Physical Audio Modeling

Modeling & Simulation Final Report

Karl Hiner, Ben Wilfong

April 2023

Github Repo: https://github.gatech.edu/bwilfong3/CSE6730_Team3
Video: https://mediaspace.gatech.edu/media/Mesh2Audio+-+Final+project+for+Modeling+%26+Simulation/1_mh9qwexo/290982512
Audio samples:
https://drive.google.com/drive/folders/1wTdc4w5yPa5-ZNjtl8z-cwkA1gmOq1Jv

Abstract: This project explores the problem of modeling synthesized sounds from cross-sections of axisymmetric objects. In order to do this, we have implemented a finite element code to create stiffness and mass matrices that describe the object of interest, and then use the eigenvalues and eigenvectors of the resulting generalized eigen-problem to produce a superposition of sine waves that mimic the sound that object would make when struck. This work improves on previous work done by Michon et al. [9] by using a 2-D axisymmetric modeling approach to provide real-time feedback through an intuitive graphical user interface. The following report details the system of interest, the conceptual model, the numerical implementation, and the results of this work.

Contents

1 Project Description

We implement a model and application for converting a 2D and 3D meshes into dynamic physical audio models, facilitating real-time user interaction for the application of excitation forces and

modification of model parameters. While our ultimate model is designed to accommodate a wide variety of volumetric objects, our primary focus is on the specialized domain of metallic bells.

An example bell and its axisymmetric cross section are given in Figure 1. Rather than modeling the entire 3D volume of the bell, we have elected to use the axisymmetry of the problem and model only the cross-section shown on the right-hand side of Figure 1, which greatly reduces the computational cost. The inputs to our model are a two-dimensional cross-section, the Young's Modulus of the material, the Poisson's ratio of the material, and the density of the material.

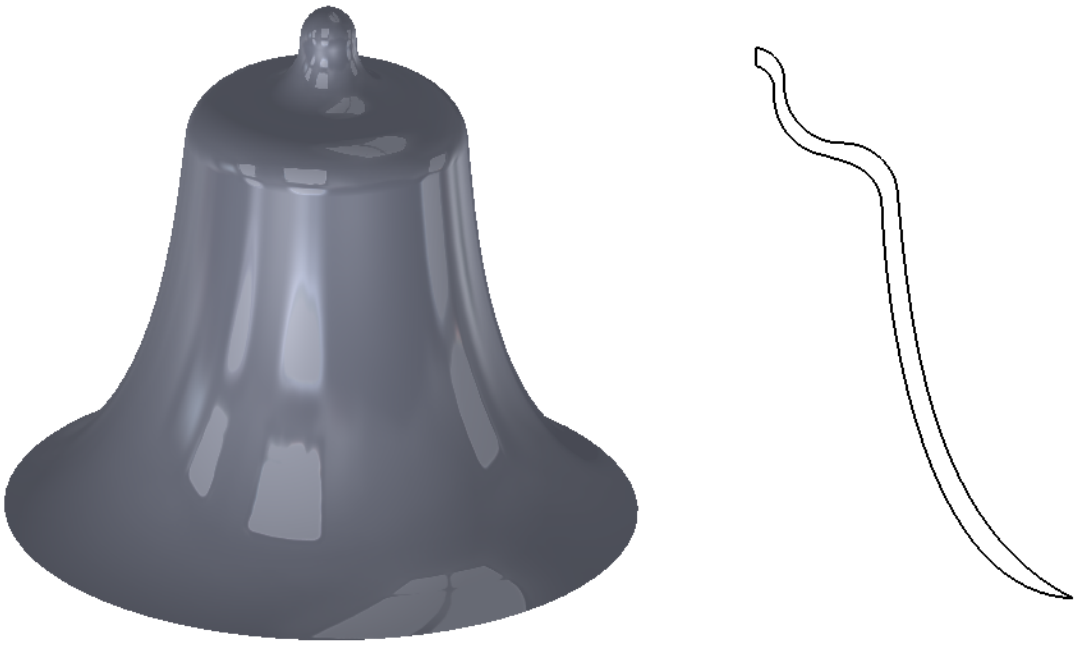

Figure 1: System of Interest

2 Literature Review

Physical audio modeling offers many benefits compared with other digital audio synthesis techniques. Physically accurate models map directly to the physical systems they approximate, making interpretation and control intuitive. They generate audio with a natural and realistic character, and they are highly general; a physical model for one instrument can be adapted to another with relative ease, and complex systems can be composed and connected in straightforward ways.

However, physical audio models are also computationally expensive compared to other audio synthesis techniques, requiring simplifications to trade off realism and physical plausibility for computational tractability.

Digital waveguides [14], for example, model 1D systems such as strings or wind instruments by iterating the 1D wave equation with an appropriately chosen wave variable (e.g., string displacement or air pressure), modeling boundaries like plectra (picks), guitar bridges, or flute tone holes as scattering junctions with digital audio filters designed to match their wave impedance. However, waveguides are limited to 1D systems with low dispersion and do not lend themselves easily to

nonlinear interactions [3].

Finite difference schemes, finite volume methods, and other time-stepping methods are more computationally costly but are much more general, offering nonlinear interactions, time-varying external interaction, and straightforward extension to two and three dimensions [3].

The technique of modal synthesis uses a source-filter model to decompose physical audio systems into an excitation signal that drives a parallel filter bank implementing the transfer functions of resonant modes in the physical system [13]. The resonant modes can be computed directly from a 3D model, and the source-filter audio model can be implemented compactly and efficiently with classical digital audio filter techniques. This approach offers a compromise between the generality and physical accuracy of finite difference schemes while still allowing for real-time sound synthesis in computationally constrained settings.

Modal synthesis is the approach we will investigate in our project. Specifically, we aim to *convert a volumetric mesh into a physical audio model*. 3D volumetric meshes are ubiquitous in graphics. The ability to interpret existing volumetric meshes as physically plausible audio generators would be of great use for digital audiovisual media production, such as in video games or movies. Additionally, the modal synthesis paradigm supports extracting audio responses from external impulse forces, enabling estimation of the audio response from object interactions, as well as real-time instrument control in the music production context.

We will model our initial implementation on the work of Michon, Martin & Smith in their *Mesh2Faust* project [9]. In this approach, a bell is modeled using the finite element method using a 3-D tetrahedral mesh. The physics of the system is modeled using the linear deformation equation with no damping,

$$\mathbf{M\ddot{x}(t) + Kx(t) = f(}\boldsymbol{t}\mathbf{)}$$

where $\mathbf{x}(\boldsymbol{t})$ is the vector of displacements of each node, $\mathbf{M}$ and $\mathbf{K}$ are the finite element mass and stiffness matrices respectively, and $\mathbf{f}(\boldsymbol{t})$ is the force vector. It is in this modeling step that we hope to greatly reduce the computational cost by modeling only a 2d cross-section of a bell using axisymmetric triangular elements [10]. Assume that solutions to this system have the form $u_i(t) = \mathbf{U}_i e^{j\omega_i t}$ where $\mathbf{U}_i \in \mathsf{R}^{2n}$ and $\omega_i \in \mathsf{R}$. Substituting this assumed solution into the linear deformation equation yields the generalized eigenvalue problem

$$\mathbf{K\Lambda = MU\Lambda},$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues and $\mathbf{U}$ is a modal matrix containing the eigenvectors $\mathbf{U}_i$ of the linear deformation equation. For this step, we intend to implement the *FEAST* [15] algorithm to find eigenpairs quickly.

Using the transformation $\mathbf{x = Uq}$, the linear deformation equation can be decoupled giving

$$\mathbf{\ddot{q} + \Lambda q = U^T f}.$$

The solutions of the homogenous system $\mathbf{\ddot{q} + \Lambda q = 0}$ are given by a set of modes

$$q_i = a_i \sin(2\pi f_i + \theta_i),$$

where a_i is the amplitude of excitation, f_i is its frequency, and θ_i is a phase shift, assumed to be zero. The a_i depend on the object and the excitement location, and the f_i are given by

$$f_i = \frac{1}{2\pi}\sqrt{\lambda_i}$$

for each eigenvalue λ_i. These outputs are used as inputs to the modal analysis mentioned previously, yielding a profile for a synthesized sound.

We hope to extend and improve on the existing work of Mesh2Faust in the following ways:

- *Improve slow compile times from mesh to modal parameters*. Mesh2Faust takes about 15 minutes to convert a mesh with ∼30k faces on a "regular laptop" [8].
- Mesh2Faust is a command line application and provides no control over generating the 3D mesh itself. We aim to create a visual interface with controls for mesh generation & material properties, as well as parameters of the modal audio generation, with a focus on bell modeling.
- Mesh2Faust utilizes full 3D finite element analysis. We will investigate using 2d axisymmetric elements to take advantage of the inherent radial symmetry in bells. This will dramatically reduce the number of vertices, the computational cost, and the size of the gains matrix required to post-process the finite element results. We expect this simplification will allow for near real-time workflows when combined with GPU acceleration. However, imposing radial symmetry may produce less realism in the final audio result, and we plan to investigate this tradeoff.
- In addition, Mesh2Faust relies on Intel's MKL library[1], which is closed-source and only compatible with Intel processors. As part of our contribution, we plan to support running on non-Intel processors, such as ARM.

3 Conceptual Model

Our conceptual model is based on that presented by Michon et. al.[9]. The overall structure of this approach is given in Figure 2. We split our model into two parts: 1) physical audio modeling and 2) finite element modeling. The details of each part follow.

3.1 Finite Element Modeling

At its core, the finite element method is one of many approaches to numerically solving differential equations. For our purposes, we will be using 2d, axisymmetric, triangular elements that are formulated to solve the linear elasticity equations in cylindrical coordinates:

$$\begin{aligned}
\frac{\partial \sigma_{rr}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{r\theta}}{\partial \theta} + \frac{\partial \sigma_{rz}}{\partial z} + \frac{1}{r}(\sigma_{rr} - \sigma_{\theta\theta}) + F_r &= \rho \frac{\partial^2 u_r}{\partial t^2}, \\
\frac{\partial \sigma_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta\theta}}{\partial \theta} + \frac{\partial \sigma_{\theta z}}{\partial z} + \frac{2}{r}\sigma_{r\theta} + F_\theta &= \rho \frac{\partial^2 u_\theta}{\partial t^2}, \\
\frac{\partial \sigma_{rz}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta z}}{\partial \theta} + \frac{\partial \sigma_{zz}}{\partial z} + \frac{1}{r}\sigma_{rz} + F_z &= \rho \frac{\partial^2 u_z}{\partial t^2}.
\end{aligned}$$

In these equations, σ_{ij} denotes a stress on the i^{th} face of a differential volume in the j^{th} direction, F_i denotes an external force in the i^{th} direction, and u_i denotes displacement in the i^{th} direction. In a static finite element analysis, the displacements u_i are found by solving a system of equations

$$\boldsymbol{K}\boldsymbol{x} = \boldsymbol{F}$$

where $\boldsymbol{K}$ is a matrix of stuffiness coefficients that relate the nodal displacements contained in $\boldsymbol{x}$ to the external forces contained in $\boldsymbol{F}$. In dynamic finite element analysis with damping ignored, the ODE

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = \boldsymbol{F} \tag{1}$$

is advanced from an initial condition $\boldsymbol{x}_0$. In this equation, $\boldsymbol{K}$ and $\boldsymbol{F}$ are as described earlier, and $\boldsymbol{M}$ is a matrix that takes element mass into account. Given that we are interested in finding frequencies and shapes of vibrations, which are time-dependent in nature, our modeling will utilize the latter of these two approaches.

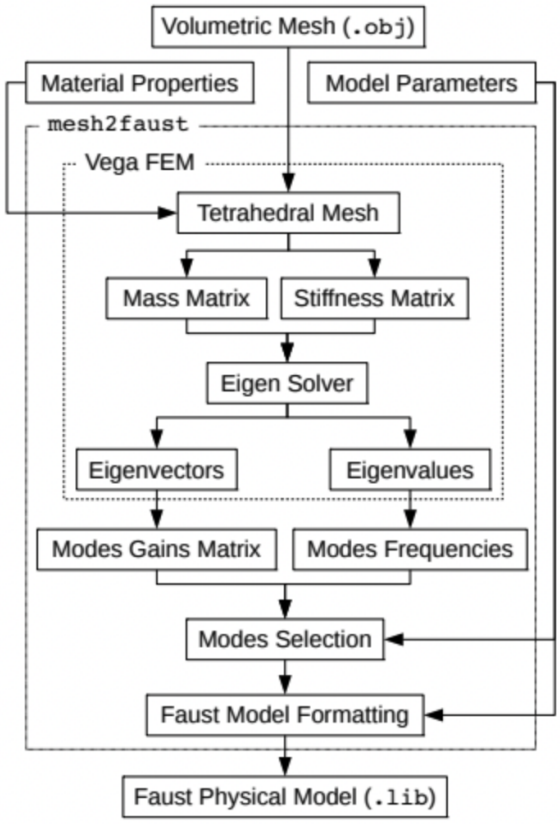

Figure 2: Program Structure [9]

To see the value of the eigenvalues and eigvenvectors assume that solutions to this system (1) have the form $u_i(t) = \mathbf{U}_i e^{j\omega_i t}$ where $\mathbf{U}_i \in \mathsf{R}^{2n}$ and $\omega_i \in \mathsf{R}$. Substituting this assumed solution into (1) yields the generalized eigenvalue problem

$$\mathbf{K\Lambda} = \mathbf{MU\Lambda},$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues and $\mathbf{U}$ is a modal matrix containing the eigenvectors $\mathbf{U}_i$ of the linear deformation equation.

Using the transformation $\mathbf{x} = \mathbf{Uq}$, the linear deformation equation can be decoupled giving

$$\ddot{\mathbf{q}} + \mathbf{\Lambda q} = \mathbf{U^T f}.$$

The solutions of the homogenous system $\ddot{\mathbf{q}} + \mathbf{\Lambda q} = \mathbf{0}$ are given by a set of modes

$$q_i = a_i \sin(2\pi f_i + \theta_i),$$

where a_i is the amplitude of excitation, f_i is its frequency, and θ_i is a phase shift, assumed to be zero. The a_i depend on the object and the excitement location, and the f_i are given by

$$f_i = \frac{1}{2\pi}\sqrt{\lambda_i}$$

for each eigenvalue λ_i. These outputs are used as inputs to the modal analysis, yielding a profile for a synthesized sound.

3.2 Physical Audio Modeling

As explained in the literature review, we find the set of modes,

$$q_i = a_i \sin(2\pi f_i + \theta_i),$$

where a_i is the amplitude of excitation, f_i is its frequency, and θ_i is a phase shift, assumed to be zero. The a_i depend on the object and the excitement location, and the f_i are given by

$$f_i = \frac{1}{2\pi}\sqrt{\lambda_i}$$

for each eigenvalue λ_i.

Each mode is implemented as an exponentially decaying sine wave, with its own frequency, gain, and resonance duration (T60). These sine waves are implemented as resonant bandpass filters, allowing any signal to excite them. Specifically, the mode filters are implemented as a parallel bank of biquad filters with transfer function

$$H(z) = g\frac{1 - z^2}{1 + \alpha_1 z^{-1} + \alpha_2 z^{-2}},$$

with

$$\begin{aligned}
\alpha_1 &= -2\tau\cos\omega \\
\alpha_2 &= \tau^2 \\
\omega &= \frac{2\pi f}{f_s} \\
\tau &= 0.001^{\frac{1}{t^{60}}},
\end{aligned}$$

where g, f, and t_{60} are the mode gain, frequency, and T60, respectively.

4 Simulation Model

The simulation of our conceptual model is naturally separated into two parts, finite element modeling and physical audio modeling. Details of each are as follows:

4.1 Finite Element Modeling

The purpose of the finite element code is to populate the stiffnes and mass matrices of the system of interest. Finite element stiffness matrices are generally computed using an energy minimization approach [6]. For a general element with displacement function matrix $\boldsymbol{B}$ and stress-strain tensor $\boldsymbol{D}$, the element stiffness matrix can be computed by evaluating

$$\boldsymbol{K}_e = \int_\Omega \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \mathrm{d}\Omega \tag{2}$$

over some domain Ω. For our purposes we use a linear mapping from (r, z) space to (s, t) space in which each triangular element is mapped as if it were a square, with two nodes sharing the same coordinate, to the unit square $[-1, 1] \times [-1, 1]$. This mapping, along with the use of axisymmetry, allows for (2) to be written as

$$\boldsymbol{K}_e = 2\pi r_c \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} ||\boldsymbol{J}|| \mathrm{d}s\mathrm{d}t$$

where r_c is the center of mass of the element. This formulation allows for a 9-point Gaussian Quadrature rule to be used to approximate the integral. For axisymmetric problems, the matrix $\boldsymbol{D}$ [2] that relates the stress $\{\sigma_r \ \sigma_z \ \sigma_\theta \ \tau_{rz}\}^T$ to the strains $\{\varepsilon_r \ \varepsilon_z \ \varepsilon_\theta \ \gamma_{rz}\}^T$ is given by

$$\begin{Bmatrix} \sigma_r \\ \sigma_z \\ \sigma_\theta \\ \tau_{rz} \end{Bmatrix} = \underbrace{\frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & 0 \\ 0 & 0 & 0 & (1-2\nu)/2 \end{bmatrix}}_{\boldsymbol{D}} \begin{Bmatrix} \varepsilon_r \\ \varepsilon_z \\ \varepsilon_\theta \\ \gamma_{rz} \end{Bmatrix},$$

where E and ν are the Young's Modulus and Poisson's ratio of the material being modeled. Details regarding the matrices $\boldsymbol{B}$ and $\boldsymbol{J}$ are provided in Appendix B.

The mass matrix [6] is given by the integral

$$\boldsymbol{M}_e = \int_\Omega \rho \boldsymbol{N}^T \boldsymbol{N} \mathrm{d}\Omega \tag{3}$$

over some domain Ω where $\boldsymbol{N}$ is the matrix of shape functions as described in Appendix B. Again, as a result of the linear mapping used to go from (r, z) space to (s, t) space and the assumption that density is uniform, (3) can be rewritten as

$$\boldsymbol{M}_e = 2\pi r_c \rho A_e \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{N}^T \boldsymbol{N} \mathrm{d}s\mathrm{d}t$$

where ρ is the density of the material, r_c is the centroid of the element, and A_e is the area of the element. Global mass and stiffness matrices are then created via a superposition of element mass and stiffness matrices.

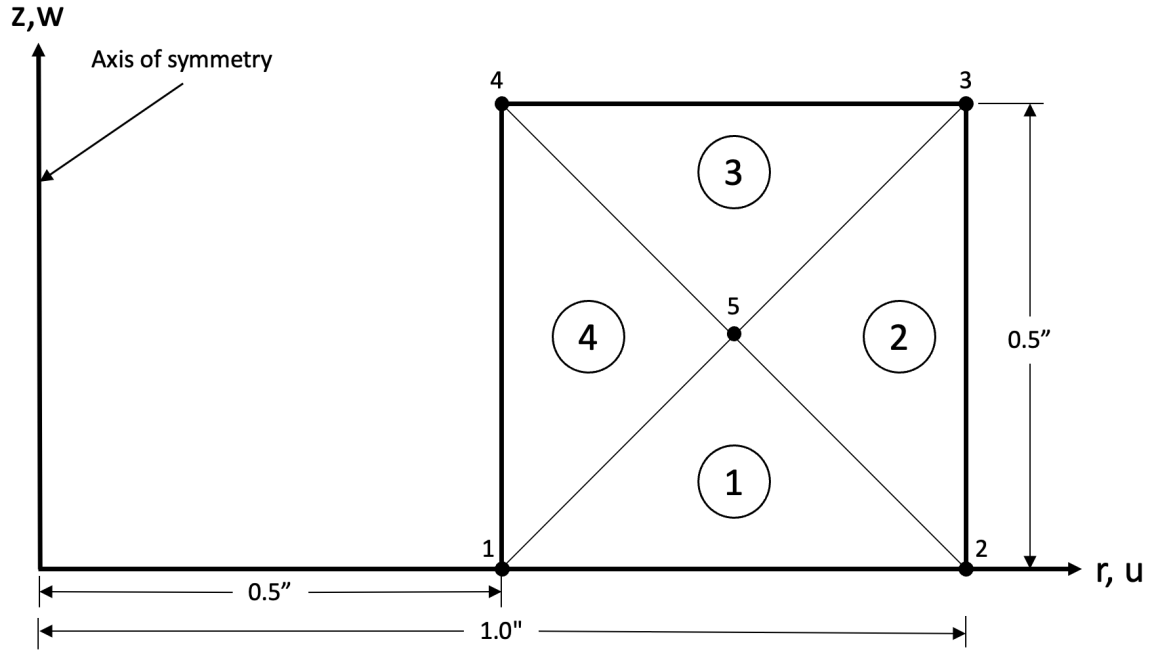

Figure 3: Stiffness Matrix Verification Case

4.2 Physical Audio Modeling

5 Validation and Verification

In order to verify the model, the stiffness and mass matrices produced by the finite element method must first be verified against trustworthy, exact calculations. For this, we use the results of [2]. This presentation presents the finite element test problem shown in Figure 3 and provides element and global stiffness matrices through exact calculations. This test case features four connected element stiffness matrices that share five nodes. Table 1 gives the maximum error, defined by the vector infinity norm

$$e = 100 \times \left|\left| \frac{K_{ij}^C - K_{ij}^E}{K_{ij}^E} \right|\right|_\infty$$

where $K_{i,j}^C$ references the stiffness matrix produced by our code and K_{ij}^E references the exact data provided by [2]. Note that exact agreement is not to be expected, as our implementation utilizes approximate Gaussian Quadrature rather than exact integration to compute the stiffness matrices. The resulting relative errors however are small, and shown to be insignificant in the validation step. Details regarding the exact and computed stiffness matrices are provided in Appendix C. The mass matrix is shown in Appendix D to match the expected result exactly, within machine precision. This is due to the Gaussian Quadrature rule used being sufficient to evaluate the polynomials formed in the product $\boldsymbol{N}^T\boldsymbol{N}$ exactly. In order to validate the finite element model, we utilize the thin walled pressure vessel case shown in Figure 4 with known theoretical solution. The hoop stress calculated by our finite element implementation agrees with a relative error of only 0.15%. This verifies that our implementation yields physical results.

Table 1: Stiffness Matrix Relative Error

Element	1	2	3	4	Global
e	2.86%	1.13%	2.86%	3.73%	3.73%

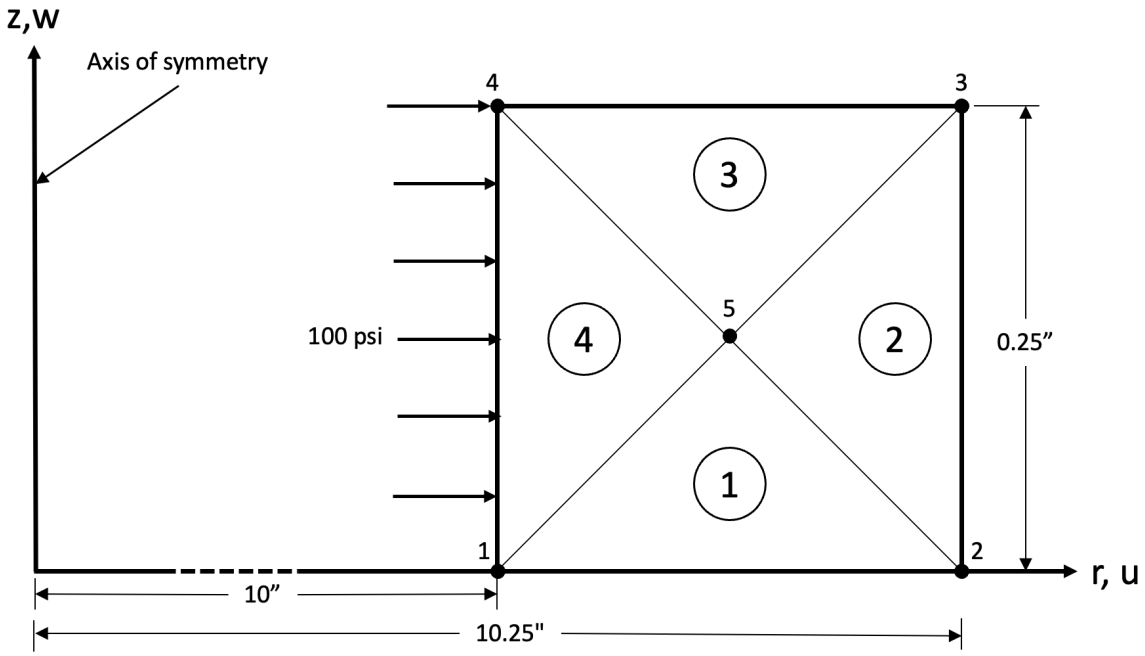

Figure 4: Validationwe Problem

6 Experimental Results

This section discusses the audio model validation. For a discussion of the finite element validation, see appendix A.

Although we would like to have completed more rigorous validation of our final audio models, our validation results are currently limited to a qualitative analysis by simply listening to the results and comparing them with their real-world counterparts.

Additionally, we qualitatively verified that, for the 3D modeling case, the results of our generated Faust DSP programs produced nearly identical to the baseline that we reproduced following mesh2faust. [8]

We have uploaded audio samples demonstrating both our 3D and 2D axisymmetric methods at [5].

These samples include audio generated by using a shaped pulse of filtered white noise (emulating a hammer strike) to excite the resonating biquad filters corresponding to the modes associated with the struck vertices in our interactive "Mesh2Audio" application.

The samples include simulations of the following:

- Three kinds of bells
- A wine glass
- The standard "Utah teapot" 3D model [16]
- A "hand drum" generated by extruding the 2D profile shown in Fig. 5 into the 3D mesh shown in Fig. 6

All 3D models except the teapot were generated by extruding the correspondingly named 2D bell cross-sections that can be found in the `app/res` directory of the repository linked in the title page.

As demonstrated in our project video, in addition to the near-instantanious generation of axisymmetric audio models, we also successfully reduced the computation time needed to:

- Generate tetrahedral volumetric meshes from triangular meshes: We consistently see speedups on the order of 10x compared to the mesh2faust implementation by using the tetgen[12] library instead of Vega[7].
- Solve the sparse eigenvalue problem to estimate the resonant modes: We often see up to 2x speedup in estimating eigenvalues through the use of the Spectra[11] library, and by using better configuration settings and reducing memory usage.

7 Discussion & Conclusion

We successfully achieved our project goals, implementing both 3D and 2D-axisymmetric physical audio models of several 3D objects, such as bells, wine glasses, non-symmetric 3D volumetric meshes from external sources, and many varieties of custom-generated axisymmmetric 3D models.

In addition, through the computational improvements discussed in the results section, and by implementing a custom mesh viewing and generating library, we achieved our main goal of enabling a rapid interactive workflow for generating realistic physical audio models.

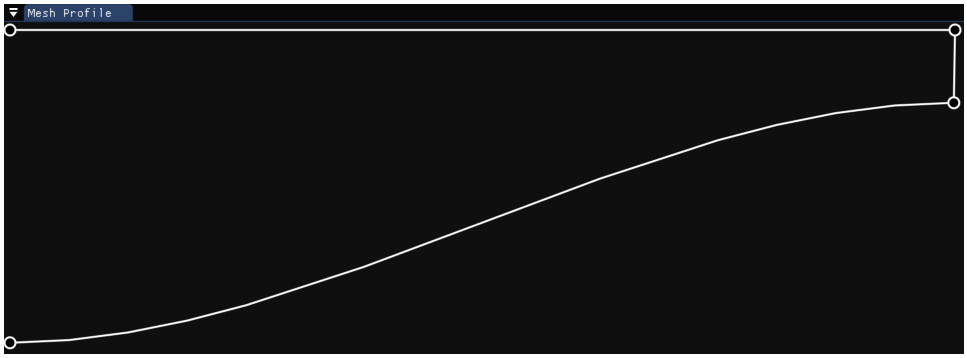

Figure 5: "Hand drum" 2D profile

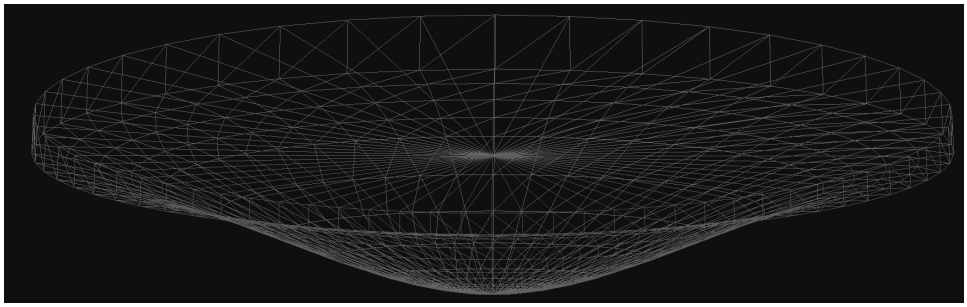

Figure 6: "Hand drum" 3D triangular mesh

We are also in the process of contributing much of our work to the open-source Faust project [4].

Although our 2D-axisymmetric physical audio modeling method is novel (to our knowledge) and generates results of moderate fidelity and realism, there is still a good deal of examination and work needed to make the quality useful enough for serious use beyond creative musical applications.

Narrowing the gap between our 2D and 3D models while maintaining the same order of computation time would enable rapid prototyping in scientific research, and speed up development in creative digital media applications such as animation, video games, Foley, and music production.

References

[1] *Accelerate Fast Math with Intel® oneAPI Math Kernel Library*. URL: https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html (visited on 03/14/2023).

[2] Mohamed Arafa. *Finite Element Method Chapter 9 Axisymmetric Elements*. 2017. URL: http://site.iugaza.edu.ps/marafa/files/FEM-Chapter-9-2017-18.pdf (visited on 04/24/2023).

[3] Stefan Bilbao et al. "The NESS Project: Physical Modeling, Algorithms and Sound Synthesis". In: *Computer Music Journal* 43 (Nov. 2019). DOI: 10.1162/comj_a_00516.

[4] *Faust - Functional programming language for signal processing and sound synthesis*. 2023. URL: https://github.com/grame-cncm/faust (visited on 02/04/2023).

[5] Karl Hiner and Ben Wilfong. *Mesh2Audio audio samples*. URL: https://drive.google.com/drive/folders/1wTdc4w5yPa5-ZNjtl8z-cwkA1gmOq1Jv.

[6] T.J.R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Civil and Mechanical Engineering. Dover Publications, 2012. ISBN: 978-0-486-13502-1. URL: https://books.google.com/books?id=cHH2n_qBK0IC.

[7] Daniel Schroeder: Jernej Barbič Fun Shing Sin. *Vega FEM Library*. 2012. URL: http://www.jernejbarbic.com/vega (visited on 02/04/2023).

[8] Romain Michon. *Romain Michon - Faust Tutorials*. URL: https://ccrma.stanford.edu/~rmichon/faustTutorials/#converting-a-mesh-to-a-faust-physical-model (visited on 03/10/2023).

[9] Romain Michon, Sara R Martin, and Julius O Smith. "MESH2FAUST: A Modal Physical Model Generator for the Faust Programming Language -Application to Bell Modeling". In: (Mar. 2021).

[10] Woolley Mitchell and Fisher. *Formulation and Experimental Veri cation of an Axisymmetric Finite-Element Structural Analysis*. 1971. URL: https://nvlpubs.nist.gov/nistpubs/jres/75C/jresv75Cn3-4p155_A1b.pdf.

[11] Yixuan Qiu. *Spectra - C++ Library For Large Scale Eigenvalue Problem*. 2023. URL: https://spectralib.org/ (visited on 02/04/2023).

[12] Hang Si. "TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator". In: *ACM Transactions on Mathematical Software* 41.2 (Feb. 2015), 11:1–11:36. ISSN: 0098-3500. DOI: 10.1145/2629697. URL: https://doi.org/10.1145/2629697 (visited on 04/26/2023).

[13] Julius O. Smith. "Physical Audio Signal Processing". In: (accessed March 2023). online book, 2010 edition. URL: https://ccrma.stanford.edu/~jos/pasp/Modal_Representation.html.

[14] Julius O. Smith. "Physical Modeling Using Digital Waveguides". In: *Computer Music Journal* 16.4 (1992), pp. 74–91. ISSN: 01489267, 15315169. URL: http://www.jstor.org/stable/3680470 (visited on 03/10/2023).

[15] Ping Tak Peter Tang and Eric Polizzi. *FEAST as a Subspace Iteration Eigensolver Accelerated by Approximate Spectral Projection*. 2013. URL: https://arxiv.org/abs/1302.0432.

[16] *Utah Teapot*. URL: https://graphics.cs.utah.edu/teapot/.

A Distribution of Work

The modeling is naturally split into two components. In the first, the finite element method is used to produce global mass and stiffness matrices which is being completed by Ben. These outputs are then used as inputs to the physical audio modeling, which Karl is completing along with the application interface.

B Finite Element Matrix Details

For a triangular finite element, the shape functions that map to the unit square are given by

$$N_1 = \frac{1}{4}(1+s)(1+t),$$
$$N_2 = \frac{1}{4}(1-s)(1+t),$$
$$N_3 = \frac{1}{2}(1-t).$$

The matrix $\boldsymbol{N}$ that relates the (r,z) coordinates to the (u,v) coordinates is given by

$$\begin{Bmatrix} r(s,t) \\ z(s,t) \end{Bmatrix} = \underbrace{\frac{1}{4}\begin{bmatrix} (1+s)(1+t) & 0 & (1-s)(1+t) & 0 & 2(1-t) & 0 \\ 0 & (1+s)(1+t) & 0 & (1-s)(1+t) & 0 & 2(1-t) \end{bmatrix}}_{\boldsymbol{N}} \begin{Bmatrix} r_1(x,y) \\ z_1(x,y) \\ r_2(x,y) \\ z_2(x,y) \\ r_3(x,y) \\ z_3(x,y) \end{Bmatrix}.$$

The associated Jacobian matrix $\boldsymbol{J}$ that describes the mapping form (r,z) space to (s,t) space is given by

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial(r(s,t))}{\partial s} & \frac{\partial(z(s,t))}{\partial s} \\ \frac{\partial(r(s,t))}{\partial t} & \frac{\partial(z(s,t))}{\partial t} \end{bmatrix} = \frac{1}{4}\begin{bmatrix} (r_1-r_2)(1+t) & (z1-z2)(1+t) \\ (r_1-r_2)s + r_1 + r_2 - 2r_3 & (z_1-z_2)s + z_1 + z_2 - 2z_3 \end{bmatrix}.$$

The derivatives of the shape functions in (x,t) space are then given by

$$\begin{Bmatrix} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_1}{\partial y} \end{Bmatrix} = \boldsymbol{J}^{-1}\begin{Bmatrix} \frac{\partial N_1}{\partial s} \\ \frac{\partial N_1}{\partial t} \end{Bmatrix} = \begin{Bmatrix} \frac{z_3-z_2}{r_1(z_3-z_2)+r_2(z_1-z_3)+r_3(z_2-z_1)} \\ \frac{r_2-r_3}{r_1(z_3-z_2)+r_2(z_1-z_3)+r_3(z_2-z_2)} \end{Bmatrix}$$
$$\begin{Bmatrix} \frac{\partial N_2}{\partial x} \\ \frac{\partial N_2}{\partial y} \end{Bmatrix} = \boldsymbol{J}^{-1}\begin{Bmatrix} \frac{\partial N_2}{\partial s} \\ \frac{\partial N_2}{\partial t} \end{Bmatrix} = \begin{Bmatrix} \frac{z_1-z_3}{r_1(z_3-z_3)+r_2(z_1-z_3)+r_3(z_2-z_1)} \\ \frac{r_1-r_3}{r_1(z_2-z_3)+r_2(z_3-z_1)+r_3(z_1-z_2)} \end{Bmatrix}$$
$$\begin{Bmatrix} \frac{\partial N_3}{\partial x} \\ \frac{\partial N_3}{\partial y} \end{Bmatrix} = \boldsymbol{J}^{-1}\begin{Bmatrix} \frac{\partial N_3}{\partial s} \\ \frac{\partial N_3}{\partial t} \end{Bmatrix} = \begin{Bmatrix} \frac{z_1-z_2}{r_1(z_2-z_3)+r_2(z_3-z_1)+r_3(z_1-z_2)} \\ \frac{r_1-r_2}{r_1(z_3-z_2)+r_2(z_1-z_3)+r_3(z_2-z_2)} \end{Bmatrix}$$

These derivatives are then used to populate the matrix $\boldsymbol{B}$

$$\begin{Bmatrix} \varepsilon_r(s,t) \\ \varepsilon_z(s,t) \\ \varepsilon_\theta(s,t)\gamma_{rz} \end{Bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} \\ \frac{N_1}{r} & 0 & \frac{N_2}{r} & 0 & \frac{N_3}{r} & 0 \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial}{\partial N_3}x \end{bmatrix}}_{\boldsymbol{B}} \begin{Bmatrix} r_1 \\ z_1 \\ r_2 \\ z_2 \\ r_3 \\ z_3 \end{Bmatrix}$$

that relates the nodal displacments in the (r,z) space to strains in the (s,t) space

C Finite Element Stiffness Matrix Verification Details

The error matrix E is given by

$$E = 100 \times \frac{|K_x^C - K_x^E|}{|K_x^E|},$$

where K_x^C is the x element matrix from our implementation and K_x^E is the x element matrix from the reference [2]. Elements of E that result in NaN due to zeros in K_x^E are replaced with zeros. The maximum relative error in an element stiffness matrix is 3.73% and the maximum relative error in the global stiffness matrix is 3.73%. Exact agreement is not to be expected as our implementation uses a 9-point 2D Gaussian Quadrature rule to compute element stiffness matrices while [2] uses exact integration. These results serve to verify both our element stiffness matrix calculation as well as our global assembly routine. The results for each matrix of interest follow.

C.1 Element 1 Validation

Exact result from [2].

$$K_1^E = 10^6 \begin{bmatrix} 54.46 & 29.45 & -31.63 & 2.26 & -29.37 & -31.71 \\ 29.45 & 61.17 & -11.33 & 33.98 & -31.72 & -95.15 \\ -31.63 & -11.33 & 72.59 & -38.52 & -20.31 & 49.84 \\ 2.26 & 33.98 & -38.52 & 61.17 & 22.66 & -95.15 \\ -29.37 & -31.72 & -20.31 & 22.66 & 56.72 & 9.06 \\ -31.74 & -95.15 & 49.84 & -95.15 & 9.06 & 190.31 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Approximate result from our implementation.

$$K_1^C = 10^6 \begin{bmatrix} 55.63 & 29.45 & -32.22 & 2.27 & -29.96 & -31.72 \\ 29.45 & 61.17 & -11.33 & 33.98 & -31.72 & -95.15 \\ -32.22 & -11.33 & 73.76 & -38.51 & -20.89 & 49.84 \\ 2.27 & 33.98 & -38.51 & 61.17 & 22.66 & -95.15 \\ -29.96 & -31.72 & -20.89 & 22.66 & 57.90 & 9.06 \\ -31.71 & -95.15 & 49.84 & -95.15 & 9.06 & 190.31 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Element-wise relative error.

$$E_1 = \begin{bmatrix} 2.15 & 0 & 1.87 & 0.44 & 2.01 & 0.03 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1.87 & 0 & 1.61 & 0.03 & 2.86 & 0 \\ 0.44 & 0 & 0.03 & 0 & 0 & 0 \\ 2.01 & 0 & 2.86 & 0 & 2.08 & 0 \\ 0.03 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \%$$

The maximum relative error is 2.86%.

C.2 Element 2 Validation

Exact result from [2].

$$K_2^E = 10^6 \begin{bmatrix} 85.75 & -46.07 & 52.52 & 12.84 & -118.92 & 33.23 \\ -46.07 & 74.77 & -12.84 & -41.54 & 45.32 & -33.23 \\ 52.52 & -12.84 & 85.74 & 46.07 & -118.92 & -33.23 \\ 12.84 & -41.54 & 46.07 & 74.77 & -45.32 & -33.23 \\ -118.92 & 45.32 & -118.92 & -45.32 & 216.41 & 0 \\ 33.23 & -33.23 & -33.23 & -33.23 & 0 & 66.46 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Approximate result from our implementation.

$$K_2^C = 10^6 \begin{bmatrix} 86.71 & -46.07 & 52.04 & 12.84 & -119.39 & 33.23 \\ -46.07 & 74.76 & -12.84 & -41.54 & 45.31 & -33.23 \\ 52.04 & -12.84 & 86.71 & 46.07 & -119.39 & -33.23 \\ 12.84 & -41.54 & 46.07 & 74.76 & -45.31 & -33.23 \\ -119.39 & 45.31 & -119.39 & -45.31 & 217.36 & 0.00 \\ 33.23 & -33.23 & -33.23 & -33.23 & 0.00 & 66.46 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Element-wise relative error.

$$E_2 = \begin{bmatrix} 1.12 & 0 & 0.91 & 0 & 0.40 & 0 \\ 0 & 0.01 & 0 & 0 & 0.02 & 0 \\ 0.91 & 0 & 1.13 & 0 & 0.40 & 0 \\ 0 & 0 & 0 & 0.01 & 0.02 & 0 \\ 0.40 & 0.02 & 0.40 & 0.02 & 0.44 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \%$$

The maximum relative error is 1.13%.

C.3 Element 3 Validation

Exact result from [2].

$$K_3^E = 10^6 \begin{bmatrix} 72.58 & 38.52 & -31.63 & 11.33 & -20.31 & -49.84 \\ 38.52 & 61.17 & -2.26 & 33.98 & -22.66 & -95.15 \\ -31.63 & -2.26 & 54.46 & -29.45 & -29.37 & 31.72 \\ 11.33 & 33.98 & -29.45 & 61.17 & 31.72 & -95.15 \\ -20.31 & -22.66 & -29.37 & 31.72 & 56.72 & -9.06 \\ -49.84 & -99.15 & 31.72 & -95.15 & -9.06 & 190.31 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Approximate result from our implementation.

$$K_3^C = 10^6 \begin{bmatrix} 73.76 & 38.51 & -32.22 & 11.33 & -20.89 & -49.84 \\ 38.51 & 61.17 & -2.27 & 33.98 & -22.66 & -95.15 \\ -32.22 & -2.27 & 55.63 & -29.45 & -29.96 & 31.72 \\ 11.33 & 33.98 & -29.45 & 61.17 & 31.72 & -95.15 \\ -20.89 & -22.66 & -29.96 & 31.72 & 57.90 & -9.06 \\ -49.84 & -95.15 & 31.72 & -95.15 & -9.06 & 190.31 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Element-wise relative error.

$$E_3 = \begin{bmatrix} 1.63 & 0.03 & 1.87 & 0 & 2.86 & 0 \\ 0.03 & 0 & 0.44 & 0 & 0 & 0 \\ 1.87 & 0.44 & 2.15 & 0 & 2.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 2.86 & 0 & 2.01 & 0 & 2.08 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \%$$

The maximum relative error is 2.86%.

C.4 Element 4 Validation

Exact result from [2].

$$K_4^E = 10^6 \begin{bmatrix} 41.53 & -21.90 & 20.39 & 0.75 & -66.45 & 21.14 \\ -21.90 & 47.57 & -0.75 & -26.43 & 36.24 & -21.14 \\ 20.39 & -0.75 & 41.53 & 21.90 & -66.45 & -21.14 \\ 0.75 & -26.43 & 21.90 & 47.57 & -36.24 & -21.14 \\ -66.45 & 36.24 & -66.45 & -36.24 & 169.14 & 0 \\ 21.14 & -21.14 & -21.14 & -21.14 & 0 & 42.28 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Approximate result from our implementation.

$$K_4^C = 10^6 \begin{bmatrix} 43.05 & -21.90 & 19.63 & 0.76 & -67.21 & 21.15 \\ -21.90 & 47.58 & -0.76 & -26.43 & 36.25 & -21.15 \\ 19.63 & -0.76 & 43.05 & 21.90 & -67.21 & -21.15 \\ 0.76 & -26.43 & 21.90 & 47.58 & -36.25 & -21.15 \\ -67.21 & 36.25 & -67.21 & -36.25 & 170.67 & 0.00 \\ 21.15 & -21.15 & -21.15 & -21.15 & 0.00 & 42.29 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Element-wise relative error.

$$E_4 = \begin{bmatrix} 3.66 & 0 & 3.73 & 1.33 & 1.14 & 0.05 \\ 0 & 0.02 & 1.33 & 0 & 0.03 & 0.05 \\ 3.73 & 1.33 & 3.66 & 0 & 1.14 & 0.05 \\ 1.33 & 0 & 0 & 0.02 & 0.03 & 0.05 \\ 1.14 & 0.03 & 1.14 & 0.03 & 0.90 & 0 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0 & 0.02 \end{bmatrix} \%$$

The maximum relative error is 3.73%.

C.5 Global Stiffness Validation

Exact result from [2].

$$K_G^E = 10^6 \begin{bmatrix} 95.99 & 51.35 & -31.63 & 2.26 & 0 & 0 & 20.39 & -0.75 & -95.82 & -52.86 \\ 51.35 & 108.74 & -11.33 & 33.98 & 0 & 0 & 0.75 & -26.43 & -67.96 & -116.3 \\ -31.63 & -11.33 & 158.34 & -84.49 & 52.52 & 12.84 & 0 & 0 & -139.2 & 83.07 \\ 2.26 & 33.98 & -84.59 & 135.94 & -12.84 & -41.54 & 0 & 0 & 67.98 & -128.4 \\ 0 & 0 & 52.52 & -12.84 & 158.33 & 84.59 & -31.63 & 11.33 & -139.2 & -83.07 \\ 0 & 0 & 12.84 & -41.54 & 84.59 & 135.94 & -2.26 & 33.98 & -67.98 & -128.4 \\ 20.39 & 0.75 & 0 & 0 & -31.63 & -2.26 & 95.99 & -51.35 & -95.82 & 52.86 \\ -0.75 & -26.43 & 0 & 0 & 11.33 & 33.98 & -51.35 & 108.74 & 67.96 & -116.3 \\ -95.82 & -67.96 & -139.2 & 67.98 & -139.2 & -67.98 & -95.82 & 67.96 & 498.99 & 0 \\ -52.86 & -116.3 & 83.07 & -128.4 & -83.07 & -128.4 & 52.86 & -116.3 & 0 & 489.36 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Approximate result from our implementation.

$$K_G^C = 10^6 \begin{bmatrix} 98.68 & 51.35 & -32.22 & 2.27 & 0.00 & 0.00 & 19.63 & -0.76 & -97.17 & -52.86 \\ 51.35 & 108.75 & -11.33 & 33.98 & 0.00 & 0.00 & 0.76 & -26.43 & -67.97 & -116.30 \\ -32.22 & -11.33 & 160.47 & -84.58 & 52.04 & 12.84 & 0.00 & 0.00 & -140.28 & 83.07 \\ 2.27 & 33.98 & -84.58 & 135.93 & -12.84 & -41.54 & 0.00 & 0.00 & 67.97 & -128.38 \\ 0.00 & 0.00 & 52.04 & -12.84 & 160.47 & 84.58 & -32.22 & 11.33 & -140.28 & -83.07 \\ 0.00 & 0.00 & 12.84 & -41.54 & 84.58 & 135.93 & -2.27 & 33.98 & -67.97 & -128.38 \\ 19.63 & 0.76 & 0.00 & 0.00 & -32.22 & -2.27 & 98.68 & -51.35 & -97.17 & 52.86 \\ -0.76 & -26.43 & 0.00 & 0.00 & 11.33 & 33.98 & -51.35 & 108.75 & 67.97 & -116.30 \\ -97.17 & -67.97 & -140.28 & 67.97 & -140.28 & -67.97 & -97.17 & 67.97 & 503.83 & 0.00 \\ -52.86 & -116.30 & 83.07 & -128.38 & -83.07 & -128.38 & 52.86 & -116.30 & 0.00 & 489.36 \end{bmatrix} \frac{\text{lb}}{\text{in.}}$$

Element-wise relative error.

$$E_G = \begin{bmatrix} 2.80 & 0 & 1.87 & 0.44 & 0 & 0 & 3.73 & 1.33 & 1.41 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 & 1.33 & 0 & 0.01 & 0 \\ 1.87 & 0 & 1.35 & 0.11 & 0.91 & 0 & 0 & 0 & 0.78 & 0 \\ 0.44 & 0 & 0.01 & 0.01 & 0 & 0 & 0 & 0 & 0.01 & 0.02 \\ 0 & 0 & 0.91 & 0 & 1.35 & 0.01 & 1.87 & 0 & 0.78 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0.01 & 0.44 & 0 & 0.01 & 0.02 \\ 3.73 & 1.33 & 0 & 0 & 1.87 & 0.44 & 2.80 & 0 & 1.41 & 0 \\ 1.33 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01 & 0.01 & 0 \\ 1.41 & 0.01 & 0.78 & 0.01 & 0.78 & 0.01 & 1.41 & 0.01 & 0.97 & 0 \\ 00 & 0 & 0.02 & 0 & 0.02 & 0 & 0 & 0 & 0 & \end{bmatrix} \%$$

The maximum relative error is 3.73%.

D Finite Element Mass Matrix Verification Details

Consider element 4 in the given validation problem. Given the matrix $\boldsymbol{N}$ as described in Appendix B, the exact mass matrix, with $\rho = 1000$, is given by

$$2000\pi r_c A \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{N}^T\boldsymbol{N}\mathrm{d}t\mathrm{d}t$$

Exact evaluation of this integral in Mathematica yeilds

$$M_4^E = 2000\pi \left(\frac{11}{12}\right)(0.0625) \begin{bmatrix} \frac{4}{9} & 0 & \frac{2}{9} & 0 & \frac{1}{3} & 0 \\ 0 & \frac{4}{9} & 0 & \frac{2}{9} & 0 & \frac{1}{3} \\ \frac{2}{9} & 0 & \frac{4}{9} & 0 & \frac{1}{3} & 0 \\ 0 & \frac{2}{9} & 0 & \frac{4}{9} & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{4}{3} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{4}{3} \end{bmatrix},$$

$$\approx \begin{bmatrix} 159.99 & 0 & 79.99 & 0 & 119.99 & 0 \\ 0 & 159.99 & 0 & 79.99 & 0 & 119.99 \\ 79.99 & 0 & 159.99 & 0 & 119.99 & 0 \\ 0 & 79.99 & 0 & 159.99 & 0 & 119.99 \\ 119.99 & 0 & 119.99 & 0 & 479.97 & 0 \\ 0 & 119.99 & 0 & 119.99 & 0 & 479.97 \end{bmatrix}.$$

The mass matrix produced by our implementation is

$$M_4^C = \begin{bmatrix} 159.99 & 0.00 & 79.99 & 0.00 & 119.99 & 0.00 \\ 0.00 & 159.99 & 0.00 & 79.99 & 0.00 & 119.99 \\ 79.99 & 0.00 & 159.99 & 0.00 & 119.99 & 0.00 \\ 0.00 & 79.99 & 0.00 & 159.99 & 0.00 & 119.99 \\ 119.99 & 0.00 & 119.99 & 0.00 & 479.97 & 0.00 \\ 0.00 & 119.99 & 0.00 & 119.99 & 0.00 & 479.97 \end{bmatrix}.$$

Quick inspection verifies that the mass matrix produced by our code is identitcal to the exact mass matrix within machine precision. This is because the 9 point Gaussian Quadrature rule is sufficiently high order to evaluate the polynomials resulting from the product $\boldsymbol{N}^T\boldsymbol{N}$ exactly.

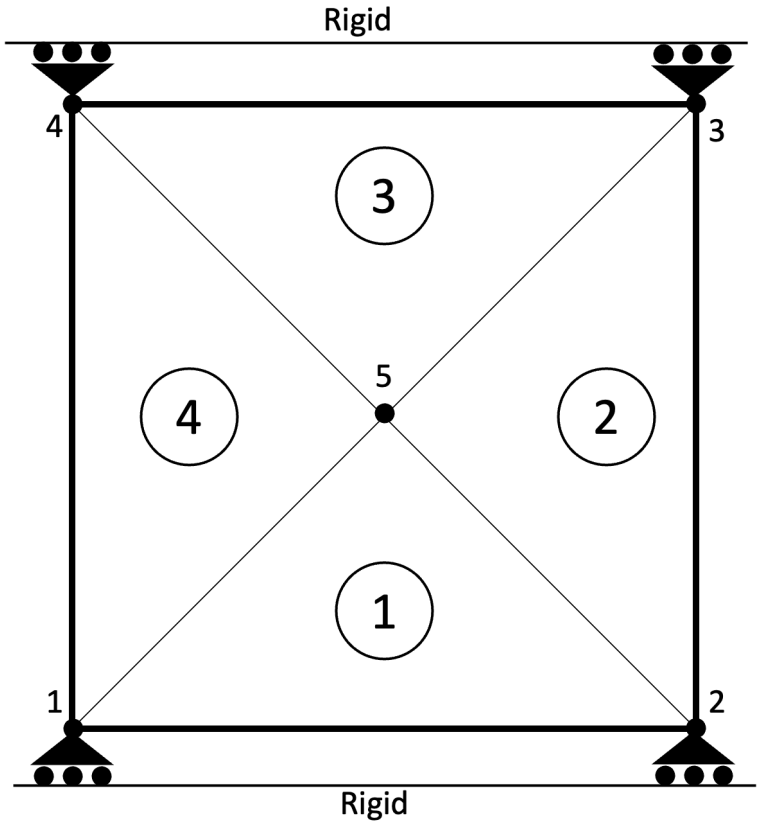

Figure 7: Constrained System

E Finite Element Validation

Figure 7 shows the constrained system that will be used to compute hoop stress. The rollers on the top and bottom are against rigid boundaries and allow motion in the horizontal, but not vertical direction. The constrained system of equations that results from this problem is

$$10^6 \begin{bmatrix} 1635.87 & -458.75 & 0 & 450.50 & -1640.96 & -910.00 \\ -458.75 & 1667.59 & 467.12 & 0 & -1662.11 & 925.11 \\ 0 & 467.12 & 1667.59 & -458.75 & -1662.11 & -925.11 \\ 450.50 & 0 & -458.75 & 1635.87 & -1640.96 & 910.00 \\ -1640.96 & -1662.11 & -1662.11 & -1640.96 & 6606.67 & 0 \\ -910.00 & 925.11 & -925.11 & 910.00 & 0 & 6606.41 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ v_5 \end{Bmatrix} = \pi(10)(0.25) \begin{Bmatrix} 100 \\ 0 \\ 0 \\ 100 \\ 0 \\ 0 \end{Bmatrix}.$$

Solution of this system yields the r and z displacements

$$\{u_1\ u_1\ u_3\ u_4\ u_5\ v_5\}^T = 1 \times 10^{-3}\{1.2415,\ 1.2415,\ 1.2415,\ 1.2415,\ 1.2415,\ 0\}^T.$$

Collecting the displacements for element 4 gives

$$\{u_1\ v_1\ u_4\ v_4\ u_5\ v_5\}^T = 1 \times 10^{-3}\{1.2415,\ 0,\ 1.2415,\ 0,\ 1.2415,\ 0\}^T.$$

Right multiplying the displacements by the matrix $\boldsymbol{B}$ yields the strains.

$$\begin{Bmatrix} \varepsilon_r \\ \varepsilon_z \\ \varepsilon_\theta \\ \tau_{rz} \end{Bmatrix} = 1 \times 10^{-3} \begin{bmatrix} -4 & 0 & -4 & 0 & 8 & 0 \\ 0 & 4 & 0 & -4 & 0 & 0 \\ 0.0784 & 0 & 0.00996 & 0 & 0.0112 & 0 \\ 4 & -4 & -4 & -4 & 0 & 8 \end{bmatrix} \begin{Bmatrix} 1.2415 \\ 0 \\ 1.2415 \\ 0 \\ 1.2415 \\ 0 \end{Bmatrix} = 1 \times 10^{-5} \begin{Bmatrix} -5.4612 \\ 0 \\ 12.3562 \\ 0 \end{Bmatrix}$$

Right multiplying this result by the stress strain matrix D yields the stresses.

$$\begin{Bmatrix} \sigma_r \\ \sigma_z \\ \sigma_\theta \\ \gamma_{rz} \end{Bmatrix} = 1 \times 10^{7} \begin{bmatrix} 4.0385 & 1.7308 & 1.7308 & 0 \\ 1.7308 & 4.0385 & 1.7308 & 0 \\ 1.7308 & 1.7308 & 4.0385 & 0 \\ 0 & 0 & 0 & 1.1538 \end{bmatrix} \begin{Bmatrix} \varepsilon_r \\ \varepsilon_z \\ \varepsilon_\theta \\ \tau_{rz} \end{Bmatrix} = \begin{Bmatrix} -66.9 \\ 1193 \\ 4044 \\ 0 \end{Bmatrix} \text{psi.}$$

The hoop stress is given by σ_θ and is equal to 4044 psi using this model. The theoretical value for hoop stress is given by the formula

$$\sigma_\theta = \frac{PD_m}{2t}$$

where P is the pressure, D_m is the average of the inner and outer diameters, and t is the wall thickness. For this problem, the theoretical hoop stress is

$$\sigma_\theta = \frac{(100)(20.25)}{(2)(0.25)} = 4050\text{psi.}$$

The relative error between the result of our finite element code with this theoretical value is 0.15%, validating this modeling approach.