Physical Audio Modeling of Passive Rigid Bodies

Real-time Interactive Audio Generation Using Linear Modal Analysis

Karl Hiner
khiner6@gatech.edu

May 3, 2024

1 Abstract

An impact on the surface of a passive rigid body produces mechanical vibrations that propagate through the material. As the excited object vibrates, its surface interacts with particles in the surrounding medium, producing longitudinal pressure waves that can be heard by a nearby listener. The nature of the audio signal heard by such a listener is determined by the dynamics of these pressure waves, and these dynamics depend on many factors — the material and geometry of the object, the excitation instrument and impact location, objects and boundaries in the environment, and even the geometry of the listener [15]. Several highly general methods are available to accurately model the dynamics of acoustic radiation in the presence of complex geometries [17] and dynamic, granular boundary conditions [11]. However, these methods are computationally intractable for settings such as video games and virtual realities, which require real-time execution on consumer hardware. This work investigates computationally efficient methods for estimating the time-varying sound pressure level at a point in a 3D environment with passive rigid bodies radiating acoustic waves in response to impact forces on their surface. By comparing results with a recently released large-scale dataset of real object impact sounds [2], we show that the well-known and efficient method of Linear Modal Analysis achieves qualitatively satisfactory results but often fails to accurately capture perceptually relevant vibrational modes using available material and geometry data. Additionally, we provide an interactive 3D application for generating modal audio models from surface meshes and evaluating the excitation audio against real-world recordings. While our quantitative evaluation is preliminary, we hope this study and accompanying application aid audio modeling researchers and interactive experience designers in developing efficient and physically accurate rigid body audio models.

2 Introduction

2.1 Acoustic Radiation

This work focuses on the dynamics of mechanical waves. However, since our broad goal is to accurately model audio in interactive environements, an understanding of acoustic radiation is fundamental. Acoustic wave propagation can be described by a scalar pressure field $P(\mathbf{x}, t)$, where $\mathbf{x}$ is a 3D position vector, and $\mathbf{v}(\mathbf{x}, t)$ is the vector velocity field, representing the velocity of the medium's particles as pressure waves propagate through it. The key material parameters are the speed of sound c_a and the medium's particle density ρ, which affects its acoustic impedance.

The governing equations of acoustic radiation can expressed as coupled first-order differential equations relating the temporal derivative of the pressure field to the spatial derivative of the velocity field:

$$\begin{aligned}\frac{\partial P}{\partial t} &= -\rho c_a^2 \nabla \cdot \mathbf{v}, \\ \frac{\partial \mathbf{v}}{\partial t} &= -\frac{1}{\rho} \nabla P\end{aligned}$$

From these equations, the wave equation can be derived (as detailed in *Schneider, 2010* [11]):

$$\nabla^2 P - \frac{1}{c_a^2} \frac{\partial^2 P}{\partial t^2} = 0$$

A solution particularly relevant for modeling sound wave dynamics is the harmonic plane wave, described by:

$$P(\mathbf{x}, t) = P_0 e^{j(\omega t - \mathbf{k} \cdot \mathbf{x})},$$

where P_0 is the amplitude of the pressure wave, ω is the angular frequency, and $\mathbf{k}$ is the wave propagation vector.

Linearity only holds generally in *small-signal acoustics*, when the amplitudes of the pressure waves are small enough that the dynamics do not significantly affect the properties of the propagating medium. Throughout this work, we assume small-scale acoustic dynamics. [1]

2.2 Linear Modal Analysis

Linear Modal Analysis (LMA) is a computational technique used to model the vibrational dynamics of structures in response to external forces, and is based on solving the linear elastodynamic equation,

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f},$$

where $\mathbf{u}(t)$ is the displacement vector of N mesh nodes, $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are the mass, damping, and stiffness matrices, and $\mathbf{f}(t)$ is the external force vector.

[1] This is a physical justification for the ubiquitous practice of modeling the combined sound pressure of multiple audio waves incident at a given point by simply summing them.

For this study, we adopt the common *Rayleigh Damping* assumption that the damping of the system is fully characterized by a linear combination of its inertial and elastic effects. Under this assumption, we express the damping matrix as

$$\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K},$$

where α and β are scalars.

LMA assumes solutions to the dynamic problem have the form $\mathbf{u}(t) = \mathbf{\Phi}\mathbf{q}(t)$, where $\mathbf{\Phi}$ is the modal matrix with $\mathbf{\Phi}_i \in \mathbb{R}^{3N}$ representing the ith "mode shape", and $\mathbf{q}(t)$ the corresponding modal amplitudes, assumed to be exponentially-decaying sinusoids. Substituting yields the eigenvalue problem

$$\mathbf{K\Phi} = \mathbf{\Lambda M \Phi}, \tag{1}$$

where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues.

This decomposition facilitates decoupling the global motion into individual modal contributions, which can be computed (and cached) independently and combined by linear superposition. The system of decoupled ordinary differential equations can be written as

$$\ddot{q}_i + 2\xi_i\omega_i\dot{q}_i + \omega_i^2 q_i = \frac{Q_i}{m_i}, \quad i = 1..N,$$

with

- $m_i = \text{diag}(\mathbf{\Phi}^T\mathbf{M\Phi})_i$, $k_i = \text{diag}(\mathbf{\Phi}^T\mathbf{K\Phi})_i$, $Q_i = (\mathbf{\Phi}^T\mathbf{f})_i$: The mass, stiffness, and force for mode i
- $\omega_i = \sqrt{\mathbf{\Lambda}_i} = \sqrt{k_i/m_i}$: The *undamped* frequency of vibration, in radians
- $\xi_i = \frac{c_i}{2m_i\omega_i} = \frac{1}{2}\left(\frac{\alpha}{\omega_i} + \beta\omega_i\right)$: The (dimensionless) modal damping factor

For a system starting from rest at $t = 0$, the dynamic response of each mode i due to forcing $\mathbf{Q}(t)$ is given by the convolution integral,

$$q_i(t) = \int_0^t e^{-\xi_i\omega_i(t-\tau)} \sin\omega_{di}(t-\tau)\frac{Q_i(\tau)}{m_i\omega_{di}}\, d\tau,$$

where $\omega_{di} = \omega_i\sqrt{1-\xi_i^2}$ is the (observed) *damped* frequency of vibration.

In other words, Linear Modal Analysis is based on the assumption that the surface vibrations of an object are a linear combination of independent oscillations at different frequencies and amplitudes. This largely holds true for highly rigid (non-deformable) objects with low-force impacts, especially since low-amplitude nonlinear surface effects may be imperceptible after "smearing out" during radiation. However, many easily perceptible nonlinear phenomena, such as sharp transients following an impact, are commonplace in real-world interactions, and are not modelled by this method.

3 Methods

3.1 Project overview

This work is a continuation of a prior project, *Mesh2Audio* [5], which generates Faust DSP code implementing a modal audio model from a given surface mesh and material properties (using a process detailed below). *Mesh2Audio* builds on the work of Michon, Martin & Smith in their *Mesh2Faust* project [7], with the following major contributions:

- Provides an OpenGL-based interface for axisymmetric mesh generation, immediate-mode Faust DSP parameter interface, interactive vertex excitation, and more.
- Implements a complete, independent 2D axisymmetric FEM model.
- Dramatically speeds up Finite Element and eigenvalue estimation.
- Many other improvements to the original *Mesh2Faust* work, which were contributed to the Faust project.

The present work further extends on *Mesh2Audio* with the following major improvements and additions:

- A complete rewrite of the interactive application from OpenGL to Vulkan with improved UI and UX.
- A comprehensive integrated *RealImpact* 3D dataset explorer supporting interactive listener position selection, playback of impact sounds at recorded vertices, and comparison of real-world impact recordings with modal model impacts.
- Add damping to the modal FEM model, resulting in major improvements to physical realism, especially for objects with short resonance durations.
- Extensive improvements and additions to mesh editing and debugging capabilities.
- Ability to create mesh primitives (cube, sphere, torus, cylinder, etc.) and generate modal audio models from them.
- Improvements to eigenvalue estimation accuracy.

In addition to the project code, we contributed the above-mentioned additions and improvements to the Faust project.

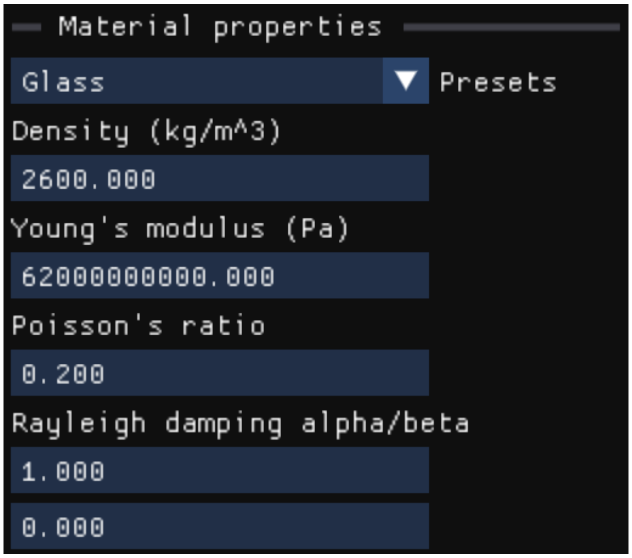
3.2 Finite Element Method

We use the tetgen library [12] to convert triangular surface meshes into tetrahedral volumetric meshes, and the Vega FEM library [6] to compute the mass and stiffness matrices from tetrahedral volumes and material properties.

We adopt the material properties (1) presented in Table 4 of Jui-Hsien Wang and Doug James' KleinPAT paper [16].

Table 1: Material parameters: ρ: density; E: Young's Modulus; ν: Poisson ratio; α and β: Rayleigh damping parameters. All parameters are in SI units.

Materials	ρ (kg/m^3)	E (Pa)	ν	α	β (1/K)
Ceramic	2700	7.2E10	0.19	6	1E-7
Glass	2600	6.2E10	0.20	1	1E-7
Wood	750	1.1E10	0.25	60	2E-6
Plastic	1070	1.4E9	0.35	30	1E-6
Iron	8000	2.1E11	0.28	5	1E-7
Polycarbonate	1190	2.4E9	0.37	0.5	4E-7
Steel	7850	2.0E11	0.29	5	3E-8



Material properties

Glass ▼ Presets

Density (kg/m^3)

2600.000

Young's modulus (Pa)

62000000000.000

Poisson's ratio

0.200

Rayleigh damping alpha/beta

1.000

0.000

Figure 1: Application UI section for material property configuration

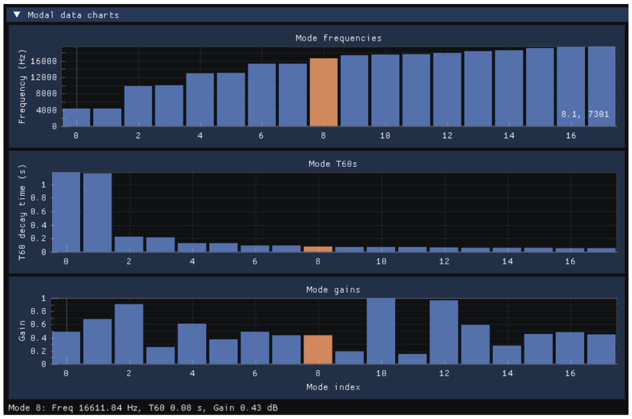
These properties are offered as presets in the application UI, and are automatically selected based on the object file name when the current selected mesh originates from a *RealImpact* object. Material properties are also configurable in the UI via text input fields 1

After computing the mass and stifness matrices, the system's eigenvalues and eigenvectors are estimated using the Spectra C++ library [9] to solve the elastodynamic eigenvalue problem (1) (using `SymGEigsShiftSolver` method, providing the lowest expected frequency of 20 Hz as the shift σ value).

Finally we compute the mode parameters from the estimated eigenvalues and eigenvectors (which are viewable in the UI for debugging purposes 2).

3.3 Integration (Audio Rendering)

As explained in the LMA section above, each resonating mode is modelled as an exponentially decaying sinusoid with an independent frequency, gain, and resonance duration. Rather than im-



```
Mode 8: Freq 16611.84 Hz, T60 0.08 s, Gain 0.43 dB
```

Figure 2: Application UI section for viewing all estimated modal parameters

plementing them as sine waves with exponential envelopes, using resonant bandpass filters allows any signal to excite the vertices (modes). Mode filters are implemented as a parallel bank of biquad filters (Fig. 1), each with transfer function (Fig. 2)

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

with $\alpha_1 = -2\tau\cos\omega,\ \alpha_2 = \tau^2,\ \omega = \frac{2\pi f}{f_s}$, and $\tau = 0.001^{\frac{1}{t_{60}}}$. Here, g is the mode gain, f the mode frequency in Hz, f_s is the sampling rate (always 48 kHz in this study), and t_{60} is the *T60 decay time* - the time it takes for the mode to decay by 60 dB from its initial amplitude.

Referring back to the *LMA* section above, the mode frequencies correspond to the eigenvalues (and the damping factor), the gains are determined from the eigenvectors and the excitation force, and the T60 values are derived from the mode damping factors.

The full audio pipeline is implemented with a generated Faust Faust (Functional Audio Stream) programming language [8] DSP program. Faust compiles the audio graph described in the DSP code to LLVM instructions (using LLVM JIT compilation), and executes the compiled graph sample-by-sample in real-time. Generated audio buffers are streamed to the native audio device using the miniaudio C library [10].

Shown below are components of the signal-flow block diagram produced by the `faust2svg` command from one such generated DSP program. The top-level signal-flow diagram (3) shows the impact "hammer" model component feeding into the modal model component.

The hammer is a simple model composed of a short pulse of white noise windowed by an attack-release envelope and smoothed by a lowpass filter, with "hardness" corresponding to pulse width and "size" corresponding to lowpass cutoff wavelength. [2] The hammer model is described by the following Faust expression (shown below in Figure 4):

[2] This is an adaptation of a method for piano hammer modeling used in commuted synthesis [14]. Despite its simplicity, the method has physical justification, but here is best regarded as a short injection of wideband energy at a mesh vertex, used to excite all modes.

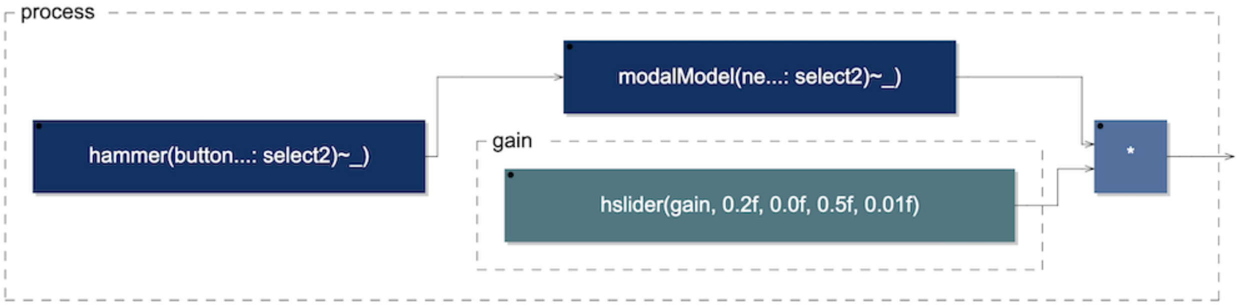

process
modalModel(ne...: select2)~_)
hammer(button...: select2)~_)
gain
hslider(gain, 0.2f, 0.0f, 0.5f, 0.01f)
*

Figure 3: Top-level Faust signal-flow diagram showing a hammer model exciting a modal model

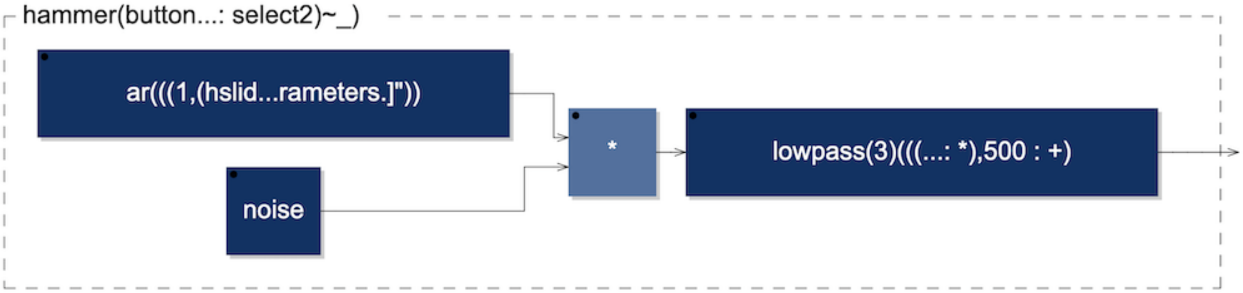

hammer(button...: select2)~_)
ar(((1,(hslid...rameters.]"))
*
lowpass(3)(((...: *),500 : +)
noise

Figure 4: Signal-flow diagram of the hammer model

```
hammer(trig, hardness, size) = en.ar(att, att, trig) * no.noise :
  fi.lowpass(3, ctoff) with {
    ctoff = (1-size)*9500 + 500;
    att = (1-hardness)*0.01 + 0.001;
  };
```

Figure 5 below shows the signal-flow diagram of the biquad filter bank, along with a detailed view of a single mode filter implementing the difference equation for the transfter function H(z) above.

A full example generated Faust DSP program is provided in the appendix A.

4 Results

While implementing the prior *Mesh2Audio* work (which focused on modeling bells), we were frustrated that, while the results *sounded* reasonably realistic, we had no way to evaluate the accuracy of the rendered audio. One of the main aspirations of this project was to address this issue. *RealImpact* was published shortly after completion of *Mesh2Audio*, introducing a generous dataset tailored to address exactly this derth of publicly available data. Producing such a dataset is challenging and resource intensive, and (to the knowledge of the authors) no dataset approaches its quality and quantity. It has been immensely valuable to have side-by-side comparisons available. However, in this project we were unable to find robust methods or metrics for quantitative evaluation, as we had originally hoped.

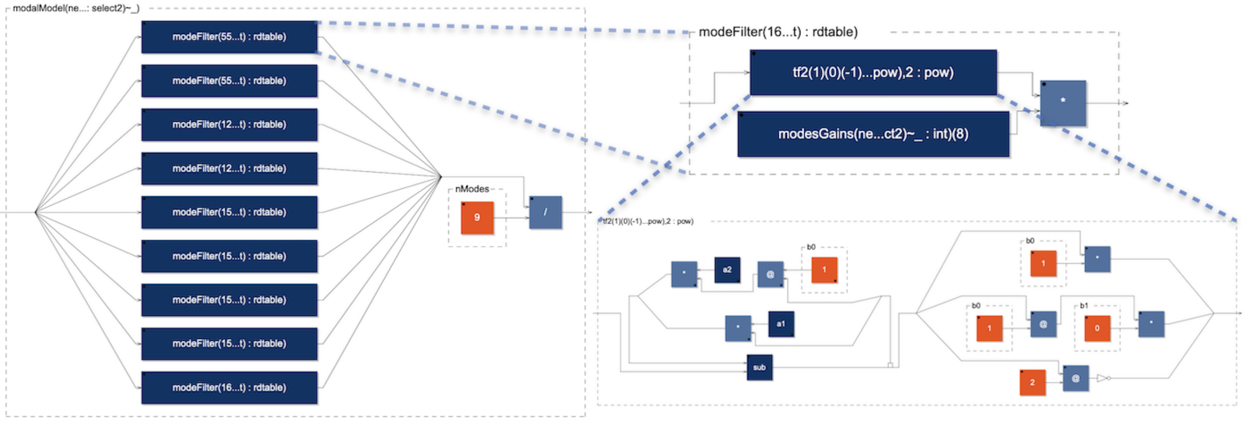

modalModel(ne...: select2)~_)
modeFilter(55...t) : rdtable)
modeFilter(55...t) : rdtable)
modeFilter(12...t) : rdtable)
modeFilter(12...t) : rdtable)
modeFilter(15...t) : rdtable)
modeFilter(15...t) : rdtable)
modeFilter(15...t) : rdtable)
modeFilter(15...t) : rdtable)
modeFilter(16...t) : rdtable)
nModes
9
/
modeFilter(16...t) : rdtable)
tf2(1)(0)(-1)...pow),2 : pow)
*
modesGains(ne...ct2)~_ : int)(8)
tf2(1)(0)(-1)...pow),2 : pow)
a2
@
b0
1
a1
sub
b0
1
b0
1
@
b1
0
2
@

Figure 5: Signal-flow diagram of the modal model. *Left:* Parallel bank of mode biquad filters. *Right:* Single mode filter implementing $H(z)$

An ideal evaluation metric would closely match subjective judgement of audio similarity. Many perceptual audio loss functions [3] and perceptually-motivated signal transforms [13] have been proposed, offering direct comparison of time or frequency-domain audio signals. However, given the specific context of modal audio synthesis, in which the entire signal is a superposition of pure sinuoid modes with known frequency, I wanted to take advantage of this known low-dimensional latent representation. There are several known methods [14] for estimating the resonant mode frequencies, gains, and decay times directly from spectra. If successfully applied, this would offer direct comparison of all estimated modal parameters with their estimated counterparts in the target real-world recorded impacts. However, these methods are complex and often unreliable. As a preliminary step, we investigated simply comparing the M estimated mode frequencies (ignoring the gains and decay times) with the top-M peak frequencies in the real-world impact spectra. However, we encountered several difficulties using even this simple approach:

- Many *RealImpact* recordings have a low signal-to-noise ratio, causing spurious false frequency maxima uncorrelated with the resonant modes.
- We observed it is very common for modal analysis to find modes in pairs with closely-spaced frequencies *degenerate modes* [1] (see 6).
- Degenerate modes pose one difficulty in matching frequencies between the generated and target spectra, but frequency-matching is a nontrivial problem, since the number of true modes is unknown, and it's not generally possible to distinguish between incorrect, missing, or spurious modes.
- Even a perfect model is only as accurate as its inputs. Generic material properties (1) may differ significantly from those of the real household objects, and the 3D scans may be missing important geometry.
- Finally, the impact recordings are affected by the recording environment and acoustic dynamics, while our modal model "records" a direct signal from the object's surface vibrations, making comparisons difficult even with a perfect model and input data.

Due to these challenges, we found that e.g. a straightforward RMSE of spectral peaks, was not only inaccurate in the absolute sense, but was not even reliable for relative comparisons. Thus, the

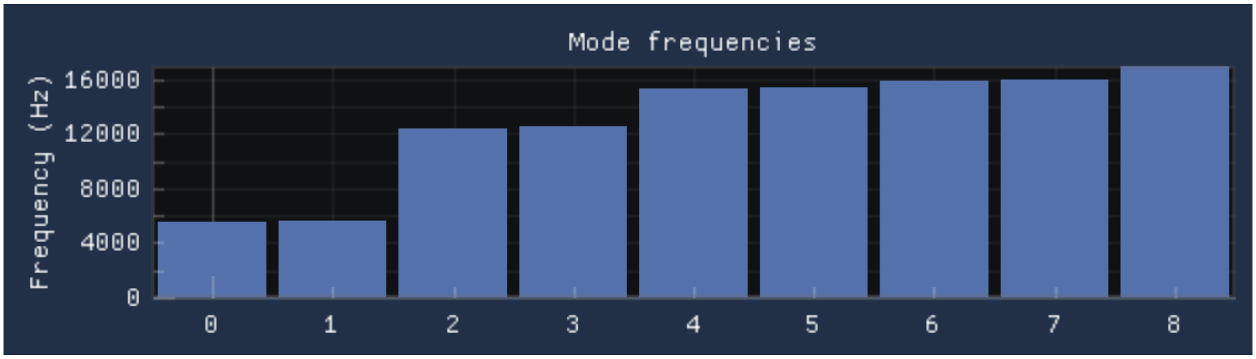


Figure 6: An example of estimated mode frequencies occuring in closely-spaced frequency pairs (*degenerate pairs*)

results we present here are mostly qualitative.

4.1 Subjective evaluation

Subjectively, the most important perceptual factor is the fundamental frequency. We observed that for almost all *RealImpact* objects, the raw FEM-estimated modal frequencies were shifted *up* in frequency. It could also be that low-freqency true modes are often simply not found. However, since simply scaling frequencies often results in reasonably close mappings to true modes in the low-frequency range, we believe the most likely explanation for this failure mode is inaccurate material parameters.

To remedy this issue, when generating a modal model for which a true recording is present, we scale all mode frequencies to set the lowest mode equal to the peak frequency in the true spectrum. This naive approach is not always accurate, but it addresses the majority of mismatches.

Another common failure mode is underestimating damping. The model, as implemented, seems to have a strong bias for resonating much longer than real-world objects, across modes. The mismatch is not extreme or consistent enough to strongly indicate a fundamental error in calculation or scaling, although it is certainly possible.

4.2 Quantitative and qualitative evaluation

The table below shows spectrum comparisons between synthesized and real-world recorded spectra 2 in response to "striking" various meshes at the same position (rounded to the resolution of the 3D-scanned meshes). The cylinders shown in the images represent recorded microphone positions, but all real-world spectra shown are from recorded from a single microphone centered near the impacted object, and the modal audio model does not implement any audio wave radiation modeling. (Again, all modal audio samples are generated by extracting estimated surface vibrations, as if recorded from a contact microphone.)

The spectra are measured from 30 frames to 3,000 frames (1/16 of a second) after impact. The 30-frame start buffer provides a short time for nonlinear contributions to decay, and the duration is long enough to accurately resolve modes as low as 32 Hz (well below the expected fundamental vibrational mode of these small household objects). A Blackman–Harris window is applied to the audio signal before applying the FFT.

It is clear from brief examination of the spectra that the synthesized results are far from quantitatively accurate reconstructions of the target impact responses. Much of the discrepency is due to high noise levels in the real-world recordings (while the synthesized results, are composed of pure sinusoidal components). But there are clearly prominent and perceptually relevant resonant modes that are unaccounted for in the synthesized audio.

The README of the project GitHub repo contains links to the corresponding audio samples. Although the synthesized samples are by no means perceptually identical to the real-world recordings, I find many to be surprisingly close, given all the inherent issues outlined above.

5 Summary and conclusions

Overall, these results show that Linear Modal Analysis/Synthesis is an effective and highly efficient method for physically plausible audio synthesis of rigid body. However, in addition to the many issues outlined above, the linear assumption has perceptible impact on physical accuracy, especially when few modes are estimated.

These experiments seem to indicate that it is possible to acheive an overall impression of perceived audio similarity while failing to achieve close spectral alignment. This is testament to the perceptual importance of non-spectral aspects such as fundamental frequency and dynamic envelope, and underlines the crucial importance of developing suitable datasets. As immersive environments such as augmented and virtual realities become increasingly capable and visual rendering fidelity improves, rigourous testing of simulated acoustics against the real-world will become increasingly important to ensure that acoustic realism keeps pace.

Continuing to develop real-world audio datasets alongside model improvements will also provide opportunities for data-driven audio generation models, for example by nudging fully differentiable physically-informed models [4] [18] towards natural audio manifolds.

References

[1] D Afolabi. Modal Interaction in Linear Dynamic Systems Near Degenerate Modes. November 1991.

[2] Samuel Clarke, Ruohan Gao, Mason Wang, Mark Rau, Julia Xu, Jui-Hsien Wang, Doug L. James, and Jiajun Wu. Realimpact: A dataset of impact sound fields for real objects, 2023.

[3] Dan Elbaz and Michael Zibulevsky. Perceptual audio loss function for deep learning, August 2017. arXiv:1708.05987 [cs].

[4] Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: Differentiable Digital Signal Processing. September 2019.

[5] Karl Hiner and Ben Wilfong. Mesh2audio, December 2023.

[6] Daniel Schroeder: Jernej Barbič, Fun Shing Sin. Vega fem library, 2012.

[7] Romain Michon, Sara R Martin, and Julius O Smith. Mesh2faust: A modal physical model generator for the faust programming language -application to bell modeling, 3 2021.

[8] Yann Orlarey, Dominique Fober, and Stéphane Letz. Faust - functional programming language for signal processing and sound synthesis, 2024.

[9] Yixuan Qiu. Spectra - c++ library for large scale eigenvalue problem, 2024.

[10] David Reid. miniaudio - an audio playback and capture library in a single source file, 2024.

[11] John B. Schneider. *Understanding the Finite-Difference Time-Domain Method*, chapter 12. 2010.

[12] Hang Si. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Transactions on Mathematical Software*, 41(2):11:1–11:36, February 2015.

[13] J.O. Smith and J.S. Abel. Bark and ERB bilinear transforms. *IEEE Transactions on Speech and Audio Processing*, 7(6):697–708, November 1999.

[14] Julius O. Smith III. *Physical Audio Signal Processing: for Musical Instruments and Audio Effects*. W3K Publishing, 2010.

[15] Daniel Starch. *Perimetry of the Localization of Sound*. Ph.d. dissertation, State University of Iowa, 1908.

[16] Jui-Hsien Wang and Doug L. James. KleinPAT: optimal mode conflation for time-domain precomputation of acoustic transfer. *ACM Transactions on Graphics*, 38(4), August 2019.

[17] Jui-Hsien Wang, Ante Qu, Timothy R. Langlois, and Doug L. James. Toward wave-based sound synthesis for computer animation. *ACM Trans. Graph.*, 37(4):109:1–109:16, 2018.

[18] Xin Wang, Shinji Takaki, and Junichi Yamagishi. Neural source-filter waveform models for statistical parametric speech synthesis, November 2019.

A Complete model Faust code

The following is a complete example Faust DSP code output, including the full modal model and parameter UI specification, excluding only tooltip help text. `mode...` floading point values are condensed with ellipses for brevity.

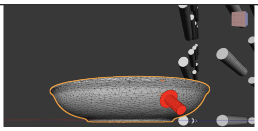
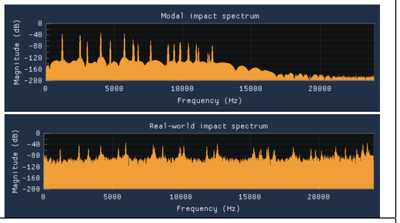
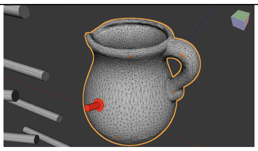
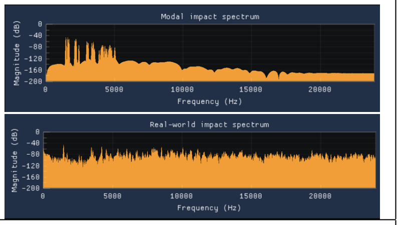
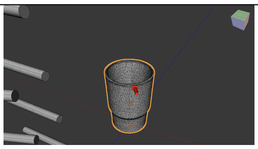
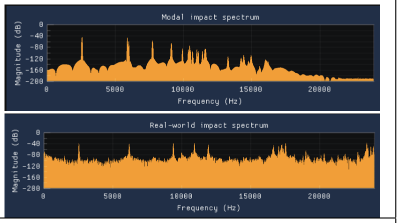
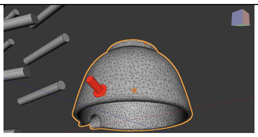
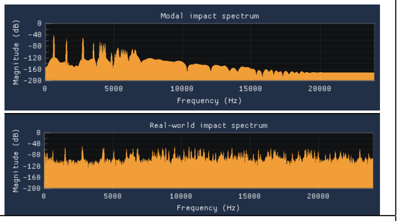
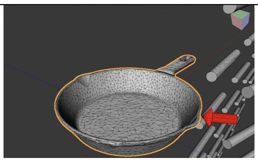
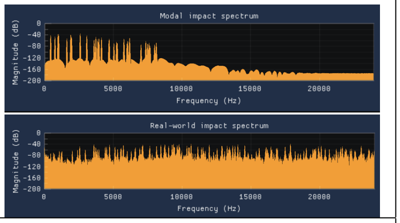
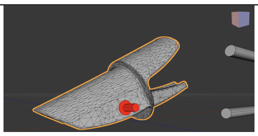
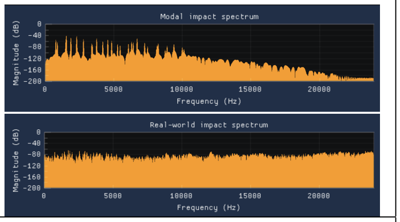
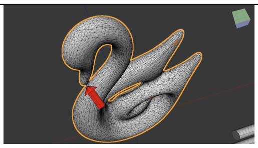
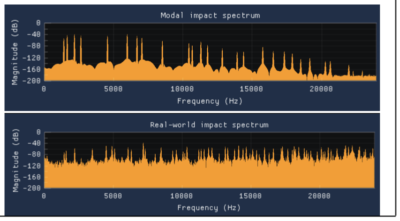
```
import("stdfaust.lib");

modalModel(freq,exPos,t60Scale) = _ <: par(
  mode,nModes,
  pm.modeFilter(modeFreqs(mode),modesT60s(mode),modesGains(int(exPos),mode))
) :> /(nModes)
with{
  nModes = 30;
  nExPos = 5;
  modeFreqRatios(n) = ba.take(n+1,(1,1.01168,2.25071,...));
  modeFreqs(mode) = freq*modeFreqRatios(mode);
  modesGains(p,n) = waveform{0.435705,0.726901,0.825358,...},
                    int(p*nModes+n) :
                    rdtable : select2(modeFreqs(n)<(ma.SR/2-1),0);
  modesT60s(n) = t60Scale * ba.take(n+1,(0.717945,0.701597,0.142672,...));
};
gate = button("Gate");
hammerHardness = hslider("Hammer hardness",0.9,0,1,0.01) : ba.sAndH(gate);
hammerSize = hslider("Hammer size",0.1,0,1,0.01) : ba.sAndH(gate);
gain = hslider("Gain[scale:log]",0.2,0,0.5,0.01);
freq = hslider("Frequency[scale:log]",2569,60,26000,1) : ba.sAndH(gate);
exPos = nentry("Excite index",2,0,4,1) : ba.sAndH(gate);
t60Scale = hslider("t60[scale:log]",1,0.1,10,0.01) : ba.sAndH(gate);

hammer(trig,hardness,size) = en.ar(att,att,trig)*no.noise :
  fi.lowpass(3,ctoff)
with{ ctoff = (1-size)*9500+500; att = (1-hardness)*0.01+0.001; };

process = hammer(gate,hammerHardness,hammerSize) :
  modalModel(freq,exPos,t60Scale)*gain;
```

Table 2: Comparison of Mesh, Modal Spectrum, and Real Spectrum for Various Objects

Object Name	Mesh	Synthesized/Real Spectra
Ceramic Koi Bowl		
Ceramic Pitcher		
Glass Cup		
Iron Mortar		
Iron Skillet		
Plastic Scoop		
Swan Small Ceramic		 Modal impact spectrum Magnitude (dB) 0 -40 -80 -120 -160 -200 0 5000 10000 15000 20000 Frequency (Hz) Real-world impact spectrum Magnitude (dB) 0 -40 -80 -120 -160 -200 0 5000 10000 15000 20000 Frequency (Hz)