Physical Audio Modeling

Modeling & Simulation Literature Review

Karl Hiner, Benjamin Wilfong

March 2023

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 [7], 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 [2].

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 [2].

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 [6]. 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 [4]. 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 $\mathbf{x(}\boldsymbol{t}\mathbf{)}$ is the vector of displacements of each node, $\mathbf{M}$ and $\mathbf{K}$ are the finite element mass and stiffness matrices respectively, and $\mathbf{f(}\boldsymbol{t}\mathbf{)}$ 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 [5]. Assume that solutions to this system have the form $u_i(t) = \mathbf{U}_i e^{j\omega_i t}$ where $\mathbf{U}_i \in \mathbb{R}^{2n}$ and $\omega_i \in \mathbb{R}$. 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*[8] 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" [3].
- 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.

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] 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.

[3] 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).

[4] 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).

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

[6] 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.

[7] 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).

[8] 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.