcal{X}_s(\omega) \\ \mathcal{X}_f(\omega) \end{Bmatrix} \approx \begin{bmatrix} \Phi_s & 0 \\ 0 & \Phi_f \end{bmatrix} \begin{Bmatrix} X_s(\omega) \\ X_f(\omega) \end{Bmatrix}. \tag{1.14}$$

Using the above substitution and multiplying equation (1.2) on the left by

$$\begin{bmatrix} \Phi_s^T & 0 \\ 0 & \Phi_f^T \end{bmatrix}, \tag{1.15}$$

the modal FRP with acoustic fluid interaction is derived:

$$\begin{bmatrix} Z_s(\omega) & i\omega A \\ i\omega A^T & Z_f(\omega) \end{bmatrix} \begin{Bmatrix} X_s(\omega) \\ X_f(\omega) \end{Bmatrix} = \begin{Bmatrix} F_s(\omega) \\ F_f(\omega) \end{Bmatrix}. \tag{1.16}$$

Here, $Z_s(\omega)$ is the structure's coefficient matrix represented by the sum in square brackets from equation (1.13), and the analogous matrix for the fluid is given by

$$Z_f(\omega) = -\left(-\omega^2 I_f + i\omega C_f + \Lambda_f\right), \tag{1.17}$$

where $A = \Phi_s^T \mathcal{A} \Phi_f$, $C_f = \Phi_f^T \mathcal{C} \Phi_f$, and $F_f(\omega) = \Phi_f^T F_f(\omega)$.

1.4 Determining Algorithm Performance

A floating point operation (FLOP) is typically considered to be a single multiplication or addition involving floating point numbers. On modern computers, a calculation which requires pairs of numbers to be multiplied together and added to a sum is usually handled by a multiplier-accumulator (MAC) unit. The inner product of two real vectors of length n uses the MAC unit n times, and costs $2n$ FLOPs. Therefore, the product of two real matrices, one of size $m \times n$ and the other of size $n \times k$, requires $2mnk$ FLOPs. Since a complex multiplication-accumulation requires four multiplications and four additions, the inner product of two complex vectors of length n is $8n$. Then, the product of two complex matrices with the same dimensions as the real matrices costs $8mnk$ FLOPs. Table 1.1 gives a summary of FLOP counts

A Description	B Description	AB FLOPs
diagonal complex	full real	$2nk$
diagonal complex	full complex	$6nk$
full complex	diagonal complex	$6mn$
full real	full real	$2mnk$
full real	full complex	$4mnk$
full complex	full complex	$8mnk$

Table 1.1: AB FLOPs for Various Forms and Types of A and B

for matrix multiplications for AB, where A is $m \times n$ and B is $n \times k$, for various forms and data types of the matrices. When A or B is diagonal, then no additions of entries in the matrices are required. When one of the matrices is real and the other is complex, then the FLOP cost is double the cost of performing a matrix multiplication when both matrices are real.

FLOP costs can be used to determine the most efficient way to accomplish a task. For example, suppose three real matrices are multiplied together as $D := ABC$ where A is $a \times b$, B is $b \times c$, and C is $c \times d$. The product D can be determined in two ways:

$$D := (AB)C \text{ or } D := A(BC).$$

If the first two matrices are multiplied first, the number of FLOPs required to compute D is $2ac(b+d)$. If the second two matrices are multiplied first, the number of FLOPs is $2bd(a+c)$. Clearly, for this task, the sizes of the matrices dictate which of the two choices requires fewer FLOPs and is, therefore, more efficient.

The performance of high performance computing (HPC) software for NVH analyses is directly related to the performance of a collection of dense linear algebra

routines. The simpler routines are contained within the Basic Linear Algebra Subprograms (BLAS) interface [31]. BLAS functionality is divided into three levels, in order of complexity:

1. BLAS level-1 subprograms perform operations on vectors, such as inner products and vector scaling.

2. BLAS level-2 subprograms perform matrix-vector operations. Since the number of operations in these subprograms is proportional to the number of matrix and vector entries, their performance is limited by memory bandwidth.

3. BLAS level-3 subprograms perform matrix-matrix operations. For square matrices of dimension n, the number of FLOPs in BLAS level-3 subprograms is $O(n^3)$, whereas the number of entries in the matrices is $O(n^2)$. This indicates that these subprograms are limited by the processor throughput. BLAS level-3 subprograms can be implemented in a way which takes advantage of parallel computing.

The Linear Algebra Package (LAPACK) contains complicated routines such as matrix factorizations and matrix solves [7]. Table 1.2 lists a few LAPACK routines which are referenced in this dissertation. In the table, A represents an $n \times n$ square, symmetric matrix, B represents an $n \times n$ square, nonsymmetric matrix, and C represents an $n \times r$ general matrix. The FLOP costs listed in Tables 1.1 and 1.2 are used to compute FLOP costs for all of the algorithms presented in this dissertation.

Name	Description	FLOPs
DSYTRD	Tridiagonal reduction of real A	$\frac{4}{3}n^3$
DORMTR	Backtransformation of real C	$2n^2r + nr$
ZSYSV	Complex "solve": $A^{-1}C$	$\frac{4}{3}n^3 + 8n^2r + O(n^2)$
ZGESV	Complex "solve": $B^{-1}C$	$\frac{8}{3}n^3 + 8n^2r + O(n^2)$

Table 1.2: FLOPs for Various LAPACK routines

1.5 Motivations and Challenges

NASTRAN is the primary commercial NVH software package that automobile engineers use to analyze FE models. After the AMLS software is used within NASTRAN to provide a substantial modal reduction, the next task is to find the solution of the modal FRP. When the structural damping matrix is present in the FRP, NASTRAN does not take advantage of any special properties of K_s. In this case, modal FRPs are solved by first forming the coefficient matrix of the modal FRP at a frequency. Next, the complex matrix system of equations is solved (using a ZSYSV or ZGESV routine) which involves factoring the coefficient matrix into two triangular matrices and performing two triangular matrix solves. This approach is very expensive as it requires $O(N_{ev}^3)$ FLOPs at *every* frequency. In many cases, using this approach to solve the modal FRP dominates NVH analyses.

This dissertation provides two new approaches for handling the modal structural damping matrix and the choice of the two approaches is primarily based on the rank of K_s. For each approach, only one $O(N_{ev}^3)$ operation is required, followed by some computations for every frequency which have a FLOP cost significantly less than

using the brute-force approach at every frequency. Modern computers on which NVH analyses are executed have processors with multiple cores per CPU and coprocessors which provide very fast computation performance. This dissertation explains how the two approaches can be implemented for optimal performance on various modern computer system architectures.

The Fast Frequency Response Solver (FastFRS) is a commercial software package created by the author which provides automobile analysts a comprehensive modal FRP solver. It is currently licensed by many automobile companies around the world, and is used in place of the NASTRAN modal FRP solver. The two approaches presented in this dissertation are implemented in FastFRS, giving it a distinct advantage in performance over other available modal FRP solvers.

1.6 The Industrial Test Suite

Several industrial models are used in this dissertation to demonstrate the effectiveness and performance of the approaches. The models differ in N_{ev}, N_f, N_{freq}, and N_c, and represent a typical range of automobile models from industry which have modal structural damping. Some information about each model is listed in Table 1.3.

1.7 Outline of the Dissertation

In this dissertation, two approaches are presented which solve modal FRPs for models with structural damping efficiently. The following is an outline of the dissertation.

Model	N_{ev}	N_{freq}	N_c	N_f
A	10351	218	1	803
B	11475	381	6	6017
C	8595	797	345	364
D	6539	401	150	857
E	15923	796	90	1088
F	12165	591	48	426

Table 1.3: Statistics of Automobile Models in the Industrial Test Suite

- Chapter 2 provides a survey of the traditional and recent methods used to solve modal FRPs with viscous and structural damping.

- Chapter 3 introduces the first new approach for handling modal structural damping. This approach uses a low rank approximation for K_s.

- Chapter 4 introduces the second approach, which treats K_s as a full rank matrix. A new complex symmetric matrix eigensolver is presented which is used in this approach.

- Chapter 5 gives an overview of implementations of the approaches which take advantage of current parallel programming paradigms.

- Chapter 6 evaluates the new approaches' solutions in terms of solution accuracy and elapsed time for the models in Table 1.3.

- Chapter 7 discusses conclusions and future work.

Chapter 2

Survey of Methods to Solve Modal FRPs

In the absence of viscous and structural damping, the coefficient matrix of the FRP in equation (1.13) becomes diagonal in the modal space:

$$[-\omega^2 I + \Lambda] X_s(\omega) = F_s(\omega). \tag{2.1}$$

The response is trivial to compute, requiring only $O(N_{ev})$ FLOPs at every frequency:

$$X_s(\omega) = \left[-\omega^2 I + \Lambda\right]^{-1} F_s(\omega).$$

However, damping must be represented in models of automobile structures to obtain accurate results. Sections of this chapter present modal FRPs having viscous and/or structural damping with increasing levels of complexity. Methods proposed by others for solving different types of modal FRPs are described. Section 2.1 identifies models of damping that result in uncoupled systems of equations in the modal FRP that, like equation (2.1), are trivial to solve. If viscous damping is represented, but the structural damping model does not produce off-diagonal terms in the modal FRP, the modal FRP has a full coefficient matrix, and it can be solved at every frequency using a special matrix formula which is discussed in Section 2.2. When structural damping is modeled in such a way that off-diagonal terms are produced in the modal FRP, the traditional approach in Section 2.3 can be used to determine the solution

to the modal FRP, but it is expensive. An alternative to the traditional approach is described in the final section of the chapter. This alternative approach is inexpensive, but the solutions of the modal FRP that result from its use are sometimes inaccurate.

2.1 Solving Uncoupled Modal FRPs

Models with globally uniform structural damping represented with the scalar parameter γ, and/or Rayleigh (proportional) damping produce an uncoupled modal FRP. With proportional damping, the FE viscous damping matrix is represented as a linear combination of the FE mass and stiffness matrices. Let

$$\mathcal{B} = \beta_0 \mathcal{M} + \beta_1 \mathcal{K},$$

where β_0 and β_1 are two scalar quantities. The coefficient matrix of the modal FRP is diagonal. The response is

$$X_s(\omega) = \left[(-\omega^2 + i\omega\beta_0)I + (1 + i\gamma + i\omega\beta_1)\Lambda\right]^{-1} F_s(\omega).$$

Globally uniform structural damping and proportional damping only offer three parameters (γ, β_0, and β_1) for modeling damping, so their versatility for accurately modeling damping in a complex structure is very limited.

2.2 Solving Modal FRPs having Non-Proportional Damping

If there is non-proportional viscous damping, but no structural damping present, then equation (1.13) becomes

$$[-\omega^2 I + i\omega B + \Lambda] X_s(\omega) = F_s(\omega).$$

In typical automobile models, viscous damping is only associated with a limited number of discrete devices such as engine mounts and shock absorbers, so the FE matrix $\mathcal{B}$ is very sparse, having only dozens of nonzero rows and columns, N_{nz}. Kim [27] successfully takes advantage of this sparsity by recognizing that even though the modal viscous damping matrix, B, is full, its rank is always equal to or less than N_{nz}. If $\mathcal{B}_{nz} \in \mathbb{R}^{N_{nz} \times N_{nz}}$ represents a condensed matrix containing only the nonzero rows and columns of $\mathcal{B}$, and $\Phi_{nz} \in \mathbb{R}^{N_{nz} \times N_{ev}}$ represents a matrix of the rows of Φ (from equation (1.12)) that correspond to nonzero columns in $\mathcal{B}$, the modal viscous damping matrix can be represented as:

$$B = \Phi_{nz}^T \mathcal{B}_{nz} \Phi_{nz}. \tag{2.2}$$

$\mathcal{B}_{nz}$ is a square matrix with an extremely small dimension, due to the sparsity of $\mathcal{B}$. Typically, $\mathcal{B}$ is a symmetric matrix and it can be decomposed using an eigenvalue decomposition (EVD): $\mathcal{B}_{nz} = \overline{U} \Sigma_{\mathcal{B}} \overline{U}^T$. If gyroscopic effects are present in the automobile model, $\mathcal{B}$ is nonsymmetric and can be decomposed using a singular value decomposition (SVD): $\mathcal{B}_{nz} = \overline{U} \Sigma_{\mathcal{B}} \overline{V}^T$. Let U and V be defined as $\Phi_{nz}^T \overline{U}$ and $\Phi_{nz}^T \overline{V}$, respectively, then

$$B = \Phi_{nz}^T \overline{U} \Sigma_{\mathcal{B}} \overline{V}^T \Phi_{nz} = U \Sigma_{\mathcal{B}} V^T \tag{2.3}$$

where the dimension of the diagonal matrix $\Sigma_{\mathcal{B}}$ is $\text{rank}(\mathcal{B}_{nz})$, and U and V are $N_{ev} \times \text{rank}(\mathcal{B}_{nz})$. Since $\overline{U}$ and $\overline{V}$ are orthogonal, U and V are not. The modal FRP becomes

$$[-\omega^2 I + \Lambda + U(i\omega \Sigma_{\mathcal{B}}) V^T] X_s(\omega) = F_s(\omega). \tag{2.4}$$

If $\mathcal{B}$ is symmetric, then $U = V$. The coefficient matrix above is a low rank modification to a diagonal matrix and can be inverted using a variation of the *Sherman-Morrison-Woodbury* (SMW) formula [23].

Formula 1 (**Sherman-Morrison-Woodbury**). *If W and $(I + Y^T W^{-1} Z)$ are invertible, then $(W + ZY^T)^{-1} = W^{-1} - W^{-1}Z(I + Y^T W^{-1} Z)^{-1} Y^T W^{-1}$.*

The SMW formula can be modified into a form useful for solving (2.4) by letting W be a diagonal matrix D, and letting $Z = P\Sigma^{1/2}$ and $Y = R\Sigma^{1/2}$, where Σ is another diagonal matrix:

$$(D + P\Sigma R^T)^{-1} = D^{-1} - D^{-1}P\Sigma^{\frac{1}{2}}(I + \Sigma^{\frac{1}{2}}R^T D^{-1} P\Sigma^{\frac{1}{2}})^{-1}\Sigma^{\frac{1}{2}}R^T D^{-1}. \tag{2.5}$$

To invert the coefficient matrix in equation (2.4) at each frequency ω using (2.5), let

$$D(\omega) := -\omega^2 I + \Lambda,$$

$$P := U,$$

$$R := V, \text{ and}$$

$$\Sigma(\omega) := i\omega\Sigma_{\mathcal{B}}.$$

Without the SMW formula, the FLOP cost of computing the response of equation (2.4) at each frequency is $O(N_{ev}^3)$. Using the SMW formula, the FLOP cost of computing the response at a frequency is $O((\text{rank}(\mathcal{B}_{nz}))^3)$, which is much smaller than the method which avoids using the SMW formula if $\text{rank}(\mathcal{B}_{nz})$ is much smaller than N_{ev}.

2.3 Traditional Approaches for Solving FRPs with Structural Damping

When structural damping is present in an automobile structure, the coefficient matrix of the modal FRP is full. Then, the modal FRP represents a complex indefinite linear system of equations. Traditionally, these kinds of problems are solved by forming the coefficient matrix and solving the matrix equation at every frequency using either an iterative or an exact approach.

2.3.1 Iterative Approaches

For large systems of equations with positive definite coefficient matrices, iterative approaches are generally preferred over exact approaches because they are typically faster and use less computer memory. However, these features of iterative approaches do not extend to systems with complex indefinite coefficient matrices. Since the convergence rate of iterative approaches depends on spectral properties of the coefficient matrix, and the coefficient matrix of interest is a complex indefinite one, iterative approaches either converge slowly, or do not converge at all. Furthermore, for those systems that do converge, a solution must be found for every right hand side (N_c) for every frequency. Iterative approaches are not practical to use to find the response of large modal FRPs with structural damping.

2.3.2 Exact Approaches

Exact approaches reliably provide solutions to complex indefinite systems of equations. They use a single factorization at each frequency to compute the solution

for all right hand sides simultaneously. Also, since the steps in exact approaches are prescribed, their FLOP costs are known. In order to avoid inevitable divisions by zero which occur while complex indefinite matrices are factored, a pivoting strategy is used [26]. The LAPACK routine, ZSYSV, provides the solution to a complex indefinite system and uses the efficient Bunch-Kaufman method [12] for the factorization's pivoting strategy. The cost of pivoting is small compared to the cost of the factorization. Using ZSYSV to solve the modal FRP in equation (1.13) involves the following steps:

1. Form the coefficient matrix, $Z_s(\omega)$, in equation (1.13) at a particular frequency:

$$Z_s(\omega) := -\omega^2 I + i\omega B + (1 + i\gamma)\Lambda + iK_s.$$

   Then, the modal FRP is $Z_s(\omega)X_s(\omega) = F_s(\omega)$.

2. Factor $Z_s(\omega)$ with the Bunch-Kaufman pivoting strategy: $Z_s = UDU^T$, where U is a unit upper triagular matrix and D is a symmetric block-diagonal matrix with 1×1 and 2×2 diagonal blocks.

3. Let a matrix $Y := U^T X_s$ and $W := DY$. Then, the response is computed through:

$$W = U^{-1} F_s$$

$$Y = D^{-1} W$$

$$X_s = U^{-T} Y.$$

From Table 1.2, the FLOP cost of computing responses with this approach is

$$\text{Cost} = N_{freq}\left(\frac{4}{3}N_{ev}^3 + 8N_{ev}^2N_c\right).$$

If B is nonsymmetric, an LU factorization (with partial pivoting) is performed, implemented in MKL in the routine ZGESV. This approach's FLOP cost is

$$\text{Cost} = N_{freq}\left(\frac{8}{3}N_{ev}^3 + 8N_{ev}^2N_c\right).$$

When the number of modes and frequencies is high, exact approaches often require more time to complete than the reduction from FE space to modal space using AMLS. For models with structural damping, the traditional approaches for solving the corresponding modal FRPs are time-consuming and expensive.

2.4 An Inadequate Approach for Solving the Modal FRP with Structural Damping

In his dissertation, Kim [27] suggests an approach for solving modal FRPs which have structural damping, but no viscous damping. It computes one $O(N_{ev}^3)$ factorization, which is used to diagonalize the coefficient matrix. In this situation, the modal FRP is

$$[-\omega^2 I + (1+i\gamma)\Lambda + iK_s]X_s(\omega) = F_s(\omega). \tag{2.6}$$

Let the $N_{ev} \times N_{ev}$ complex symmetric matrix, C, be defined as

$$C := (1+i\gamma)\Lambda + iK_s. \tag{2.7}$$

Next, its eigenvalue decomposition is computed:

$$C = \Phi_C \Lambda_C \Phi_C^T. \tag{2.8}$$

When Φ_C is scaled so that $\Phi_C^T \Phi_C = I$, the solution of the modal FRP becomes:

$$X_s(\omega) = \Phi_C \left[-\omega^2 I + \Lambda_C \right]^{-1} \Phi_C^T F_s(\omega). \tag{2.9}$$

The critical step in the proposed approach is the determination of Φ_C and Λ_C in equation (2.8). This type of matrix decomposition is unusual. The diagonalization of complex symmetric matrices is a topic which has not received much attention because of the following [8], [9], [43]:

- Complex Hermitian matrices are encountered much more often in practice than complex symmetric matrices.
- A straightforward approach to the tridiagonal reduction of a complex symmetric matrix can encounter numerical instability that results in significant loss of accuracy.
- Complex symmetric matrices do not benefit from all of the advantageous properties of real symmetric matrices. For instance, a complex symmetric matrix cannot be reduced to diagonal form by a unitary similarity transformation.

Because of the possibility of encountering instability, the symmetry of complex symmetric matrices is ordinarily not exploited in finding its EVD, but instead an EVD process for general nonsymmetric complex matrices is used. However, Kim explores the potential for exploiting the symmetry of C in finding its EVD, using a procedure which is analogous to the most efficient procedure for finding the EVD of *real* symmetric matrices. The procedure for real symmetric matrices consists of these steps:

1. Reduce the matrix to tridiagonal form using a sequence of unitary Householder reflections.

2. Compute the eigenvalues and eigenvectors of the tridiagonal matrix. The eigenvalues of the tridiagonal matrix are the same as those of the original matrix, because the first step uses similarity transformations.

3. Recover the eigenvectors of the original matrix using a sequence of the Householder reflections used in the first step.

In complex matrix arithmetic, similarity transformations with unitary matrices are ordinarily preferred because they are norm-preserving. However, in this case, since C is complex symmetric rather than Hermitian, unitary transformations do not preserve symmetry. Instead, a sequence of unitary Householder reflections that annihilates all entries below the subdiagonal, for example, does not annihilate the entries above the superdiagonal, so the matrix becomes upper Hessenberg rather than tridiagonal. This increases computational cost significantly. A similarity transformation that preserves symmetry can be accomplished with a complex matrix which is not unitary, but is instead a complex orthogonal (CO) matrix [14]. CO matrices can be used to reduce the complex symmetric matrix C to complex symmetric tridiagonal form.

2.4.1 Complex Orthogonal Rotations

A sequence of CO rotation matrices can be used to tridiagonalize complex symmetric matrices. A CO rotation G, which is the complex analogue of the Givens

rotation, is defined as:

$$G = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}.$$

The angle θ is allowed to be complex; therefore, $\sin(\theta)$ and $\cos(\theta)$ can be complex. The values of $\sin\theta$ and $\cos\theta$ are related to one another through the relationship $\sin^2(\theta) + \cos^2(\theta) = 1$, which means that G is an orthonormal matrix because $G^T G = GG^T = I$.

A single CO rotation matrix is used to annihilate a single entry from a column. For example, the entries in G which annihilate the second entry in the column $\begin{Bmatrix} a & b \end{Bmatrix}^T$ through

$$G \begin{Bmatrix} a \\ b \end{Bmatrix} = \begin{Bmatrix} \alpha \\ 0 \end{Bmatrix}$$

are $\cos(\theta) = \frac{a}{\alpha}$ and $\sin(\theta) = \frac{-b}{\alpha}$, with $\alpha = \sqrt{a^2 + b^2}$. The value of θ itself is not typically computed. In general, when G is $N_{ev} \times N_{ev}$, it takes the form

$$G = \begin{array}{c} \\ \\ p-1 \\ p \\ \\ N_{ev} \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & p-1 & p & & N_{ev} \end{array} \\ \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cos(\theta) & -\sin(\theta) & \cdots & 0 \\ 0 & \cdots & \sin(\theta) & \cos(\theta) & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix} \end{array}. \tag{2.10}$$

The position p represents the row number of the entry which will be annihilated using the entry at row number $(p-1)$.

When the entry being annihilated is in an $N_{ev} \times N_{ev}$ matrix, the only rows affected by the multiplication are $p-1$ and p. CO rotations can reduce a complex symmetric matrix to tridiagonal form, but the entries are annihilated individually. It is much more economical to use reflections instead of rotations for tridiagonal reduction.

2.4.2 Complex Symmetric Tridiagonal Reduction

Two strategies have been presented that perform complex symmetric matrix tridiagonal reductions using reflections [9], [21]. Both strategies progress from one corner of the matrix along the diagonal to the opposite corner, annihilating entries of the matrix outside of the tridiagonal, one row/column pair at a time.

The first strategy, the "splitting method", involves annihilating the real part of a row/column pair using a real Householder reflection followed by annihilating the imaginary part of that same row/column pair (but reduced in length by 1 so that the real part's annihilation is retained) with another real Householder reflection. Next, the remaining entry is annihilated by a CO rotation [9].

Another strategy keeps the real and imaginary parts together and is analogous to performing real Householder transformations to obtain the tridiagonal matrix. By modifying La Budde's Method [29] to accommodate complex symmetric matrices, Gansterer shows that a complex symmetric matrix can be tridiagonalized with CO reflections [21]. La Budde's Method is preferred to the "splitting method" because it requires less memory throughput. Let a Householder-analogous CO reflection be

defined as:

$$H := I - \frac{2}{\beta}\mathbf{v}\mathbf{v}^T, \text{ where } \beta = \mathbf{v}^T\mathbf{v} \text{ and } \mathbf{v} \in \mathbb{C}^n. \tag{2.11}$$

The vector $\mathbf{v}$ is chosen so that $H\mathbf{x}$ produces a vector with zeroes in all but the first entry. For this to be true, $\mathbf{v}$ must be $\mathbf{x} \pm \hat{\mathbf{e}}_1\sqrt{\mathbf{x}^T\mathbf{x}}$ where $\hat{\mathbf{e}}_1$ is a Euclidean unit vector. Then,

$$\beta = 2(\mathbf{x}^T\mathbf{x} \pm x_1\sqrt{\mathbf{x}^T\mathbf{x}}) \tag{2.12}$$

and to maximize accuracy, the sign of the second term above is selected to minimize $\frac{2}{\beta}$. For real arithmetic, this decision ensures that the scalar coefficient $\frac{2}{\beta}$ is bounded. However, because $\mathbf{x}$ is complex, it is possible for $\mathbf{x}^T\mathbf{x}$ to be zero when $\mathbf{x}$ is nonzero. A simple example of such an $\mathbf{x}$ is

$$\mathbf{x} = \begin{Bmatrix} 1+i \\ 1-i \end{Bmatrix}.$$

This causes "catastrophic cancellation," which results in β being zero or nearly zero, and instability in the reflection matrix H. The possibility of catastrophic cancellation exists in both the splitting method and the modified La Budde's method.

2.4.3 Catastrophic Cancellation

Suppose during the tridiagonalization of a complex symmetric matrix of size N_{ev}, catastrophic cancellation is encountered at row/column κ. The form of the

matrix at this point is:

$$\left[\begin{array}{ccccccccccc}
\alpha_1 & \beta_1 & & & & & & & \cdots & 0 \\
\beta_1 & \alpha_2 & \beta_2 & & & & & & & \vdots \\
 & \beta_2 & \ddots & \ddots & & & & & & \\
 & & \ddots & \alpha_{\kappa-1} & \beta_{\kappa-1} & & & & & \\
 & & & \beta_{\kappa-1} & \alpha_\kappa & x_1 & x_2 & x_3 & \cdots & x_r \\
 & & & & x_1 & \times & \times & \times & \cdots & \times \\
 & & & & x_2 & \times & \times & \times & \cdots & \times \\
 & & & & x_3 & \times & \times & \times & \cdots & \times \\
\vdots & & & & \vdots & \vdots & \vdots & \vdots & \ddots & \times \\
0 & \cdots & & & x_r & \times & \times & \times & \times & \times
\end{array}\right] \tag{2.13}$$

where $r = N_{ev} - \kappa$ and the vector $\mathbf{x} = \begin{Bmatrix} x_1 & x_2 & \cdots & x_r \end{Bmatrix}^T$ corresponds to the vector exhibiting catastrophic cancellation. The part of the matrix that has been tridiagonalized (the entries α_j and β_j where $j = 1, \ldots, \kappa - 1$) is called the "tridiagonal tail". The beginning of the tail corresponds to entries near α_1; the end of the tail corresponds to entries near α_κ.

In [9], Bar-On and Ryaboy identify catastrophic cancellation and call it a "breakdown". They conclude that this problem is very unlikely to occur, as it indicates that the real and imaginary parts of $\mathbf{x}$ have identical magnitudes and are orthogonal. Bar-On and Ryaboy present an algorithm which is to be used when a breakdown is encountered during tridiagonalization.

In Algorithm 1, a CO rotation is symmetrically applied at the beginning of the

Algorithm 1: Bar-On and Ryaboy's method for removing "breakdowns"

Given : A partially reduced matrix of the form equation (2.13), with a breakdown at row/column κ

1 Perform one sweep of the complex variant of the QL algorithm from the beginning to the end of the tridiagonal tail.
2 The final CO rotation in the sweep fills in row/column $\kappa - 1$ with nonzeroes.
3 The fill-in row/column is annihilated.

tridiagonal tail to create a "bulge" - a new nonzero entry just outside of the tail. When another CO rotation matrix is used to annihilate this bulge, a new bulge appears one step down the tail. Repeating this process, which causes the bulge to move along the tail, is called "bulge chasing" [42]. All together, the creation of the bulge, followed by bulge chasing constitutes one sweep of the QL algorithm [15]. When the bulge reaches the unreduced part of the matrix, instead of creating a bulge, the last CO rotation matrix replaces the previously-annihilated row/column with nonzeroes, which must be annihilated again. Kim [27] suggests using La Budde's Method to tridiagonalize the matrix and Algorithm 1 to avoid breakdowns. The similar matrices C and T are related to one another through

$$C = QTQ^T \tag{2.14}$$

with

$$Q^TQ = I. \tag{2.15}$$

Considering the possibility of breakdowns, the CO matrix Q is

$$Q = \prod_{i=1}^{N_{ev}-2} K_i, \text{ where } K_i = \begin{cases} G_i H'_{i-1} \hat{H}_i, & \text{if a breakdown is present} \\ H_i, & \text{otherwise.} \end{cases} \tag{2.16}$$

Model	$e_o(Q)$	$e_t(Q,T,C)$
A	4.43×10^{-3}	1.75×10^{-2}
B	3.77×10^{-3}	6.49×10^{-3}
C	4.76×10^{-4}	2.83×10^{-3}
D	2.32×10^{-4}	3.46×10^{-3}
E	1.20×10^{-2}	2.85×10^{-1}
F	1.56×10^{-2}	1.19×10^{-2}

Table 2.1: Quality of Tridiagonalization

The matrix H_i is defined in equation 2.11 and the product of the CO rotations used in a sweep of complex QL is represented by G_i. The sweep causes the reduction to retreat by one row/column and H'_{i-1} is the CO reflection that annihilates the filled-in entries. Finally, $\hat{H}_i$ is the new CO reflection that is used after the breakdown is removed.

In order to determine the validity of the Q and T matrices, two metrics are used. The first checks the orthogonality of Q from equation (2.15):

$$e_{\mathrm{o}}(Q) := \max_{i,j} |\delta_{ij} - (Q^T Q)_{ij}|. \tag{2.17}$$

The second verifies that C can be recovered using Q and T by measuring the difference between QTQ^T and C with respect to the diagonal entries of C from equation (2.14):

$$e_{\mathrm{t}}(Q,T,C) := \max_{i,j} \left| \frac{(QTQ^T)_{ij} - C_{ij}}{\sqrt{C_{ii}C_{jj}}} \right|. \tag{2.18}$$

The results from tridiagonalizing the C matrices in the Industrial Test Suite using Q from equation (2.16) are found in Table 2.1. This table shows that the quality of the tridiagonal reduction is substantially worse than machine precision error. In all

of the cases, when the resulting Q and T matrices are used to compute the EVD of C, the quality of the resulting solution to the modal FRP from equation (2.9) is too poor to be used in industry. The cause of this imprecision is identified and a method for overcoming this problem is discussed in Chapter 4.

Chapter 3

Low Rank Approach

The structure of a typical automobile is made of mostly one material: steel. Consequently, analysts are able to represent structural damping for most of the structure with a single global structural damping coefficient γ. Structural damping for components made of other materials is represented in $\mathcal{K}_s$. When the number of elements contributing to $\mathcal{K}_s$ is small, then there are several null rows and columns in $\mathcal{K}_s$. This means that $\mathcal{K}_s$ can be low rank which implies that K_s in equation (1.13) has the potential of being low rank.

3.1 Low Rank Representation

An $m \times n$ matrix A with $m \geq n$ can be factored using a standard tool, the singular value decomposition (SVD): $A = U\Sigma V^T$, where the orthogonal matrices U and V are $m \times m$ and $n \times n$, respectively, and Σ is $m \times n$. The nonzeroes in Σ are located on the diagonal of the upper $n \times n$ part of Σ. The factorization can be written as a sum of n rank-1 matrices:

$$A = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T,$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ are the ith columns of U and V, respectively, and σ_i is the (i, i) entry in Σ. If there are only r nonzero values in Σ, where $r < n$, then the factorization can

be expressed as a sum of r rank-1 matrices instead. A can be approximated by using fewer than r rank-1 matrices, possibly resulting in savings in computation.

Given a matrix A as described above and a matrix norm $||\cdot||$, let the low rank approximation (LRA) of A of rank r be defined as

$$\hat{A} = \arg\min_{\bar{A}} \|\bar{A} - A\|, \text{ subject to } \text{rank}(\bar{A}) = r \leq n.$$

The Eckart-Young-Mirsky theorem [24] states that if the matrix norm is the Frobenius norm, then the truncated SVD can be used to create an approximation to A with a lower rank.

Theorem 1 (**Eckart-Young-Mirsky**). *Let A be an $m \times n$ matrix with $m \geq n$. Define the singular value decomposition as $A = U\Sigma V^T$, where $U^TU = I$, $V^TV = I$, and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. If $\hat{\Sigma}$ is defined as $\hat{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k, 0, \ldots, 0)$, then the rank* k *matrix $\hat{A} = U\hat{\Sigma}V^T$ is such that*

$$||A - \hat{A}||_F = \min_{\text{rank}(\bar{A}) \leq k} ||A - \bar{A}||_F = \sqrt{\sigma_{k+1}^2 + \ldots + \sigma_n^2}.$$

For a symmetric matrix, such as the modal structural damping matrix, the singular values are the absolute values of the eigenvalues of the matrix, and each column of U is equal to the corresponding column of V, to within a sign. The SVD becomes essentially an eigenvalue decomposition (EVD), with eigenvalues sorted by absolute value and $U = V$. An LRA of A of rank-r that is optimal in the sense of the Frobenius norm can be obtained by setting eigenvalues with magnitudes smaller than a threshold to zero, effectively truncating the sum of rank-1 matrices used to approximate A. Because FE structural damping matrices can have a low rank, it

may be possible to approximate the modal structural damping matrices with LRAs. Unfortunately, the values in $\mathcal{K}_s$ vary greatly in magnitude from one model to another, with larger values exerting disproportionate influence on eigenvalue decompositions, which then causes the eigenvalue ranges and distributions to vary across models. For the Industrial Test Suite, the maximum eigenvalues of K_s vary from $O(10^5)$ to $O(10^{12})$. A standard cutoff value for determining LRAs using Theorem 1 requires that the eigenvalue ranges of all modal structural damping matrices have some consistency. This can be accomplished by using weighting matrices to scale the rows and columns of K_s, which can diminish the influence of the extreme values on the EVD, thereby exposing the underlying structure of K_s. A theorem of Markovsky [35] tells how an optimal LRA of a matrix weighted on both sides can be obtained:

Theorem 2 (**Two-Sided Weighted LRA**). *Define the two-sided weighted matrix $L = W_l A W_r$, where the square matrices W_l and W_r are the left and right weighting matrices. Let $\hat{L}^*$ represent the optimal unweighted LRA of L of rank-k. Then,*

$$\hat{A}^* = W_l^{-1} \hat{L}^* W_r^{-1} \tag{3.1}$$

is a solution of the following two-sided weighted LRA problem

$$\min_{\text{rank}(\bar{A}) \leq k} ||W_l(A - \bar{A})W_r||_F.$$

Determining the most appropriate weighting matrices that produce optimal LRAs can be complicated. A common approach is to use weighting matrices that have a special structure such as diagonal or block-diagonal so that the inverses in equation (3.1) are easy to compute [34].

3.2 Determining the Rank of the Structural Damping Matrix

When there is no viscous damping, the modal FRP from equation (1.13) becomes

$$[-\omega^2 I + (1 + i\gamma)\Lambda + iK_s]X_s(\omega) = F_s(\omega). \tag{3.2}$$

If all of the structural damping in the automobile model can be represented by the global structural damping factor, then equation (3.2) becomes

$$[-\omega^2 I + \Lambda + i\gamma\Lambda]X_s(\omega) = F_s(\omega)$$

and Λ's direct impact on the structural damping term is obvious. The structural damping of models that have this simple damping can be compared to one another easily by comparing the models' γ values. This is accomplished by removing the structural damping term's dependence on Λ. When the structural damping of the model can not be represented by a single γ, then matrix K_s must be used as in equation (3.2). Just as the level of structural damping γ can be found from matrix $\gamma\Lambda$ by dividing by the eigenvalues in Λ, matrix K_s can similarly be interpreted relative to the eigenvalues by multiplying on its left and right by $\Lambda^{-1/2}$. Equation (3.2) can be written as

$$\{-\omega^2 I + \Lambda^{1/2}\left[I + i(\gamma I + \Lambda^{-1/2}K_s\Lambda^{-1/2})\right]\Lambda^{1/2}\}X_s(\omega) = F_s(\omega)$$

which suggests that a suitable choice for the left and right weighting matrices for a weighted LRA of K_s is $\Lambda^{-1/2}$.

Since some of the modes in the frequency response can be rigid body modes, it is important to be mindful while forming the weighting matrix W that some entries

in Λ can be zero or nearly zero. Then, W is the pseudoinverse of $\Lambda^{1/2}$:

$$W := \text{diag}\left(w_1, \ldots, w_{N_{ev}}\right), \text{ where } w_i = \begin{cases} \frac{1}{\sqrt{\Lambda_{i,i}}}, & \text{if } \Lambda_{i,i} > 0 \\ 1, & \text{otherwise.} \end{cases} \tag{3.3}$$

Let L represent the two-sided weighted modal structural damping matrix:

$$L = W K_s W \tag{3.4}$$

and let its eigenvalue decomposition be determined to be

$$L = \Phi_L \Lambda_L \Phi_L^T \tag{3.5}$$

where $\Phi_L^T \Phi_L = I$ and $\Lambda_L = \text{diag}(\overline{\lambda}_1, \overline{\lambda}_2, \ldots, \overline{\lambda}_{N_{ev}})$. We find that a K_s matrix and its corresponding weighted matrix L generally have similar ranks,but dramatically different eigenvalue ranges. The new eigenvalue ranges of the weighted modal structural damping matrices are similar across models, and a consistent method can now be used to determine the ranks of L matrices. Table 3.1 shows how the maximum eigenvalues of the models in the Industrial Test Suite are much more similar for L than for K_s.

After the values in Λ_L are ordered so that $|\bar{\lambda}_1| > |\bar{\lambda}_2| > \cdots > |\bar{\lambda}_{N_{ev}}|$, the first k entries are used to form $\hat{\Lambda}_L = \text{diag}(\overline{\lambda}_1, \overline{\lambda}_2, \ldots \overline{\lambda}_k)$. Next, let $\hat{L}$ represent a rank-k approximation of L:

$$L \approx \hat{L} = \hat{\Phi}_L \hat{\Lambda}_L \hat{\Phi}_L^T, \tag{3.6}$$

where $\hat{\Phi}_L$ contains the first k columns of Φ_L. Let the arbitrary value τ_{LRA} represent an LRA tolerance, which can be used for any automobile model's structural damping

Model	Max e'value of K_s	Max e'value of L
A	2.11×10^{6}	1.00×10^{-1}
B	4.11×10^{5}	1.99×10^{-2}
C	4.81×10^{12}	2.04×10^{0}
D	4.51×10^{8}	1.34×10^{-1}
E	8.80×10^{6}	1.50×10^{-1}
F	4.08×10^{6}	2.86×10^{-1}

Table 3.1: Comparing Maximum Eigenvalues of the K_s to $L = WK_sW$

matrix. Using Theorem 1, the relative difference between the original structural damping matrix and its LRA through the weighting matrix can be written as:

$$\frac{||W(K_s - \hat{K}_s)W||_F}{||WK_sW||_F} = \frac{||L - \hat{L}||_F}{||L||_F} = \frac{\min\limits_{\text{rank}(\bar{L}) \leq k} ||L - \hat{L}||_F}{||\Lambda_L||_F} < \tau_{\text{LRA}}.$$

The above equation reduces to

$$\frac{\sqrt{\sum\limits_{r=k+1}^{N_{ev}} \overline{\lambda}_r^2}}{\sqrt{\sum\limits_{r=1}^{N_{ev}} \overline{\lambda}_r^2}} < \tau_{\text{LRA}}. \tag{3.7}$$

Therefore, the rank of the weighted modal structural damping matrix, N_{LRA} is

$$N_{\text{LRA}}(\tau_{\text{LRA}}) := \text{ minimum } k \text{ such that } \sum_{r=k+1}^{N_{ev}} \overline{\lambda}_r^2 < \tau_{\text{LRA}}^2 \sum_{r=1}^{N_{ev}} \overline{\lambda}_r^2. \tag{3.8}$$

The summation on the right hand side of equation (3.8) is constant, therefore, once Λ_L is known, the LRA's rank is trivial to compute as a function of the common tolerance τ_{LRA}.

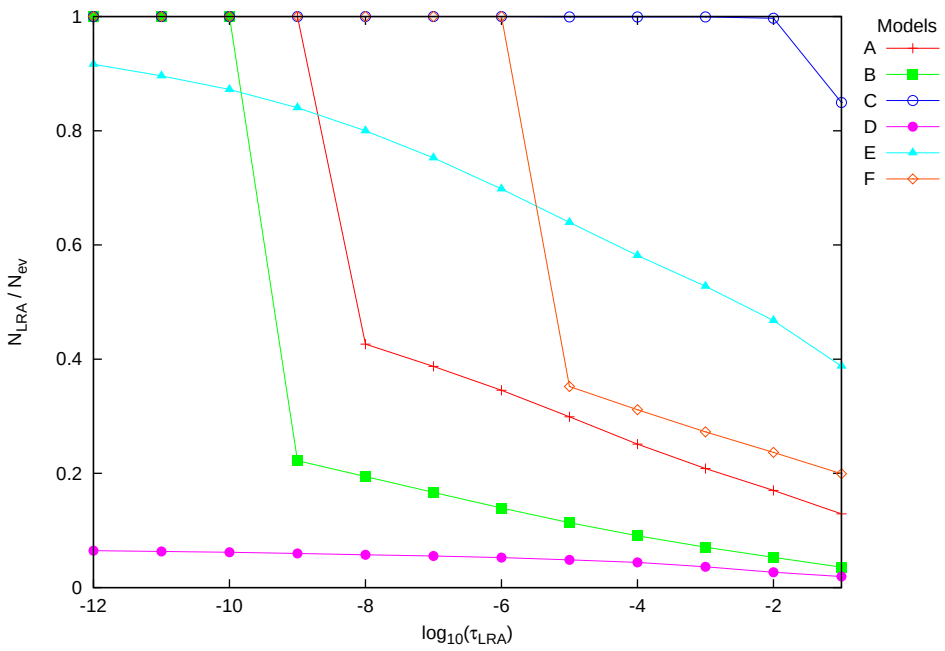


Figure 3.1: The Weighted LRAs for the Industrial Test Suite

Figure 3.1 shows how the ranks of the modal structural damping matrices in the Industrial Test Suite (as a ratio of each model's N_{ev}) decrease as τ_{LRA} increases. For some models, such as Model B and Model D, a very stringent τ_{LRA} produces a very low rank approximation. In contrast, Model C is required to be represented as full rank unless τ_{LRA} is set to a large enough value to result in an inaccurate approximation.

Let the LRA of K_s be represented by $\hat{K}_s$. The weighting matrices transform $\hat{L}$ into $\hat{K}_s$ through the following:

$$\hat{K}_s = (W^{-1}\hat{\Phi}_L)\hat{\Lambda}_L(W^{-1}\hat{\Phi}_L)^T. \tag{3.9}$$

The modal structural damping matrix is now written as

$$K_s \approx \hat{K}_s = J_s \hat{\Lambda}_L J_s^T \tag{3.10}$$

where the non-orthogonal matrix $J_s = W^{-1}\hat{\Phi}_L$ is $N_{ev} \times N_{\text{LRA}}$. In the low rank approach, the matrix K_s is replaced with its LRA, $\hat{K}_s$, whose rank is N_{LRA}.

3.3 Solving Modal FRPs using the Low Rank Approach

In order to determine the rank of K_s, the eigenvalues of the weighted LRA of K_s must be computed. The most effective way of accomplishing this task is to first use Householder reflections to tridiagonalize the real symmetric matrix L from equation (3.4), where W is defined in equation (3.3). This yields the real symmetric tridiagonal matrix

$$T := Q^T L Q, \tag{3.11}$$

where Q is an orthogonal matrix containing the product of the Householder reflections used to annihilate entries outside the tridiagonal. Householder reduction is performed in the LAPACK routine, DSYTRD, with the FLOP cost of $\frac{4}{3}N_{ev}^3$. Since equation (3.11) represents a similarity transformation, the eigenvalues of T are the same as those of L.

If L has a very small N_{LRA}, then shortly after N_{LRA} rows and columns of L have been annihilated, the unreduced matrix will be zero. A potentially substantial savings in FLOPs can be gained if this situation is detected and used to stop the tridiagonal reduction process early.

The eigenvalue decomposition of T is

$$T = \Phi_T \Lambda_L \Phi_T^T.$$

which can be computed by the MR^3 algorithm [17] with $O(N_{ev}^2)$ FLOPs. The entries in Λ_L and columns of Φ_T are sorted according to the absolute value of the eigenvalues. Then, given τ_{LRA}, equation (3.8) is used to determine N_{LRA} very inexpensively. The first N_{LRA} columns of Φ_T form the truncated $\hat{\Phi}_T$ and the corresponding rows and columns of Λ_L are used to form $\hat{\Lambda}_L$, so that $T \approx \hat{T} = \hat{\Phi}_T \hat{\Lambda}_L \hat{\Phi}_T^T$. The matrix J_s is recovered through

$$J_s = W^{-1} \left(Q \hat{\Phi}_T \right). \tag{3.12}$$

The most costly operation in equation (3.12) is the backtransformation step, contained within the parentheses, which can be computed using the LAPACK routine, DORMTR. From Table 1.2, its FLOP cost is $2N_{ev}^2 N_{\text{LRA}} + N_{ev} N_{\text{LRA}}$. The total FLOP cost of finding the LRA of the weighted modal structural damping matrix is

$$\text{C}_{\text{LRA}} = \frac{4}{3} N_{ev}^3 + 2 N_{ev}^2 N_{\text{LRA}}. \tag{3.13}$$

Now, using the LRA of K_s, equation (3.2) becomes

$$\left[-\omega^2 I + (1 + i\gamma)\Lambda + i J_s \hat{\Lambda}_L J_s^T \right] X_s(\omega) = F_s(\omega). \tag{3.14}$$

This modal FRP can be solved efficiently using the SMW formula in equation (2.5), with the substitutions

$$D(\omega) := -\omega^2 I + (1 + i\gamma)\Lambda \in \mathbb{C}^{N_{ev} \times N_{ev}},$$

$$P = R := J_s \in \mathbb{R}^{N_{ev} \times N_k}, \text{ and}$$

$$\Sigma := i\hat{\Lambda}_L \in \mathbb{C}^{N_k \times N_k},$$

Step	Task	FLOP Cost
(1)	$D := D(\omega)^{-1}$, $\Sigma := \Sigma(\omega)^{1/2}$	$N_{ev} + N_k$
(2)	$G_1 := P^T D F_s$	$4N_{ev}N_cN_k + 6N_{ev}N_c$
(3)	$G_1 := \Sigma G_1$	$6N_kN_c$
(4)	$G_2 := P^T D P$	$2N_{ev}N_k(N_k+1)$
(5)	$G_2 := I + \Sigma G_2 \Sigma$	$6N_k^2 + N_k$
(6)	$G_1 := G_2^{-1} G_1$	$\frac{4}{3}N_k^3 + 8N_k^2N_c$
(7)	$G_1 := \Sigma G_1$	$6N_kN_c$
(8)	$F_s := PG_1 - F_s$	$4N_{ev}N_kN_c$
(9)	$X_s := -DF_s$	$6N_{ev}N_c$
Total	$X_s := (D + P\Sigma P^T)^{-1} F_s$	$\frac{4}{3}N_k^3 + 2N_{ev}N_k^2 + 8N_cN_k^2 + 8N_{ev}N_cN_k$

Table 3.2: Symmetric SMW Operations per Frequency for Low Rank Approach

where $N_k = N_{\text{LRA}}$. The solution to equation (3.14) is

$$\begin{aligned} X_s(\omega) &= (D(\omega) + P\Sigma P^T)^{-1} F_s(\omega) \\ &= [D^{-1} - D^{-1}P\Sigma^{1/2}(I + \Sigma^{1/2}P^T D^{-1} P \Sigma^{1/2})^{-1}\Sigma^{1/2}P^T D^{-1}]F_s. \end{aligned} \tag{3.15}$$

$D(\omega)$ and Σ are complex diagonal matrices, so the operations D^{-1} and $\Sigma^{1/2}$ are very inexpensive. All of the matrices, except for D, are frequency independent. But, because $D(\omega)$ is used throughout equation (3.15), the solution must be computed for every frequency. The work required to execute the matrix operations in equation (3.15) is broken up into steps with each corresponding FLOP cost in Table 3.2. The steps are chosen to minimize the amount of complex arithmetic. This means that Steps (2), (4), and (8) involve matrix multiplications of purely real matrices by complex matrices, which are not standard BLAS subprograms. Also, the G_2 matrix in

Steps (4), (5), and (6), is symmetric and only the diagonal and either the upper or lower triangle of the matrix is maintained.

The total FLOP cost of computing the solution of the modal FRP of equation (3.2) is equal to the sum of the significant operations in computing the LRA of K_s (dominated by the tridiagonal reduction and backtransformation steps) and the cost of computing the response at a single frequency, listed in Table 3.2, multiplied by the number of frequencies:

$$\text{Cost} = \text{C}_{\text{LRA}} + N_{freq}\left(\frac{4}{3}N_k^3 + 2N_{ev}N_k^2 + 8N_cN_k^2 + 8N_{ev}N_cN_k\right). \tag{3.16}$$

When the rank of K_s is low, the cost of computing the LRA is insignificant compared to the total cost of finding the responses at every frequency.

3.3.1 Low Rank Structural Damping and Viscous Damping

When a model has both structural and viscous damping, and K_s is represented with an LRA, it is possible to use the SMW formula to solve the FRP efficiently. The two damping matrices are factored according to equations (2.3) and (3.10). Then, equation (1.13) is reformulated as

$$\left\{-\omega^2 I + (1+i\gamma)\Lambda + \begin{bmatrix} U & J_s \end{bmatrix} \begin{bmatrix} i\omega\Sigma_{\mathcal{B}} & \\ & i\hat{\Lambda}_L \end{bmatrix} \begin{bmatrix} V^T \\ J_s^T \end{bmatrix}\right\} X_s(\omega) = F_s(\omega). \tag{3.17}$$

Step	Task	FLOP Cost
(1)	$D := D(\omega)^{-1}$, $\Sigma := \Sigma(\omega)^{1/2}$	$N_{ev} + N_k$
(2)	$G_1 := R^T D F_s$	$4N_{ev}N_cN_k + 6N_{ev}N_c$
(3)	$G_1 := \Sigma G_1$	$6N_kN_c$
(4)	$G_2 := R^T D P$	$2N_{ev}N_k(2N_k+1)$
(5)	$G_2 := I + \Sigma G_2 \Sigma$	$12N_k^2 + N_k$
(6)	$G_1 := G_2^{-1} G_1$	$\frac{8}{3}N_k^3 + 8N_k^2N_c$
(7)	$G_1 := \Sigma G_1$	$6N_kN_c$
(8)	$F_s := PG_1 - F_s$	$4N_{ev}N_kN_c$
(9)	$X_s := -DF_s$	$6N_{ev}N_c$
Total	$X_s := (D + P\Sigma R^T)^{-1} F_s$	$\frac{8}{3}N_k^3 + 4N_{ev}N_k^2 + 8N_cN_k^2 + 8N_{ev}N_cN_k$

Table 3.3: Nonsymmetric SMW Operations per Frequency for Low Rank Approach

Equation (3.17) can then be solved using the SMW formula with the substitutions

$$\begin{aligned} D(\omega) &:= -\omega^2 I + (1 + i\gamma)\Lambda \in \mathbb{C}^{N_{ev} \times N_{ev}}, \\ P &:= \begin{bmatrix} U & J_s \end{bmatrix} \in \mathbb{R}^{N_{ev} \times N_k}, \\ R &:= \begin{bmatrix} V & J_s \end{bmatrix} \in \mathbb{R}^{N_{ev} \times N_k}, \text{ and} \\ \Sigma(\omega) &:= \begin{bmatrix} i\omega\Sigma_b & \\ & i\hat{\Lambda}_L \end{bmatrix} \in \mathbb{C}^{N_k \times N_k} \end{aligned} \tag{3.18}$$

where $N_k = \text{rank}(\mathcal{B}_{nz}) + N_{\text{LRA}}$. If the modal viscous damping matrix is symmetric, $P = R$, and the response at each frequency is computed using the steps outlined in Table 3.2. The factorization in equation (2.3) is very inexpensive, since $\mathcal{B}_{nz}$ is a very small matrix. The total FLOP cost is given in equation (3.16), where N_k is defined as the sum of the ranks of $\mathcal{B}_{nz}$ and K_s.

If the modal viscous damping matrix is not symmetric, $P \neq R$, and the re-

sponse is computed with a different set of steps, outlined in Table 3.3. The total FLOP cost is

$$\text{Cost} = \text{C}_{\text{LRA}} + N_{freq}\left(\frac{8}{3}N_k^3 + 4N_{ev}N_k^2 + 8N_cN_k^2 + 8N_{ev}N_cN_k\right). \tag{3.19}$$

3.4 Using the Low Rank Approach with Structure-Fluid Interaction

The low rank approach can be extended to solve modal FRPs with structure and fluid interaction in one of two ways, depending on the form of the acoustic fluid matrices.

3.4.1 General Acoustic Fluid Formulation

A partitioned solution approach is used when the modal fluid "damping" matrix is full, which means that Z_f in equation (1.17) is also full. First, the matrix equation is cast as a system of equations:

$$\begin{aligned} Z_sX_s + i\omega AX_f &= F_s \\ i\omega A^TX_s + Z_fX_f &= F_f. \end{aligned} \tag{3.20}$$

All of the matrices in the system above, except the area matrix A, are frequency-dependent. The first equation is rearranged into

$$X_s = Z_s^{-1}\left(F_s - i\omega AX_f\right) \tag{3.21}$$

and used in the second equation:

$$\left(Z_f + \omega^2 A^TZ_s^{-1}A\right)X_f = F_f - i\omega A^TZ_s^{-1}F_s.$$

The above equation contains the products $Z_s^{-1}A$ and $Z_s^{-1}F_s$ which can be combined into $Z_s^{-1}\begin{bmatrix}A & F_s\end{bmatrix}$. These products can then be computed using the SMW formula once per frequency with the substitutions in equation (3.18) with $N_k = \text{rank}(\mathcal{B}_{nz}) + N_{\text{LRA}}$. Let $\bar{Z}_f := Z_f + \omega^2 A^T Z_s^{-1} A$ and $\bar{F}_f := F_f - i\omega A^T Z_s^{-1} F_s$. Then, the acoustic fluid solution is

$$X_f = \bar{Z}_f^{-1} \bar{F}_f. \tag{3.22}$$

The number of acoustic fluid modes is typically much smaller than the number of structural modes; therefore, equation (3.22) is relatively inexpensive to solve. After X_f is determined at a particular frequency, equation (3.21) is used to solve for X_s. Table 3.4 outlines all of the steps and the cost of finding the structural and acoustic fluid responses at a particular frequency, when modal viscous damping is symmetric. The FLOP cost of finding the structural and acoustic fluid responses at a single frequency, considering only the cubic terms is:

$$\begin{aligned} C_{\text{LR1}} = {} & N_{ev}(8N_f N_k + 8N_c N_k + 12N_f N_c) \\ & + N_k^2 \left(\frac{4}{3} N_k + 2N_{ev} + 8N_f + 8N_c \right) \\ & + N_f^2 \left(4N_{ev} + \frac{4}{3} N_f + 8N_c \right). \end{aligned} \tag{3.23}$$

The total FLOP cost is

$$\text{Cost} = \text{C}_{\text{LRA}} + N_{freq} C_{\text{LR1}}. \tag{3.24}$$

3.4.2 Diagonal Acoustic Fluid Formulation

If the FE fluid "damping" is Rayleigh (proportional) damping, the SMW formula can be used to solve the modal FRP efficiently. Let the FE fluid "damping"

Step	Task	FLOP Cost
(1)	$D := D(\omega)^{-1}$, $\Sigma := \Sigma(\omega)^{1/2}$	$N_{ev}+N_k$
(2)	$G_3 := P^T DA$	$2N_{ev}N_f(2N_k+1)$
(3)	$G_4 := P^T DF_s$	$4N_{ev}N_cN_k+6N_{ev}N_c$
(4)	$\begin{bmatrix} G_3 & G_4 \end{bmatrix} = \Sigma \begin{bmatrix} G_3 & G_4 \end{bmatrix}$	$6N_k(N_f+N_c)$
(5)	$G_2 := P^T DP$	$2N_{ev}N_k(N_k+1)$
(6)	$G_2 := I + \Sigma G_2 \Sigma$	$6N_k^2+N_k$
(7)	$\begin{bmatrix} G_3 & G_4 \end{bmatrix} := G_2^{-1} \begin{bmatrix} G_3 & G_4 \end{bmatrix}$	$\frac{4}{3}N_k^3+8N_k^2(N_f+N_c)$
(8)	$\begin{bmatrix} G_3 & G_4 \end{bmatrix} := \Sigma \begin{bmatrix} G_3 & G_4 \end{bmatrix}$	$6N_k(N_f+N_c)$
(9)	$\begin{bmatrix} \bar{A} & F_s \end{bmatrix} := P \begin{bmatrix} G_3 & G_4 \end{bmatrix} - \begin{bmatrix} A & F_s \end{bmatrix}$	$4N_{ev}N_k(N_f+N_c)$
(10)	$\bar{A} := -D\bar{A}, F_s := -DF_s$	$6N_{ev}(N_f+N_c)$
(11)	$Z_f := \omega^2 I_f - i\omega C_f - \Lambda_f + \omega^2 A^T \bar{A}$	$4N_f^2(N_{ev}+1)$
(12)	$F_f := F_f - i\omega A^T F_s$	$4N_fN_{ev}N_c$
(13)	$X_f := Z_f^{-1} F_f$	$\frac{4}{3}N_f^3+8N_f^2N_c$
(14)	$X_s := F_s - i\omega \bar{A} X_f$	$8N_{ev}N_fN_c$

Table 3.4: Low Rank Approach Operations per Frequency for General Fluid Matrices

in equation (1.2) be represented as a linear combination of the FE fluid "mass" and "stiffness" matrices.

$$\mathcal{C} := \beta_0 \mathcal{E} + \beta_1 \mathcal{H}, \tag{3.25}$$

where β_0 and β_1 are two scalar values. (If the FE fluid "damping" is not modeled at all, both scalars equal 0.) The modal FRP is represented by equation (1.17), where Z_s is the coefficient matrix from equation (1.13) and

$$Z_f = (\omega^2 - i\omega\beta_0)I + (1 - i\omega\beta_1)\Lambda_f. \tag{3.26}$$

Since Z_f is diagonal, the modal FRP in equation (1.16) can be solved using the SMW formula. Using the system of equations in equation (3.20), rearrange the second equation into

$$X_f = Z_f^{-1}\left(F_f - i\omega A^T X_s\right) \tag{3.27}$$

and use it in the first equation:

$$\left(Z_s + \omega^2 A Z_f^{-1} A^T\right) X_s = F_s - i\omega A Z_f^{-1} F_f. \tag{3.28}$$

The $AZ_f^{-1}A^T$ term is a rank-N_f update to Z_s. From this perspective, the SMW formula can be used to solve for X_s with the substitutions:

$$\begin{aligned}
D(\omega) &:= -\omega^2 I + (1+i\gamma)\Lambda \in \mathbb{C}^{N_{ev}\times N_{ev}},\\
P &:= \begin{bmatrix} J_s & U & A \end{bmatrix} \in \mathbb{R}^{N_{ev}\times N_k},\\
\Sigma(\omega) &:= \begin{bmatrix} i\hat{\Lambda}_L & & \\ & i\omega\Sigma_B & \\ & & \omega^2 Z_f^{-1}(\omega) \end{bmatrix} \in \mathbb{C}^{N_k\times N_k}, \text{ and}\\
F(\omega) &:= F_s - i\omega A Z_f^{-1} F_f \in \mathbb{C}^{N_{ev}\times N_c}
\end{aligned}$$

when the viscous damping matrix is symmetric and where $N_k = \text{rank}(\mathcal{B}_{nz}) + N_{\text{LRA}} + N_f$. The steps needed to compute X_s and X_f for the symmetric $\mathcal{B}_{nz}$ case are outlined in Table 3.5. The FLOP cost of computing the response at a single frequency is

$$\begin{aligned}
C_{\text{LR2}} = {} & N_{ev}(8N_f N_c + 8N_c N_k)\\
& + N_k^2\left(\frac{4}{3}N_k + 2N_{ev} + 8N_c\right).
\end{aligned} \tag{3.29}$$

The total FLOP cost is

$$\text{Cost} = \text{C}_{\text{LRA}} + N_{freq} C_{\text{LR2}}. \tag{3.30}$$

Step	Task	FLOP Cost
(1)	$Z_f := \omega^2 I_f - i\omega C_f - \Lambda_f$	$6N_f$
(2)	$F_f := Z_f^{-1} F_f$	$6N_f N_c$
(3)	$F_s := F_s - i\omega A F_f$	$4N_{ev} N_f N_c$
(4)	$D := D(\omega)^{-1}$, $\Sigma := \Sigma(\omega)^{1/2}$	$N_{ev}+N_k$
(5)	$G_1 := P^T D F_s$	$4N_{ev} N_c N_k+6N_{ev} N_c$
(6)	$G_1 := \Sigma G_1$	$6N_k N_c$
(7)	$G_2 := P^T D P$	$2N_{ev} N_k(N_k+1)$
(8)	$G_2 := I + \Sigma G_2 \Sigma$	$6N_k^2+N_k$
(9)	$G_1 := G_2^{-1} G_1$	$\frac{4}{3}N_k^3+8N_k^2 N_c$
(10)	$G_1 := \Sigma G_1$	$6N_k N_c$
(11)	$F_s := P G_1 - F_s$	$4N_{ev} N_k N_c$
(12)	$X_s := -D F_s$	$6N_{ev} N_c$
(13)	$X_f := A^T X_s$	$4N_f N_{ev} N_c$
(14)	$X_f := Z_f^{-1} X_f$	$6N_f N_c$
(15)	$X_f := F_f - i\omega X_f$	$5N_{ev} N_c$

Table 3.5: Low Rank Approach Operations per Frequency for Diagonal Fluid Matrices

Chapter 4

Complex Symmetric Approach

If the rank of the modal structural damping matrix is high, it may be preferable to treat the matrix as if it has full rank and use the complex symmetric approach described in Section 2.4 to solve the modal FRP. However, before this approach can be used, the loss of precision in Q and T demonstrated in Table 2.1 from tridiagonalizing C (where C is defined in equation (2.7)) must be addressed.

Closer inspection of the $\mathbf{x}^T\mathbf{x}$ inner products, which are used to form the complex orthogonal (CO) reflections, makes the underlying issue more apparent. If $\mathbf{x} = \mathbf{y} + i\mathbf{z}$, then

$$\mathbf{x}^T\mathbf{x} = \mathbf{y}^T\mathbf{y} - \mathbf{z}^T\mathbf{z} + i2\mathbf{y}^T\mathbf{z}, \tag{4.1}$$

and the real part of the result can suffer from cancellation error if $\mathbf{y}^T\mathbf{y}$ and $\mathbf{z}^T\mathbf{z}$ are nearly equal. The precision of the difference between the two is reduced by the number of digits of agreement [22]. For example, suppose a machine is able to store floating point numbers with four digits of precision and let a complex vector be defined as

$$\mathbf{x} = \begin{Bmatrix} 10.01 + i10.00 \\ 10.00 - i10.00 \end{Bmatrix}.$$

Then, using the machine's four digit limitation, the following steps compute $\mathbf{x}^T\mathbf{x}$.

$$\begin{aligned}\mathbf{x}^T\mathbf{x} &= [(100.2 + 100.0 - (100.0 + 100.0) + i2(100.1 - 100.0)] \\ &= 0.2 + i0.2\end{aligned}$$

Since there are three digits of agreement between $\mathbf{y}^T\mathbf{y}$ and $\mathbf{z}^T\mathbf{z}$, three digits of precision are lost in the result. The author of this dissertation calls this type of imprecision due to cancellation error a "cancellation event" (CE). A CE occurs when $\mathbf{x}^T\mathbf{x} \approx \mathbf{0}$ in a relative sense, and a metric which is used to detect its existence is derived in the next section.

4.1 The Cancellation Event

CO reflections are used to reduce the complex symmetric matrix C to tridiagonal form. Let a vector $\mathbf{x}$ represent nonzero entries of C at and below a subdiagonal. Then, a CO reflection annihilates the entries in $\mathbf{x}$ below the subdiagonal. By symmetry, the entries to the right of the corresponding superdiagonal are also annihilated simultaneously. It is convenient to scale the reflection vector $\mathbf{v}$, which is used to construct the CO reflection, so that its first entry is 1. This allows the essential part of $\mathbf{v}$ to be stored below the subdiagonal of the column which is being annihilated. In the process of doing this scaling, the amount of cancellation error that is introduced becomes evident, and a means for detecting its presence is derived. After scaling, the vector $\mathbf{v}$ is

$$\mathbf{v} := \frac{1}{x_1 \pm \sqrt{\mathbf{x}^T\mathbf{x}}} \left(\mathbf{x} \pm \hat{\mathbf{e}}_1\sqrt{\mathbf{x}^T\mathbf{x}}\right) \tag{4.2}$$

and the scalar value $\frac{2}{\beta}$ (with $\beta = \mathbf{v}^T\mathbf{v}$) becomes

$$\frac{2}{\beta} = \frac{\left(x_1 \pm \sqrt{\mathbf{x}^T\mathbf{x}}\right)^2}{\mathbf{x}^T\mathbf{x} \pm x_1\sqrt{\mathbf{x}^T\mathbf{x}}} = 1 \pm \frac{x_1}{\sqrt{\mathbf{x}^T\mathbf{x}}}. \tag{4.3}$$

By definition, the CO reflection is orthonormal:

$$H^TH = \left(I - \frac{2}{\beta}\mathbf{v}\mathbf{v}^T\right)\left(I - \frac{2}{\beta}\mathbf{v}\mathbf{v}^T\right) = I - \frac{4}{\beta}\mathbf{v}\mathbf{v}^T + \frac{4}{\beta^2}(\mathbf{v}^T\mathbf{v})\mathbf{v}\mathbf{v}^T = I. \tag{4.4}$$

If there are large entries in H, then the orthonormality property above relies on cancellation to hold true. Thus, the sign in front of the $\sqrt{\mathbf{x}^T\mathbf{x}}$ term is chosen to minimize the magnitude of $\mathbf{v}$ which minimizes the magnitude of entries in H.

When the CO reflection is applied to $\mathbf{x}$, the result is

$$H\mathbf{x} = \left(I - \frac{2}{\beta}\mathbf{v}\mathbf{v}^T\right)\mathbf{x} = \mathbf{x} - \frac{2\mathbf{v}^T\mathbf{x}}{\mathbf{v}^T\mathbf{v}}\mathbf{v} = \mathbf{x} - \mathbf{x} \mp \hat{\mathbf{e}}_1\sqrt{\mathbf{x}^T\mathbf{x}} = \mp\hat{\mathbf{e}}_1\sqrt{\mathbf{x}^T\mathbf{x}} \tag{4.5}$$

which shows that the corresponding subdiagonal entry in the tridiagonal tail becomes $\mp\sqrt{\mathbf{x}^T\mathbf{x}}$. The product $\mathbf{x}^T\mathbf{x}$ is needed to compute $\mathbf{v}$ and $\frac{2}{\beta}$, which are used to compute a CO reflection, and a subdiagonal entry in T. If a lot of precision is lost in the product $\mathbf{x}^T\mathbf{x}$, then equations (2.14) and (2.15) lose their validity.

The loss of precision due to cancellation in the real part of the product $\mathbf{x}^T\mathbf{x}$ in equation (4.1) can be expressed as the ratio of $\mathbf{y}^T\mathbf{y}$ (or $\mathbf{z}^T\mathbf{z}$) to the difference $\mathbf{y}^T\mathbf{y} - \mathbf{z}^T\mathbf{z}$, in which the cancellation has taken place. In the complex quantity $\mathbf{x}^T\mathbf{x}$, if $\mathbf{y}^T\mathbf{z} \gg \mathbf{y}^T\mathbf{y} - \mathbf{z}^T\mathbf{z}$, the imaginary part dominates the real part and the loss of precision in the real part is of less importance. With this in mind, it is appropriate to express the loss of precision due to cancellation as the ratio of $\mathbf{y}^T\mathbf{y}$ (or $\mathbf{z}^T\mathbf{z}$) to the magnitude of $\mathbf{x}^T\mathbf{x}$. A measure of the severity of the loss of precision due to

cancellation is the number of digits lost in a cancellation event, and this is given by the quantity

$$\begin{aligned} \text{CE value} &= \log_{10} \frac{\mathbf{y}^T\mathbf{y}}{\sqrt{(\mathbf{y}^T\mathbf{y} - \mathbf{z}^T\mathbf{z})^2 + (2\mathbf{y}^T\mathbf{z})^2}} \\ &= \log_{10} \frac{\mathbf{y}^T\mathbf{y}}{|\mathbf{x}^T\mathbf{x}|}. \end{aligned} \tag{4.6}$$

When the CE value is greater than a chosen CE tolerance, τ_{CE}, which represents the acceptable number of digits lost, a CE is said to be encountered in forming the product $\mathbf{x}^T\mathbf{x}$.

The CE value calculated in equation (4.6) does not depend on the numerical precision of the real or imaginary parts of $\mathbf{x}$. Furthermore, if, when a CE is encountered, a higher precision is used to compute $\mathbf{x}^T\mathbf{x}$, then $\mathbf{x}^T\mathbf{x}$ will still be approximately zero. When $\mathbf{x}^T\mathbf{x}$ is approximately zero, then small variations in the computation of $\mathbf{x}^T\mathbf{x}$ produces large variations in the value of $\frac{2}{\beta}$ in equation (4.3), and the entries in the resulting CO reflection H can become very large. If there are very large entries in H, then the accuracy with which $H^TH = I$ is satisfied will be limited because of the cancellation that will be required to produce only ones and zeros in I. This loss of accuracy will also be seen in an orthogonal matrix formed as a product of reflection matrices.

To understand the character of the CE values encountered during the tridiagonal reduction of a large matrix, it is helpful to plot graphs of these values versus the column number. Figure 4.1 shows the CE values encountered while tridiagonalizing the complex symmetric matrix from Model E. For this implementation, the tridiago-

nal reduction progresses from the lower-right corner up toward the upper-left corner. It is evident that the cancellation event is not an infrequent occurrence.

Figure 4.2 shows the distribution of cancellation event values encountered during the tridiagonal reduction of a complex symmetric matrix containing random entries. This matrix produces a graph that also has a wide variation of CE values. The effect of CEs is not negligible. The tridiagonal reduction procedure exhibits both instances of very high cancellation and from frequent occurrences of low cancellation. The cancellation event has the character of a chance event, and each row/column's CE value cannot be predicted before the tridiagonal reduction begins.

The matrix C has a complex diagonal and is purely imaginary outside of its diagonal. The first row/column in C which is annihilated has no cancellation since it is purely imaginary. The first CO reflection vector and scalar, defined in equations (4.2) and (4.3), are purely real because the $\mathbf{x}$ which is used to create them is purely imaginary. This means that the CO reflection which annihilates the first row/column of C is purely real. After this CO reflection is applied to C, the real part of the non-diagonal entries in the new C matrix are different from zero, but remain dominated by their imaginary counterparts. After several rows/columns are annihilated, the real and imaginary parts of C are more similarly matched, and it is only then that a cancellation event is encountered. The figures reflect this observation. At the end of the tridiagonalization process, the annihilated vectors are shorter and are less likely to encounter agreement to many digits; therefore, the end of the tridiagonalization process is usually free of cancellation, which is also reflected in the figures.

Algorithm 1, described in Section 2.4.3, is useful for avoiding infrequent break-

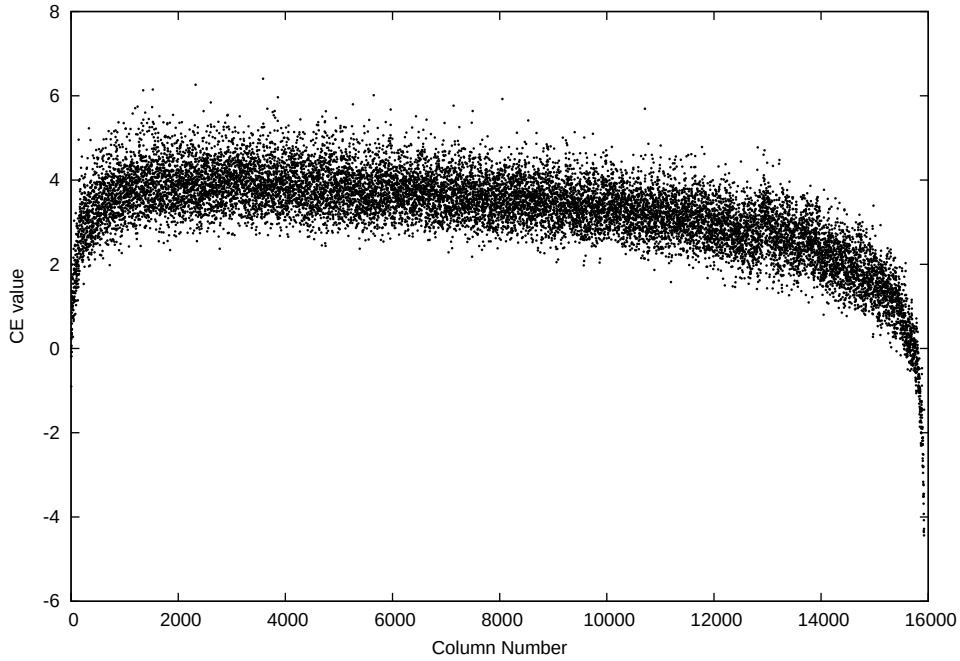


Figure 4.1: CE Values for C from Model E

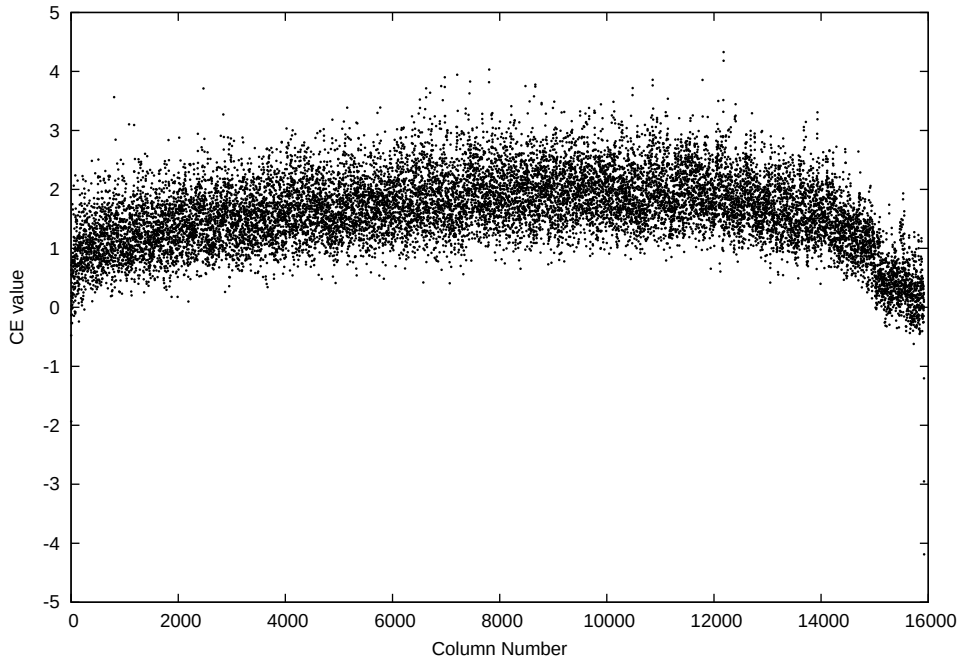


Figure 4.2: CE Values for C with Random Entries

downs ($\mathbf{x}^T\mathbf{x} = 0$ when $\mathbf{x} \neq \mathbf{0}$). However, cancellation events are not infrequent and since the first step in the algorithm is a variant of the QL algorithm, the repeated use of Algorithm 1 causes the subdiagonal entries in the tridiagonal tail to approach zero. As the tridiagonal tail becomes more diagonalized, the algorithm becomes less able to remove CEs.

4.2 Characteristics of CE Removal Methods

The method presented by Bar-On and Ryaboy is ineffective at removing many CEs encountered while tridiagonalizing complex symmetric matrices from automobile models. This section identifies characteristics that would be needed for a CE removal method to be successful. With these characteristics in mind, an effective CE removal method has been developed and is presented in the next section.

Application of LaBudde's Method for real matrices to tridiagonal reduction of complex symmetric matrices entails annihilating the part of each successive row/column of C outside the tridiagonal using a CO reflection. In the complex symmetric case, just before forming and applying every CO reflection, the CE value associated with forming the $\mathbf{x}^T\mathbf{x}$ product for the reflection must be checked, according to equation (4.6). If the number of digits lost in computing $\mathbf{x}^T\mathbf{x}$ is greater than the CE tolerance, then the row/column possesses a CE. Before any more work is done to tridiagonalize the matrix, the CE must be removed. The Implicit Q Theorem [23] provides some valuable information about symmetric tridiagonal reduction which must be kept in mind in developing a successful CE removal strategy.

Theorem 3 (**Implicit Q Theorem**). *Suppose* $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ *and* $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$

are orthogonal matrices such that $Q^T AQ = T$ *and* $V^T AV = S$ *are unreduced, tridiagonal matrices and* A *is an* $n \times n$ *symmetric matrix. If* $\mathbf{v}_1 = \mathbf{q}_1$*, then* $\mathbf{v}_i = \pm\mathbf{q}_i$ *and* $t_{i,i-1} = \pm s_{i,i-1}$ *for* $i = 2 \ldots n$*.*

The first column $\mathbf{q}_1$ of Q may be chosen arbitrarily, but it determines the remaining columns of Q, to within a sign, and the subdiagonal of T, to within a sign. Equation (4.5) shows that when entries in a row/column are annihilated with a CO reflection, the resulting subdiagonal entry in the tridiagonal matrix is $\mp\sqrt{\mathbf{x}^T\mathbf{x}}$. Therefore, if $\mathbf{x}^T\mathbf{x}$ is found to be too small for a particular row/column, it can only be altered by modifying the first column $\mathbf{q}_1$ in some way. The first step of Algorithm 1 is a complex QL sweep which changes the first column of Q. This change enables the algorithm to modify the value of $\mathbf{x}^T\mathbf{x}$ and the CE can be removed. However, its tendency to annihilate the subdiagonal entries in the tridiagonal tail makes it impractical for repeated use.

It is possible to change the CE value of $\mathbf{x}$ by changing the numerator of the fraction in equation (4.6). This can be accomplished by reverting the unreduced matrix to its state before the previous row/column was annihilated and re-annihilating the previous row/column using an alternate annihilation scheme. However, this method will not modify the first column $\mathbf{q}_1$, and the denominator of the fraction, $\mathbf{x}^T\mathbf{x}$, which corresponds to the square of the subdiagonal entry which remains after $\mathbf{x}$ is annihilated, will remain unchanged. If $\mathbf{x}^T\mathbf{x}$ is very small, then this method will never be successful at removing CEs.

If a method uses a bulge chasing (implicit QR) algorithm [42], it is likely that

the complex sines and cosines (the values that make up the CO rotations as seen in equation (2.10)) will have magnitudes greater than 1. The sines and cosines are used to form the updated Q that has a new first column. So, in any CE removal method, the magnitudes of entries in Q must be monitored to ensure that those entries do not become huge, because if they do, then Q will rely on cancellation to keep equation (2.15) valid.

Finally, a CE removal method must reliably be able to remove the CE in one of its first attempts. If many attempts must be made for each CE, the time required for this will cause the tridiagonal reduction to become a greater bottleneck for finding the eigenvalue decomposition of C.

4.3 A New Cancellation Event Removal Method

Using the characteristics described in section 4.2, a new method is presented which removes cancellation events efficiently. This method is not a variation of the QR algorithm; therefore it may not share the inherent tendency of the QR algorithm to zero subdiagonal and superdiagonal entries in the tridiagonal tail. Nearly every time this method is used, the CE is removed on its first attempt.

Suppose the tridiagonal reduction of an $N_{ev} \times N_{ev}$ complex, symmetric matrix C progresses from the top-left corner of C to the bottom-right corner of C, and during this process a CE is found at row/column κ. Let the matrix at this state be

represented with $\bar{C}$. The form of $\bar{C}$ is

$$\bar{C} = \begin{bmatrix}
\alpha_1 & \beta_1 & & & & & & & \cdots & 0 \\
\beta_1 & \alpha_2 & \beta_2 & & & & & & & \vdots \\
 & \beta_2 & \ddots & \ddots & & & & & & \\
 & & \ddots & \alpha_{\kappa-1} & \beta_{\kappa-1} & & & & & \\
 & & & \beta_{\kappa-1} & \alpha_\kappa & x_1 & x_2 & x_3 & \cdots & x_r \\
 & & & & x_1 & \times & \times & \times & \cdots & \times \\
 & & & & x_2 & \times & \times & \times & \cdots & \times \\
 & & & & x_3 & \times & \times & \times & \cdots & \times \\
\vdots & & & & \vdots & \vdots & \vdots & \vdots & \ddots & \times \\
0 & \cdots & & & x_r & \times & \times & \times & \times & \times
\end{bmatrix} \tag{4.7}$$

where $r = N_{ev} - \kappa$. Let $\bar{C}$ be partitioned as

$$\bar{C} = \begin{bmatrix} T_1 & \hat{O}_1 & O \\ \hat{O}_1^T & T_2 & \hat{O}_2 \\ O & \hat{O}_2^T & \bar{C}' \end{bmatrix}. \tag{4.8}$$

The square matrix $\bar{C}'$ is the unreduced part of $\bar{C}$ and is of dimension $(N_{ev} - \kappa + 1) \times (N_{ev} - \kappa + 1)$. The cancellation event is situated in the first row/column of $\bar{C}'$. The tridiagonal tail is separated into a small $p \times p$ tridiagonal matrix T_1 at the beginning of the tail, and tridiagonal matrix T_2 which is of dimension $(\kappa - p - 1)$ and contains the remainder of the tail. The O matrices are null, and submatrices $\hat{O}_1$ and $\hat{O}_2$ each have only one nonzero entry, having the form

$$\hat{O}_1 = \begin{bmatrix} \tilde{0} & O \\ \bar{C}_{p,p+1} & \tilde{0}^T \end{bmatrix} \text{ and } \hat{O}_2 = \begin{bmatrix} \tilde{0} & O \\ \bar{C}_{\kappa-1,\kappa} & \tilde{0}^T \end{bmatrix}.$$

Since the CO reflections are applied symmetrically, $\bar{C}'$, T_1, and T_2 are symmetric. The following subsections describe the steps in the new method which together remove a cancellation event in the first row/column of $\bar{C}'$.

4.3.1 Step 1: The Small Tridiagonal Eigenvalue Decomposition

The first step in the CE removal method is to find the eigenvalue decomposition of T_1

$$T_1 \Phi_{T_1} = \Phi_{T_1} \Lambda_{T_1}$$

and scale Φ_{T_1} so that $\Phi_{T_1}^T \Phi_{T_1} = I$ and $\Phi_{T_1}^T T_1 \Phi_{T_1} = \Lambda_{T_1}$. The EVD above is determined using a procedure developed by Cullum and Willoughby [15]. Define a complex orthogonal matrix $\Upsilon_\kappa \in \mathbb{C}^{N_{ev} \times N_{ev}}$ that has the form

$$\Upsilon_\kappa = \begin{bmatrix} \Phi_{T_1} & O \\ O & I \end{bmatrix}. \tag{4.9}$$

When the matrix $\bar{C}$ is transformed into a similar matrix $\bar{C}^{(1)}$ through

$$\bar{C}^{(1)} := \Upsilon_\kappa^T \bar{C} \Upsilon_\kappa, \tag{4.10}$$

because of the orthogonality of eigenvectors in Φ_{T_1} with respect to T_1, the upper-left corner of $\bar{C}^{(1)}$ has an arrow shape. The T_1 matrix has been diagonalized, with entries equal to the eigenvalues of T_1. The first column of $\hat{O}_1$ is now completely nonzero, equal to the last column of $\Phi_{T_1}^T$, scaled by $\bar{C}_{p,p+1}$. Graphically, the first $(\kappa - 1)$ rows

and columns of $\bar{C}^{(1)}$ have the form

$$
\begin{array}{c}
 \\
 \\
 \\
 \\
p \\
 \\
 \\
 \\
\kappa-1
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc}
 & & & & p & & & & \kappa-1
\end{array} \\
\begin{bmatrix}
\lambda_1 & 0 & & & & + & & & \\
0 & \lambda_2 & 0 & & & + & & & \\
 & 0 & \ddots & \ddots & & \vdots & & & \\
 & & \ddots & \lambda_{p-1} & 0 & + & & & \\
 & & & 0 & \lambda_p & \times & & & \\
+ & + & \cdots & + & \times & \times & \times & & \\
 & & & & & \times & \times & \ddots & \\
 & & & & & & \ddots & \ddots & \times \\
 & & & & & & & \times & \times
\end{bmatrix}
\end{array}
$$

where "$\times$" represents a nonzero entry. Entries marked with "0" were nonzero but have become zero and "+" represents new nonzero entries. The $\lambda_1, \lambda_2, \ldots, \lambda_p$ values are the diagonal entries of Λ_{T_1}.

4.3.2 Step 2: Annihilate Most of the Arrow

Zha's algorithm [46] is a two-way chasing algorithm that transforms real, symmetric arrowhead matrices to tridiagonal form using Givens rotations. The second step in the CE removal method uses this algorithm, modified to use CO rotations, to convert the arrowhead form to nearly tridiagonal form. Completely restoring the tridiagonal matrix would have no effect on removing CEs, therefore, all but the final step of Zha's algorithm is used. The series of rotations is halted when there is one bulge entry remaining. Let the product of the CO rotation matrices used in this step

be called Ω_κ, and the new similar matrix, $\bar{C}^{(2)}$, be defined as

$$\bar{C}^{(2)} = \Omega_\kappa^T \bar{C}^{(1)} \Omega_\kappa. \tag{4.11}$$

Now, the first $(\kappa - 1)$ rows and columns of $\bar{C}^{(2)}$ have the form

$$\begin{array}{c} \\ \\ \\ \\ \\ p \\ \\ \\ \\ \kappa-1 \end{array}
\begin{array}{c}
\begin{array}{ccccccccc} & & & & p & & & & \kappa-1 \end{array} \\
\left[\begin{array}{ccccccccc}
\times & \times & & & & & & & \\
\times & \ddots & \ddots & & & & & & \\
 & \ddots & \times & \times & & & & & \\
 & & \times & \times & 0 & + & & & \\
 & & & 0 & \times & \times & & & \\
 & & & + & \times & \times & \times & & \\
 & & & & & \times & \times & \ddots & \\
 & & & & & & \ddots & \ddots & \times \\
 & & & & & & & \times & \times
\end{array}\right]
\end{array}.$$

The form of the matrix resembles a tridiagonal matrix at a step during a bulge-chasing algorithm. Zha's algorithm chases bulges toward the beginning of the tridiagonal tail. The "chase" must now be directed toward the end of the tridiagonal tail; however, because of the "0" next to the bulge, the first CO rotation used to chase in the new direction behaves like a permutation matrix. The matrices Υ_κ in Step 1 and Ω_κ in Step 2 successfully modify the first column $\mathbf{q}_1$ of Q which means that the product $\mathbf{x}^T\mathbf{x}$ is modified due to the Implicit Q Theorem.

4.3.3 Step 3: Permute Rows and Columns p and $p+1$

Let a permutation matrix P_κ be defined as an $N_{ev} \times N_{ev}$ identity matrix, but with rows and columns p and $p+1$ swapped:

$$P_\kappa = \begin{array}{c} \\ p \\ p+1 \\ \\ N_{ev} \end{array} \begin{array}{c} \begin{array}{cccccc} & & p & p+1 & & N_{ev} \end{array} \\ \left[\begin{array}{cccccc} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & \cdots & 0 \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{array}\right] \end{array}.$$

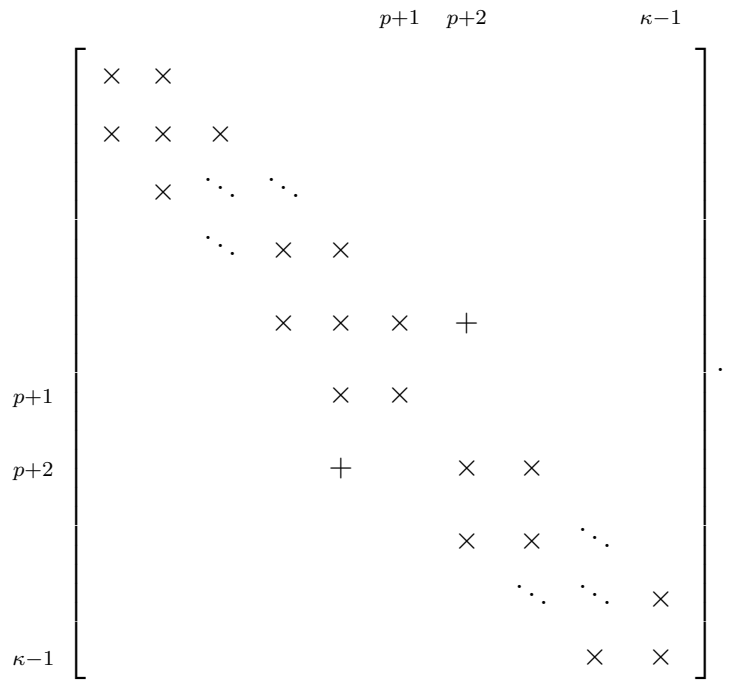
This matrix is used to form another similar matrix $\bar{C}^{(3)}$ through the multiplication

$$\bar{C}^{(3)} := P_\kappa \bar{C}^{(2)} P_\kappa. \tag{4.12}$$

The matrix $\bar{C}^{(3)}$ is identical to $\bar{C}^{(2)}$ except that rows and columns p and $p+1$ are exchanged. The first p rows and columns of $\bar{C}^{(3)}$ are tridiagonal. The bulge which was permuted is now chased toward the end of the tridiagonal tail.

4.3.4 Step 4: Bulge Chasing and Fill-In Row/Column

After the transformation using P_κ, the first $\kappa - 1$ rows and columns of $\bar{C}^{(3)}$ have the form:



Using CO rotations, the bulge, which begins at $\bar{C}^{(3)}_{p,p+2}$ and $\bar{C}^{(3)}_{p+2,p}$, is chased along the tridiagonal to the end of the tail. Just before the last bulge is annihilated, the matrix

has the form

$$
\begin{array}{c}
 \\
 \\
 \\
 \\
 \\
 \\
\kappa \\
 \\
 \\
 \\
N_{ev}
\end{array}
\begin{array}{c}
\begin{array}{ccccccccccc}
 & & & & & \kappa & & & & & N_{ev}
\end{array} \\
\left[\begin{array}{ccccccccccc}
\times & \times & & & & & & & & & \\
\times & \ddots & \ddots & & & & & & & & \\
 & \ddots & \times & \times & & & & & & & \\
 & & \times & \times & \times & + & & & & & \\
 & & & \times & \times & \times & & & & & \\
 & & & + & \times & \times & \times & \times & \cdots & \times & \\
 & & & & & \times & \times & \times & \cdots & \times & \\
 & & & & & \times & \times & \times & \cdots & \times & \\
 & & & & & \vdots & \vdots & \vdots & \ddots & \vdots & \\
 & & & & & \times & \times & \times & \cdots & \times &
\end{array}\right]
\end{array}.
$$

The final CO rotation replaces $\bar{C}_{\kappa-1,\kappa+1:N_{ev}}$ and $\bar{C}_{\kappa+1:N_{ev},\kappa-1}$ with nonzeroes. Let the product of the CO rotation matrices used to chase the bulge be Γ_κ and define the similar matrix $\bar{C}^{(4)}$ as

$$\bar{C}^{(4)} = \Gamma_\kappa^T \bar{C}^{(3)} \Gamma_\kappa. \tag{4.13}$$

4.3.5 Step 5: Annihilate the Fill-In Row/Column

Let the values of $\sin(\theta)$ and $\cos(\theta)$ in the CO rotation matrix which annihilates the final bulge in $\bar{C}^{(3)}$ be s_f and c_f. The form of $\bar{C}^{(4)}$ is

$$
\begin{array}{c}
\\
\\
\\
\\
\\
\kappa \\
\\
\\
\\
N_{ev}
\end{array}
\begin{array}{c}
\begin{array}{ccccccccccc}
 & & & & & & \kappa & & & & N_{ev}
\end{array} \\
\left[\begin{array}{ccccccccccc}
\times & \times & & & & & & & & & \\
\times & \ddots & \ddots & & & & & & & & \\
 & \ddots & \times & \times & & & & & & & \\
 & & \times & \times & \times & & & & & & \\
 & & & \times & \times & \times & + & + & \cdots & + & \\
 & & & & \times & \times & \times & \times & \cdots & \times & \\
 & & & & + & \times & \times & \times & \cdots & \times & \\
 & & & & + & \times & \times & \times & \cdots & \times & \\
 & & & & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \\
 & & & & + & \times & \times & \times & \cdots & \times &
\end{array}\right]
\end{array}.
$$

which resembles the form of $\bar{C}$, except that the tridiagonal reduction has retreated by one row/column. The troublesome vector at row/column κ in $\bar{C}^{(4)}$ is now $c_f x$, where

$$\mathbf{x} = \left\{ x_1 \quad x_2 \quad \cdots \quad x_{N_{ev}-\kappa} \right\}^T$$

corresponds to entries $\bar{C}_{\kappa+1:N_{ev},\kappa}$. Let the fill-in row/column be represented by $\mathbf{x}_f$ which contains the new nonzeroes:

$$\mathbf{x}_f = \left\{ \bar{C}^{(4)}_{\kappa,\kappa-1} \quad s_f x_1 \quad s_f x_2 \quad \cdots \quad s_f x_{N_{ev}-\kappa} \right\}^T.$$

The vector $\mathbf{x}_f$ must be annihilated in order for this CE removal attempt to continue. If a CO reflection is used to annihilate $\mathbf{x}_f$, the remaining subdiagonal entry

is $\mp\sqrt{\mathbf{x}_f^T\mathbf{x}_f}$, according to equation (4.5). The Implicit Q theorem states that the first column of Q determines the subdiagonal entries in the tridiagonal matrix, each within a sign. Therefore, since the first column of Q is set in Steps 1 and 2, then regardless of the annihilation method used to annihilate $\mathbf{x}_f$, the subdiagonal entry at position $(\kappa, \kappa-1)$ will become $\mp\sqrt{s_f^2\mathbf{x}^T\mathbf{x} + \left(\bar{C}^{(4)}_{\kappa,\kappa-1}\right)^2}$. The method used to annihilate entries in $\mathbf{x}_f$ also updates the troublesome vector at row/column κ and experience has shown that this updated vector at row/column κ rarely possesses a cancellation event. The values in the updated vector, and, therefore, the CE value of the updated vector, are dependent on the method used to annihilate $\mathbf{x}_f$.

Let the CO matrix used to annihilate the fill-in be represented by X_κ. The new $\bar{C}^{(m)}$ matrix is defined as

$$\bar{C}^{(m)} := X_\kappa^T \bar{C}^{(4)} X_\kappa. \tag{4.14}$$

The matrix is now returned to its original form in equation (4.7) having an unreduced submatrix and a tridiagonal tail. All of the values in $\bar{C}^{(m)}$ are different from those in $\bar{C}$. In the unlikely circumstance that a CE still exists in row/column κ, the changes from this attempt can be discarded, the size of p can be increased by one, and steps 1 through 5 can be repeated. Because the number of operations required to remove a cancellation event is $O(\kappa^2)$, different values of p can be tried inexpensively to find one that is satisfactory. The number of rows in the first column of Q which are modified by this CE removal method is equal to p, so different values of p will always produce different vectors for row/column κ.

4.4 The Complete CE Removal Method

Putting equations (4.10) through (4.14) together, the modified $\bar{C}$ matrix is

$$\bar{C}^{(m)} := X_\kappa^T \Gamma_\kappa^T P_\kappa \Omega_\kappa^T \Upsilon_\kappa^T \bar{C} \Upsilon_\kappa \Omega_\kappa P_\kappa \Gamma_\kappa X_\kappa. \tag{4.15}$$

This method removes CEs for any row/column which is longer than $p+2$. The matrix Q which tridiagonalizes C (as seen in equation (2.14)) while considering the possibility of CEs is

$$Q = \prod_{i=1}^{N_{ev}-2} K_i, \text{ where } K_i = \begin{cases} \Upsilon_i \Omega_i P_i \Gamma_i X_i \hat{H}_i, & \text{if a CE exists} \\ H_i, & \text{otherwise.} \end{cases} \tag{4.16}$$

The matrix H_i is defined in equation 2.11 and $\hat{H}_i$ is the CO reflector used to annihilate the new CE-free row/column. When this process is used to remove CEs encountered while tridiagonalizing C matrices from industrial models, one attempt with a value of $p = 5$ removes the CE nearly every time.

The sum of the FLOPs required to perform the first four steps of the CE removal process is small because the operations are performed on the tridiagonal tail and p is small. The only appreciable cost in removing a CE is from annihilating the fill-in row/column. This FLOP cost is $\delta(N_{ev} - \kappa)^2$, where κ is the number of the row/column with a CE and δ is set by the method of annihilation. The value of δ for two methods of annihilation is derived in the next chapter, in Section 5.2. Since the CE is a chance event, it is not possible to know the values of κ before the tridiagonal reduction process begins. Therefore, the average FLOP cost is computed which is the sum of the FLOP costs of removing CEs at every row/column in the matrix, divided

by the total number of rows/columns which are annihilated:

$$\frac{1}{N_{ev}-2}\sum_{\kappa=1}^{N_{ev}-2}\delta(N_{ev}-\kappa)^2 \approx \frac{\delta}{3}N_{ev}^2. \tag{4.17}$$

All of the steps taken to remove CEs use CO matrices, and the CEs are removed between applications of CO reflections. The process reduces the complex symmetric matrix C into a tridiagonal matrix, T, through $T = Q^T C Q$, where Q is defined in equation (4.16). Next, the eigenvalue decomposition of T is solved: $T = \Phi_T \Lambda_T \Phi_T^T$. Since T and C are similar matrices, their eigenvalues are the same.

After the eigenvector matrix, Φ_T, of the tridiagonal matrix is determined, Φ_T must be backtransformed into the eigenvector matrix of C, Φ_C. During the backtransformation, the matrices, Υ_κ, Ω_κ, P_κ, Γ_κ, and X_κ, must be applied in the reverse order for every row/column which has a CE removed to provide the correct update to Q. Since the eigenvector matrix Φ_T is full and general, these CE updates are more costly to perform than in the tridiagonal reduction. However, the process can be easily parallelized by dividing Φ_T into column panels to speed up computation. Table 4.1 provides a list of the steps required to apply the CE removal matrices during the backtransformation as well as the corresponding FLOP cost to perform each step. The value of δ is set by the method of annihilation, which is derived in Section 5.2.

Taking only the highest order cubic terms, the average cost per CE removed in the backtransformation is

$$\frac{1}{N_{ev}-2}\sum_{\kappa=1}^{N_{ev}-2}\delta N_{ev}(N_{ev}-\kappa) + 28N_{ev}(\kappa-p-1) \approx \left(14+\frac{\delta}{2}\right)N_{ev}^2. \tag{4.18}$$

Step	Operation	FLOP Cost
1	Apply X_κ	$\delta N_{ev}(N_{ev}-\kappa)$
2	Apply Γ_κ	$28N_{ev}(\kappa-p-1)$
3	Apply P_κ	neglible
4	Apply Ω_κ	$14p^2N_{ev}$
5	Apply Υ_κ	$8p^2N_{ev}$
Total	1 CE Update	$\delta N_{ev}(N_{ev}-\kappa) + 28N_{ev}(\kappa-p-1) + 22p^2N_{ev}$

Table 4.1: CE-related FLOPs during $\Phi_C = Q\Phi_T$

4.5 The CE Tolerance

Rounding error has an effect on real arithmetic in all matrix computations due to the limitation of representing analytical numbers as floating point numbers on computers. For example, for a large, real vector, $\mathbf{x}$, it is not unusual for the result of $\mathbf{x}^T\mathbf{x}$ to differ in the last one or two digits when entries in $\mathbf{x}$ have been reordered. Therefore, since the CE value is a measure of the number of digits lost in the computation of the complex $\mathbf{x}^T\mathbf{x}$, a tolerance of less than 2.0 is unrealistic.

Let N_{CE} represent the number of CEs encountered and removed. The ratio

$$\alpha_{\text{CE}} = \frac{N_{\text{CE}}}{N_{ev}} \tag{4.19}$$

is a simple measure which can be used to compare the frequency at which CEs are fixed across industrial models. Figure 4.3 plots α_{CE} versus the CE tolerance, τ_{CE}, for the Industrial Test Suite models. Even though the cancellation event is a chance event, the figure shows that there is some consistency in the frequency of CEs based on tolerance. As τ_{CE} is lowered, more CEs are removed, and the resulting matrices

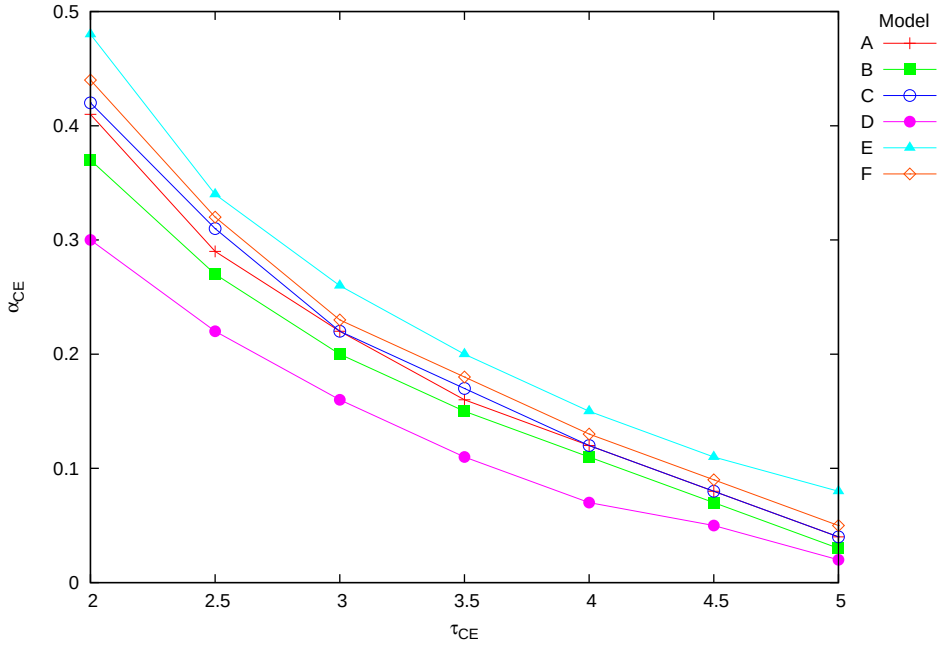


Figure 4.3: Frequency of CEs Removed for Industrial Test Suite

Q and T become more valid in equations (2.14) and (2.15). The error measurements e_o and e_t defined in equations (2.17) and (2.18) are plotted for each model in the Industrial Test Suite against the CE tolerances, in Figures 4.4 and 4.5.

4.6 The Complex Symmetric Matrix Eigensolver

The complex symmetric matrix eigensolver (CSMES) finds the eigenvalues and eigenvectors of a complex symmetric matrix C through the following steps.

1. The rows and columns of C are annihilated, one at a time, using CO reflections with a total FLOP cost of $\frac{16}{3}N_{ev}^3$. (The tridiagonal reduction requires

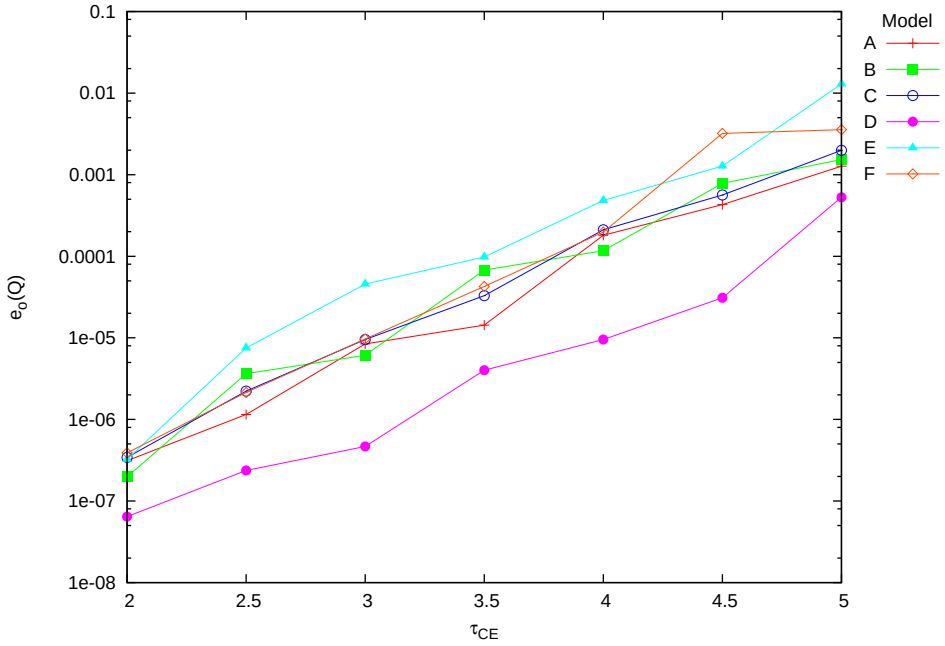


Figure 4.4: Effect of Removing CEs on $e_o(Q)$

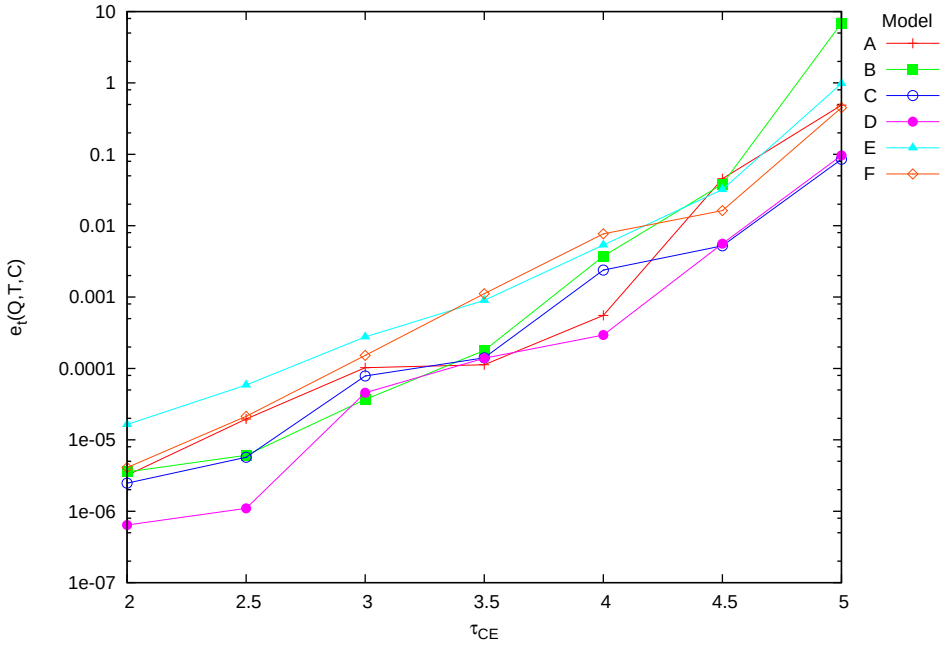


Figure 4.5: Effect of Removing CEs on $e_t(Q,T,C)$

$\frac{2}{3}N_{ev}^3$ complex multiplications and $\frac{2}{3}N_{ev}^3$ complex additions.) Just before each row/column is annihilated, the CE value of the row/column is compared with a given CE tolerance. If the CE value is greater than the tolerance, the CE removal method is used before the tridiagonal reduction continues.

2. The eigenvalues of the complex symmetric tridiagonal matrix, T, are determined using Cullum and Willoughby's CMTQL1 algorithm, which is a complex orthogonal extension of the EISPACK routine, IMTQL1 [15]. The eigenvectors of T are determined using inverse iteration. The total cost of performing these operations is $O(N_{ev}^2)$.

3. The eigenvalues of C are the same as the eigenvalues of T. The eigenvectors of C are recovered by implicitly applying each CO reflection from the right in reverse order for a total cost of $8N_{ev}^3 + 4N_{ev}^2$ FLOPs. The matrices used to remove CEs in the tridiagonal reduction are applied between the appropriate CO reflections, in reverse order. When Q is defined according to equation (4.16), the eigenvector matrices are related to one another through $\Phi_C = Q\Phi_T$. This process is known as the complex symmetric backtransformation.

The improvement in Q and T by fixing CEs directly impacts the quality of the EVD of the complex symmetric matrix. The change in e_o and e_t, plotted against CE tolerances for the Industrial Test Suite are plotted in Figures 4.6 and 4.7.

The total FLOP cost of CSMES, considering only the cubic terms, is displayed in Table 4.2. Using equation (4.19), the total number of FLOPs for CSMES is

$$\mathrm{C_{CSMES}} = \left[\frac{40}{3} + \left(\frac{5\delta}{6} + 14\right)\alpha_{CE}\right] N_{ev}^3. \tag{4.20}$$

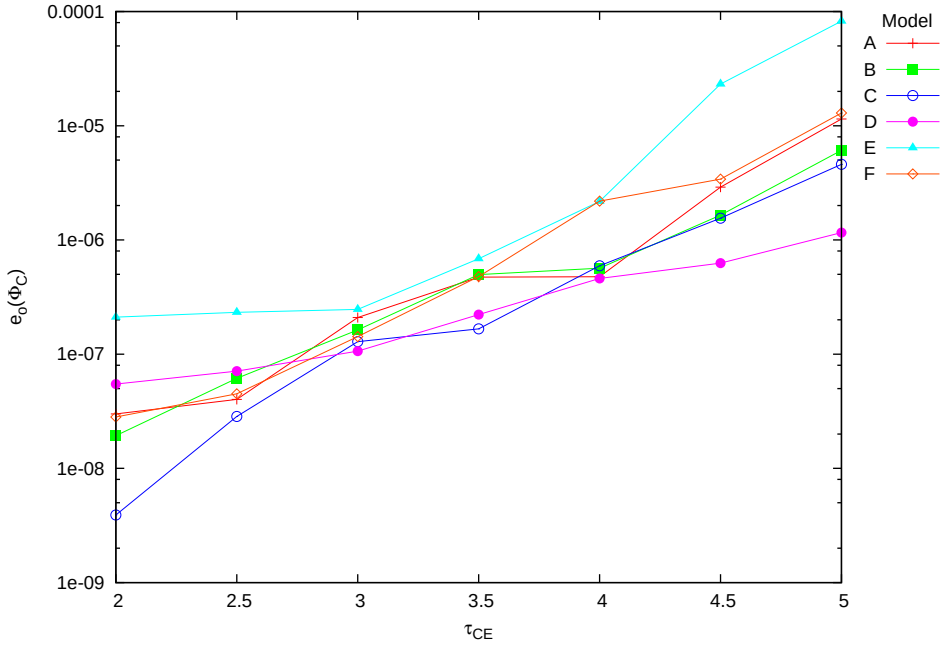


Figure 4.6: Effect of Removing CEs on $e_o(\Phi_C)$

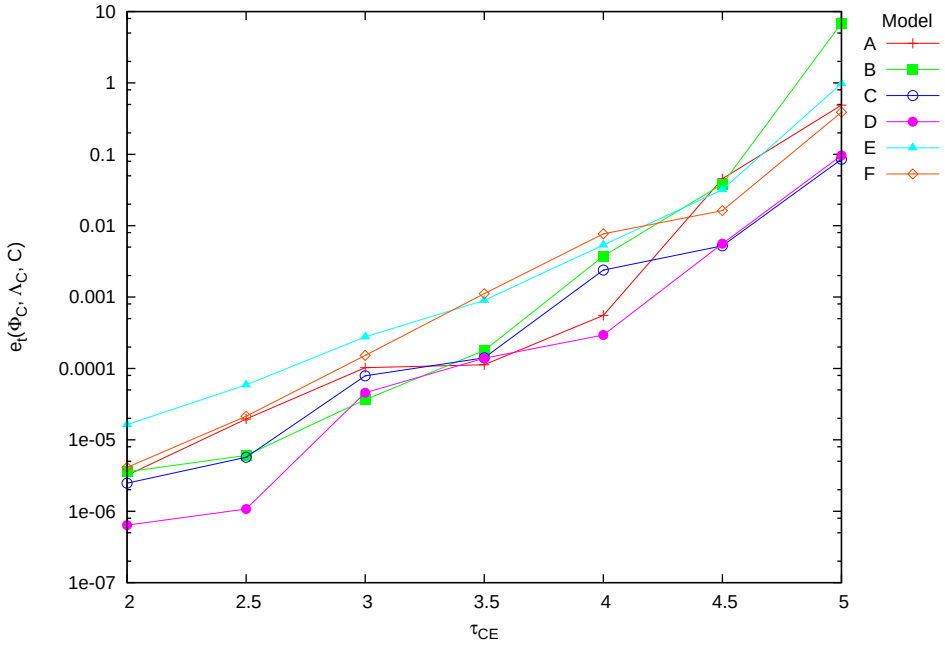


Figure 4.7: Effect of Removing CEs on $e_t(\Phi_C, \Lambda_C, C)$

Operation	FLOP Cost
Row/Column Annihilation	$\frac{16}{3}N_{ev}^3$
CE fixes	$\frac{\delta}{3}N_{ev}^2 N_{CE}$
Backtransformation	$8N_{ev}^3 + 4N_{ev}^2$
Applying CE fixes	$\left(14 + \frac{\delta}{2}\right) N_{ev}^2 N_{CE}$
Total	$\frac{40}{3}N_{ev}^3 + \left(\frac{5\delta}{6} + 14\right) N_{ev}^2 N_{CE}$

Table 4.2: CSMES FLOPs

Figure 4.3 can be used to give an estimate for α_{CE} before the tridiagonal reduction begins based on the CE tolerance.

4.7 Solving Modal FRPs using the Complex Symmetric Approach

The complex symmetric matrix C is defined in equation (2.7) and its eigenvalues and eigenvectors are computed using CSMES:

$$C = \Phi_C \Lambda_C \Phi_C^T, \text{ where } \Phi_C^T \Phi_C = I.$$

The complex symmetric approach uses the results from CSMES to transform the full coefficient matrix of the modal FRP into the sum of a diagonal and, in most cases, a low rank matrix, and the SMW formula is used (unless there is no low rank matrix) to find the response at every frequency.

4.7.1 Modal FRP with Only Structural Damping

When there is no viscous damping in the model, the use of CSMES completely diagonalizes the coefficient matrix of the modal FRP. The response is given in equation

(2.9) and is straightfoward to compute. The total FLOP cost is the cost of performing CSMES on the matrix C in equation (4.20) and the matrix products involving Φ_C, performed at every frequency:

$$\text{Cost} = \text{C}_{\text{CSMES}} + 16N_{ev}^2 N_c N_{freq}. \tag{4.21}$$

4.7.2 Modal FRP with Viscous and Structural Damping

When viscous and structural damping are represented in the model, the modal FRP in equation (1.13) becomes

$$[-\omega^2 I + i\omega B + C]X_s(\omega) = F_s(\omega). \tag{4.22}$$

Let $X_s(\omega) := \Phi_C Y(\omega)$ and premultiply equation (4.22) by Φ_C^T. The viscous damping matrix is cast in a compact form in equation (2.3). If the modal viscous damping matrix is nonsymmetric, its SVD is $B = U\Sigma_{\mathcal{B}}V^T$ and the modal FRP becomes

$$[-\omega^2 I + i\omega\Phi_C^T U\Sigma_{\mathcal{B}}V^T\Phi_C + \Lambda_C]Y(\omega) = \Phi_C^T F_s. \tag{4.23}$$

The above equation is solved efficiently using the SMW formula, with the substitutions

$$\begin{aligned} D(\omega) &:= -\omega^2 I + \Lambda_C \in \mathbb{C}^{N_{ev}\times N_{ev}}, \\ P &:= \Phi_C^T U \in \mathbb{C}^{N_{ev}\times N_k}, \\ R &:= \Phi_C^T V \in \mathbb{C}^{N_{ev}\times N_k}, \text{ and} \\ \Sigma(\omega) &:= i\omega\Sigma_{\mathcal{B}} \in \mathbb{C}^{N_k\times N_k}, \end{aligned} \tag{4.24}$$

where $N_k = \text{rank}(\mathcal{B}_{nz})$. Then, the response is

$$X_s(\omega) = \Phi_C \left[D + P\Sigma R^T\right]^{-1} \Phi_C^T F_s.$$

Step	Task	FLOP Cost
(1)	$F := \Phi_C^T F$	$8N_{ev}^2 N_c$
(2)	$D := D^{-1/2}$, $\Sigma := \Sigma^{1/2}$	$N_{ev} + N_k$
(3)	$P := DP\Sigma$, $R := DR\Sigma$	$24N_{ev}N_k$
(4)	$F := DF$	$6N_{ev}N_c$
(5)	$G_1 := R^T F$	$8N_{ev}N_k N_c$
(6)	$G_2 := R^T P$	$8N_k^2 N_{ev}$
(7)	$G_2 := I + G_2$	N_k
(8)	$G_1 := G_2^{-1} G_1$	$\frac{8}{3}N_k^3 + 8N_k^2 N_c$
(9)	$F := PG_1 - F$	$8N_{ev}N_k N_c$
(10)	$Y := -DF$	$6N_{ev}N_c$
(11)	$X := \Phi_C Y$	$8N_{ev}^2 N_c$

Table 4.3: Nonsymmetric SMW Operations per Frequency for Complex Symmetric Approach

Since the rank of $\mathcal{B}_{nz}$ is typically very small, the time required to compute P and R is small when compared to the CSMES execution time. The steps needed to compute the above response at a frequency and the FLOP costs which correspond to each step are listed in Table 4.3. The steps in the table are selected to maximize matrix reuse since neither P nor R is purely real. The rank of the condensed FE viscous damping matrix is typically very small (less than 20), so steps (1) and (11) are the most costly steps to compute the response at a frequency. All steps involving the viscous damping matrices have very small dimensions, so their FLOP costs are negligible. Therefore, the total FLOP cost is given in equation (4.21).

If the modal viscous damping matrix is symmetric, then this symmetry can be exploited in the SMW formula with D, P, and Σ set to the values in equation (4.24),

Step	Task	FLOP Cost
(1)	$F := \Phi_C^T F$	$8N_{ev}^2 N_c$
(2)	$D := D^{-1/2}$, $\Sigma := \Sigma^{1/2}$	$N_{ev} + N_k$
(3)	$P := DP\Sigma$	$12N_{ev}N_k$
(4)	$F := DF$	$6N_{ev}N_c$
(5)	$G_1 := P^T F$	$8N_{ev}N_k N_c$
(6)	$G_2 := P^T P$	$4N_k^2 N_{ev}$
(7)	$G_2 := I + G_2$	N_k
(8)	$G_1 := G_2^{-1} G_1$	$\frac{4}{3}N_k^3 + 8N_k^2 N_c$
(9)	$F := PG_1 - F$	$8N_{ev}N_k N_c$
(10)	$X := -DF$	$6N_{ev}N_c$
(11)	$X := \Phi_C Y$	$8N_{ev}^2 N_c$

Table 4.4: Symmetric SMW Operations per Frequency for Complex Symmetric Approach

but $R = P$. The response is

$$X_s(\omega) = \Phi_C \left[D + P\Sigma P^T \right]^{-1} \Phi_C^T F_s.$$

The FLOP cost of computing a response at a frequency for this symmetric case is listed in Table 4.4. The most costly steps, again, are the first and last steps, which do not depend on P. Therefore, the FLOP cost of computing the modal FRP is the same as the nonsymmetric case, given in equation (4.21).

4.8 Using the Complex Symmetric Approach with Structure-Fluid Interaction

When the model has viscous and structural damping and acoustic fluid elements, the modal FRP can be solved using the complex symmetric approach. The frequency independent work is performed first: CSMES computes Φ_C and Λ_C and for symmetric viscous damping matrices, the EVD from equation (2.3) is used to compute U and $\Sigma_{\mathcal{B}}$. Let $X_s(\omega) = \Phi_C Y(\omega)$, let $A_c = \Phi_C^T A$, and premultiply equation (1.16) by

$$\begin{bmatrix} \Phi_C^T & \\ & I \end{bmatrix}. \tag{4.25}$$

This gives a new FRP:

$$\begin{bmatrix} \bar{Z}_s(\omega) & i\omega A_c \\ i\omega A_c^T & Z_f(\omega) \end{bmatrix} \begin{Bmatrix} Y(\omega) \\ X_f(\omega) \end{Bmatrix} = \begin{Bmatrix} \Phi_C^T F_s(\omega) \\ F_f(\omega) \end{Bmatrix} \tag{4.26}$$

where $\bar{Z}_s(\omega) = -\omega^2 I + \Lambda_C + i\omega \Phi_C^T U \Sigma_{\mathcal{B}} U^T \Phi_C$.

4.8.1 General Acoustic Fluid Formulation

When the modal fluid "damping" matrix is full, Z_f in equation (4.26) is also full. A partitioned approach is appropriate as it casts the modal FRP into one that is solved using the SMW formula. First, the FRP is written as a system of equations:

$$\begin{aligned} \bar{Z}_s Y + i\omega A_c X_f &= \Phi_C^T F_s \\ i\omega A_c^T Y + Z_f X_f &= F_f. \end{aligned} \tag{4.27}$$

Rearranging the first equation into

$$Y = \bar{Z}_s^{-1} \left[\Phi_C^T F_s - i\omega A_c X_f \right] \tag{4.28}$$

and plugging the above into the second equation gives the fluid response:

$$X_f = \left[Z_f + \omega^2 A_c^T \bar{Z}_s^{-1} A_c\right]^{-1} \left[F_f - i\omega A_c^T \bar{Z}_s^{-1} (\Phi_C^T F_s)\right]. \tag{4.29}$$

Then, the structural response is

$$X_s = \Phi_C \left[\bar{Z}_s^{-1} (\Phi_C^T F_s) - i\omega \bar{Z}_s^{-1} A_c X_f\right]. \tag{4.30}$$

The terms $\bar{Z}_s^{-1}(\Phi_C^T F_s)$ and $\bar{Z}_s^{-1} A_c$ are computed through the following

$$\left[D + P\Sigma P^T\right]^{-1} \begin{bmatrix} \Phi_C^T F_s & A_c \end{bmatrix} \tag{4.31}$$

using the SMW formula with the substitutions from equation (4.24). The steps to compute the fluid and structural responses in equations (4.29) and (4.30) which rely on the SMW formula are listed in Table 4.5.

Taking only the cubic terms, the FLOP cost per frequency is

$$\begin{aligned} C_{\text{CE1}} = {} & N_{ev}\left(16N_{ev}N_c + 16N_fN_k + 16N_cN_k + 12N_fN_c\right) \\ & + N_k^2\left(\frac{4}{3}N_k + 4N_{ev} + 8N_f + 8N_c\right) \\ & + N_f^2\left(4N_{ev} + \frac{4}{3}N_f + 8N_c\right). \end{aligned} \tag{4.32}$$

Then, the total FLOP cost is the sum of the CSMES cost, the cost of forming A_c, and the cost of all frequency dependent work for all frequencies. The number of structural modes is usually much larger than the number of frequencies and the number of fluid modes, causing the CSMES computation to become the primary FLOP cost.

$$\text{Cost} = \text{C}_{\text{CSMES}} + 4N_{ev}^2 N_f + N_{freq} C_{\text{CE1}}$$

Step	Task	FLOP Cost
(1)	$F_s := \Phi_C^T F_s$	$8N_{ev}^2 N_c$
(2)	$D := D^{-1/2}$, $\Sigma := \Sigma^{1/2}$	$N_{ev}+N_k$
(3)	$P := DP\Sigma$	$12N_{ev}N_k$
(4)	$G_3 := P^T D A_c$	$4N_{ev}N_f(2N_k+1)$
(5)	$G_4 := P^T D F_s$	$8N_{ev}N_cN_k+6N_{ev}N_c$
(6)	$G_2 := P^T P$	$4N_k^2 N_{ev}$
(7)	$G_2 := I + G_2$	N_k
(8)	$\begin{bmatrix} G_3 & G_4 \end{bmatrix} := G_2^{-1} \begin{bmatrix} G_3 & G_4 \end{bmatrix}$	$\frac{4}{3}N_k^3+8N_k^2(N_f+N_c)$
(9)	$\bar{A} := DA_c$, $F_s := DF_s$	$6N_{ev}(N_f+N_c)$
(10)	$\begin{bmatrix} \bar{A} & F_s \end{bmatrix} := P \begin{bmatrix} G_3 & G_4 \end{bmatrix} - \begin{bmatrix} \bar{A} & F_s \end{bmatrix}$	$8N_{ev}N_k(N_f+N_c)$
(11)	$\bar{A} := -D\bar{A}, F_s := -DF_s$	$6N_{ev}(N_f+N_c)$
(12)	$Z_f := \omega^2 I_f - i\omega C_f - \Lambda_f + \omega^2 A_c^T \bar{A}$	$4N_f(N_f N_{ev}+1)$
(13)	$F_f := F_f - i\omega A_c^T F_s$	$4N_f N_{ev} N_c$
(14)	$X_f := Z_f^{-1} F_f$	$\frac{4}{3}N_f^3+8N_f^2 N_c$
(15)	$F_s := F_s - i\omega \bar{A} X_f$	$8N_{ev}N_f N_c$
(16)	$X_s := \Phi_C F_s$	$8N_{ev}^2 N_c$

Table 4.5: Complex Symmetric Approach Operations per Frequency for General Fluid Matrices

4.8.2 Diagonal Acoustic Fluid Formulation

If the FE fluid "damping" is proportional damping, or not present at all, Z_f (represented in equation (3.26)) is diagonal and the SMW formula is used to solve the modal FRP efficiently. Using the system of equations in equation (4.27), the second equation is rearranged into

$$X_f = Z_f^{-1}\left(F_f - i\omega A_c^T Y\right) \tag{4.33}$$

and plugged into the first equation which gives the following:

$$\left[\bar{Z}_s + \omega^2 A_c Z_f^{-1} A_c^T\right] Y = \Phi_C^T F_s - i\omega A_c Z_f^{-1} F_f. \tag{4.34}$$

The above equation is solved efficiently for Y using the SMW formula with the substitutions:

$$\begin{aligned} D(\omega) &:= -\omega^2 I + \Lambda_C \in \mathbb{C}^{N_{ev}\times N_{ev}}, \\ P &:= \begin{bmatrix} \Phi_C^T U & A_c \end{bmatrix} \in \mathbb{C}^{N_{ev}\times N_k}, \\ \Sigma(\omega) &:= \begin{bmatrix} i\omega\Sigma_{\mathcal{B}} & \\ & \omega^2 Z_f^{-1} \end{bmatrix} \in \mathbb{C}^{N_k\times N_k}, \text{ and} \\ F(\omega) &:= \Phi_C^T F_s - i\omega A_c Z_f^{-1} F_f \end{aligned} \tag{4.35}$$

where $N_k = \text{rank}(\mathcal{B}_{nz}) + N_f$ and the viscous damping matrix is symmetric. Then, Y is

$$Y = \left[\bar{Z}_s + \omega^2 A_c Z_f^{-1} A_c^T\right]^{-1}\left[\Phi_C^T F_s - i\omega A_c Z_f^{-1} F_f\right]. \tag{4.36}$$

After Y is computed using the SMW formula, the fluid response is determined from equation (4.33) and the structural response is $X_s = \Phi_C Y$. The steps to compute the

responses are listed in Table 4.6. The total number of FLOPs per frequency is

$$\begin{aligned} C_{\text{CE2}} = N_{ev}\left(16N_{ev}N_c + 16N_fN_c + 16N_kN_c\right) \\ + N_k^2\left(\frac{4}{3}N_k + 8N_c + 4N_{ev}\right). \end{aligned} \tag{4.37}$$

The total FLOP cost is

$$\text{Cost} = \text{C}_{\text{CSMES}} + 4N_{ev}^2N_f + N_{freq}C_{\text{CE2}}.$$

4.9 Solving Modal FRPs having Low Frequency Modes

With enforced motion techniques, base motion is used to specify the displacement, velocity, or acceleration at particular grid points. This is typically accomplished by setting the masses at the grid points to be very large, which produces models with low frequency modes. Also, if the structure is unconstrained, or if mechanism modes are present, then some of the modes will be rigid body modes. All of these situations produce a Λ matrix containing diagonal values which are low or nearly zero [11]. The complex symmetric matrix, C, is formed from Λ in equation (2.7). Consequently, when low frequency modes are present in the model, C becomes ill-conditioned and the eigenvalue decomposition from CSMES is imprecise [27]. To avoid the ill-conditioning, the modal FRP is partitioned into a low frequency part and a high frequency part. The number of low frequency modes is very small, typically less than ten. For a model with symmetric viscous damping (with $B = U\Sigma_{\mathcal{B}}U^T$) and structure-fluid interaction, let the number of low frequency modes be represented by N_l and the number of high frequency modes be N_h and $N_{ev} = N_l + N_h$. Partition C,

Step	Task	FLOP Cost
(1)	$Z_f := \omega^2 I_f - i\omega C_f - \Lambda_f$	$6N_f$
(2)	$F_s := \Phi_C^T F_s, F_f := Z_f^{-1} F_f$	$8N_{ev}^2 N_c + 6N_f N_c$
(3)	$D := D^{-1/2}, \Sigma := \Sigma^{1/2}$	$N_{ev} + N_k$
(4)	$P := DP\Sigma$	$12N_{ev}N_k$
(5)	$G_2 := P^T P$	$4N_k^2 N_{ev}$
(6)	$G_2 := G_2 + I$	N_k
(7)	$F_s := F_s - i\omega A_c F_f$	$8N_{ev}N_f N_c$
(8)	$F_s := DF_s$	$6N_{ev}N_c$
(9)	$G_1 := P^T F_s$	$8N_{ev}N_k N_c$
(10)	$G_1 := G_2^{-1} G_1$	$\frac{4}{3}N_k^3 + 8N_k^2 N_c$
(11)	$F_s := PG_1 - F_s$	$8N_{ev}N_k N_c$
(12)	$Y := -DF_s$	$6N_{ev}N_c$
(13)	$X_s := \Phi_C Y$	$8N_{ev}^2 N_c$
(14)	$X_f := A_c^T Y$	$8N_f N_{ev} N_c$
(15)	$X_f := Z_f^{-1} X_f$	$6N_f N_c$
(16)	$X_f := F_f - i\omega X_f$	$4N_{ev}N_c$

Table 4.6: Complex Symmetric Approach Operations per Frequency for Diagonal Fluid Matrices

U, X_s, A, and F_s into:

$$C = \begin{matrix} \\ N_l \\ N_h \end{matrix} \overset{\begin{matrix} N_l & N_h \end{matrix}}{\left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{12}^T & C_{22} \end{array}\right]}, \; U = \begin{matrix} \\ N_l \\ N_h \end{matrix} \overset{\begin{matrix} \text{rank}(\mathcal{B}) \end{matrix}}{\left[\begin{array}{c} U_1 \\ \hline U_2 \end{array}\right]},$$

$$X_s = \begin{matrix} \\ N_l \\ N_h \end{matrix} \overset{\begin{matrix} N_c \end{matrix}}{\left[\begin{array}{c} X_1 \\ \hline X_2 \end{array}\right]}, \; A = \begin{matrix} \\ N_l \\ N_h \end{matrix} \overset{\begin{matrix} N_f \end{matrix}}{\left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right]}, \text{ and } F_s = \begin{matrix} \\ N_l \\ N_h \end{matrix} \overset{\begin{matrix} N_c \end{matrix}}{\left[\begin{array}{c} F_1 \\ \hline F_2 \end{array}\right]}.$$

The modal viscous damping matrix is

$$B = \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{12}^T & B_{22} \end{array}\right] = \left[\begin{array}{c|c} U_1 \Sigma_{\mathcal{B}} U_1^T & U_1 \Sigma_{\mathcal{B}} U_2^T \\ \hline U_2 \Sigma_{\mathcal{B}} U_1^T & U_2 \Sigma_{\mathcal{B}} U_2^T \end{array}\right]. \tag{4.38}$$

Then, the modal FRP becomes:

$$\begin{bmatrix} -\omega^2 I + C_{11} + i\omega B_{11} & C_{12} + i\omega B_{12} & i\omega A_1 \\ C_{12}^T + i\omega B_{12}^T & -\omega^2 I + C_{22} + i\omega U_2 \Sigma_{\mathcal{B}} U_2^T & i\omega A_2 \\ i\omega A_1^T & i\omega A_2^T & Z_f \end{bmatrix} \begin{Bmatrix} X_1 \\ X_2 \\ X_f \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_f \end{Bmatrix}.$$

CSMES is computed on the high frequency part of C, which has good conditioning: $C_{22} = \Phi_C \Lambda_C \Phi_C^T$. Let $X_2(\omega) = \Phi_C Y(\omega)$ and premultiply the above equation by

$$\begin{bmatrix} I & & \\ & \Phi_C^T & \\ & & I \end{bmatrix}$$

which becomes

$$\begin{bmatrix} Z_{11}(\omega) & Z_{12}(\omega) & i\omega A_1 \\ Z_{12}^T(\omega) & \bar{Z}_{22}(\omega) & i\omega A_c \\ i\omega A_1^T & i\omega A_c^T & Z_f(\omega) \end{bmatrix} \begin{Bmatrix} X_1 \\ Y \\ X_f \end{Bmatrix} = \begin{Bmatrix} F_1 \\ \Phi_C^T F_2 \\ F_f \end{Bmatrix}, \tag{4.39}$$

where

$$\begin{aligned} A_c &:= \Phi_C^T A_2, \\ Z_{11}(\omega) &:= -\omega^2 I + C_{11} + i\omega B_{11}, \\ \bar{Z}_{22}(\omega) &:= -\omega^2 I + \Lambda_C + i\omega \Phi_C^T U_2 \Sigma_{\mathcal{B}} U_2^T \Phi_C, \text{ and} \\ Z_{12}(\omega) &:= \left(C_{12} + i\omega B_{12}\right) \Phi_C. \end{aligned} \tag{4.40}$$

If $\bar{C}_{12} := C_{12}\Phi_C$ and $\bar{B}_{12} := B_{12}\Phi_C$, then Z_{12} is formed at each frequency as the linear combination of constant matrices: $Z_{12}(\omega) = \bar{C}_{12} + i\omega\bar{B}_{12}$. When the acoustic fluid matrices are diagonal, the fluid part can be included as a rank-N_f update to the structure. Let

$$\begin{aligned} D(\omega) &:= -\omega^2 I + \Lambda_C \in \mathbb{C}^{N_h \times N_h}, \\ P &:= \begin{bmatrix} \Phi_C^T U & A_c \end{bmatrix} \in \mathbb{C}^{N_h \times N_k}, \text{ and} \\ \Sigma(\omega) &:= \begin{bmatrix} i\omega\Sigma_{\mathcal{B}} & \\ & \omega^2 Z_f^{-1} \end{bmatrix} \in \mathbb{C}^{N_k \times N_k}, \end{aligned}$$

where $N_k = \text{rank}(\mathcal{B}) + N_f$.

Using the techniques presented in previous sections, it can be shown that equation (4.39), for the situation in which Z_f is diagonal, is solved using the steps listed in Table 4.7. Because $N_h \approx N_{ev}$ and $N_h \gg N_l$, the total FLOP cost is the same as equation (4.8.2). For determining FLOP costs in the complex symmetric approach, the possibility of low frequency modes can be ignored.

Step	Task	FLOP Cost
(1)	$F_2 := \Phi_C^T F_2(\omega), F_f := Z_f^{-1} F_f(\omega)$	$8N_h^2N_c+6N_fN_c$
(2)	$F_2 := F_2 - i\omega A_c F_f$	$8N_hN_fN_c$
(3)	$Z_{12} := \bar{C}_{12} + i\omega\bar{B}_{12}$	$4N_lN_h$
(4)	$\bar{Z}_{12} := Z_{12} + \omega^2 A_1 Z_f^{-1} A_c^T$	$4N_fN_h(N_l+1)$
(5)	$D := D^{-1/2}(\omega), \Sigma := \Sigma^{1/2}(\omega)$	N_h+N_k
(6)	$P := DP\Sigma$	$12N_hN_k$
(7)	$\bar{Z}_{12} := \bar{Z}_{12}D, F_2 := DF_2$	$6N_hN_c+6N_lN_h$
(8)	$G_2 := I+P^TP$	$4N_k^2N_h+N_k$
(9)	$\begin{bmatrix} G_3 & G_4 \end{bmatrix} := P^T \begin{bmatrix} F_2 & \bar{Z}_{12}^T \end{bmatrix}$	$8N_kN_h(N_c+N_l)$
(10)	$\begin{bmatrix} G_3 & G_4 \end{bmatrix} := G_2^{-1} \begin{bmatrix} G_3 & G_4 \end{bmatrix}$	$\frac{4}{3}N_k^3+8N_k^2(N_c+N_l)$
(11)	$\begin{bmatrix} F_2 & \bar{Z}_{12}^T \end{bmatrix} := P \begin{bmatrix} G_3 & G_4 \end{bmatrix} - \begin{bmatrix} F_2 & \bar{Z}_{12}^T \end{bmatrix}$	$8N_kN_h(N_c+N_l)$
(12)	$\bar{Z}_{12} := -\bar{Z}_{12}D, F_2 := -DF_2$	$6N_cN_h+6N_lN_h$
(13)	$A_{2c} := A_c Z_f^{-1}, A_{1c} := A_1 Z_f^{-1}$	$6N_hN_f+2N_lN_f$
(14)	$F_f := F_f - i\omega A_{2c}^T F_2$	$8N_fN_hN_c$
(15)	$A_{1c} := \bar{Z}_{12}A_{2c} - A_{1c}$	$8N_lN_hN_f$
(16)	$Z_{11} := Z_{11} - Z_{12}\bar{Z}_{12}^T - \omega^2 A_1 A_{1c}^T$	$8N_l^2N_h+4N_l^2N_f$
(17)	$F_1 := F_1 - i\omega A_1 F_f - Z_{12}F_2$	$8N_lN_hN_c+4N_lN_fN_c$
(18)	$X_1 := Z_{11}^{-1}F_1$	$\frac{4}{3}N_l^3+8N_l^2N_c$
(19)	$X_f := F_f + i\omega A_{1c}^T X_1$	$8N_fN_lN_c$
(20)	$Y := F_2 - \bar{Z}_{12}^T X_1$	$8N_hN_lN_c$
(21)	$X_2 := \Phi_C Y$	$8N_h^2N_c$

Table 4.7: Complex Symmetric Approach Operations per Frequency for Diagonal Fluid Matrices, where Low Frequency Modes are Present

Chapter 5

Parallel Implementation of the New Approaches

The low rank and complex symmetric approaches efficiently provide the solutions to modal FRPs having structural and localized viscous damping. This chapter explores the ways that the approaches can be implemented to take advantage of modern computing technology. During the last several decades of the 20^{th} century, software performance improvements were automatically gained by the faster clock speeds of successive processors, without requiring software developers to modify code substantially to accommodate new types of hardware. This "free lunch", as described by Sutter [40], ended near the beginning of this current century due to processor hardware limitations such as an inability to dissipate the ever-increasing amount of heat, power demands, and problems with leaking voltage. Hardware manufacturers are addressing this processor speed plateau problem by creating CPU processors with multiple cores and by creating many-core coprocessors. In order to take advantage of computers with these hardware configurations, programmers must now implement algorithms with concurrency in mind. Concurrency is the programming paradigm in which expensive computations are distributed among computing resources and executed simultaneously. This may require coordination between the resources.

One way of increasing CPU throughput is to increase the number of computing

cores. It is common now to install two or more multi-core CPUs on one motherboard so that they share access to system memory, disk drives, and hardware linked through the peripheral component interconnect (PCI) bus. CPU processors carry out the instructions of a computer program through a CPU process, which is subdivided into sequences of instructions called CPU threads. A single CPU process is shared among all of the multi-core CPUs and tends to have peak performance when the number of CPU threads is equal to the total number of CPU cores. A programming interface which takes advantage of this type of configuration, supporting shared memory multiprocessing, is called OpenMP. The parallel processing in OpenMP is a "fork-join" type in which execution at a designated point is subdivided into parallel tasks performed by the CPU threads. After the parallel tasks are completed at another designated point, the threads merge and a single thread continues working on the serial tasks. Intel's Math Kernel Library (MKL) provides implementations of BLAS and LAPACK which have been optimized for Intel multi-core processors. MKL's BLAS level-3 subprograms and those LAPACK routines which rely on BLAS level-3 subprograms are capable of multi-threaded parallelism. Unfortunately, the additional cores in CPU processors do not improve the memory bandwidth of the computer. In some situations, additional cores in a CPU processor decrease the memory bandwidth per core. The result is that it is memory bandwidth, and not processor throughput, which is the performance bottleneck for many operations.

Computing speeds are also increased through the use of many-core coprocessors which offer a dramatic improvement in computational performance over multi-core CPUs. Modern coprocessors used for floating point arithmetic contain hundreds

or thousands of computing cores with very fast memory bandwidth and are connected to the processors through, for example, the PCI bus. Though there are examples of modern coprocessors including Intel's Xeon Phi and AMD's Firestream, Nvidia's Tesla graphics processing units (GPUs) are primarily chosen for NVH analyses because their hardware and accompanying software are much more mature than the other options. Nvidia provides a programming interface called CUDA which gives programmers the ability to implement algorithms which can take advantage of the thousands of computing cores on GPUs [4]. Compared to CPU cores, GPU cores are "lightweight", performing only simple tasks. However, the thousands of GPU cores working together can solve dense linear algebra operations at a much higher rate than the multi-core CPUs.

Nvidia also provides a thorough implementation of BLAS called CUBLAS [3]. The BLAS level-3 subprograms in CUBLAS perform many times faster than their MKL counterparts. The memory bandwidth is higher for GPUs than CPU cores; however, the BLAS level-2 subprograms on GPUs are also limited by memory bandwidth. There is no implementation of LAPACK for GPUs provided by Nvidia. But, there are open source and commercial software packages, such as libFLAME [44], CULA Dense [1], and MAGMA [6], which provide some routines found in LAPACK. Unfortunately, these packages do not offer the specific type of functionality needed to implement the two approaches presented in this dissertation, so specialized CUDA "kernel" routines were created to provide the missing functionality.

Using OpenMP, GPU routines can be executed in parallel, with each CPU thread launching CUBLAS and CUDA kernel routines on separate GPUs simultane-

ously. All kernels, whether in CUBLAS or written by a developer of more specialized software can only operate on matrix data that have been transferred to the GPU. Currently, the maximum on-board memory of any GPU is 24 GB, which means that all matrix data for solving modal FRPs cannot completely reside on the GPUs. Matrix data is transmitted to a GPU from the CPU cores through the PCI bus. So, GPUs should only be used when the cost of transferring the matrix data between the GPUs and CPU cores is much less than the cost of performing the operation with the CPU cores.

Automobile companies typically perform HPC NVH analyses on single node machines with one or two multi-core CPU processors; however, it is becoming common for the companies to incorporate one to four GPUs in their machines as well. Therefore, a machine ("Machine A") with these attributes was built in order to simulate an engineer's computing experience. Machine A has two 8-core 3.1 GHz Sandy Bridge Intel E5-2687W processors and four K20c "Kepler" GPUs. The maximum theoretical FLOP rate and memory bandwidths of the computational hardware of Machine A are given in Table 5.1, taken from data found in [2] and [5]. All of the performance results reported in this and the next chapter reflect those of implementations executed on Machine A.

5.1 The WY Representation

The low rank and complex symmetric approaches require a tridiagonal reduction of a symmetric $N_{ev} \times N_{ev}$ matrix. One way the reduction performances are improved is by taking advantage of the "WY representation", introduced by Bischof

	CPU Processors	1 K20 GPU
Number of Computing Cores	16	2496
Memory Bandwidth (GB/sec)	51.2	208
Compute Performance (GFLOP/sec)	396.8	1170
Memory (GB)	256	5

Table 5.1: Theoretical Hardware Maximums for Machine A

and Van Loan [10]. This representation works by aggregating many operations, reducing the number of memory accesses which improves the overall performance. During the tridiagonal reduction process, successive row/column pairs are annihilated from one corner of the matrix to the other using reflections, leaving a tridiagonal matrix. A reflection is defined as $H = I - \frac{2}{\beta}\mathbf{v}\mathbf{v}^T$, where $\beta = \mathbf{v}^T\mathbf{v}$. The vector $\mathbf{v}$ is determined by the row/column which is annihilated, $\mathbf{x}$, through equation (4.2). Let A represent the symmetric matrix which is tridiagonalized in either approach. If $\mathbf{x}$ is the first row/column of A, then multiplying HA annihilates all of $\mathbf{x}$ below the first subdiagonal of A, while AH annihilates all of $\mathbf{x}$ to the right of the first super-diagonal of A. Multiplying HAH annihilates the row and column simultaneously, leaving entries in the diagonal and subdiagonal that belong to the tridiagonal matrix. The transformed matrix, A', is

$$A' := HAH = A - \frac{2}{\beta}A\mathbf{v}\mathbf{v}^T - \frac{2}{\beta}\mathbf{v}\mathbf{v}^T A + \frac{4}{\beta^2}\left(\mathbf{v}^T A\mathbf{v}\right)\mathbf{v}\mathbf{v}^T.$$

The above equation can be rearranged into a more compact form by first setting

$$\mathbf{w} := A\mathbf{v} \tag{5.1}$$

which is a symmetric matrix-vector multiplication. Next, inexpensive operations update $\mathbf{w}$ into

$$\mathbf{w} := \frac{2}{\beta}\left(\mathbf{w} - \frac{\mathbf{w}^T\mathbf{v}}{\beta}\mathbf{v}\right). \tag{5.2}$$

Then, the product HAH can be written as

$$A' := A - \mathbf{w}\mathbf{v}^T - \mathbf{v}\mathbf{w}^T. \tag{5.3}$$

A reflection is a rank-1 modification of the identity matrix, and its two-sided application to A is a rank-2 update on A, which is a BLAS level-2 operation. Then, A becomes completely tridiagonal using $(N_{ev}-2)$ rank-2 updates.

The WY representation allows the product of r reflections to be combined into a single rank-r modification of the identity matrix. This permits the tridiagonalization process to be constructed into a blocked algorithm. Many $\mathbf{w}$ and $\mathbf{v}$ vectors are grouped together into matrices and these matrices are used to apply updates simultaneously in efficient BLAS level-3 subprograms. The backtransformation performance is also improved using the WY representation. In this process, matrices containing $\mathbf{v}$ and $\mathbf{w}$ vectors are used to backtransform the tridiagonal eigenvector matrix into the matrix of eigenvectors of A using BLAS level-3 subprograms.

There are two alternate methods for accumulating reflections, which, like the WY representation, are rank-r modifications of the identity matrix. The compact WY representation exploits the connection between the $\mathbf{v}$ and $\mathbf{w}$ vectors in order to save computer memory and more efficiently create the matrices needed in the blocked algorithm [39]. The UT representation is a modification of the compact WY representation which can offer a further improvement in performance since its

matrix operations are computed in a different order [38]. The relative performance of these two methods depends on the performance of particular BLAS subprograms, which is a topic outside of the scope of this dissertation [45]. For simplicity, the WY representation is used in this dissertation to accumulate reflections during the symmetric matrix tridiagonal reduction process.

5.2 Annihilating the Fill-In Row/Column in the CE Removal Method

In the complex symmetric approach, the final step in the CE removal method is the annihilation of the fill-in row/column, which is described in Section 4.3.5. The fill-in row/column corresponds to entries below the first subdiagonal and to the right of the first superdiagonal of the unreduced part of the complex symmetric matrix. These entries can be annihilated, for example, using a CO reflection or a series of CO rotations. Let the size of the unreduced part of the matrix be n.

When a CO reflection is used, all of the entries in the row/column are annihilated simultaneously. The previous section shows that the main cost in computing and applying a single reflection is a matrix-vector multiplication in equation (5.1) and a rank-2 update in equation (5.3). The matrix-vector multiplication requires n^2 scalar multiplications and n^2 scalar additions. The rank-2 update, which is only applied on the diagonal and the lower triangle of the matrix because of symmetry, also requires n^2 scalar multiplications and n^2 scalar additions. Since a complex multiplication requires 6 FLOPs and a complex addition requires 2 FLOPs, the total number of FLOPs needed to annihilate the fill-in row/column is $16n^2$.

When a series of CO rotations is used, the entries in the fill-in row/column are annihilated sequentially, which is a process described in Section 2.4.1. When a rotation is applied from the right, two columns of the unreduced matrix are modified. Each column becomes a linear combination of itself and the other affected column. By symmetry, when the rotation is applied from the left, the two corresponding rows of the unreduced matrix are modified. Each row becomes a linear combination of itself and the other affected row.

One application of a CO rotation from either side requires $4n$ scalar multiplications and $2n$ scalar additions. The application of a CO rotation from both sides requires twice as many operations; however, since the matrix is symmetric, only the diagonal and the upper or lower triangle of the matrix is referenced, which reduces the number of operations by half. Since there are approximately n entries in the row/column that are annihilated with CO rotations, the number of required scalar multiplications is $4n^2$ and the number of required scalar additions is $2n^2$. The total number of FLOPs needed to annihilate the fill-in row/column using CO rotations is $28n^2$.

Table 5.2 provides a summary of the FLOP cost results from this section. The value of δ used in the total FLOP cost calculations in equations (4.17) and (4.18) is set by the method used to annihilate the fill-in row/column. If a CO reflection is used, $\delta = 16$, and if a series of CO rotations is used, $\delta = 28$.

	1 CO Reflection	$(n-2)$ CO Rotations
Addition	$12n^2$	$24n^2$
Multiplication	$4n^2$	$4n^2$
Total	$16n^2$	$28n^2$

Table 5.2: FLOP Cost for Annihilating the Fill-In Row/Column

5.3 Real Symmetric Tridiagonal Reduction

The weighted structural damping matrix, L, is defined in equation (3.4). During the determination of an LRA of L, the first dominating process is its reduction to tridiagonal form, as seen in equation (3.11). This section describes the various ways that this process is parallelized on machines with multiple CPU cores, multiple CPU cores with a single GPU, or multiple CPU cores with multiple GPUs. Algorithm 2 is a pseudocode which shows how real symmetric tridiagonal reduction, blocked to take advantage of the WY representation, is accomplished. It serves as a reference for the following subsections.

In Algorithm 2, after a default algorithmic block size b is set, the reduction process is performed over $N = \lceil (N_{ev} - 2)/b \rceil$ blocks. Here, the ceiling notation ($\lceil \cdot \rceil$) represents the ceiling function, or the least integer upper bound. Since L is symmetric, computations and updates in line 19 are only performed on the lower triangle of L for each block. The "for loop" at line 2 iterates over blocks of L. Within this loop is another "for loop" at line 6, which iterates over columns within a block of L. At an iteration of the inner "for loop", i is the index of a column $\mathbf{x}$. The part of $\mathbf{x}$ below the subdiagonal is annihilated with the reflection which is implicitly created and applied

Algorithm 2: Blocked Real Symmetric Tridiagonalization

Data: $L \in \mathbb{R}^{N_{ev} \times N_{ev}}$ is symmetric; b is block size; $V, W \in \mathbb{R}^{N_{ev} \times b}$

$N = \lceil (N_{ev} - 2)/b \rceil$
for $j = 1 : N$ **do** `// loop over blocks`
  $V = 0, W = 0$
  $s = (j-1)b + 1$
  $t = \min(s+b-1, N_{ev} - 2)$
  for $i = s : t$ **do** `// loop over columns`
    if $i > s$ **then** `// make x current`
      `updatex`(L, i, V, W)
    $\mathbf{x} = L(i+1 : N_{ev}, i)$
    $[\mathbf{v}, \sigma, \tau] =$ `buildv(x)`
    $\mathbf{w} = L(i+1 : N_{ev}, i+1 : N_{ev})\mathbf{v}$
    $L(i+1, i) = \sigma$
    if $i > s$ **then** `// make w current`
      `updatew`$(\mathbf{w}, i, \mathbf{v}, V, W)$
    $\mathbf{w} = \tau \left[\mathbf{w} - \left(\frac{1}{2} \tau \mathbf{w}^T \mathbf{v} \right) \mathbf{v} \right]$
    $V(i+1 : N_{ev}, i-s) = \mathbf{v};\ W(i+1 : N_{ev}, i-s) = \mathbf{w}$
    $L(i+2 : N_{ev}, i) = \mathbf{v}(2 : N_{ev} - i + 1),\ z(i) = \tau$
  $V_0 = V(t+1 : N_{ev}, :);\ W_0 = W(t+1 : N_{ev}, :)$
  $L(t+1 : N_{ev}, t+1 : N_{ev}) \mathrel{-}= \left(V_0 W_0^T + W_0 V_0^T \right)$ `// update outside of block`
  if `checkempty`(L) **then** `// Quit early if possible`
    $L(t+1 : N_{ev}, t+1 : N_{ev}) = 0$
    $\mathbf{z}(t+1 : N_{ev}) = 0$
    quit

Result: The tridiagonal matrix is stored in the diagonal and subdiagonal entries of L. The lower triangle of L stores the reflection vectors and $\mathbf{z}$ stores the reflection scalars τ.

Algorithm 3: `buildv`($\mathbf{x}$)

input : $\mathbf{x}$ vector of length n
output: Reflection vector $\mathbf{v}$ and scalars $\tau = 2/\mathbf{v}^T\mathbf{v}$, $\sigma = \sqrt{\mathbf{x}^T\mathbf{x}}$, such that $H\mathbf{x} = \sigma\hat{\mathbf{e}}_1$, where $H = I - \tau\mathbf{v}\mathbf{v}^T$.

$\sigma = \sqrt{\mathbf{x}^T\mathbf{x}}$
if $(\mathbb{R}(x(1) * \sigma) > 0)$ **then**
  $\sigma = -\sigma$
$\tau = 1 - x(1)/\sigma$
$\alpha = 1/(x(1) - \sigma)$
$\mathbf{v} = \alpha\mathbf{x}$
$v(1) = 1$

in lines 10 through 15. In line 10, `buildv` (displayed in Algorithm 3) takes $\mathbf{x}$ as input to create the reflection vector, $\mathbf{v}$, and reflection scalar, $\tau = 2/\mathbf{v}^T\mathbf{v}$, which are used to create the reflection. The i-th subdiagonal entry of the resulting tridiagonal matrix is also created in `buildv` and stored in the subdiagonal of L.

After $\mathbf{v}$ is computed, $\mathbf{w}$ is computed using a matrix-vector multiplication at line 11, which corresponds to equation (5.1), and is then modified in line 15, which corresponds to equation (5.2). Once $\mathbf{v}$ and $\mathbf{w}$ are determined, equation (5.3) could be used to update the unreduced matrix, but this is an expensive technique. Instead, the WY representation is used to make several $\mathbf{v}$ and $\mathbf{w}$ updates simultaneously. The $\mathbf{v}$ and $\mathbf{w}$ vectors are stored as columns in the V and W matrices. Outside of the algorithmic block, the V and W updates are performed with the matrix multiplications in line 19, which is a rank-$2b$ update. Within the block, the updates are never performed explicitly. In line 8, `updatex` updates the $\mathbf{x}$ vector with the accumulated V and W matrices to create the current $\mathbf{x}$ vector. In line 14, `updatew` is used to update the $\mathbf{w}$ vector with the V and W matrices. These two algorithms are displayed

Algorithm 4: `updatex`(A, i, V, W)

Data: A is $n \times n$; V and W are $n \times b$
$\mathbf{x} = A(i{:}n, i)$
$\mathbf{x} = \mathbf{x} - V(i{:}n, :)W(i, :)^T - W(i{:}n, :)V(i, :)^T$
$A(i{:}n, i) = \mathbf{x}$
Result: The ith column of A is updated with the current V and W matrices.

Algorithm 5: `updatew`$(\mathbf{w}, i, \mathbf{v}, V, W)$

Data: V and W are $n \times b$, $\mathbf{w}$ is length $n - i$
$\mathbf{p} = W(i{+}1{:}n, :)^T\mathbf{v}$
$\mathbf{w} = \mathbf{w} - V(i{+}1{:}n, :)\mathbf{p}$
$\mathbf{p} = V(i{+}1{:}n, :)^T\mathbf{v}$
$\mathbf{w} = \mathbf{w} - W(i{+}1{:}n, :)\mathbf{p}$
Result: $\mathbf{w}$ is updated with the current V and W matrices.

in Algorithms 4 and 5 and require few FLOPs since they operate on matrices and vectors with small dimensions. The costly steps in Algorithm 2 are the matrix-vector multiplication (in line 11) and the application of the rank-$2b$ update of V and W to the unreduced matrix after a block is annihilated (in line 19).

The trace of L is equal to the sum of the eigenvalues of L, and, because the eigenvalues of L are also the eigenvalues of the tridiagonal matrix, T, the trace of T is equal to the trace of L. If L is of low rank, then at some point during the reduction process, the trace of the tridiagonal tail will be equal to the trace of L. Then, the tridiagonal tail represents a submatrix of the complete tridiagonal matrix, with all of the other values in the tridiagonal matrix equal to zero. The eigenvalue decomposition of the current tridiagonal tail sufficiently represents the eigenvalue decomposition of the complete tridiagonal matrix. Hence, the tridiagonal reduction

process can be halted before it finishes, potentially saving many FLOPs. Line 20 in Algorithm 2 represents a test on the unreduced part of L which decides whether or not the tridiagonal reduction can be halted. This test can be as simple as comparing the trace of the current tridiagonal tail with the original trace of L.

5.3.1 Multi-Core Implementation

The LAPACK routine which performs real symmetric matrix tridiagonalization is called DSYTRD. The performance of all LAPACK routines depends on the performance of the BLAS subprograms they use. In this case, the efficiency of DSYTRD depends on the efficiency of DSYMV, the symmetric matrix-vector multiplication which is used for nearly every column of L, and DSYR2K, the symmetric matrix rank-$2b$ update which is used for nearly every algorithmic block.

Figure 5.1 shows the rate of performance in GFLOP per second versus the number of given CPU threads for DSYMV and DSYR2K for Machine A. DSYMV is a BLAS level-2 subprogram which is limited by memory bandwidth. Its number of FLOPs is proportional to the amount of matrix and vector data on which it operates. For this operation, the processor is not efficient because the rate that matrix data is supplied to the processor cannot match the rate of computation of the CPU cores. The figure shows that there is no improvement in performance for DSYMV as more CPU threads are added. DSYR2K, on the other hand, is a BLAS level-3 subprogram. It is compute-bound, not limited by memory bandwidth, and its performance improves as the number of threads is increased. This figure is helpful in understanding the parallel performance of DSYTRD, which is shown in Figure 5.2.

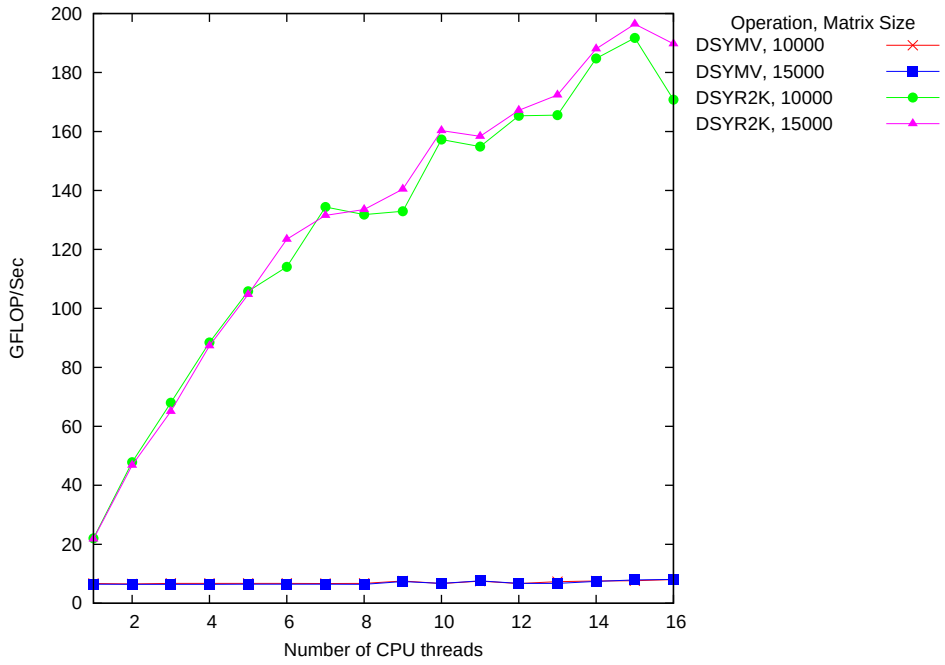


Figure 5.1: Performance of MKL DSYMV and DSYR2K ($b = 100$)

For low numbers of CPU threads, the highly parallelizable nature of DSYR2K dominates the behavior of DSYTRD in Figure 5.2. After about eight threads, however, the improvement in performance stalls due to the memory bandwidth limitation of DSYMV. The consequence of this limitation is a highly irregular and inconsistent rate of performance for higher numbers of CPU threads. Better parallelism can only be attained if the memory bandwidth limitation in the DSYMV operation is ameliorated, which is possible through the higher memory bandwidth of GPUs. Since DSYTRD is incapable of detecting whether or not the tridiagonal reduction process can halt early for a low rank matrix, a multi-core implementation of Algorithm 2 was created which performs this check and stops the tridiagonalization if possible.

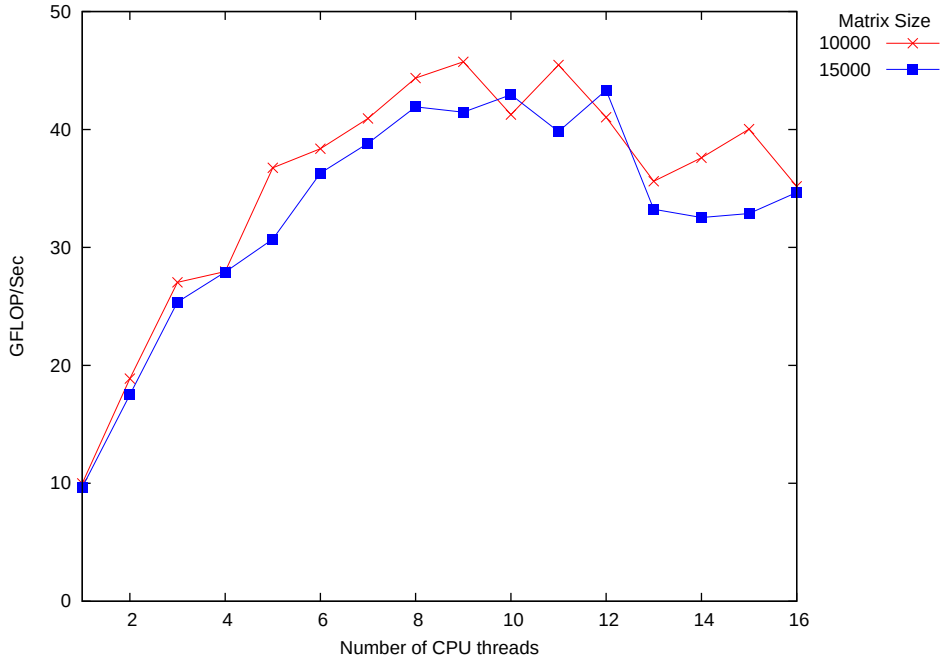


Figure 5.2: Performance of MKL DSYTRD

5.3.2 Single-GPU Implementation

CUBLAS takes advantage of the thousands of GPU cores and high memory bandwidth to provide very efficient DSYMV and DSYR2K routines. Recently, an option of DSYMV was released in CUBLAS which relies on "atomic" operations to give a more efficient DSYMV. Table 5.3 compares the performance of the 16-core MKL BLAS DSYMV with the new CUBLAS DSYMV. The CUBLAS DSYR2K is more efficient than MKL's DSYR2K as seen in Table 5.4. When the results in the two tables are compared, it is evident that although the GPU rates are faster than the MKL rates, the DSYMV routine remains the bottleneck for tridiagonal reduction in a GPU implementation.

Size	MKL 16-core	CUBLAS w/atomic
10000	7.80	53.1
15000	8.01	51.5

Table 5.3: Parallel Performance of DSYMV (GFLOP/Sec)

Size	MKL 16-core	CUBLAS
10000	170.8	657.1
15000	189.7	666.3

Table 5.4: Parallel Performance of DSYR2K (GFLOP/Sec)

It is straightforward to implement Algorithm 2 to use CUBLAS DSYMV and DSYR2K. Unfortunately, both of these efficient CUBLAS routines require the entire symmetric matrix to be held in memory, which means that about half of the matrix is stored and never referenced. Since the amount of memory on a GPU is limited, there is a maximum matrix size for which the entire matrix can fit on a GPU and can be tridiagonalized with a GPU implementation. A scheme which performs tridiagonal reduction on very large matrices and yet still takes advantage of GPUs is described in the next subsection.

5.3.3 Multi-GPU Implementation

For an implementation of the tridiagonalization process to use multiple GPUs efficiently, each GPU must be made to do an equal share of the work, and this will require the matrix data to be distributed approximately equally among the GPUs.

Larger matrices can be accommodated if only the lower triangle is stored in GPU memory. Then the heights of the stored portions of the various columns will differ. The matrix data can be distributed nearly equally among GPUs by allocating columns to them in a cyclic manner. Finally, if a matrix is too large to fit among the given GPUs, then a minimum portion of the matrix is stored in CPU memory. In this case, the CPU cores cooperate to perform the computations and updates for their local data.

In Algorithm 2, the annihilation of each column requires a matrix-vector multiplication which involves all of the remaining columns in the matrix. This distribution assigns columns of L to GPUs in a round-robin fashion, demonstrated below.

$$L = \begin{bmatrix} | & | & & | & | & | & \\ 1 & 2 & \cdots & g & 1 & 2 & \cdots \\ | & | & & | & | & | & \end{bmatrix}$$

The entries of L above the diagonal are not referenced, so storing these entries on GPUs wastes precious GPU memory. But, if the columns are stored on different GPUs, then the matrix data is not contiguous and BLAS level-3 operations cannot be used. Therefore, a block-packed scheme is used to store the 1-D cyclic columns on each GPU. For example, suppose a 16×16 L matrix is distributed among 2 GPUs. Figure 5.3 shows which elements of L GPU #2 owns, for a block-packed storage of width 3. Each block of local matrix data (corresponding to a rectangle in the table) is stored independently, but each block's data is stored contiguously. Matrix entries corresponding to global data above the diagonal are set to zero. This padding constitutes a negligible amount of storage space compared to the amount of storage space required to hold the values below and at the diagonal.

$L_{2,2}$	0	0					
$L_{3,2}$	0	0					
$L_{4,2}$	$L_{4,4}$	0					
$L_{5,2}$	$L_{5,4}$	0					
$L_{6,2}$	$L_{6,4}$	$L_{6,6}$					
$L_{7,2}$	$L_{7,4}$	$L_{7,6}$					
$L_{8,2}$	$L_{8,4}$	$L_{8,6}$	$L_{8,8}$	0	0		
$L_{9,2}$	$L_{9,4}$	$L_{9,6}$	$L_{9,8}$	0	0		
$L_{10,2}$	$L_{10,4}$	$L_{10,6}$	$L_{10,8}$	$L_{10,10}$	0		
$L_{11,2}$	$L_{11,4}$	$L_{11,6}$	$L_{11,8}$	$L_{11,10}$	0		
$L_{12,2}$	$L_{12,4}$	$L_{12,6}$	$L_{12,8}$	$L_{12,10}$	$L_{12,12}$		
$L_{13,2}$	$L_{13,4}$	$L_{13,6}$	$L_{13,8}$	$L_{13,10}$	$L_{13,12}$		
$L_{14,2}$	$L_{14,4}$	$L_{14,6}$	$L_{14,8}$	$L_{14,10}$	$L_{14,12}$	$L_{14,14}$	0
$L_{15,2}$	$L_{15,4}$	$L_{15,6}$	$L_{15,8}$	$L_{15,10}$	$L_{15,12}$	$L_{15,14}$	0
$L_{16,2}$	$L_{16,4}$	$L_{16,6}$	$L_{16,8}$	$L_{16,10}$	$L_{16,12}$	$L_{16,14}$	$L_{16,16}$

Figure 5.3: Local data for GPU #2 of a 16×16 distributed matrix, with block-packed storage of width 3.

Each GPU has its own memory space, so any data needed by one GPU, but stored on another, must be communicated across the PCI bus. When the sum of data stored on multiple GPUs is required, each GPU communicates the needed data to the CPU cores, which then perform the sum. The result of the sum is communicated back to the GPUs, if necessary. This heterogeneous approach is essential for the GPUs to operate independently, yet coordinate when necessary. The barrier routines in OpenMP and the CUDA routines which halt CPU execution until certain GPU tasks are completed, are used to ensure that all of the data in the various locations stay current.

Size	MKL 16-core	1 GPU	2 GPUs	3 GPUs	4 GPUs
10000	7.891	20.57	38.89	59.37	69.34
15000	8.090	23.13	44.57	67.07	95.08

Table 5.5: Parallel Performance of DSYMV (GFLOP/Sec)

Size	MKL 16-core	1 GPU	2 GPUs	3 GPUs	4 GPUs
10000	169.4	557.7	956.8	1295	1521
15000	185.3	610.1	1077	1462	1771

Table 5.6: Parallel Performance of DSYR2K (GFLOP/Sec)

CUBLAS does not provide DSYMV and DSYR2K routines compatible with this distribution of the matrix data, since consecutive columns of the matrix are on different GPUs. Instead, matrix-vector multiplications and rank-2k updates are performed locally by looping over blocks of matrix data and launching CUBLAS or specially created CUDA kernels to carry out the arithmetic. For DSYMV, each GPU computes its portion of the matrix-vector multiplication. Then, the complete product is obtained by taking the sum of all of the local products from each GPU.

Table 5.5 shows the improvement in performance for an implementation of the DSYMV operation using multiple GPUs on Machine A. The multi-GPU DSYR2K is implemented by having each GPU independently update its local block of matrix data with the pertinent parts of the V and W matrices using the CUBLAS DGEMM routine. Table 5.6 shows the performance of DSYR2K on Machine A.

A machine built for NVH analyses could potentially have CPU cores which

together match the performance of a single GPU. In this case, better performance may be gained by assigning some columns of L to the CPU cores. Also, the models in the Industrial Test Suite all have L matrices which can fit on one GPU, but this trend is not likely to continue. Historically, the rate at which GPU memory increases with each generation of GPUs is slower than the rate at which memory demands increase as more structural modes are included in analyses.

5.4 Real Backtransformation

After the weighted structural damping matrix is reduced to tridiagonal form, the eigensolution is computed for the tridiagonal matrix. A subset of the eigenpairs of the tridiagonal matrix is used to create the LRA of L. The nonzero eigenvalues of L are simply the retained eigenvalues of the tridiagonal matrix. The eigenvectors of L are computed by backtransforming the retained eigenvectors of the tridiagonal matrix by applying the Householder reflections used to create the tridiagonal matrix. This process is represented within the parentheses of equation (3.12). Forming Q, which is the product of the Householder reflections, explicitly as the product of the individual reflections is expensive, so, the WY representation is used to implicitly apply the updates to $\hat{\Phi}_T$ by blocks. A pseudocode which shows how this backtransformation is accomplished is presented in Algorithm 6.

In the algorithm, $\hat{\Phi}_T$ is the $N_{ev} \times N_{\text{LRA}}$ matrix which comprises the retained eigenvectors of the tridiagonal matrix. The backtransformation is performed over algorithmic blocks of size b. During the tridiagonal reduction, the reflection vectors $\mathbf{v}$ are stored in the lower triangle of L (below the subdiagonal), and the reflection

Algorithm 6: Blocked Real Backtransformation

Data: $L \in \mathbb{R}^{N_{ev} \times N_{ev}}$ is symmetric; b is block size; $V, W \in \mathbb{R}^{N_{ev} \times b}$; $\hat{\Phi}_T \in \mathbb{R}^{N_{ev} \times N_{\text{LRA}}}$

1 $N = \lceil (N_{ev} - 2)/b \rceil$
2 **for** $j = N : 1 : -1$ **do**
3 $V = 0, W = 0$
4 $s = (j-1)b + 1$
5 $t = \min(s+b-1, N_{ev} - 2)$
6 **for** $i = t : s : -1$ **do**
7 $\mathbf{v}(2 : N_{ev} - i + 1) = L(i+2 : N_{ev}, i)$
8 $v(1) = 1$
9 $\tau = z(i)$
10 $\mathbf{w} = \tau \mathbf{v}$
11 **if** $i < t$ **then**
12 $\mathbf{w} = (I + WY^T)\mathbf{w}$
13 $V(i+1 : N_{ev}, i-s+1) = \mathbf{v};\ W(i+1 : N_{ev}, i-s+1) = \mathbf{w}$
14 $\hat{\Phi}_T = (I + WY^T)\hat{\Phi}_T$

Result: $\hat{\Phi}_T$ is overwritten with $Q\hat{\Phi}_T$, where Q is the product of the Householder reflections, formed from the reflection vectors stored below the subdiagonal of L and the reflection scalars stored in $\mathbf{z}$.

scalars $\tau = 2/\mathbf{v}^T\mathbf{v}$ are stored in $\mathbf{z}$. The backtransformation algorithm refers to these stored values and computes each corresponding $\mathbf{w}$ in lines 10-12. The reflections are applied to $\hat{\Phi}_T$ in reverse order, from the left, using the V and W matrices through matrix-matrix multiplications, which are very efficient. Although the FLOP cost of the backtransformation process is higher than the tridiagonal reduction process, it does not have a memory bandwidth limitation, since the most costly step is a matrix-matrix multiplication. Thus, the backtransformation process is much more amenable to parallelization than the tridiagonal reduction process. The MKL LAPACK routine which performs real backtransformation for multi-core CPUs is called DORMTR.

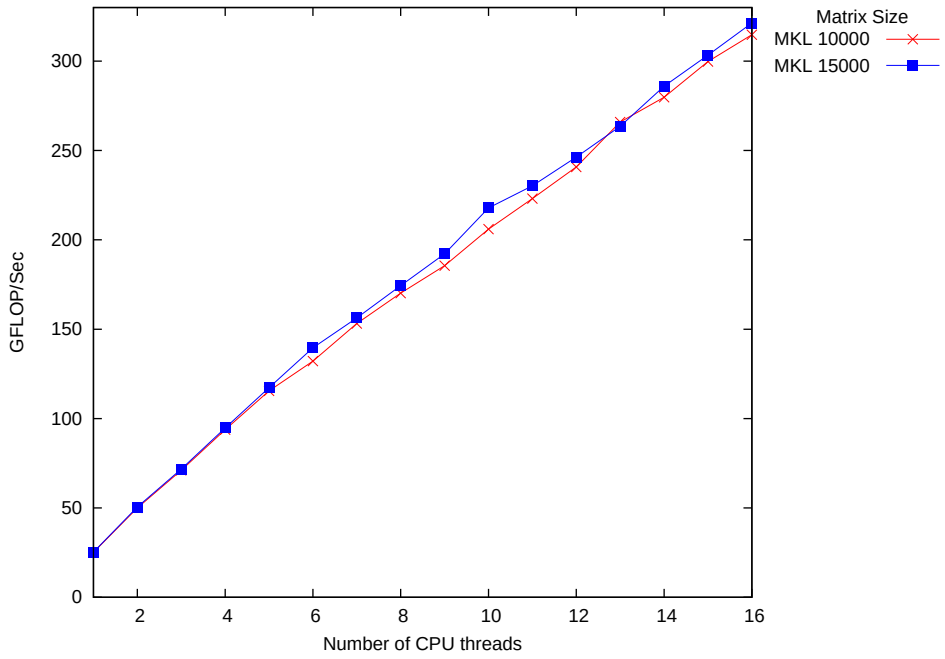


Figure 5.4: Performance of MKL DORMTR

Since its dominant step involves matrix-matrix multiplications, it is able to use all of the given CPU cores as seen in Figure 5.4.

It is straightforward to implement Algorithm 6 to take advantage of any number of GPUs in addition to the CPU cores. Since the updates to $\hat{\Phi}_T$ are applied from the left, $\hat{\Phi}_T$ is divided into panels and each panel is backtransformed independently. The number of panels is equal to the number of GPUs, plus one if the CPU cores participate in the computation. Because the GPUs and the CPU cores perform operations at different rates, the width of the panel allocated to each GPU is typically larger than the width allocated to the CPU cores.

5.5 Complex Symmetric Tridiagonal Reduction

The complex symmetric approach uses CSMES to find the eigenvalues and eigenvectors of C as defined in equation (2.7). CSMES's first task is to reduce C to tridiagonal form, accomplished using CO reflections which sequentially annihilate rows/columns of C. Unfortunately, the tridiagonal reduction of complex symmetric matrices suffers from an imprecision issue, mentioned at the end of Section 2.4. The imprecision is caused by cancellation events, which frequently appear when CO reflections are formed.

Algorithm 7 is a pseudocode which demonstrates how complex symmetric tridiagonal reduction is performed over algorithmic blocks. For convenience, in this dissertation, an implementation of this algorithm is called ZSYTRD. Most of the algorithm resembles that of the tridiagonal reduction of real symmetric matrices. The differences between the algorithms are due to the existence of CEs. Since CEs are not removed by a simple CO reflection, it is not possible to extend the WY representation to include all of the additional matrices in equation (4.15) used to remove CEs. As a result, when a CE appears, the current algorithmic block must be ended so that the CE can be removed. The CEs are not known until they are discovered in the tridiagonalization process. Hence, the "for loops" in Algorithm 2 are replaced with "while loops" in Algorithm 7 to accommodate the variable widths of the algorithmic blocks.

Within the inner "while loop", just before a column $\mathbf{x}$ is used to create the CO reflection, its CE value is determined using the routine `CEvalue`, which is listed in Algorithm 8 (described in equation (4.6)). If the CE value is larger than τ_{CE}, then

Algorithm 7: CE-aware Blocked CO Symmetric Tridiagonalization

Data: $C \in \mathbb{C}^{N_{ev} \times N_{ev}}$ is symmetric; b is default block size; $V, W \in \mathbb{C}^{N_{ev} \times b}$, τ_{CE} is the CE tolerance, J is the CE data struct

$r = N_{ev} - 2$
$s = 1$
while $r > 0$ **do**
  $V = 0, W = 0$
  $t = \min(s+b-1, N_{ev} - 2)$
  $\mathbf{x} = C(s+1{:}N_{ev}, s)$
  if `CEvalue(x)` $> \tau_{\text{CE}}$ **then**
    `removeCE`$(C, s, \tau_{\text{CE}}, J)$
  $i = s$
  while $i \leq t$ **do**
    if $i > s$ **then**
      $x_0 = C(i{:}N_{ev}, i)$
      `updatex`(C, i, V, W)
    $\mathbf{x} = C(i+1{:}N_{ev}, i)$
    if `CEvalue(x)` $> \tau_{\text{CE}}$ **then**
      $t := i - 1$
      $C(i{:}N_{ev}, i) = x_0$
    else
      $[\mathbf{v}, \sigma, \tau]$ = `buildv(x)`
      $\mathbf{w} = C(i+1{:}N_{ev}, i+1{:}N_{ev})\mathbf{v}$
      $C(i+1, i) = \sigma$
      if $i > s$ **then**
        `updatew`$(\mathbf{w}, i, V, W)$
      $\mathbf{w} = \tau \left[\mathbf{w} - \left(\frac{1}{2}\tau \mathbf{w}^T \mathbf{v}\right) \mathbf{v}\right]$
      $V(i+1{:}N_{ev}, i-s) = \mathbf{v};\ W(i+1{:}N_{ev}, i-s) = \mathbf{w}$
      $C(i+2{:}N_{ev}, i) = \mathbf{v}(2{:}N_{ev}-i+1),\ z(i) = \tau$
      $i = i + 1$
  $V_0 = V(t+1{:}N_{ev}, :);\ W_0 = W(t+1{:}N_{ev}, :)$
  $C(t+1{:}N_{ev}, t+1{:}N_{ev}) = C(t+1{:}N_{ev}, t+1{:}N_{ev}) - V_0 W_0^T - W_0 V_0^T$
  $r = r - t + s - 1;\ s = s + t - s + 1$

Result: The tridiagonal matrix is stored in the diagonal and subdiagonal entries of C. The lower triangle of C stores the reflection vectors and $\mathbf{z}$ stores the reflection scalars τ.

Algorithm 8: `CEvalue`(x)

input : $\mathbf{x}$ vector of length n
output: The cancellation event value, γ
$\beta = |\mathbf{x}^T\mathbf{x}|$
$\alpha = 0$
for $i = 1 : n$ **do**
  $t = \mathbb{R}(x(i))$
  $\alpha = \alpha + t^2$
$\gamma = \log_{10}(\alpha/\beta)$

the CE is removed before more tridiagonalization continues. The loop is exited, the work performed by `updatex` is reverted in line 17, and the accumulated V and W matrices are used to update the unreduced matrix in line 29. The outcome of these steps is that a CE can only be at the beginning of an algorithmic block.

A CE is detected in line 7 and removed in line 8 using `removeCE` which is displayed in Algorithm 9. The `removeCE` routine performs the steps outlined in Section 4.3 to remove a CE at column κ. Usually, this process removes the CE with a single attempt, but if is unsuccessful, the matrix is reverted back to its original state, the size of the small tridiagonal matrix in line 5 is increased by one, and the process is restarted.

The multiplication in line 7 of `removeCE` causes the first column of Q to be modified, which, according to Theorem 3, implies that the κ-th subdiagonal value of the tridiagonal matrix is modified. This causes the denominator in equation (4.6) to be modified which produces a new CE value. The CE value can also be modified by altering the numerator in the equation. In line 11, the fill-in column is annihilated using a chosen matrix X. This X can represent, for instance, a CO reflection or

Algorithm 9: `removeCE`(C, κ, τ_{CE}, J)

input : $C \in \mathbb{C}^{N_{ev} \times N_{ev}}$ with CE at column κ, CE tolerance τ_{CE}
output: The transformed C with no CE at column κ

`// Information about these steps is found in Section 4.3.`

$p = 5$
$p_{max} = 10$
$\mathbf{x} = C(\kappa{+}1 : N_{ev}, \kappa)$
while `CEvalue` $(\mathbf{x}) > \tau_{\text{CE}}$ **do**
  $T_1 = C(1:p, 1:p)$
  Solve: $T_1 \Phi_{T_1} = \Phi_{T_1} \Lambda_{T_1}$
  `// Define` Υ `according to equation` (4.9).
  $C' = \Upsilon^T C \Upsilon$
  `// The top-left corner of` C' `has an arrowhead form. Let` Ω `represent the product of the CO rotations which transform` C' `so that it has a tail with a bulge.`
  $C' = \Omega^T C \Omega$
  `// Permute the bulge to progress toward the unreduced part of` C'.
  $C' = P^T C P$
  `// Let` Γ `represent the product of the CO rotations which chase the bulge to the unreduced part of` C'.
  $C' = \Gamma^T C \Gamma$
  `// The last CO rotation fills in the previous column. Annihilate this column using` X.
  $C' = X^T C' X$
  $\mathbf{x} = C'(\kappa{+}1 : N_{ev}, \kappa)$
  $p = p + 1$
  if $p > p_{max}$ **then** break
$C = C'$

`// Store the CE data in the struct,` J.

$J.\Upsilon_\kappa = \Upsilon$
$J.\Omega_\kappa = \Omega$
$J.P_\kappa = P$
$J.\Gamma_\kappa = \Gamma$
$J.X_\kappa = X$

the product of CO rotations. Different choices of X can modify the numerator of (4.6) to produce different CE values. A single CO reflection which annihilates a column is more efficient than using many CO rotations; however, the CO rotations can be applied across rows or columns of the unreduced matrix in parallel, giving it a performance advantage over using a single CO reflection.

Equation (5.4) shows the status of the complex symmetric matrix just before the final entry in the fill-in column is annihilated. In an implementation of this process, only the lower or upper triangle of the matrix needs to be referenced, but the complete matrix is shown to provide insight into this step. In the equation, CO rotations have annihilated entries in the fill-in row/column which are marked with "0", and have modified the unreduced part of the matrix. The CO rotations applied from the left have modified rows of the unreduced matrix, marked with "$\times_l$", and CO rotations applied from the right have modified columns of the matrix, marked with "$\times_r$". The "$\times_{lr}$" entries represent values which have been modified by CO rotations

from both the left and the right.

$$
\begin{array}{c}
 \\
 \\
 \\
 \\
 \\
\kappa \\
 \\
 \\
 \\
N_{ev}
\end{array}
\begin{array}{c}
\begin{array}{ccccccccccc}
 & & & & & \kappa & & & & & N_{ev}
\end{array} \\
\left[\begin{array}{ccccccccccc}
\times & \times & & & & & & & & & \\
\times & \ddots & \ddots & & & & & & & & \\
 & \ddots & \times & \times & & & & & & & \\
 & & \times & \times & \times & & & & & & \\
 & & & \times & \times & \times & + & 0 & \cdots & 0 & \\
 & & & & \times & \times & \times_r & \times_r & \cdots & \times_r & \\
 & & & & + & \times_l & \times_{lr} & \times_{lr} & \cdots & \times_{lr} & \\
 & & & & 0 & \times_l & \times_{lr} & \times_{lr} & \cdots & \times_{lr} & \\
 & & & & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \\
 & & & & 0 & \times_l & \times_{lr} & \times_{lr} & \cdots & \times_{lr} &
\end{array}\right].
\end{array}
\tag{5.4}
$$

Before the final entry in the fill-in row/column, marked with "+", is annihilated, the current values in row/column κ (which holds the CE) can be computed without referencing the rest of the unreduced matrix. But, when the last CO rotation is applied from the left and right, row/column κ becomes dependent on the current values of row/column κ+1, which means that in order to compute the modified row/column κ, every entry in the fill-in row/column must be annihilated. Then, the success of a CE removal attempt is only known after the fill-in column has been completely annihilated and its rotations have updated the unreduced part of the matrix. The updates to the unreduced matrix are BLAS level-2 subprograms, which are limited by memory bandwidth. The primary cost of removing a CE is the annihilation of the fill-in column.

With a lower CE tolerance, more CEs are detected and removed, and the algorithmic blocks become more segmented. So, as τ_{CE} is lowered, ZSYR2K performs rank-2b updates to the unreduced matrix outside of the algorithmic block with thinner V and W matrices. The ZSYR2K routine is less efficient when the width of the updating matrices is reduced, and it must be called more often as the number of algorithmic blocks becomes higher.

5.5.1 Multi-Core Implementation

The performance of a multi-core implementation of ZSYTRD depends on the performance of the multi-core complex symmetric matrix-vector multiplication, the rank-2b update, and CE removals. The BLAS subprogram which performs complex symmetric matrix-vector multiplication is called ZSYMV. For a chosen matrix size, this routine executes four times as many FLOPs as DSYMV, yet complex matrix data is only twice as large as real matrix data. This suggests that the memory bandwidth limitation is more relaxed for ZSYMV than for DSYMV. Unfortunately, MKL's BLAS ZSYMV does not appear to take advantage of this observation. Its performance with 1 thread and 16 threads are indistinguishable from one another, much like DSYMV. Therefore, Algorithm 10 is used to perform the multiplication and takes advantage of multiple threads, given a workspace. Figure 5.5 shows the improvement in parallel performance of an implementation of Algorithm 10 over MKL ZSYMV. As expected, the BLAS level-3 MKL ZSYR2K subprogram takes advantage of multiple CPU cores very well which is also seen in Figure 5.5.

The most costly step in removing cancellation events is the annihilation of

Algorithm 10: Efficient Complex Symmetric Matrix-Vector Multiplication

Data: C is $n \times n$ and symmetric; $\mathbf{x}$ and $\mathbf{y}$ are vectors of length n; w_0 is the default block width, N_t is the number of OpenMP threads used in the parallel region, $\mathbf{y}^t$ is thread t's local copy of $\mathbf{y}$.

$N = \lceil n/w_0 \rceil$
begin parallel region
$t =$ a thread's rank
$\mathbf{y}^t = \mathbf{0}$
for $b = 1$ ***to*** N **do**
  if $b > 1$ **then**
    $s = w_0(b-1)$
    $w = \min(w_0, n-s+1)$
    for $j = 1$ ***to*** w **do**
      for $i = 1$ ***to*** s **do**
        $c = C(i, j+s)$
        $y^t(i) \mathrel{+}= cx(j+s)$
        $y^t(j+s) \mathrel{+}= cx(i)$
      end
    end
  end
  for $j = 1$ ***to*** w **do**
    for $i = 1$ ***to*** $j-1$ **do**
      $c = C(i+s, j+s)$
      $y^t(i+s) \mathrel{+}= cx(j+s)$
      $y^t(j+s) \mathrel{+}= cx(i+s)$
    end
  end
end
end parallel region
$y = \sum_{t=1}^{N_t} y^t$
Result: $\mathbf{y} = C\mathbf{x}$

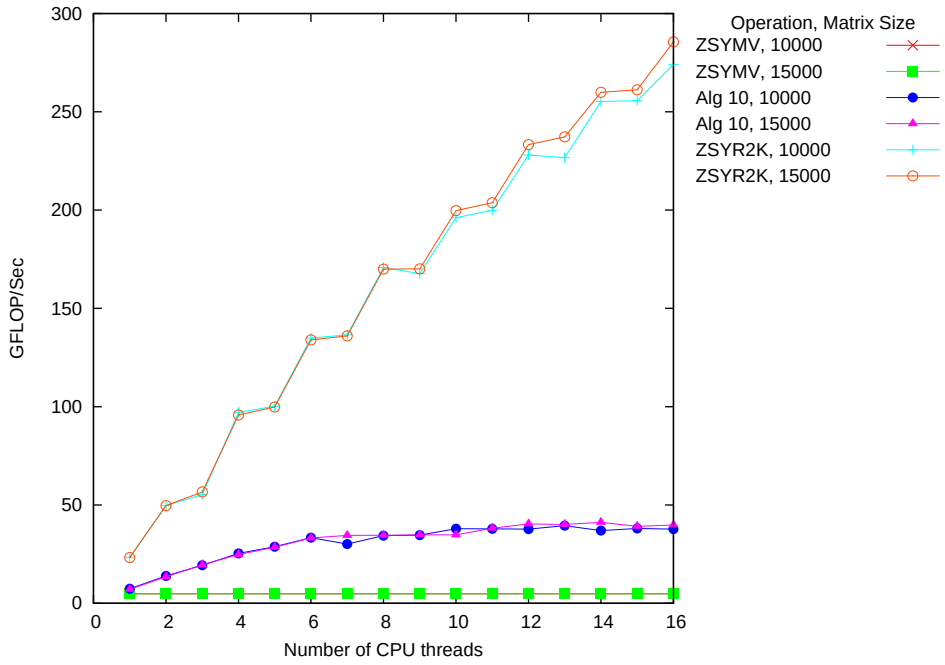


Figure 5.5: Performance of MKL ZSYMV, Alg. 10, and MKL ZSYR2K

the fill-in column using CO rotations. When a rotation is created, multiple CPU threads work independently, applying the rotation to sections of the rows or columns of the unreduced matrix. Also, to prepare for the unlikely event that the CE removal attempt is unsuccessful, the CPU threads copy the values from the unreduced part of C to the unreferenced triangle of C in parallel before the fill-in rotations are applied. If the attempt is unsuccessful, the CPU threads reinstate the original values in parallel before the next CE removal attempt begins.

Size	MKL 16-cores	Alg 10 16-cores	CUBLAS w/atomic
10000	4.82	40.9	59.6
15000	4.83	41.8	61.6

Table 5.7: Parallel Performance of Single GPU ZSYMV (GFLOP/Sec)

Size	MKL 16-core	CUBLAS
10000	276.0	857.7
15000	285.5	866.6

Table 5.8: Parallel Performance of Single GPU ZSYR2K (GFLOP/Sec)

5.5.2 Single-GPU Implementation

A new option of ZSYMV has been made available in the latest version of CUBLAS which uses atomic operations to provide a more efficient complex symmetric matrix-vector multiplication. Table 5.7 compares the performance of three versions of ZSYMV: MKL ZSYMV, the ZSYMV from Algorithm 10 and the new CUBLAS option.

The CUBLAS ZSYR2K is substantially more efficient than MKL's ZSYR2K as seen in Table 5.8. An implementation of the complex symmetric tridiagonal reduction process which takes advantage of a single GPU is straightforward to create. Unlike the real symmetric case, the upper half of the matrix is not an unreferenced part of the matrix as it can be used to store the lower part of the matrix in case a CE removal attempt fails. Unfortunately, since the matrix data is complex, the maximum matrix size for which a matrix can fit on a single GPU is much lower than the maximum size

Size	1 GPU	2 GPUs	3 GPUs	4 GPUs
10000	35.20	70.03	101.2	128.5
15000	36.95	72.44	107.8	137.7

Table 5.9: Parallel Performance of Multi-GPU ZSYMV (GFLOP/Sec)

for real tridiagonal reduction.

5.5.3 Multi-GPU Implementation

As with the real case, the multi-GPU implementation uses block-packed storage (described in Section 5.3.3) to store the lower triangle of the matrix. If a CE removal attempt fails, the original values in the unreduced part of the matrix must be reinstated, in order to begin a new attempt. So, additional GPU memory must be allocated to store the original matrix values before a CE removal attempt is made. Since a CE could exist in nearly any annihilated column, the amount of additional GPU memory is equal to the amount of GPU memory used to store the current matrix data.

Table 5.9 shows how the global ZSYMV performance improves when more GPUs participate in the computation. ZSYR2K performance is also improved as additional GPUs share the workload as seen in Table 5.10, but the ZSYMV operation remains a bottleneck for complex symmetric tridiagonalization.

The CE removal process first operates on the tridiagonal tail, which comprises vector data, and these operations are performed efficiently by the CPU. When the fill-in column is annihilated, the procedure of applying the CO rotations to the unreduced

Size	1 GPU	2 GPUs	3 GPUs	4 GPUs
10000	557.7	956.8	1295	1521
15000	610.1	1077	1462	1771

Table 5.10: Parallel Performance of Multi-GPU ZSYR2K (GFLOP/Sec)

matrix is complicated, since the matrix data is distributed across the GPUs. This operation is broken up into three steps, in order to minimize the amount of data transfers and the number of synchronizations among computing resources.

The first step in the operation is annihilating the fill-in column with CO rotations. Since the amount of matrix data is small, this step is solely performed by the CPU. The $\sin\theta$ and $\cos\theta$ values which were used to create the CO rotations are transferred to all the GPUs in anticipation of the next step. The second step is creating the CO rotations from the given $\sin\theta$ and $\cos\theta$ values and applying them from the left to the unreduced matrix. Each CO rotation causes a row of the matrix to become a linear combination of itself and a neighboring row. Each column stored on a GPU contains consecutive rows of matrix data; therefore, all of the GPUs perform this step independently of one another efficiently. The third step is applying the CO rotations from the right to the unreduced matrix. Each CO rotation causes a column of the matrix to become a linear combination of itself and a neighboring column. Since adjacent columns of the matrix are stored on different GPUs, this step is carried out most efficiently by transferring each GPU's part of the unreduced matrix to the CPU, arranging the columns of the matrix into the correct order, and applying the CO rotations using the CPU. After the CO rotations are applied, the updated unreduced matrix is redistributed to the GPUs.

The latest GPUs are capable of receiving data while they transmit other data and the data transfers can be conducted without disrupting the CPU's computations. In order to exploit these GPU features which improve the performance of the third step, the unreduced matrix is divided into panels and the updates from the right are performed on successive panels. While a panel is updated from the right by CO rotations by the CPU, the columns of the next panel are transmitted to the CPU and the previous panel is redistributed to the GPUs.

5.6 Complex Backtransformation

After the eigenvectors and eigenvalues of the complex symmetric tridiagonal matrix are determined, the eigenvectors of the tridiagonal matrix are backtransformed into the eigenvectors of C through $\Phi_C = Q\Phi_T$. Algorithm 11 is a pseudocode for this process which resembles the algorithm for real backtransformation. Line 16 determines when the CE updates (listed in Table 4.1) are applied. In line 17, the matrices which were used to remove CEs in the tridiagonal process are applied to Φ_C in reverse order in `updateCE`, displayed in Algorithm 12. Except for the computations which pertain to the CE updates, the parallel implementations of complex backtransformation are similar to the implementations of real backtransformation. Since every update, including those in `updateCE`, is performed from the left, the N_{ev} columns of Φ_C are subdivided into panels, where each GPU (and the CPU cores, if used) is responsible for the backtransformation of one panel. Similar to the case for real backtransformation, each panel is backtransformed independently.

Algorithm 11: CE-aware Blocked Complex Backtransformation

Data: $C \in \mathbb{C}^{N_{ev} \times N_{ev}}$ is symmetric; b is maximum block width from tridiagonal reduction; $V, W \in \mathbb{C}^{N_{ev} \times b}$; $\Phi_T \in \mathbb{C}^{N_{ev} \times N_{ev}}$, J is a struct which holds CE data

```
1  N = number of blocks in tridiagonal reduction
2  t = N_ev − 2
3  for j = N : 1 : −1 do
4      V = 0, W = 0
5      σ = width of block j
6      s = t − σ + 1
7      for i = t : s : −1 do
8          v(2 : N_ev − i + 1) = L(i + 2 : N_ev, i)
9          v(1) = 1
10         τ = z(i)
11         w = τv
12         if i < t then
13             w = (I + WY^T)w
14         V(i + 1 : N_ev, i − s + 1) = v;  W(i + 1 : N_ev, i − s + 1) = w
15     Φ_T = (I + WY^T)Φ_C
16     if CE was present before next block then
17         updateCE(Φ_C, J, s)
18     t = t − σ
```

Result: Φ_T is overwritten with $Q\Phi_T$, where Q is the product of the CO reflections, formed from the reflection vectors stored below the subdiagonal of C and the reflection scalars stored in $\mathbf{z}$, and the CE data stored in J.

5.7 The Fast Frequency Response Solver (FastFRS)

The two new approaches for solving modal FRPs are implemented by the author of this dissertation in the commercial software package, FastFRS, which is currently licensed by several automobile companies around the world. FastFRS is designed to take advantage of multiple CPU cores on a single node machine. A newly

Algorithm 12: `updateCE`(Φ, J, κ)

input : Φ, struct J, column κ which had a CE in tridiagonal reduction process

output: The transformed Φ, having been updated with CE matrices

`// Information about these steps is found in Section 4.4.`

1 $\Phi = J.X_\kappa \Phi$
2 $\Phi = J.\Gamma_\kappa \Phi$
3 $\Phi = J.P_\kappa \Phi$
4 $\Phi = J.\Omega_\kappa \Phi$
5 $\Phi = J.\Upsilon_\kappa \Phi$

released version of FastFRS can take advantage of multiple GPUs in addition to the CPU cores. There are two parts to every FastFRS job:

- The "Set Up" part of FastFRS includes all of the frequency independent work for solving a modal FRP. First, the singular value decomposition of B is determined as seen in equation (2.3). If B is symmetric, an eigenvalue decomposition is used instead. Next, either the LRA of K_s is computed for the low rank approach or CSMES finds the eigensolution of C for the complex symmetric approach. Operations in this part create the constant matrices which are used to compute responses using the SMW formula. All of the available CPU cores and GPUs coordinate to compute the needed eigenvalue decompositions or matrix multiplications. Sections 5.3 through 5.6 describe the parallel implementations for this part. For the low rank approach, if K_s is low rank, the cost of this step is negligible since the tridiagonal process can be halted early. For the complex symmetric approach, CSMES is usually more costly than its frequency dependent work. Additionally, some matrices must be transformed through Φ_C, such

as U and A.

- After the decomposition of either approach is determined, the response is computed using the SMW formula (with a partitioned approach, if necessary). Since D in every SMW formula variation (Table 3.2, 3.3, 4.3, or 4.4) is frequency-dependent, the SMW formula must be solved at every frequency. This is called the "Frequency Sweep" part of FastFRS. The frequency range is divided into groups of response frequencies, and each group is assigned to either a GPU with a managing CPU thread, or just a single CPU thread. Because of this, the threads with GPUs finish computing a single response much faster than their CPU-only counterparts. FastFRS uses a scheduler to assign frequencies to CPU cores, which gives preference to the CPU threads with a GPU. In the low rank approach, this is the dominant part of FastFRS.

Figure 5.6 gives an overview of how FastFRS takes advantage of c CPU cores and g GPUs. The Set Up part of FastFRS uses all the CPU cores and GPUs to compute the necessary frequency independent matrix products and decompositions. All of the CPU cores function as a single team, coordinating with the multiple GPUs to complete the tasks. In the Frequency Sweep part of FastFRS, the c threads operate independently, with the first g threads computing responses at a higher rate than the other threads. A scheduler attempts to keep all of the threads active until the Frequency Sweep is completed. Thus, FastFRS utilizes all available resources in order to provide solutions of modal FRPs very efficiently.

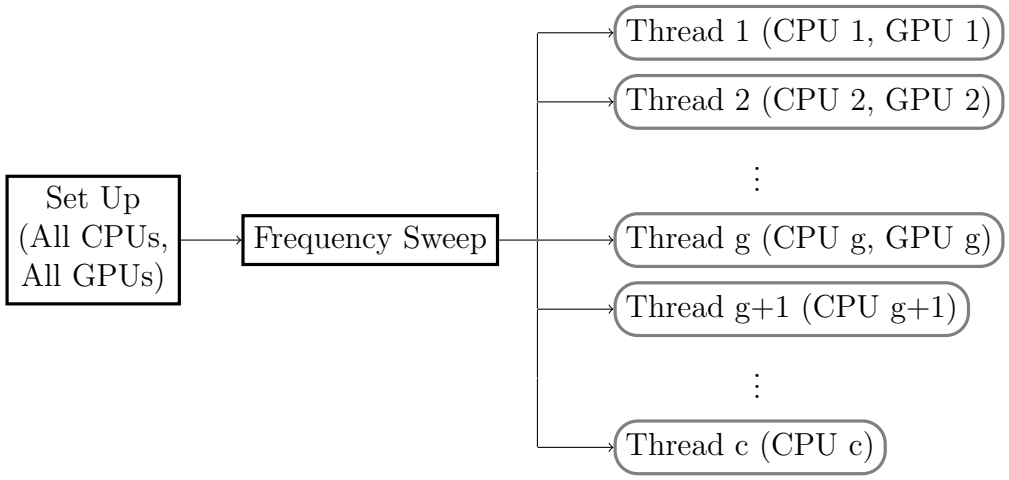


Figure 5.6: Parallel Tasks for FastFRS

Chapter 6

Results

The automobile models in the Industrial Test Suite in Table 1.3 vary in the numbers of structural modes, acoustic fluid modes, load cases, and frequency ranges of interest. Together, they encapsulate a spectrum of modal frequency response problems (FRPs) that an analyst of noise, vibration, and harshness (NVH) might encounter. Machine A (described in Chapter 5) is a single-node, multi-core, computing machine with multiple graphics processing units (GPUs), similar to the high performance computing (HPC) machines that automobile companies use for frequency response analyses. This chapter presents the accuracy and performance results for the low rank and complex symmetric approaches used to solve the modal FRPs of the Industrial Test Suite models as executed on Machine A.

In both new approaches, a pre-defined tolerance needs to be established. For the low rank approach, the value of the low rank approximation (LRA) tolerance, τ_{LRA}, determines the rank of the weighted modal structural damping matrix as shown in equation (3.8). If τ_{LRA} is equal to zero, then the complete, rather than partial, eigensolution represents the weighted modal structural damping matrix. The representation of K_s is no longer an approximation, and the low rank approach becomes an exact, full-rank method. For the complex symmetric approach, the value of the can-

cellation event (CE) tolerance, τ_{CE}, determines the quality of the complex symmetric tridiagonal reduction. Before a column is annihilated in the reduction, the column's CE value from equation (4.6) is computed. Then, if the CE value is greater than τ_{CE}, the CE is removed before the reduction process continues. As τ_{CE} approaches zero, the cancellation error in the eigenvalue decomposition is driven to zero and the accuracy of the solution approaches the full accuracy offered by machine precision. When the tolerances of both approaches are lowered, the solutions of modal FRPs using either approach align with the exact solutions. But, when the tolerances are lowered, the total FLOP costs of the approaches increase. The values for the tolerances that result in acceptable frequency response accuracy at minimal computational cost cannot be known without experimentation.

One heuristic method to determine the appropriate tolerance for either approach is to compare approximate solutions obtained using different tolerance values to the exact solution. The tolerance can then be chosen as the highest value that gives a solution close enough to the exact solution. Computing exact solutions is expensive, so it may be more efficient to compare approximate solutions as a tolerance is reduced, and choose a tolerance value beyond which improvement in solution accuracy is negligible. After the tolerances are set, it is important to periodically verify that the chosen values continue to result in acceptable accuracy. The company that was responsible for distributing the FastFRS software concluded after testing that setting τ_{LRA} to 0.001 and τ_{CE} to 3.5 would reliably produce acceptable solution accuracy for the low rank and complex symmetric approaches, respectively.

Typically, an NVH analyst uses frequency response solutions to give guidance

for design decisions or optimizations. The accuracy of the solutions is determined "visually" through a graph, rather than numerically to a certain number of digits. With an appropriate tolerance, an approximate method is considered acceptable when their graphs visually match.

An analysis of the accuracy of the low rank and complex symmetric approaches requires a virtually exact solution to which the solutions to modal FRPs can be compared, and a method for comparing the solutions. In Section 6.1, the FastFRS brute-force approach for obtaining exact solutions of modal FRPs is presented. Section 6.2 shows how the results of modal FRPs solved in two different ways can be compared to one another, and describes a technique that is used in industry to compute a measure of the difference between them. Then, Sections 6.3 and 6.4 present accuracy results of the low rank and complex symmetric approaches, respectively. In Section 6.5, the FastFRS solution strategy, which gives an analyst guidance for determining whether the low rank approach or the complex symmetric approach offers faster performance, is presented. The strategy is used on the models in the Industrial Test Suite and suggests the approach which is favored for each model. The performance results of the low rank approach are shown in Section 6.6 and those for the complex symmetric approach are shown in Section 6.7. All of these were collected from executions on Machine A using all of the CPU cores and none, some, or all of the available GPUs. Finally, Section 6.8 confirms that the favored approaches, selected using the FastFRS solution strategy in Section 6.5, are the appropriate choices.

6.1 The FastFRS Brute-Force Approach

A brute-force approach is implemented in FastFRS which provides the exact solutions to modal FRPs. This approach is expensive because it requires $O(N_{ev}^3)$ FLOPs at each frequency, but it can be used to determine the appropriate tolerances for the low rank and complex symmetric approaches. For models with structure-fluid interaction, the solutions are computed in one of two ways. If the acoustic fluid matrices are full, the entire coefficient matrix in equation (1.13) is formed and the coefficient matrix is factored to solve the modal FRP at every frequency. The total FLOP cost of this approach is

$$\text{Cost} = \left[\frac{4}{3}\left(N_{ev}+N_f\right)^3 + 8\left(N_{ev}+N_f\right)^2 N_c\right] N_{freq}. \tag{6.1}$$

For models with diagonal acoustic fluid matrices, like Models D, E, and F in the Industrial Test Suite, it is best to use a partitioned solution approach to take advantage of the fluid matrices being diagonal. The steps which solve the modal FRP in this manner are listed in Table 6.1. The total FLOP cost using this form of the brute-force approach is

$$\text{Cost} = \left[\frac{4}{3}N_{ev}^3 + 12N_{ev}N_fN_c + 4N_{ev}^2N_f + 8N_{ev}^2N_c\right] N_{freq} \tag{6.2}$$

which is less than the cost in equation (6.1). For this chapter and the next, the "brute-force approach" specifically refers to either of the two methods above (depending on the whether the acoustic fluid matrices are diagonal) which provide exact solutions and are implemented in FastFRS.

Step	Task	FLOP Cost
(1)	$F_f := Z_f^{-1} F_f(\omega)$, $\bar{A} := A Z_f^{-1}(\omega)$	$4N_f N_c + 2N_{ev} N_f$
(2)	$F_s := F_s - i\omega A F_f$	$4N_{ev} N_f N_c$
(3)	$Z_s := Z_s + \omega^2 A \bar{A}^T$	$4N_{ev}^2 N_f$
(4)	$X_s := Z_s^{-1} F_s$	$\frac{4}{3} N_{ev}^3 + 8N_{ev}^2 N_c$
(5)	$X_f := F_f - i\omega \bar{A}^T X_s$	$8N_{ev} N_f N_c$

Table 6.1: FastFRS Brute-Force Approach Operations per Frequency for Modal FRPs with Diagonal Acoustic Fluid Matrices

6.2 Comparing FRFs

FastFRS returns solutions of FRPs in modal space, so the solutions are back-transformed to FE space through equation (1.14) to be of use to analysts. These FE solutions are examined by plotting the displacements, velocities, or accelerations of a particular grid point and load case combination as a function of frequency. These types of graphs are called frequency response function (FRF) graphs. Since there is an FRF for every degree of freedom for every load case, an FRP could potentially lead to hundreds of millions of FRFs. However, analysts are typically only interested in the responses in a relatively small number of degrees of freedom of the automobile, so only the rows of Φ_s which correspond to those degrees of freedom are used in the transformation. This reduces the number of necessary FRFs significantly.

The number of FRFs in a set of FRFs is equal to the number of degrees of freedom of interest times the number of given load cases. The purpose of comparing two sets of FRFs is to determine whether or not the different methods used to create the FRFs yield equivalent responses. This can be accomplished by systematically

plotting the FRF from each set corresponding to the same degree of freedom and load case on a graph and noting the differences, which is a cumbersome task. Instead of examining every pair of FRFs, it is sufficient to first determine which pair of FRFs has the greatest difference in values over all frequencies. Then, if the "worst case" FRF pair is virtually indistinguishable on an FRF plot, then the two sets of FRFs are considered to be equivalent, and the methods used to compute the responses are considered to be equivalent.

In the NVH community, an informal measure of the difference between two computed FRFs has been developed and is used often for evaluating solution accuracy. The measurement takes into account the differences in the FRFs over all frequencies, without allowing a single large difference to dominate the calculation. The numerator of the error measurement is a sum of the differences of the squares of the response values, and the denominator is the sum of the averages of the response values, squared. Let $\mathbf{R}$ represent a vector containing the responses sorted by frequency for a particular degree of freedom and load case combination from a set of FRFs. Let $\mathbf{S}$ represent the corresponding responses from a second set of FRFs. The FRF error measure is computed in the following way.

$$\text{FRF error measure } = \begin{cases} 0 & \text{if } \mathbf{R} = \mathbf{S} = \mathbf{0} \\ \dfrac{\sum_{i=1}^{N_{freq}} \left| R_i^2 - S_i^2 \right|}{\sum_{i=1}^{N_{freq}} \left(\dfrac{R_i + S_i}{2} \right)^2} & \text{otherwise} \end{cases} \tag{6.3}$$

The "worst case" FRF pair is the one with the highest FRF error measure.

For example, suppose that an analyst is interested in comparing solutions obtained from using the low rank approach with solutions obtained from using the brute-force approach. After FastFRS produces a set of FRFs from using each approach, the analyst has two sets of FRFs to compare. An FRF from the first set has a corresponding FRF in the second set. Let **R** represent a single FRF from the first set (created from the low rank approach) and **S** represent the corresponding FRF from the second set (created from the brute-force approach). Then, equation (6.3) is used to compute the error measure between those two particular FRFs. The process is repeated for every FRF over all degrees of freedom and all load cases in order to determine which FRF pair is the "worst case".

6.3 Accuracy of the Low Rank Approach

In the low rank approach, τ_{LRA} determines the rank of the weighted modal structural damping matrix. Figures 6.1 and 6.2 demonstrate the accuracy of the low rank approach using the τ_{LRA} that is currently used in industry. Equation (6.3) was used to find the grid point and load case combination with the greatest difference between the low rank approach and the brute-force approach for two sample models in the Industrial Test Suite. These "worst case" FRFs are shown in Figure 6.1, which plots velocity responses versus the frequencies for Model D, and Figure 6.2, which plots displacement responses for Model F. Both figures show that the FRFs from the low rank approach are virtually indistinguishable from their brute-force approach counterparts. The other models in the Industrial Test Suite produce similarly indistiguishable "worst case" plots, which demonstrates that using the standard τ_{LRA}

leads to acceptable solutions of modal FRPs.

6.4 Accuracy of the Complex Symmetric Approach

The CE tolerance specifies the number of digits which are permitted to be lost in the application of a CO reflection (defined in equation (2.11)) during the tridiagonal reduction of the complex symmetric matrix, C. The precision of the eigenvalues and eigenvectors of C from CSMES is determined by τ_{CE} which influences the accuracy of the solutions of the modal FRP. Equation (6.3) was used to determine the grid point and load case corresponding to the greatest difference between the response computed from the complex symmetric and brute-force approaches. The accuracy of the complex symmetric approach is shown in two figures. Figure 6.3 plots the "worst case" FRF graphs of acceleration vs. frequency for the complex symmetric and brute-force approaches for Model E. The same grid point and load case used to plot Figure 6.2 are used to plot the FRFs in Figure 6.4. The FRF plots are practically indistinguishable from one another, demonstrating that the complex symmetric approach, using the standard value of τ_{CE}, leads to acceptable solutions of the modal FRP.

6.5 The FastFRS Solution Strategy

Equipped with the two efficient approaches for solving modal FRPs with structural damping, it is useful to know a priori which approach will perform faster for a particular modal FRP. For instance, if the modal structural damping matrix has a high rank, then the low rank approach has an even higher FLOP cost than the

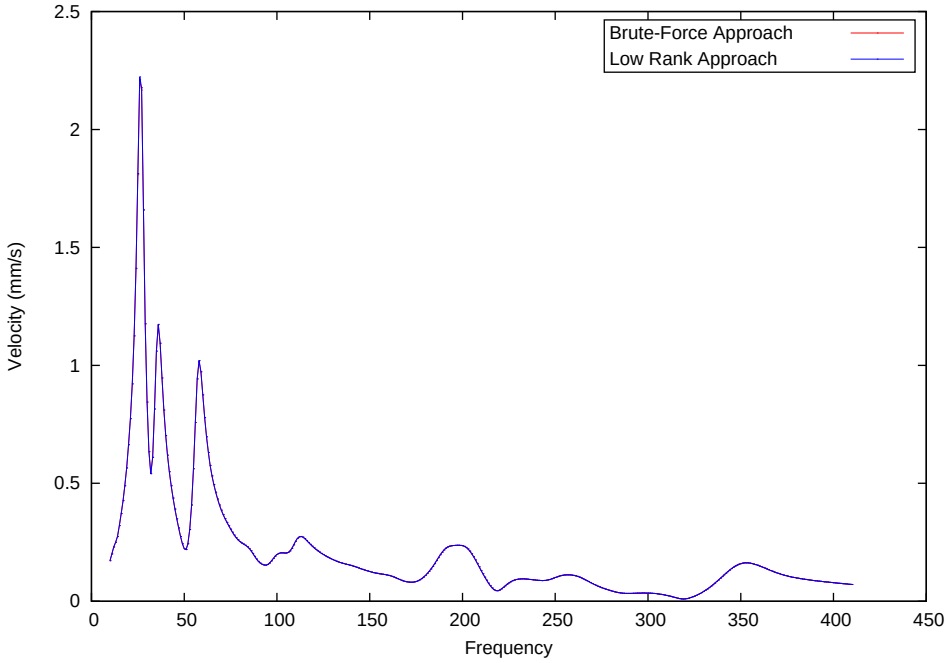


Figure 6.1: Comparing "Worst Case" FRFs for Model D

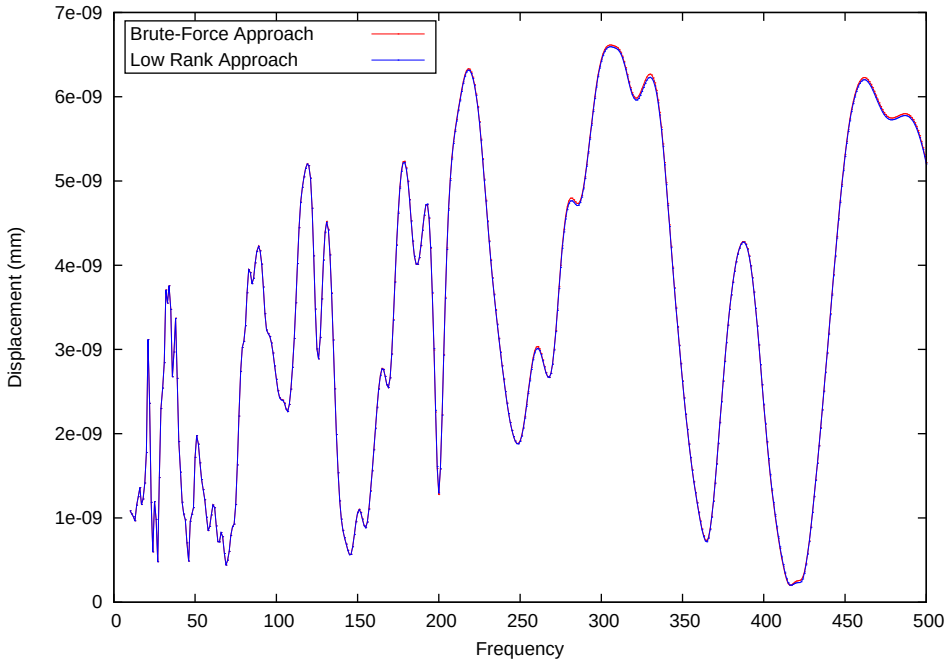


Figure 6.2: Comparing "Worst Case" FRFs for Model F

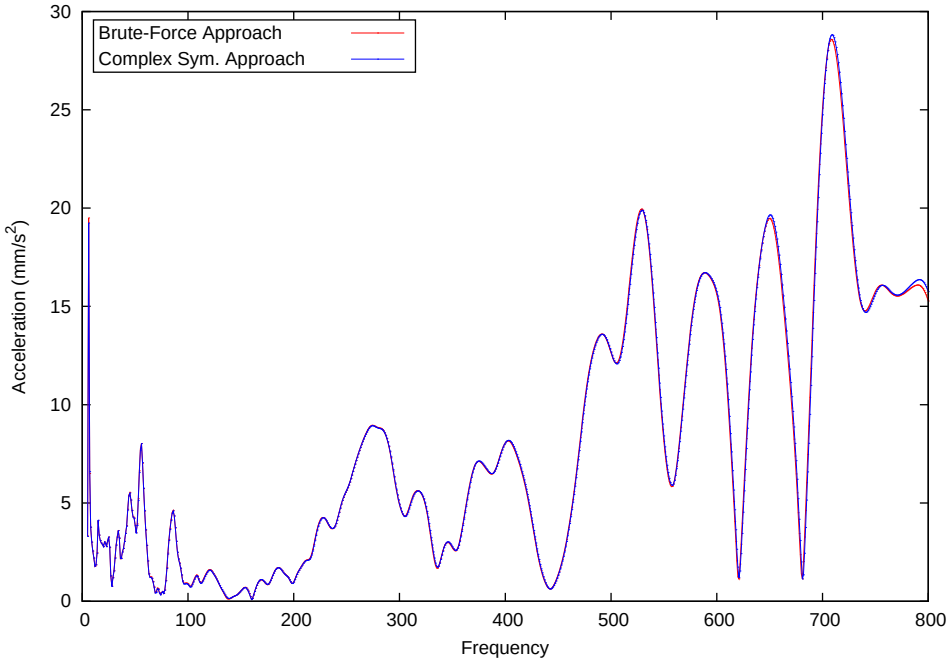


Figure 6.3: Comparing "Worst Case" FRFs for Model E

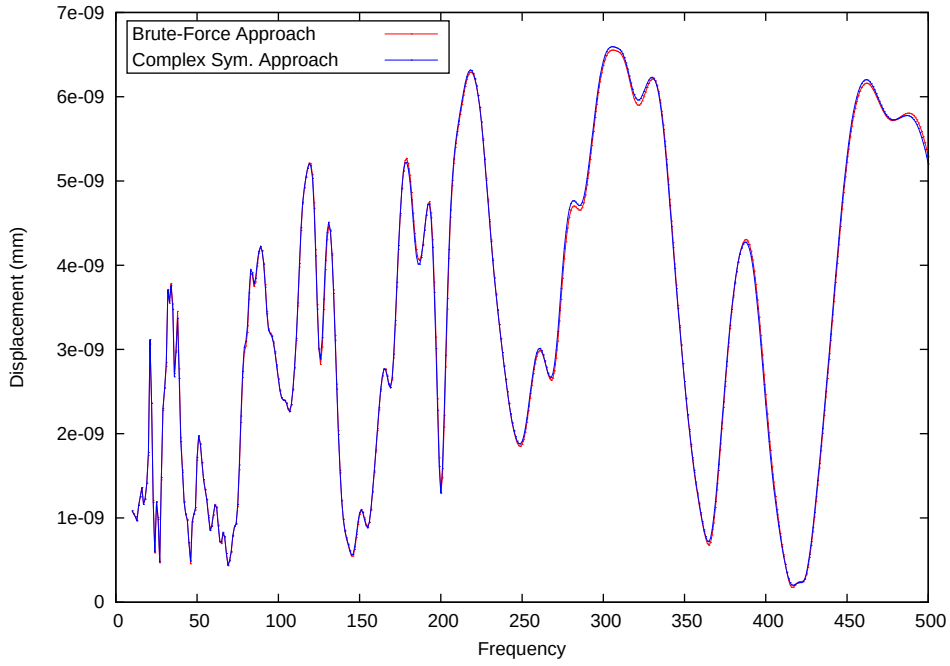


Figure 6.4: Comparing "Worst Case" FRFs for Model F

brute-force approach, and the complex symmetric approach should be chosen. If the rank of the modal structural damping matrix is very low, then using CSMES in the complex symmetric approach is unnecessary and is much more expensive than the low rank approach. This section describes a strategy that provides some guidance for determining which approach offers faster performance. The strategy relies on both machine-independent calculations, such as the rank of the modal structural damping matrix and the FLOP costs of each approach, and FLOP rate estimates for each significant part of FastFRS, which are machine-specific values. The strategy is then demonstrated on the models in the Industrial Test Suite.

1. The first step in this strategy is to determine the rank of the weighted K_s. This process is not costly when compared to the other operations in FastFRS. Also, when K_s is of low rank, the tridiagonal reduction process is stopped early and the rank is determined very quickly.

2. Second, the FLOP costs for the Set Up and the Frequency Sweep parts of FastFRS are computed for both approaches. Since the LRA of K_s has already been determined in the first step, the remaining Set Up cost for the low rank approach is zero. For the Set Up part of the complex symmetric approach, the only unknown value needed to compute the FLOP cost of CSMES is α_{CE}, which is defined in equation (4.19). Figure 4.3 suggests that even though the CE is a chance event, the total number of CEs encountered has some consistency. So, for an estimate, let $\alpha_{\text{CE}} = 0.20$, which is the upper bound which has been observed for many models, including those in the Industrial Test Suite. The cost

of CSMES, C_{CSMES}, is computed using equation (4.20). Next, the FLOP costs are computed for each of the remaining parts of FastFRS for the two new approaches. The cost of computing the low rank approach's frequency-dependent operations for a single frequency in the Frequency Sweep depends on whether the acoustic fluid matrices are diagonal. The cost is either C_{LR1} or C_{LR2}, found in equations (3.23) and (3.29), respectively. The cost of computing the complex symmetric approach's frequency-dependent operations for a single frequency in the Frequency Sweep is either C_{CE1} or C_{CE2}, found in equations (4.32) and (4.37), respectively. For convenience, the FLOP cost results presented in this chapter are reported in GFLOPs.

3. The FLOP costs in the previous step are computed without taking into account that some of them will be executed in parallel. They do not account for machine-specific parameters such as the rate at which FLOPs are executed simultaneously by CPU threads in parallel, and memory bandwidth. Therefore, the third step is determining estimates of the FLOP rates of each part of FastFRS for a particular machine. Let an estimated FLOP rate for CSMES be represented with S_{CSMES} and an estimated FLOP rate of the Frequency Sweep of both approaches be represented with S_{FS}. In order to have consistency with the FLOP costs from the second step, the FLOP rates are recorded in GFLOPs per second.

4. The final step in the FastFRS solution strategy is to estimate the total time in seconds that each approach will take to compute the responses. Since the

rank of the weighted K_s is determined in the first step in the strategy, the set up performance of the low rank approach is not included in the total time estimates. These estimates are simply:

$$\text{Low Rank Estimate } = \begin{cases} \frac{C_{\text{LR1}} * N_{freq}}{S_{\text{FS}}}, & \text{if } C_f \text{ is full} \\ \frac{C_{\text{LR2}} * N_{freq}}{S_{\text{FS}}}, & \text{otherwise} \end{cases} \tag{6.4}$$

$$\text{Complex Symmetric Estimate } = \frac{C_{\text{CSMES}}}{S_{\text{CSMES}}} + \begin{cases} \frac{C_{\text{CE1}} * N_{freq}}{S_{\text{FS}}}, & \text{if } C_f \text{ is full} \\ \frac{C_{\text{CE2}} * N_{freq}}{S_{\text{FS}}}, & \text{otherwise.} \end{cases} \tag{6.5}$$

The approach with the lower time estimate is the one which is favored, providing the solutions to the modal FRP more efficiently.

The estimated FLOP rate S_{CSMES} does not vary much across models for a particular machine since the frequency at which CEs are encountered and removed during the tridiagonalization process is somewhat consistent across models. Unfortunately, the values of S_{FS} are less consistent across models than S_{CSMES}, since there is much variation in the dimensions of the many matrices involved in the Frequency Sweep.

The models in the Industrial Test Suite are used to demonstrate the FastFRS solution strategy. First, every model's N_{LRA} is determined using the tolerance τ_{LRA} using the process described in Section 3.2. These values are listed in Table 6.2. Next, FLOP costs for each significant part of FastFRS are computed. Models B and C do not have purely diagonal acoustic fluid matrices, so their Frequency Sweep costs are computed using C_{LR1} and C_{CE1}. The acoustic fluid matrices of Models A, D, E, and F are diagonal, so their Frequency Sweep costs are computed using C_{LR2} and C_{CE2}.

Model	N_{ev}	N_{LRA}	C_{LR1} or C_{LR2}	C_{CSMES}	C_{CE1} or C_{CE2}
A	10351	2137	213.2	22331	29.40
B	11475	848	2480	30421	1972
C	8585	8594	2972	12784	425.7
D	6539	227	26.10	5629.2	150.5
E	15923	8438	4230	81281	493.0
F	12165	3350	443.8	36245	130.6

Table 6.2: Computed Costs Needed for the FastFRS Solution Strategy (GFLOPs)

Table 6.2 shows FLOP costs for each significant part of FastFRS for the models in the Industrial Test Suite.

In order to compute the estimated times for each approach, the FLOP rates specific to Machine A are needed. For simplicity, only the 0-GPU situation is considered for this exercise. The Frequency Sweep operations are primarily compute-bound and not limited by the memory bandwidth of the CPU cores. Therefore, the theoretical maximum compute performance rate for Machine A, listed in Table 5.1, which is 396.8 GFLOP/sec, is used as the value of the FLOP rate, S_{FS}. Since CSMES is limited by memory bandwidth, its FLOP rate, S_{CSMES}, is determined experimentally, and found to be 32.1 GFLOP/sec, on average. Using these values, the estimated times for each CPU-only approach are calculated and displayed in Table 6.3. The final column in the table shows which approach is favored based on estimated time. The FastFRS Solution strategy predicts that when only the CPU cores are used, Models A, B, D, and F should be solved using the low rank approach, and Models C and E should be solved using the complex symmetric approach.

Model	Low Rank Est. Time (sec)	Complex Sym. Est. Time (sec)	Favored Approach
A	113	711	Low Rank
B	2290	2770	Low Rank
C	5740	1220	Complex Sym.
D	25.3	322	Low Rank
E	8170	3480	Complex Sym.
F	636	1320	Low Rank

Table 6.3: Estimated CPU-only Performance Time for Each Approach

6.6 Performance of the Low Rank Approach

The low rank approach first determines the LRA of K_s, according to equation (3.10). The most costly step in this process is the tridiagonal reduction of the weighted modal structural damping in equation (3.11). DSYTRD is an LAPACK routine which provides tridiagonal reduction, but it assumes that the input matrix has full rank. If the matrix is low rank, then before the reduction is completed, the unreduced part of the matrix will be populated with zeroes, and the reduction can be halted early. Therefore, a specialized implementation is used in FastFRS which is capable of detecting whether the reduction process can end before the last column is annihilated. This implementation is capable of taking advantage of all of the CPU cores and any number of GPUs present on a computing machine.

Table 6.4 compares performance results of computing the LRA of the structural damping matrix using DSYTRD and the specialized implementation. The 'Using DSYTRD' column reports the total decomposition time when DSYTRD is used as the

Model	Using DSYTRD	Using "Stop Early" Implementation				
		0-GPUs	1-GPU	2-GPUs	3-GPUs	4-GPUs
A	59.3	33.3	16.2	16.7	12.2	10.7
B	75.9	21.0	12.2	11.8	9.06	7.99
C	45.8	47.5	36.7	38.7	33.2	31.3
D	12.3	2.41	2.50	2.83	2.23	1.94
E	232.5	224.8	97.7	93.6	70.7	61.1
F	99.5	67.1	30.2	29.9	21.9	19.1

Table 6.4: Performance of $K_s = J_s \hat{\Lambda}_L J_s^T$ Decomposition (seconds)

reduction routine. The remaining columns report the total decomposition time when the specialized implementation is used as the reduction routine. The performance of the specialized implementation improves as more GPUs are used in the reduction. The only model which does not benefit from the specialized implementation is Model C. With the value of τ_{LRA} that is used in industry, Model C's K_s is found to be full rank, as is confirmed in Figure 3.1. Therefore, the higher time in the 0-GPUs case for this model is due to the overhead of checking whether the process can be finished early.

The Frequency Sweep is the dominant part of FastFRS for the low rank approach. In the Frequency Sweep, each CPU thread independently computes responses for specific frequencies until the list of frequencies is exhausted. Each available GPU is assigned to a managing CPU thread. CPU threads with GPUs are able to compute responses much more quickly than the CPU threads without GPUs. For example, Model B is a model in the Industrial Test Suite which has a structural damping ma-

trix with low rank, according to Figure 3.1. Its modal FRP may be solved faster using the low rank approach than the complex symmetric approach. It has a full acoustic fluid "damping" matrix; therefore, the corresponding operations which compute the response at a single frequency with the low rank approach are displayed in Table 3.4. The performance of each step in the table depends on whether the work is performed by a solitary CPU thread or a CPU thread with a GPU. Table 6.5 breaks down the performance for each significant step listed in Table 3.4 for Machine A. Steps 2,3 and 11,12 are shown together since their operations are aggregated in order to improve BLAS and CUBLAS performance. The final column in Table 6.5 shows the "speedup", which is the CPU-only time divided by the CPU+GPU time. The speedup is an useful way to evaluate GPU performance. For instance, the speedup of step 9 is 42.4, which means that for that step in Table 3.4, the GPU has the performance of about 42 CPU cores. In step 14, two matrices are multiplied together and the matrix product width is N_c. Since BLAS and CUBLAS routines are most efficient when the matrices on which they operate are square, and for this model, N_c is small, step 14 has the lowest speedup.

Table 6.6 shows the speeds, in solutions per second, at which responses are computed by a single CPU thread for all models in the Industrial Test Suite. S_c represents the rate at which a CPU thread without a GPU computes responses, and S_g represents the rate for a CPU thread with a GPU. The speedup results in the last column shows the benefit of including GPUs in the Frequency Sweep. The structural damping matrix from Model C has full rank and most of the matrices involved in the Frequency Sweep are square. Since CUBLAS matrix multiplications are most efficient

Step in Table 3.4	CPU-only (sec)	CPU+GPU (sec)	Speedup
2, 3	17.3	0.330	52.5
5	1.08	0.0449	24.1
7	3.66	0.0938	39.0
9	13.9	0.327	42.4
11,12	82.4	2.86	28.7
13	14.9	1.90	7.84
14	0.439	0.414	1.06

Table 6.5: Model B Performance of Individual Steps in Frequency Sweep

Model	S_c	S_g	Speedup (S_g/S_c)
A	0.0768	1.21	15.8
B	0.00759	0.140	18.4
C	0.00530	0.138	26.0
D	0.345	3.57	10.3
E	0.00387	0.0803	20.7
F	0.0381	0.625	16.4

Table 6.6: Speed of CPU cores and GPU (solutions/sec)

when the matrices are square, Model C has the highest speedup in the table.

Once a response at a single frequency has been computed by one CPU thread with and without a GPU, it is possible to accurately predict the overall speed with which responses are computed by all of the CPU cores and GPUs. If a machine has c CPU cores and g GPUs, the speed of the machine, in terms of frequency response solutions per unit time, can be represented simply as a function of g as

$$S(g) \approx gS_g + (c-g)S_c = cS_c + g(S_g - S_c). \tag{6.6}$$

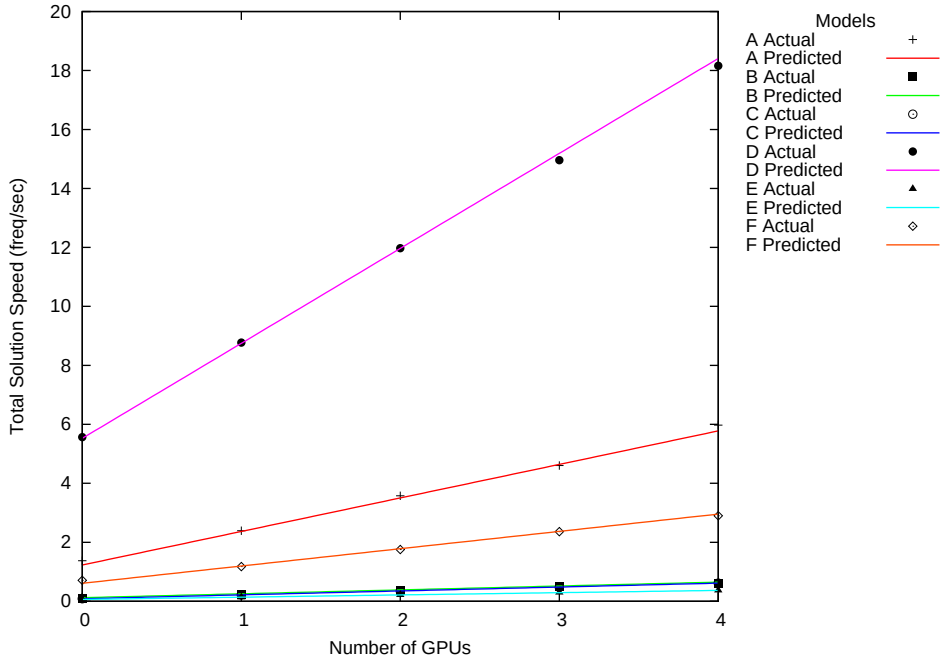


Figure 6.5: Predicted and Actual Speedup

In the latter representation, the speed begins with cS_c, the speed of the machine without GPUs, and increases linearly with the number of GPUs. In FastFRS, good load balancing across all CPU threads is maintained in the Frequency Sweep when the frequency scheduler is equipped with the values of S_c and S_g to properly assign frequencies to the CPU threads. Figure 6.5 plots equation (6.6) for each model, based on the values in Table 6.6 on Machine A. The points near or on each line represent the total response rates based on execution time. The agreement between the theoretical and measured speeds indicates that the performance is close to ideal.

The semi-log plot in Figure 6.6 shows the performance of the low rank approach compared to the CPU-only brute-force approach for all models in the Industrial Test

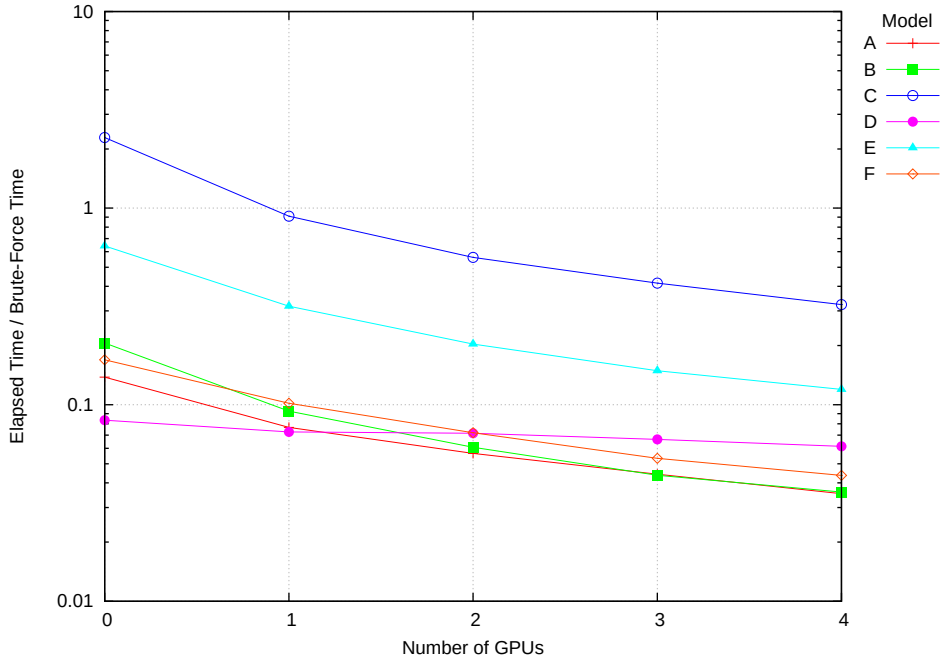


Figure 6.6: Performance Improvement of the Low Rank Approach over the Brute-Force Approach

Suite. In the figure, the y-axis is the overall elapsed time using the low rank approach divided by the time using the CPU-only brute-force approach, and the x-axis is the number of GPUs used in the low rank approach. For all models, except C and E, the improvement in performance is dramatic. When two GPUs are used, the modal FRPs from Models A, B, D, and F are solved more than ten times faster with the low rank approach than with the CPU-only brute-force approach. Model C's modal FRP is not solved faster using the low rank approach since its structural damping matrix has full rank. For this model, the low rank approach requires more operations per frequency than the brute-force approach.

6.7 Performance of the Complex Symmetric Approach

In the complex symmetric approach, the matrix C is formed from the stiffness matrix, the global structural damping factor, and K_s, as seen in equation (2.7). The eigenvalues and eigenvectors of C are computed using CSMES, described in Section 4.6, and the modal FRP is transformed into an equation which is solved efficiently using the SMW formula.

6.7.1 CSMES Performance

The two primary routines in CSMES are ZSYTRD, which performs the tridiagonal reduction of C to T, and ZORMTR, which backtransforms the eigenvectors of T to the eigenvectors of C. A secondary routine in CSMES computes the eigenvectors and eigenvalues of T and is much more efficient than ZSYTRD and ZORMTR because its FLOP cost is $O(N_{ev}^2)$. The performance of ZSYTRD is limited by the machine's memory bandwidth since a matrix-vector multiplication is required $N_{ev}-2$ times in the tridiagonal reduction process, and removing CEs requires many more costly BLAS level-2 operations. Every CE that is removed in ZSYTRD has a corresponding set of updates which must be applied in ZORMTR; therefore, as more CEs are removed, the performance of ZSYTRD and ZORMTR degrade. When τ_{CE} is set to the standard level used in industry, the number of CEs removed in the tridiagonal reduction process is usually between ten and twenty percent of N_{ev} regardless of the size of C.

Model A's structural damping matrix appears to have a high rank, as seen in Figure 3.1. When its matrix C is created and tridiagonalized in ZSYTRD using the

Operation	0-GPUs	1-GPU	2-GPUs	3-GPUs	4-GPUs
ZSYTRD	350.4	228.1	167.7	142.4	127.4
ZORMTR	302.5	197.9	124.5	113.3	94.5

Table 6.7: ZSYTRD and ZORMTR Performance for Model A (sec)

standard CE tolerance, it is found that CEs must be removed for 16% of its columns. Table 6.7 shows the time spent in the two primary routines used in CSMES for sample Model A. When two or more GPUs are used in ZSYTRD, there is an additional cost of transferring matrix data between the GPUs and the CPU cores every time a CE is removed. If the data transfer speed between the GPUs and the CPU is improved, then the multi-GPU ZSYTRD performance will improve. Although CPUs and GPUs do not need to communicate with one another during the ZORMTR process, the ZORMTR performance is hindered by the matrix updates which correspond to CEs which were removed in ZSYTRD.

6.7.2 Comparing CSMES to ZGEEV

An alternative to CSMES in LAPACK is a routine called ZGEEV which computes the eigenvalues and eigenvectors of complex *general* matrices. This routine assumes that the input matrix is square, but nonsymmetric. Instead of a reduction to tridiagonal form, ZGEEV reduces the matrix to upper (or lower) Hessenberg form, in which all of the lower triangle except the subdiagonal (or all of the upper triangle except the superdiagonal) is annihilated. In the Hessenberg reduction, the reflections are complex Householder reflections, where $H = I - \tau \mathbf{v}\mathbf{v}^H$. The vector $\mathbf{v}$ and the subdiagonal entries in the resulting Hessenberg matrix are determined from

$\mathbf{x}^H\mathbf{x}$ instead of $\mathbf{x}^T\mathbf{x}$. Therefore, the cancellation event is not present in this type of reduction. However, the symmetry of C is not exploited at all in this process, and all of the unreduced portion of C must be referenced and modified when each column is annihilated. The process of computing the eigenvalues and eigenvectors of the Hessenberg matrix is more costly than computing the eigenvalues and eigenvectors of the tridiagonal matrix in CSMES since a Hessenberg matrix has many more nonzeros than a tridiagonal matrix. The final step of ZGEEV backtransforms the eigenvectors of the Hessenberg matrix to the eigenvectors of C. ZGEEV is the standard approach for computing the eigenvalues and eigenvectors of complex symmetric matrices.

CSMES takes advantage of the symmetry of C, but CEs must be removed during the tridiagonal reduction process. By contrast, ZGEEV ignores the symmetry, but encounters no CEs. The FLOP cost of the CSMES approach depends on α_{CE}, the fraction of columns of C in which CEs are encountered that must be removed. By leaving α_{CE} as a variable and equating the FLOP cost of CSMES with the FLOP cost of ZGEEV, the value of α_{CE} for which the two costs are equal can be determined. At this particular value of α_{CE}, the benefit of exploiting symmetry is exactly balanced by the added cost of removing CEs. The cost of CSMES, C_{CSMES} which is derived in Section 4.6 is

$$C_{\text{CSMES}} = \left[\frac{40}{3} + \left(\frac{5\delta}{6} + 14\right)\alpha_{\text{CE}}\right] N_{ev}^3. \tag{6.7}$$

The value of δ in the equation above is a function of the type of CO matrices that are used to annihilate fill-in columns during the CE removal process. In CSMES, each fill-in column is annihilated with CO rotations, which corresponds to δ equal to 24. According to [7], the FLOP cost of performing ZGEEV is $26.33N_{ev}^3$. Then, equating

the two FLOP costs and cancelling N_{ev} from both sides,

$$\frac{40}{3} + \left(\frac{5 \times 28}{6} + 14 \right) \alpha_{\text{CE}} = 26.33. \tag{6.8}$$

This equation can be solved to obtain the value $\alpha_{\text{CE}} = 0.35$, which indicates that when more than 35% of the columns of C have CEs removed, ZGEEV has a lower FLOP cost than CSMES. As seen in Figure 4.3, an α_{CE} value that high corresponds to a very low CE tolerance for models in the Industrial Test Suite. The CE tolerance used in industry yields α_{CE} values that are much lower than 0.35.

Figure 6.7 shows the performance results for computing eigenvalues and eigenvectors of C for the models in the Industrial Test Suite. The standard value of $\tau_{\text{CE}} = 3.5$ was used. In the figure, CSMES is compared with the only alternative, the CPU-only ZGEEV routine, beginning with the CPU-only case (with no GPUs) and increasing the number of GPUs. Without GPUs, the time taken by CSMES is between 26% and 36% of the ZGEEV time. As GPUs are added, which ZGEEV does not presently allow, performance continues to improve significantly. When 4 GPUs are used, the time spent in CSMES is between 10% and 14% of the ZGEEV time. For all models, since α_{CE} varies between 0.10 and 0.20, CSMES requires fewer FLOPs than ZGEEV, so the superior performance of CSMES shown in the figure is not surprising.

6.7.3 Frequency Sweep

After CSMES is finished in the Set Up part of FastFRS, some matrix multiplications must be performed at every frequency in the Frequency Sweep. Table 6.8

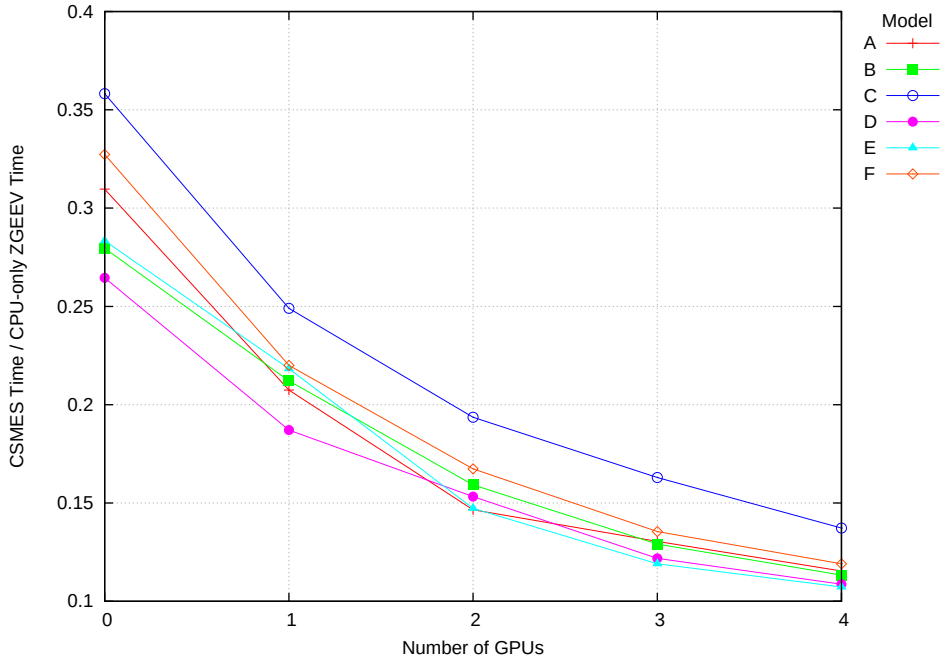


Figure 6.7: Performance Improvement of CSMES over ZGEEV

shows that these operations take very little time compared to the CSMES time for Model A.

6.7.4 Total Performance

The performance of the complex symmetric approach compared to the CPU-only brute-force approach is shown in the semi-log plot in Figure 6.8. The axis ranges are the same as the axis ranges in the plot in Figure 6.6. For all cases, the complex symmetric approach is faster than the brute-force approach. For example, the CPU-only complex symmetric approach completed its work on Model B 3.84 times faster than the brute-force approach, and the 4-GPU implementation was 16.45 times faster

Operation	0-GPUs	1-GPU	2-GPUs	3-GPUs	4-GPUs
Set Up (w/CSMES)	686.1	460.6	326.4	290.8	257.2
Freq. Sweep	35.5	25.9	21.0	19.0	17.6

Table 6.8: Model A Performance Breakdown (sec)

Model	Brute-Force Total Time (s)	Low Rank Total Time (s)	Complex Sym. Total Time (s)	Faster Approach
A	1406	194.2	724.9	Low Rank
B	17620	3630	4580	Low Rank
C	4147	9478	931.2	Complex Sym.
D	968.1	80.70	302.0	Low Rank
E	18620	11950	3584	Complex Sym.
F	5555	939.8	1321	Low Rank

Table 6.9: Total CPU-only Performance Time for Each Approach

than the brute-force approach.

6.8 The FastFRS Solution Strategy, Revisited

The FastFRS Solution Strategy attempts to predict which new approach will solve the modal FRP faster. Table 6.3 lists the results of using the strategy on the models in the Industrial Test Suite, and predictions are made in the last column. Table 6.9 lists the actual performance time for each model using the two new approaches, and confirms that the predictions in the previous table were correct.

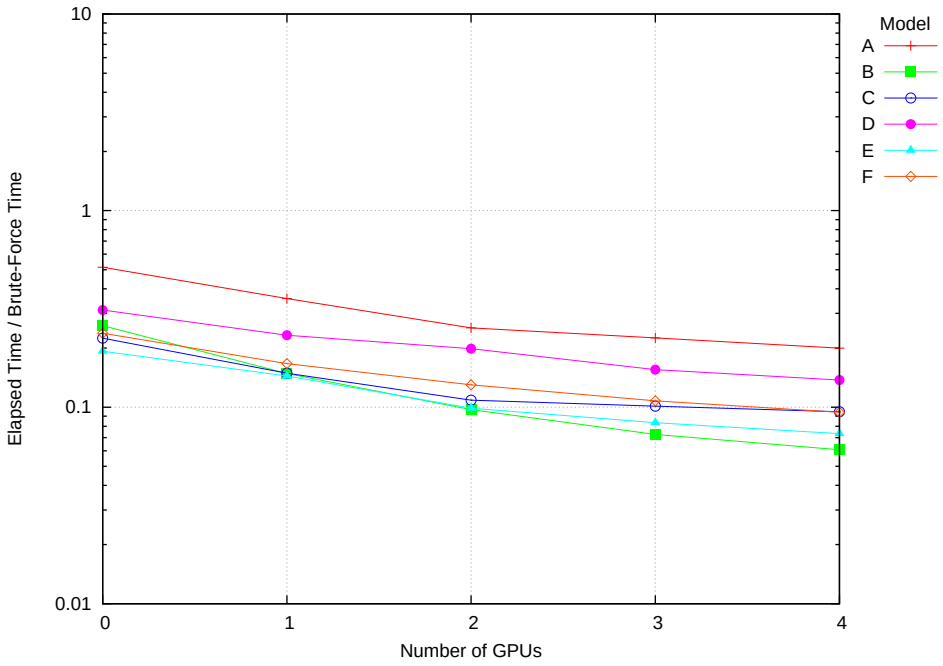


Figure 6.8: Performance Improvement of the Complex Symmetric Approach over Brute-Force Approach

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The classical modal approach is very effective for vibration analysis of extremely high dimension finite element (FE) discretizations of complicated structures, but accurate modeling of damping can reduce the benefits of this approach. For automobiles, viscous damping is typically associated with only a few discrete components, such as engine mounts and shock absorbers, and results in a low rank damping matrix that can be handled easily. However, structural damping permeates the automobile's structure, and different structural materials typically have different levels of structural damping.

An efficient approach is to establish a global structural damping level, γ, for the predominant structural material, because this results in a structural damping matrix equal to the stiffness matrix multiplied by a scalar. The modes are orthogonal with respect to this matrix as they are with respect to the stiffness matrix, so the contribution of this matrix to the modal frequency response problem (FRP) is diagonal. Then other materials are modeled by representing their deviation from the global structural damping level by another matrix $\mathcal{K}_s$, but $\mathcal{K}_s$ contributes a fully populated matrix to the modal FRP.

This dissertation presents two new approaches for solving modal FRPs whose coefficient matrix is full because of structural damping. The approaches modify the modal FRP by putting its full coefficient matrix in a form that is more amenable to efficient solution. These approaches rely on some preparatory computation that requires $O(N_{ev}^3)$ operations, followed by frequency response calculations that requires only $O(N_{ev}^2)$ operations at each frequency. The approaches are implemented in the commercial software package, FastFRS, which is currently licensed by automobile companies around the world.

7.1.1 Chapter 3: The Low Rank Approach

The rank of $\mathcal{K}_s$ is determined by the number of finite elements composed of different materials from the predominant material whose structural damping is reflected by γ. If the number of finite elements with structural damping different from γ is very small, then the $\mathcal{K}_s$ matrix is of low rank. The modal structural damping matrix K_s is full, but it is also of low rank, and the low rank approach outlined in Chapter 3 is used to solve the modal FRP. This approach relies on an accurate low rank approximation (LRA) of K_s. However, the entries in K_s vary greatly in magnitude, making it difficult to determine the rank of K_s reliably. The variation in magnitudes of values in K_s is related to the variation among values in the modal stiffness matrix Λ. The pseudoinverse of the square root of the diagonal matrix Λ is an appropriate choice of a weighting matrix for scaling rows and columns of K_s. When K_s is rescaled in this manner, determining its rank for the modal FRP becomes straightforward and yields results that are consistent across a wide variety of

automobile models. A tolerance is used to determine the rank of the weighted matrix, and K_s is represented with an LRA. The LRA of K_s makes it possible to solve the modal FRP efficiently using the Sherman-Morrison-Woodbury (SMW) formula.

7.1.2 Chapter 4: The Complex Symmetric Approach

If there are a large number of finite elements composed of different materials from the predominant material, then $\mathcal{K}_s$ is of high rank. The modal K_s matrix also has a high rank, and the complex symmetric approach, described in Chapter 4, is used to solve the modal FRP. In this approach, a complex symmetric matrix, C, is created, which is defined in equation (2.7), and its eigenvalue decomposition (EVD) is determined using the complex symmetric matrix eigensolver (CSMES), which is presented in Chapter 4. When there is no viscous damping represented in the model, the eigenvectors of C diagonalize the coefficient matrix and the modal FRP is trivial to solve. When viscous damping is represented in the model, the coefficient matrix is transformed into a diagonal plus low rank matrix, which is solved efficiently using the SMW formula.

In the first step in CSMES, a non-unitary complex orthogonal (CO) matrix is used to reduce C to symmetric tridiagonal form. The CO matrix is the product of CO reflections, which sequentially annihilate many entries in a row and the corresponding column of C. A consequence of using CO matrices for tridiagonal reduction is that when a reflection is formed and applied, there can be cancellation between terms in the product $\mathbf{x}^T\mathbf{x}$ that reduces numerical precision in the EVD of C. Chapter 4 shows how this cancellation event (CE) is detected by comparing the result of equation (4.6)

with a specified tolerance. Next, a CE removal method is described which typically removes the CE on its first attempt, but allows for additional attempts if they are needed. After C is reduced to the tridiagonal matrix T, the EVD of T is computed. Because T was obtained using similarity transformations, the eigenvalues of T are the same as those of C, but the eigenvectors of T must be backtransformed to obtain those of C. These eigenvectors are then used to transform the full coefficient matrix of the modal FRP into a form that is more amenable to efficient solution.

7.1.3 Chapter 5: Parallel Implementations of the New Approaches

Currently, automobile companies conduct NVH analyses on single node computers with multiple cores per processor. It is becoming common for some companies to install multiple GPUs on their computers because GPUs are capable of performing computations much more efficiently than multi-core processors. Chapter 5 discusses the various ways that the low rank and complex symmetric approaches are implemented to take advantage of these timely parallel computing technologies.

In the low rank approach, after K_s is weighted with the pseudo-inverse of the square root of Λ, the weighted matrix is tridiagonalized using Householder reflections. If K_s is of low rank, then in the course of the tridiagonal reduction, the portion of K_s that has not yet been tridiagonalized can become null. If this happens, many FLOPs can be saved by stopping the reduction process early, if the magnitudes of unreduced entries are examined. This feature is included in the multi-core, single GPU, and multi-GPU implementations of this process.

For the multi-GPU implementation, the columns of K_s are distributed nearly

equally among the GPUs. To ensure good load-balancing during tridiagonalization, the columns are distributed in a 1-D cyclic fashion, and in order to take advantage of GPU cores most efficiently, local matrix data is stored on each GPU in block-packed form. Data which is stored on one GPU but is needed by another is communicated across the peripheral component interconnect (PCI) bus by the CPU.

When matrix C is of full rank, the complex symmetric approach is used, and the unreduced part of C is never null during the tridiagonalization process. As each CO reflection is formed for annihilating a row and column of C, it must be checked to see if a CE has occurred. If a CE is detected, all of the CE removal steps that must be applied to the portion of the matrix that has been made tridiagonal by then are most efficiently performed by the CPU cores. The last CE removal step is applying the CO rotations, which were constructed for annihilating the entries in the fill-in column, to the as-yet unreduced part of the matrix. The performance of this step is limited by memory bandwidth.

The parallel implementations of the tridiagonal reduction process of the complex symmetric approach resemble the implementations of the low rank approach, except for the steps pertaining to the CE removal method. When an attempt is made to remove a CE, the CPU cores conduct all of the steps up to the application of CO rotations to the unreduced part of the matrix which were used to annihilated fill-in columns. For the CPU-only and single GPU implementations, the application of CO rotations is carried out without any synchronizations or coordinations between computing resources. For the multi-GPU implementation, this step is complicated, since the unreduced part of the matrix is distributed among all of the GPUs. Applications

of CO rotations are conducted on successive panels of the matrix, and simultaneously, the next panel is gathered from the GPUs and the previous panel is distributed back to the GPUs.

Both the low rank approach and the complex symmetric approach require a backtransformation process which transforms an eigenvector matrix for the tridiagonal matrix to one for the matrix that was tridiagonalized. In this process, panels of the eigenvector matrix are assigned to each of the GPUs and to the CPU cores. The panels are backtransformed independently. For the complex symmetric approach, the CO matrices used to remove the CEs in the tridiagonal reduction process must be applied between the appropriate algorithmic blocks in order to perform the backtransformation correctly.

In the Frequency Sweep part of FastFRS, the solution of the modal FRP is computed at every frequency. For this part, each available GPU is assigned to a managing CPU thread, and a scheduler assigns frequencies both to managing CPU threads and to CPU threads without GPUs during the Frequency Sweep until the list is exhausted. The scheduler ensures that the CPU threads with GPUs are given precedence in an attempt to use all of the available resources most efficiently.

7.1.4 Chapter 6: Results

Machine A has similar specifications to the computers that automobile companies use for NVH analyses. The parallel implementations described in Chapter 5 were performed on Machine A and the results are presented in Chapter 6. The accuracy and performance of the two approaches are explored by comparing responses from

the new approaches with responses from the brute-force approach for models in the Industrial Test Suite.

The accuracy of each new approach is demonstrated through the use of an error measure and frequency response function (FRF) graphs. First, for one of the new approaches, the error measure in equation (6.3) is used to identify the grid point and load case combination whose FRF differs the most from a corresponding FRF created from the brute-force approach. Next, the FRFs from the new approach and from the brute-force approach for this "worst case" are plotted on the same graph. The FRF plots from the new approaches and the brute-force approach are found to be virtually indistinguishable for the sample models in the Industrial Test Suite. These graphs demonstrate that the tolerances chosen for each approach lead to FRFs which are virtually identical to FRFs created using the brute-force approach.

Chapter 6 presents the performance of the multi-core (no GPU), single-GPU, and multi-GPU implementations of the low rank and complex symmetric approaches on Machine A. For both approaches, the performance is always improved as more GPUs are included in the Set Up and Frequency Sweep parts of FastFRS. For the low rank approach, the time required for the Set Up part of FastFRS is usually negligible compared to the time required for the Frequency Sweep part. In contrast, for the complex symmetric approach, most of the time is spent in CSMES, which is in the Set Up part of FastFRS.

The low rank approach is much more efficient than the brute-force approach for all models in the Industrial Test Suite, except for Model C, whose K_s matrix is of full rank, so the low rank approach is not appropriate. For Model A, the CPU-only

implementation of the low rank approach is 7.2 times faster than the implementation of the CPU-only brute-force approach. When expensive computations are performed by the GPUs instead of the managing CPU threads, the improvement in performance is dramatic. When 4 GPUs are used, the implementation of the low rank approach on Model A becomes 28.3 times faster than the CPU-only brute-force approach.

For all of the models in the Industrial Test Suite, the complex symmetric approach is more efficient than the brute-force approach. For example, Model E computes the response 5.2 times faster using the CPU-only complex symmetric approach than using the CPU-only brute-force approach. When 4 GPUs are used, the performance is 13.6 times faster. The improvement in performance is not as dramatic for the complex symmetric approach as the low rank approach because CSMES performance is limited by memory bandwidth.

Chapter 6 presents a solution strategy for determining in advance which new approach will compute the solution of a modal FRP more efficiently. This strategy uses the FLOP cost for the significant operations for each approach, with some machine-specific parameters, to estimate the total time each approach may take. The solution strategy correctly predicts which of the two new approaches performs faster for each model in the Industrial Test Suite.

7.2 Future Work

The low rank and complex symmetric approaches provide solutions to modal FRPs more efficiently than the traditional brute-force approach. The following list proposes some ways to extend the capabilities of the two new approaches.

1. Vargas [41] explored the possibility of obtaining a reformulation of the sum of the complex modal stiffness and structural damping matrices as a diagonal matrix plus a matrix of minimal rank. An algorithm for computing a "diagonal plus low rank" (DPLR) representation was developed, along with an iterative algorithm for using an inexact DPLR approximation in the solution. The DPLR algorithm is impractical to use in FastFRS for industrial models due to its slow convergence, but its performance may be improved by using a weighting matrix to transform the structural damping matrix as described in Chapter 3.

2. In this dissertation, it is assumed that the materials have frequency independent damping. If the structural damping is frequency-dependent and can be represented as the sum of a frequency dependent diagonal matrix and a matrix of low rank, then the techniques in Chapter 3 could be extended to accommodate this situation. Also, it is assumed that the acoustic fluid "damping" matrix is of full rank. If it is a low rank matrix, then the acoustic fluid part of the modal FRP could be represented as a diagonal plus low rank matrix and its inversion could be computed quickly using the SMW formula.

3. The CE removal method presented in Chapter 4 is used in FastFRS. However, another method could exist which may be more efficient when implemented on another parallel technology. Any method which has the characteristics described in Section 4.2 is an acceptable CE removal method and should be investigated.

4. The technological trends may change to prompt automobile companies to rely on multi-node computers using Message Passing Interface or other coprocessors

such as Intel's Phi to conduct NVH analyses. It will be important to reevaluate the implementations of the low rank and complex symmetric approaches in order to support these different technologies.

Bibliography

[1] EM Photonics: CULA Dense. Available at `http://www.culatools.com/dense`, Accessed: 2015-03-31.

[2] Intel Corporation. Xeon Processor E5-2687W Product Specifications (2014).

[3] NVIDIA Corporation. Nvidia cuBLAS Library User Guide, Version 6.5. Available at `http://docs.nvidia.com/cuda/cublas/index.html`, Accessed: 2015-03-31.

[4] NVIDIA Corporation. Nvidia CUDA C Programming Guide, Version 6.5. Available at `http://docs.nvidia.com/cuda/cuda-c-programming-guide`, Accessed: 2015-03-31.

[5] NVIDIA Corporation. Tesla K20c GPU Product Specifications (2014).

[6] Emmanuel Agullo, Jim Demmel, Jack Dongarra, Bilel Hadri, Jakub Kurzak, Julien Langou, Hatem Ltaief, Piotr Luszczek, and Stanimire Tomov. Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects. *Journal of Physics: Conference Series*, 180(1), August 2009.

[7] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[8] Ilan Bar-On and Marcin Paprzycki. High performance solution of the complex symmetric eigenproblem. *Numerical Algorithms*, 18(2):195–208, 1998.

[9] Ilan Bar-On and Victor Ryaboy. Fast diagonalization of large and dense complex symmetric matrices, with applications to quantum reaction dynamics. *SIAM J. Sci. Comput.*, 18(5):1412–1435, 1997.

[10] Christian Bischof and Charles van Loan. The wy representation for products of householder matrices. *SIAM J. Sci. Stat. Comput.*, 8(1):2–13, January 1987.

[11] K. Blakely. *MSC/Nastran User's Guide: Basic Dynamic Analysis, Version 68*. The MacNeal-Schwendler Corporation, 1993.

[12] J. R. Bunch and L. Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Math. Comp.*, pages 63–179, January 1977. ctr127.

[13] Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Julien Langou, and Stanimire Tomov. The impact of multicore on math software. In *In PARA 2006, Umea Sweden*, 2006.

[14] B. D. Craven. Complex symmetric matrices. *Journal of the Australian Mathematical Society*, 10(3-4):341–354, 1969.

[15] Jane K. Cullum and Ralph A. Willoughby. A ql procedure for computing the eigenvalues of complex symmetric tridiagonal matrices. *SIAM J. Matrix Anal. Appl.*, 17(1):83–109, 1996.

[16] James W. Demmel and Michael T. Heath Y. Applied numerical linear algebra. In *Society for Industrial and Applied Mathematics*. SIAM, 1997.

[17] Inderjit Singh Dhillon. *A New O (N(2)) Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*. PhD thesis, Berkeley, CA, USA, 1998. UMI Order No. GAX98-03176.

[18] J. J. Dongarra, Jeremy Du Croz, Sven Hammarling, and I. S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Softw.*, 16(1):1–17, March 1990.

[19] G.C. Everstine. A symmetric potential formulation for fluid-structure interaction. *Journal of Sound and Vibration*, 79(1):157 – 160, 1981.

[20] D.J. Ewins. *Modal testing: theory, practice, and application*. Mechanical engineering research studies: Engineering dynamics series. Research Studies Press, 2000.

[21] W. N. Gansterer, A. R. Gruber, and C. Pacher. Non-splitting tridiagonalization of complex symmetric matrices. In *ICCS '09: Proceedings of the 9th International Conference on Computational Science*, pages 481–490, Berlin, Heidelberg, 2009. Springer-Verlag.

[22] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23(1):5–48, March 1991.

[23] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[24] G.H. Golub, Alan Hoffman, and G.W. Stewart. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its Applications*, 8889(0):317 – 327, 1987.

[25] Bruce Hendrickson, Elizabeth Jessup, and Christopher Smith. Toward an efficient parallel eigensolver for dense symmetric matrices. *SIAM J. Sci. Comput.*, 20(3):1132–1154, January 1999.

[26] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002.

[27] Chang-wan Kim. *Frequency Response Computation for Complex Structures with Damping and Acoustic Fluid*. PhD thesis, University of Texas at Austin, Austin, TX, USA, 2004. UMI Order No. GAX98-03176.

[28] A. L. Kimball and D. E. Lovell. Internal Friction in Solids. *Physical Review*, 30:948–959, December 1927.

[29] C.D. La Budde. The reduction of an arbitrary real square matrix to tri-diagonal form using similarity transformations. *Math. Comput.*, 17:433–437, 1963.

[30] Cornelius Lanczos. An iterative method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards, Sect. B.*, 45:225–280, 1950.

[31] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for fortran usage. *ACM Trans. Math. Softw.*, 5(3):308–323, September 1979.

[32] R. B. Lehoucq. The computation of elementary unitary matrices. *ACM Trans. Math. Softw.*, 22(4):393–400, December 1996.

[33] Franklin T. Luk and Sanzheng Qiao. Using complex-orthogonal transformations to diagonalize a complex symmetric matrix. In *in Advanced Signal Processing: Algorithms, Architectures, and Implementations VII*, pages 418–425, 1997.

[34] Ivan Markovsky. *Low Rank Approximation*. Springer London, 2011.

[35] Ivan Markovsky and Sabine Van Huffel. Left vs right representations for solving weighted low-rank approximation problems. *Linear Algebra and its Applications*, 422(23):540 – 552, 2007.

[36] L. Meirovitch. *Principles and techniques of vibrations*. Prentice Hall, 1997.

[37] Z. Osinski. *Damping of Vibrations*. Taylor & Francis, 1998.

[38] Chiara Puglisi. Modification of the householder method based on the compact wy representation. *SIAM J. Sci. Stat. Comput.*, 13(3):723–726, May 1992.

[39] Robert Schreiber and Charles van Loan. A storage-efficient wy representation for products of householder transformations. *SIAM J. Sci. Stat. Comput.*, 10(1):53–57, January 1989.

[40] Herb Sutter. The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software. *Dr. Dobb's Journal*, 30(3), 2005.

[41] David Vargas. Diagonal plus low rank approximation of matrices for solving modal frequency response problems. Master's thesis, University of Texas at Austin, Austin, TX, USA, 2010.

[42] D. S. Watkins and L. Elsner. Chasing algorithms for the eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 12(2):374–384, 1991.

[43] J. H. Wilkinson, editor. *The Algebraic Eigenvalue Problem*. Oxford University Press, Inc., New York, NY, USA, 1988.

[44] Field G. Van Zee, Ernie Chan, Robert A. van de Geijn, Enrique S. Quintana-Orti, and Gregorio Quintana-Orti. The libflame library for dense matrix computations. *Computing in Science and Engineering*, 11(6):56–63, 2009.

[45] Field G. Van Zee, Robert A. van de Geijn, Gregorio Quintana-Ortí, and G. Joseph Elizondo. Families of algorithms for reducing a matrix to condensed form. *ACM Trans. Math. Softw.*, 39(1):2, 2012.

[46] Hong Yuan Zha. A two-way chasing scheme for reducing a symmetric arrowhead matrix to tridiagonal form. *Journal of Numerical Linear Algebra with Applications*, 1(1):49–57, 1992.

Vita

Jeremiah Fletcher Palmer earned Bachelor of Science degrees in Physics and Mathematics from Harding University in Searcy, AR, in May 2000. He then taught High School Physics, Algebra II, and Calculus at Brentwood Christian School in Austin, TX, for two years. In Fall 2002, he enrolled in graduate school at The University of Texas at Austin and earned his Masters of Science in Computational and Applied Mathematics in May 2004. In June 2004, he began his work in the Engineering Mechanics doctoral program under Dr. Bennighof. He and Dr. Bennighof invented the FastFRS software which is currently licensed by automobile companies throughout the world. The latest version, FastFRS 2.0, was released in Fall 2013 and an implementation which takes advantage of multiple GPUs will be released in Spring 2015.

Email address: jeremiahpalmer@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.