Simulation of a Timpani Drum

Done by:

Kitty Li
Yifei Zhu

Advisors:

Brennan Sprinkle
Charles S. Peskin

December 22, 2022

Contents

1 Introduction

A percussion musical instrument makes sound by the vibration of its body, and the air inside of and around it amplify the sound. Since the sound of a plucked string instrument can be simulated by applying a wave equation on a string, it is reasonable to simulate the sound of a percussion instrument by applying a wave equation on its body.

To solve a partial differential equation on an irregular surface in 3D, we choose to use the Finite Element Method (FEM) to solve a wave equation on a meshed surface of a Timpani drum body. In this project, we want to first ignore all effects of the air, and then consider the damping of the oscillation. However, we will not consider any other effect of the air, such as the amplification effect.

2 Equations

The most important equation that is related to sound-producing is the wave equation. In our model, approximating a Timpani drum as a cylinder, we will be using a simple one dimensional wave equation,

$$u_{tt} = c^2 u_{xx}, \tag{1}$$

and a one dimensional wave equation with a damping term,

$$u_{tt} + k u_t = c^2 u_{xx} \tag{2}$$

where u can be thought of as the displacement of the vibration, and k is a damping constant chosen by us depending on how long we would want the wave to travel on the Timpani body.

3 Numerical Method

3.1 Mesh Generation

The simulation of wave equation requires the discrete approximation of a continuous surface in 3-d space. In specific, we need to generate a triangular mesh in the desired shape.

In our case, we idealize the shape of a timpani to be an unit cylinder and generate a triangular mesh to fit this shape. To represent the mesh points, we need to effectively parameterize the surface. Since a cylinder is essentially two disks as top and bottom, and one tube as the side surface, in our practice we parameterize these surfaces separately and manage to "glue" them together.
Parameterization of the tube:

$$u \in [0,1], v \in [0,1] \Rightarrow x = \cos(2\pi u), y = \sin(2\pi u), z = 2v - 1$$

Parameterization of the disk:

$$u \in [0,1], v \in [0,1] \Rightarrow x = v\cos(2\pi u), y = v\sin(2\pi u), z = \pm 1$$

Such parametrization is easily done using basic knowledge about multivariate calculus and geometric processing. Then, by adjusting the partition in u and v, we could effectively adjust the fineness of the whole mesh.
In standard approach, we could "glue" two meshes together by creating a degenerate triangle connecting any pair of mesh points from two meshes. A degenerate triangle, in this case, is a triangle such that at least two of the points are identical. Usually such connecting won't impact any physical or numerical simulation relying on those meshes. But unfortunately, the FEM program we used in the next part doesn't allow degenerate triangles. So in practice we just connect together the tube and the disks manually by setting up links and triangles between points on the boundary.

3.2 Finite Element Method (FEM)

The reason why we chose to use FEM is use a discrete version of the Laplacian operator for triangular meshed surfaces, converting a partial differential equation to a linear problem. The derivation of a discrete Laplacian that is from the Stanford graphics course[1] will be briefly introduced here.

We first look at a standard Poisson problem,

$$\triangle u = f \tag{3}$$

Since it is hard to directly discretize the Poisson equation, we first consider a different way of presenting this equation with the concept of a weak solution, integrating both sides after mutiplying a test function ϕ.

$$\int_M \phi \triangle u \, dA \; = \int_M \phi f \, dA \tag{4}$$

We want to look at only one small part of the surface at a time, and therefore we can choose ϕ to be a hat function that equals 1 at associated vertex v_i and equals 0 elsewhere.

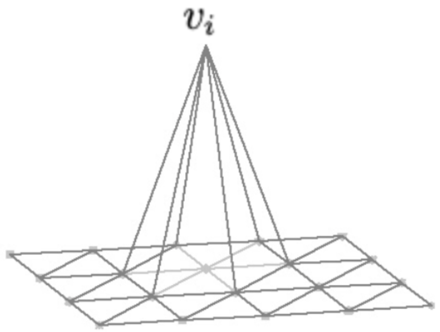

Figure 1: Associated vertex v_i

If we know the value of $u(x)$ on each vertex, we have $u(v_i) = a_i$. Then, we can approximate u by this summation,

$$u(x) = \sum_i a_i \phi_i(x) \tag{5}$$

and similarly, if we set $f(v_i) = b_i$, we can approximate f by this summation,

$$f(x) = \sum_i b_i \phi_i(x) \tag{6}$$

Therefore, when we look at the left hand side of Eq.4, we can use integration by parts to eventually arrive at a single array vector $\vec{a}$ times a list of unknown coefficients, which we define as matrix L.

$$\begin{aligned}\int_M \phi_i \triangle u \, dA &= -\int_M \nabla\phi_i \nabla u \, dA \\ &= -\int_M \nabla\phi_i \nabla(\sum_j a_j\phi_j) \, dA \\ &= -\sum_j a_j \int_M \nabla\phi_i \nabla\phi_j \, dA \\ &= \sum_j a_j L_{ij} \\ &= L\vec{a}\end{aligned}$$

Similarly, we have $M\vec{b}$ if we do the same calculation to the right hand side and denote the unknown coefficients as matrix M. Then, we arrive at the linear problem $L\vec{a} = M\vec{b}$.

Here, we used cotangent formula for Laplacian matrix L,

$$L_{ij} = \begin{cases} \frac{1}{2}\sum_{i\ j}(\cot\alpha_j + \cot\beta_j), & \text{if } i = j \\ -\frac{1}{2}(\cot\alpha_j + \cot\beta_j), & \text{if } i \sim j \\ 0, & \text{otherwise} \end{cases}$$

The main idea of the cotangent formula is to only look at the sections that are close enough since when we do the integration, sections that are too far away integrates to 0. So we are only looking at neighboring sections when vertices i and j are adjacent or when $i = j$. When $i = j$, we are considering the case when the two test functions are defined on the same vertex v_i. The same logic applies to the formula we use for calculating the mass matrix M.

$$M_{ij} = \begin{cases} \frac{\text{one-ring area}}{6}, & \text{if } i = j \\ \frac{\text{adjacent area}}{12}, & \text{if } i \neq j \end{cases}$$

Using the above formulas, we are able to compute the Laplacian matrix and the Mass matrix, or the discrete laplacian.

To solve a wave equation, we first rewrite the u_{tt} term as v_t, which then allows us to arrive at a new equation,

$$v_t = c^2 \triangle u, \tag{7}$$

or with damping,

$$v_t = c^2 \triangle u - kv, \tag{8}$$

in which the Δ is equivalent to $M^{-1}L$.

Numerically, we have a system that is to be updated every time step dt,

$$\begin{aligned} v^{n+1} - v^n &= c^2 \triangle u^n dt, \\ u^{n+1} - u^n &= v^{n+1} dt. \end{aligned}$$

or with damping,

$$\begin{aligned} v^{n+1} - v^n &= (c^2 \triangle u^n - kv^n) dt, \\ u^{n+1} - u^n &= v^{n+1} dt. \end{aligned}$$

4 Validation

4.1 Solving spherical harmonics on a unit sphere

We want to first validate our FEM code for finding the Laplacian matrix and Mass matrix. The easiest way to test the Laplacian and Mass matrices is to solve spherical harmonics on a spherical mesh[2] since applying a Laplace operator on them results in a constant times itself.
To be specific, for a given value of l, the Laplace's spherical harmonics $Y_\ell^m(\theta, \varphi)$ has $2l+1$ independent angular solutions:

$$Y_\ell^m(\theta, \varphi) = N e^{im\varphi} P_\ell^m(\cos\theta) \tag{9}$$

where m takes values from $-l$ to l. These solutions are a product of trigonometric functions which satisfies

$$r^2 \nabla^2 Y_\ell^m(\theta, \varphi) = -\ell(\ell+1) Y_\ell^m(\theta, \varphi). \tag{10}$$

In the unit sphere case, $r = 1$, then the Laplacian of spherical harmonics is just a constant $-l(l+1)$ times itself. Therefore it would be very convenient to compare the analytic results and one generated by the FEM program.

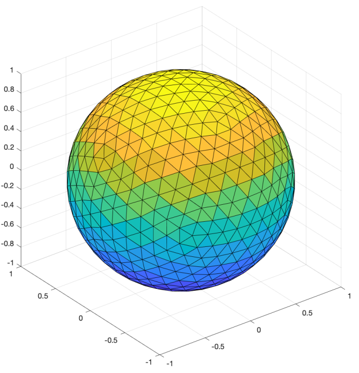

Figure 2: Meshed surface of a sphere

Using the code to compute the Laplcian and Mass matrices and plugging in both a and b in the system $L\vec{a} = M\vec{b}$, which are the known approximations for the spherical harmonics, we are able to plot the loglog plot using the error and the number of triangles on the meshed surface.

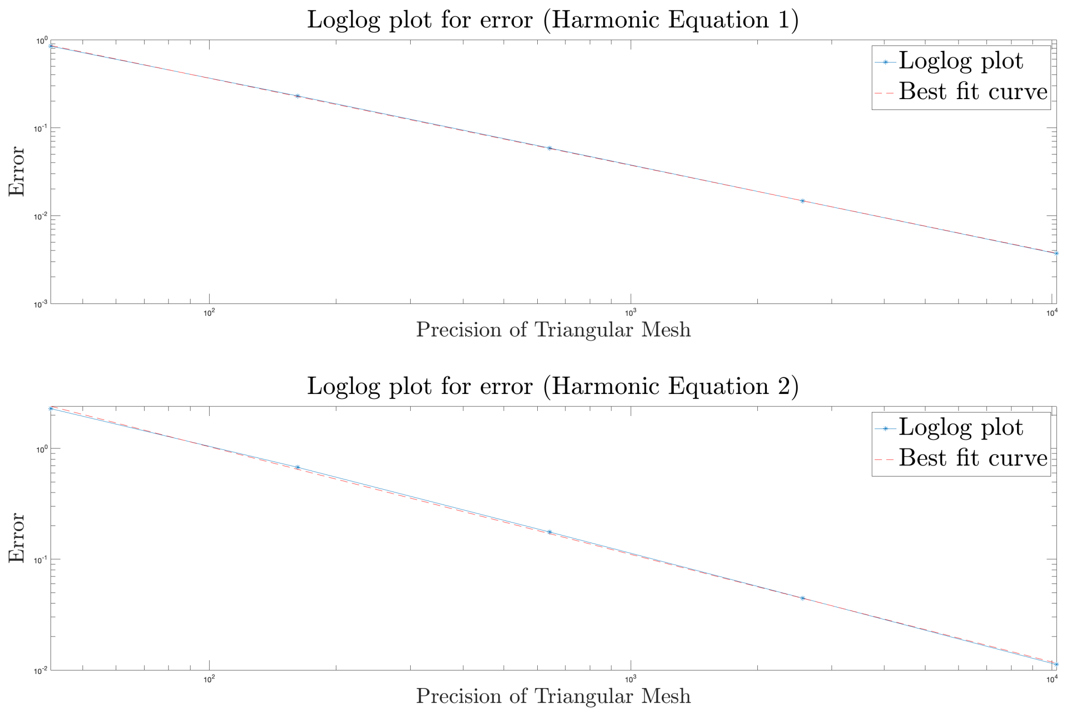

Figure 3: Loglog plot of error

From the plot, we can see that there is a definite power relationship between the number of triangles and the error, which is already known to be true for the finite element method. Therefore, we can conclude that the code for calculating Laplacian and Mass matrices work reasonably fine.

4.2 Waveform comparison

Besides validating the Laplacian and Mass matrices, we also compared the simulation result, which will be talked about with more details in the next section, with the actual Timpani Drum, both playing a $C2$ note. The actual Timpani Drum sound source is from Logic Pro X.

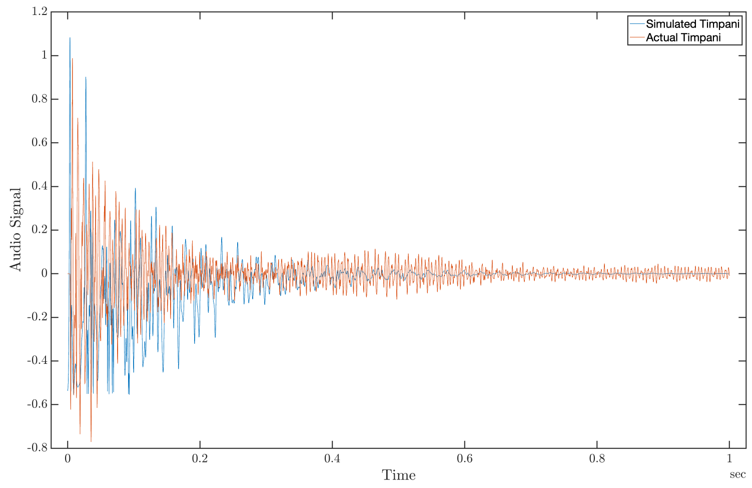

Figure 4: Simulated C2 note versus actual C2 note

After modifying the volume of the simulated $C2$ note, we can see that the simulated note and the actual note agree with each other greatly.

5 Results and Discussion

5.1 Video

The video of our final simulation can be seen here(with damping) and here(without damping). We could see from the video that the undamped case vibrates with nearly constant range, and the damped case becomes still around $t = 0.4$. This difference conforms to our physical instinct and could explain the decay in loudness and waveform range of the sound generated by damped timpani.

5.2 Audio

The parameter c in the wave equation determines the wave frequency and thus the pitch we hear from the waveform. When the mesh is the same, the wavelength doesn't change and we can get sound of desired frequency by tuning the value of c. With Kitty's perfect pitch, we are able to identify a note's frequency just by hearing it and looking up the frequency table. Therefore we can get a precise control of c to make music.

We manage to properly change the c in the wave equation to create musical notes of different pitches (frequencies). By properly arranging their lengths and rhythms we created a simulated version of *Twinkle twinkle little star*. For comparison, Kitty played the same melody using the sampled (recorded) sound of a real timpani in the digital audio workstation software Logic Pro X. The audio is here. Comparing these two audios, we could notice the significant difference in their timbres. Timbre of the real timpani clearly comes from the air resonance within its volume. But our simulated timpani doesn't model the air inside directly. So the simulated audio sounds a little "flatter", or "thinner", as if a vibrating rubber band.

5.3 Parameter study on mesh precision

When we introduce the cylinder mesh we generated, we mentioned that by changing the number of partitions in u and v, we could change the mesh precision. Then, we're interested in the problem of relationship between audio output and precision of the mesh that it was run on. Keeping other parameters equal, including the wave speed c and time-step dt, we run several simulations using different mesh precision. The merged audio output file is here(Please turn down your volume as this could be loud). Here the first audio is generated with the coarsest mesh while the last audio is with the finest.

The waveform of this merged sound is shown in the following graph:

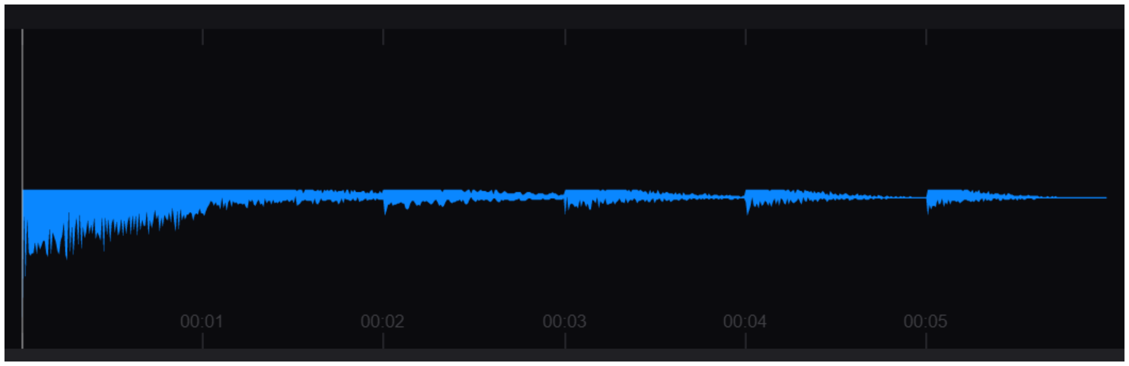

Figure 5: Waveform of audio output from different mesh precision. Horizontal axis: time; Vertical axis: audio signal

From the audio and the corresponding waveform, it's clear that the first audio generated by the coarsest mesh has the most noise in it, just like a synthesizer. It is also the loudest one. As we take finer mesh, the audio sounds more like a real timpani drum than a synthesizer. We believe that this phenomenon shows that the more mesh points we have, the closer our simulation is to the analytic solution.

6 Summary and Conclusions

In this project we successfully simulated a timpani by running the wave equations on a 2-dimensional cylindrical surface in 3-dimensional space. We generated the mesh of the cylindrical surface by parametric processing and ran a finite element method simulation based on the mesh for the propagation of the wave equations. By running the spherical harmonic function on the unit sphere and comparing it with the analytical value, we observe that the error decreases with increasing mesh precision, thus successfully verifying the correctness of the FEM program. Finally, we fine-tuned the value of the parameter c, which represents the wave speed in the program. By trying different values of the parameter c, we successfully generated a melody with the simulated timpani.

7 Bibliography

[1] Lu, T, Discrete Laplacian, from https://graphics.stanford.edu/courses/cs468-13-spring/assets/lecture12-lu.pdf

[2] wil. (n.d.). Icosphere. MathWorks. Retrieved October 27, 2022, from https://www.mathworks.com/matlabcentral/fileexchange/50105-icosphere

[3] Chen, L, INTRODUCTION TO FINITE ELEMENT METHODS

[4] Tahvanainen, Henna Matsuda, Hideto Shinoda, Ryo, Numerical simulation of the acoustic guitar for virtual prototyping, (2019).

Appendix for Video and Audio Files

Video files
Audio files