Interactive Acoustic Transfer Approximation for Modal Sound

DINGZEYU LI, YUN FEI and CHANGXI ZHENG
Columbia University

Current linear modal sound models are tightly coupled with their frequency content. Both the modal vibration of object surfaces and the resulting sound radiation depend on the vibration frequency. Whenever the user tweaks modal parameters to adjust frequencies the modal sound model changes completely, necessitating expensive recomputation of modal vibration and sound radiation.

We propose a new method for interactive and continuous editing as well as exploration of modal sound parameters. We start by sampling a number of key points around a vibrating object, and then devise a compact, low-memory representation of frequency-varying acoustic transfer values at each key point using Prony series. We efficiently precompute these series using an adaptive frequency sweeping algorithm and volume-velocity-preserving mesh simplification. At runtime, we approximate acoustic transfer values using standard multipole expansions. Given user-specified modal frequencies, we solve a small least-squares system to estimate the expansion coefficients, and thereby quickly compute the resulting sound pressure value at arbitrary listening locations. We demonstrate the numerical accuracy, the runtime performance of our method on a set of comparisons and examples, and evaluate sound quality with user perception studies.

Categories and Subject Descriptors: I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*

General Terms: Numerical Methods, Interactive Design

Additional Key Words and Phrases: Modal Vibration, Acoustic Transfer, Prony's Method, Asymptotic Waveform Evaluation

1. INTRODUCTION

Modal sounds are widely used for synthesizing plausible solid-object sounds synchronized with computer-simulated animations (e.g., see [van den Doel et al. 2001; O'Brien et al. 2002; Zheng and James 2011; Ren et al. 2013]). The standard pipeline consists of two steps: (i) integration of surface vibrations followed by (ii) the computation of sound radiation. The former produces surface motions that are driven by external forces and vibrate at individual modal frequencies, while the latter accounts for wave phenomena such as diffraction and interference that can recognizably change the sound's timbre [James et al. 2006]. Both steps are closely coupled with modal vibration frequencies.

The most expensive step of generating a modal sound is computing sound radiation. For every vibration mode with a frequency ω, it can be computed by solving a frequency-domain wave equation, the *Helmholtz equation*,

$$\nabla^2 p(\boldsymbol{x}) + k^2 p(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \Omega, \tag{1}$$

where $p(\boldsymbol{x})$ is the acoustic transfer value at $\boldsymbol{x}$, $k = \omega/c$ is the wavenumber of the corresponding vibration mode, and c is the speed of sound. To accelerate this step, various methods have been devised [Ciscowski and Brebbia 1991]. In computer graphics, James et al. [2006] introduced precomputed acoustic transfer (PAT), wherein, after hours of precomputation of equivalent point sources, fast runtime transfer evaluation is achieved at any listening location. However, in all of these methods, the entire Helmholtz solution needs to be recomputed whenever the frequency ω is changed.

The tight dependence of modal sound radiation on its vibration frequencies as well as its expensive (pre-)computation of sound radiation give rise to many difficulties when one starts to tweak model sound parameters for desirable sound effects. In practice, tuning parameters is almost unavoidable, as the material parameters (e.g., the Young's modulus) are measured and tabulated in a range of values, and there are no generally accepted damping values [Adhikari and Woodhouse 2001]. Both kinds of parameters directly affect modal frequencies (see §3), which have been found critical for achieving desired sound characteristics [Klatzky et al. 2000]. Unfortunately, when the user changes these parameters and thus the frequencies, it becomes necessary to recompute the entire modal sound, leading to a rather inefficient parameter tuning cycle.

In light of this, we propose a new method that *decouples* surface modal vibration and acoustic transfer evaluation from modal vibration frequencies. It allows the user to freely change modal frequencies at runtime, and quickly synthesize resulting sounds at an arbitrary listening location.

At first glance, one simple approach to enable runtime editing of modal frequencies is to precompute individual modal sound models using a set of frequency samples, and rely on runtime interpolation to approximate with user-specified vibration frequencies. Unfortunately, it is unclear how we should interpolate between different models of modal sound. Moreover, the acoustic transfer $p(\boldsymbol{x})$ is highly oscillatory with respect to the vibration frequency (see §5 and Figure 6). As a result, such an approach will need lots of frequency samples for plausible runtime interpolation, causing a prohibitively long precomputation time and an overwhelming memory footprint.

In our approach, we first select a set of key positions around a vibrating object. At every key position we precompute a frequency-

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

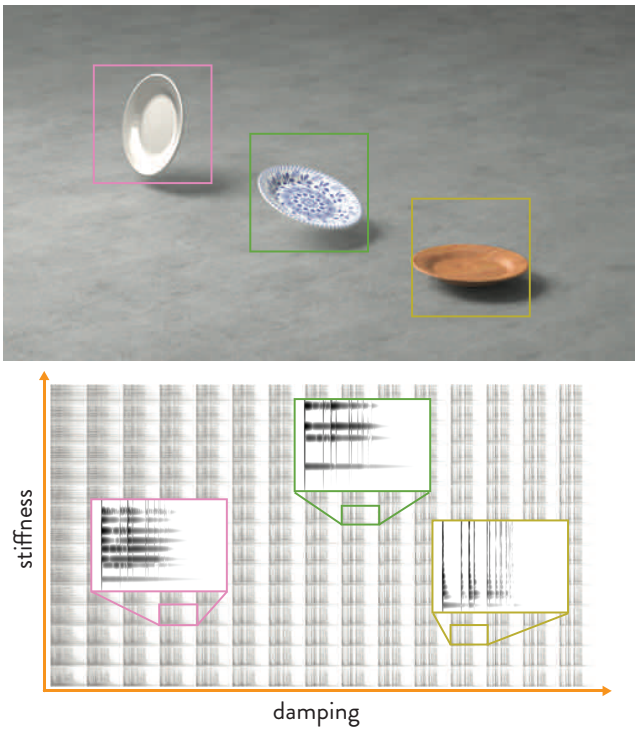


Fig. 1: **Parameter Exploration using our method:** With our precomputed information, we are able to explore the space of modal sound parameters at runtime, achieving numerous sound effects (bottom) synchronized with a physics-based animation. The three spectrograms highlighted in the colored boxes correspond to (left to right) metal, porcelain, and wood materials shown on the top. We refer to this link for all 225 sounds shown here. http://www.cs.columbia.edu/cg/interactivetransfer/

sweeping transfer function compactly represented using Prony series. We accelerate the precomputation by devising two key techniques: (i) we solve Helmholtz equations only at a few carefully selected frequency samples. And for each frequency sample, we build an asymptotic Padé approximant in frequency domain to evaluate transfer values at nearby frequencies. (ii) To speed up Helmholtz solves, we propose a frequency-adaptive mesh simplification algorithm; for low-frequency Helmholtz solves, we simplify the mesh more aggressively in exchange for larger computational speedup. At runtime, given a user-specified modal frequency, we represent the resulting transfer function at any spatial position using an acoustic multipole expansion. We first evaluate key-position transfer values, which are in turn used to construct a small least-squares problem to estimate the multipole expansion coefficients. An overview of our proposed method is shown in Figure 2.

With our proposed precomputation technique, we are able to generate various sound effects without expensive recomputation. This greatly eases the parameter tuning for different sound characteristics, whether one desires high-pitch long-ringing metal sounds or low-tone quickly damped wood-like sounds. We can explore the parameter space and quickly hear the sound feedback. The resulting sounds are almost identical to the ones using expensive full recomputation, both qualitatively and numerically.

Our technique also enables more control of sound characteristics for animators wishing to add synchronized modal sounds. We explore examples of nonlinear time-varying modal sound effects using user-guided nonphysical change of frequencies.

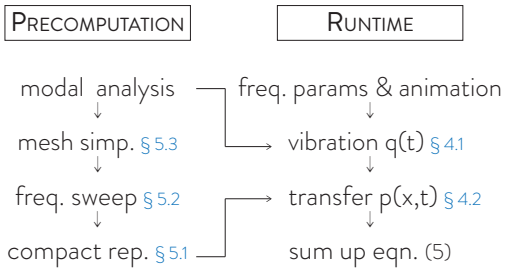


Fig. 2: **Overview:** given an input mesh, we efficiently perform the precomputation (left). At runtime, given a rigid animation, we can then quickly synthesize resulting rigid-body sounds with user-specified frequency parameters (right).

2. RELATED WORK

The computer graphics community has a long history of synthesizing synchronized sound effects for computer animation [Takala and Hahn 1992]. Modal sound models, based on linear modal analysis, have been widely used to generate plausible contact sounds [van den Doel and Pai 1996] synchronized with physics-based simulation. They are often constructed using recorded sounds [van den Doel et al. 2001; Ren et al. 2013] or linear modal analysis [Pentland and Williams 1989; O'Brien et al. 2002]. More recent development has used modal vibration for synthesizing rigid fracture sound [Zheng and James 2010], deformable sound [Zheng and James 2011], and fast interactive sound [Raghuvanshi and Lin 2006; Bonneel et al. 2008; Ren et al. 2010]. But all these methods are closely coupled with vibration frequencies, and none of them enable fast user editing of modal sound parameters with acoustic transfer functions.

The object's geometry can significantly affect modal sound radiation and change the sound's timbre in a spatially varying way, as demonstrated by James et al. [2006]. Unfortunately, computing sound radiation for all vibration modes is very expensive. To improve the performance, existing methods [James et al. 2006; Chadwick et al. 2009; Zheng and James 2011] assume fixed modal vibration frequencies, and precompute an efficient representation of acoustic transfer functions. The precomputation can take many hours. Once it is finished, the user can evaluate transfer values at an arbitrary position almost instantaneously. However, whenever the user adjusts vibration frequencies, the entire acoustic transfer representation needs to be recomputed. [Corbett et al. 2007] developed a system to acquire near-field acoustic transfer field from recorded sounds and synthesize spatial sounds interactively. Yet, this approach relied on an automated measuring system, in which the measurement is closely coupled with each object's specific geometry and material. Different from these approaches, our method only relies on precomputation and allows the user to change modal sound parameters at runtime and still enjoys the high quality of sound synthesis.

For fast estimation of sound wave radiation, O'Brien et al. [2001] adopted the *Rayleigh method* which assigns a monopole on each surface element and summed the sound radiation from all monopoles. This is essentially a first-order approximation of sound radiation, neglecting the fact that the shaped structure constitutes a scatter for the radiated sound. Furthermore, they considered time delays from the monopoles to the listener. In contrast, we solve the Helmholtz radiation equation but ignore the time delays. We

also note that the difference of resulting sounds using the Rayleigh method and the Helmholtz solution has been shown in [James et al. 2006].

On the other hand, when simulating the acoustics of rooms and concert halls, the dimensions of the rooms or obstacles are many times larger than the sound wavelength. As a result, a variety of geometric acoustical methods have been developed [Funkhouser et al. 1999; Tsingos et al. 2001; Tsingos et al. 2002], analogous to the geometric optics in image rendering. The geometric acoustic methods enjoy fast performance while producing plausible results. However, when the characteristic dimension of objects becomes comparable to the wavelength, as in our problems, the wave diffraction begins to play an import part, necessitating the modeling of sound wave behaviors usually described by near-field transfer functions. A few previous works have demonstrated the importance of transfer functions [James et al. 2006; Corbett et al. 2007] for typical rigid object sounds in computer animations.

There exist several example-based methods on estimating sound parameters from input audio clips. Early work by Pai et al. [2001] estimated rigid modal sound models from recorded sounds and measurements. More recently, Lloyd et al. [2011] analyzed the short-time Fourier Transform and identify the strongest peaks in the spectrogram to estimate modal parameters. Ren et al. [2013] computed a set of features from given examples of audio clips, and used them to optimize modal sound parameters. While they can produce high-quality parameter estimation, they are not focused on acoustic transfer functions of vibration modes. With our model, we are able to explore modal sound parameter space straightforwardly (see §7) and synthesize resulting sounds that take into account the sound radiation effects.

The frequency-varying sound radiation problem has been studied in many engineering applications. Fast frequency sweep methods are most closely related to our method [Pillage and Rohrer 1990; Lenzi et al. 2013]. The basic idea is to compute Helmholtz solutions at a few key frequencies and interpolate/extrapolate Helmholtz radiation at intermediate points. However, these methods aim to produce engineering accuracy rather than high performance at runtime. Although they provide approximations of frequency-varying Helmholtz solutions, it is nontrivial to evaluate transfer values at an arbitrary point at runtime. In contrast, we aim for fast evaluation of acoustic transfer at any spatial and frequency point. To this end, we propose to use a Prony series representation [Hauer et al. 1990] constructed using adaptive frequency sweeping.

In addition to the sound synthesis from physically based simulation, there exist numerous audio processing software. Almost all these tools rely on signal processing algorithms to edit sound effects such as frequency modulation [Chowning 1973], reverberation [Smith 1985], and spectrum adjustment [Strawn 1987], or use stochastic sound models and granular synthesis methods to produce natural sound textures [Cook 2002]. However, these methods lack *automatic synchronization* with computer-simulated animation, and often need to store a large database of sound effects. Our model is complementary to those tools, enabling automatic audiovisual synchronization using physical simulation.

Our method utilizes frequency-adaptive mesh simplification to accelerate the individual Helmholtz solves. There are numerous methods for surface mesh simplification (see a survey by Luebke [2001]). Among them, our method is based on the edge collapse algorithms [Garland and Heckbert 1997], which coarsen a mesh through a sequence of edge collapse operations. In particular, we augment the method introduced in [Hoppe 1999] and [Lindstrom and Turk 1998] to preserve mesh volume as well as volume vibration velocity. The latter is an important quantity to preserve sound radiation power. Consequently, the optimization problem for edge collapse becomes significantly harder: rather than solving a linear system, we need to solve a quadratically constrained quadratic programming (QCQP) problem, for which we propose a staggered iterative algorithm.

The idea of using geometric simplification for efficient acoustic computation has been used in the research of room acoustical modeling. The input CAD models are often simplified architectural models in an exchange for faster computation. Siltanen et al. [2008] proposed a geometry reduction method based on volumetric reconstruction using a modified Marching Cubes algorithm. Further, geometrical acoustical simulation has adopted level-of-detail approaches to adaptively select polygon meshes used in the computation [Tsingos et al. 2007; Pelzer and Vorländer 2010]. The adaptivity of these approaches is based on incident sound waves for sound auralization. While these approaches mostly focus on room acoustics, our method is concerned with sound radiation produced by the modal vibration of an object. Therefore, the adaptivity of our method is based on the modal vibration frequencies of the object.

3. MODAL SOUND PRELIMINARY

Before presenting the details of our method, we briefly review the widely used modal sound model (see [Shabana 1991; O'Brien et al. 2002; James et al. 2006] for details) and clarify the frequency-related parameters that can be freely changed in our model.

Modal Vibration First, a solid object vibration is approximated by a linear vibration equation,

$$\mathsf{M}\ddot{\boldsymbol{u}} + \mathsf{D}\dot{\boldsymbol{u}} + \mathsf{K}\boldsymbol{u} = \boldsymbol{f}_{ext}, \tag{2}$$

where M, K, and D are respectively the mass, stiffness, and damping matrices depending on the object materials, $\boldsymbol{u} \in \mathbb{R}^{3n}$ describes the finite element nodal displacement with n nodes, and $\boldsymbol{f}_{ext} \in \mathbb{R}^{3n}$ is the external force driving the vibration. The damping matrix D is usually approximated using the *Rayleigh damping* model [Shabana 1991], i.e., $\mathsf{D} = \alpha\mathsf{M}+\beta\mathsf{K}$, where the scalars α and β are user-specified parameters. Linear modal analysis then solves a generalized eigenvalue problem $\mathsf{KU} = \mathsf{MUS}$ to compute a modal shape matrix U and a diagonal eigenvalue matrix S. The former describes the vibration pattern of each mode while the latter indicates the square of undamped natural frequencies, i.e., $\mathsf{S}_{i,i} = \omega_i^2$. Substituting $\boldsymbol{u} = \mathsf{U}\boldsymbol{q}$ and then premultiplying U on both sides of (2) decouples the system into a set of 1D second-order ordinary differential equations (ODEs), each of which is an ODE describing the modal vibration of a single mode i, namely,

$$\ddot{\boldsymbol{q}}_i + d_i\dot{\boldsymbol{q}}_i + \omega_i^2\boldsymbol{q}_i = \mathsf{U}_i^T\boldsymbol{f}_{ext}, \tag{3}$$

where d_i is the damping parameter of mode i, and U_i is the i-th column of U.

Sound Radiation A vibration mode with an observed frequency ω produces propagating sound waves that have a wavenumber $k = \omega/c$. A standard tool to model its sound radiation is the Helmholtz equation (1), which is coupled with surface modal vibration through a Neumann boundary condition defined on the object surface S,

$$\frac{\partial p}{\partial \boldsymbol{n}} = -i\omega\rho\boldsymbol{v} \quad \text{on } S, \tag{4}$$

where i is the imaginary unit, ρ is the air density, and $\boldsymbol{v}$ is the mode's time-harmonic vibration velocity along the surface normal direction, computed as $\boldsymbol{v} = i\omega(\boldsymbol{n} \cdot \tilde{\boldsymbol{u}}_i)$, where $\boldsymbol{n} \cdot \tilde{\boldsymbol{u}}_i$ is the normal-direction modal displacement of a mode i. Note we also use i to represent mode index when there is no ambiguity. Solving (1) for each mode i results in a complex-valued transfer function $p_i(\boldsymbol{x})$. Finally, following the approximation in [James et al. 2006], the sound wave at a listening position $\boldsymbol{x}$ is computed as a weighted summation of all audible vibration modes,

$$s(\boldsymbol{x}) = \sum_i |p_i(\boldsymbol{x})| \boldsymbol{q}_i(t). \tag{5}$$

This expression is accurate up to a phase, ignoring the time delay of sound propagation. The Helmholtz solution describes modal sound radiation affected by the object's own geometry, and ignores environment acoustics. This is sufficient in our problem because environment acoustics are independent from a modal sound model; if the environment is known, one can easily feed the resulting sound of our model to any sound auralization methods (e.g., [Tsingos et al. 2001; Raghuvanshi et al. 2010; Mehra et al. 2013]) to add room acoustic effects.

Frequency-Related Parameters As observed by Klatzky et al. [2000], two sets of parameters are of particular importance for achieving desired sound characteristics: *vibration frequencies*, ω_i, that determine sound pitch, and *damping coefficients*, d_i, that affect the timbre of particular materials. For instance, a small damping value results in the long ringing sounds that metal or porcelain objects often produce, whereas a large damping value tends to produce sounds more like wood or stone. We note that the damping coefficients d_i are also frequency-related; they affect the observed *damped natural frequency* through the relationship $\tilde{\omega}_i = \sqrt{\omega_i^2 - d_i^2/4}$. This frequency value is used for computing the wavenumber k, and thus affects the Helmholtz solution. In standard modal sound models, the user can change material parameters such as Young's modulus to adjust ω_i, and change α and β values in Rayleigh damping to control d_i. In our implementation, we change the scales of Young's modulus, which we call stiffness scales, and damping scales. They form a 2D parameter space (see Figure 1), in which the modal shape matrix U remains constant. We also explore the examples that allow the user to change ω_i and d_i directly and individually (see §7).

4. RUNTIME SOUND SYNTHESIS

In this section, we introduce our runtime sound synthesis algorithm while deferring the precomputation details until §5 (see the flow chart of our model in Figure 2). At runtime, we take as input an animation sequence, the user-specified ω_i and d_i, and the contact forces that appear on the right-hand side of (2) to drive the surface vibration. Given a listening location $\boldsymbol{x}$, it computes surface modal vibration $\boldsymbol{q}_i$ for every mode (in §4.1), and evaluates the transfer function $p_i(\boldsymbol{x})$ (in §4.2). The final sound is computed using the superposition (5) of individual modes. For simplicity of presentation, we describe the sound synthesis algorithm for a single mode. An outline of our runtime algorithm is shown in Algorithm 1.

4.1 Vibration Integration for $\boldsymbol{q}_i(t)$

We first solve the decoupled 1D modal vibration equation (3), where the external force $\boldsymbol{f}_{ext}$ is the contact forces resulting from the simulated animation. Many previous methods [Hamming 1983; James and Pai 2002] solve this inhomogeneous second-order ODE using a digital Infinite Impulse Response (IIR) filter. Our implementation uses the fourth-order Runge-Kutta method [Press et al. 2007], since we found it has comparable performance but higher accuracy than the digital IIR filter, especially when the user specifies time-varying ω_i and d_i values, as in the examples of §7 (see Figure 3).

Algorithm 1 Runtime Sound Computation for Mode i

Require: frequency ω_i of mode i and listening position $\boldsymbol{x}$
1: **procedure** SOUNDEVAL($i, \boldsymbol{x}$)
2: Compute $q_i(t)$ at audio rate by integrating (3) ▷ §4.1
3: Compute transfer $p(\omega_i)$ at key positions using (9) ▷ §5.1
4: Estimate M_n^m using a least-squares solve (7) ▷ §4.2
5: Compute transfer $p(\boldsymbol{x})$ using M_n^m and (6)
6: Compute $\sum_i |p(\boldsymbol{x})| q_i(t)$ at the audio rate
7: **end procedure**

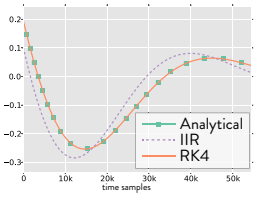


Fig. 3: **Comparison between Runge-Kutta and IIR filter:** Given a time-varying vibration equation, fourth-order Runge-Kutta (RK4) integrator (orange) offers higher accuracy against the IIR filter (purple), which was used in previous methods.

4.2 Transfer $p_i(\boldsymbol{x})$ Estimation via Least-Squares

A main challenge for runtime sound synthesis is the evaluation of transfer values $p_i(\boldsymbol{x})$. This is because p_i, the solution of the Helmholtz equation (1), is frequency-dependent. Whenever the user changes frequency parameters, we need to update $p_i(\boldsymbol{x})$, but solving the Helmholtz equation from scratch is impractical computationally very expensive.

Multipole Approximation To allow the user to freely change the listening location $\boldsymbol{x}$ while editing a sound, we need a compact representation of $p(\boldsymbol{x})$ in the spatial domain. Similar to [Zheng and James 2010], we represent a Helmholtz solution $p(\boldsymbol{x})$ using a single point multipole expansion [Gumerov and Duraiswami 2004], which takes an expansion form,

$$p_i(\boldsymbol{x}) \approx ik \sum_{n=0}^{N} \sum_{m=-n}^{n} S_n^m(\boldsymbol{x}, \bar{\boldsymbol{x}}_0) M_n^m(\omega). \tag{6}$$

Here $\bar{\boldsymbol{x}}_0$ is the expansion center near the object. In practice, we always place it at the object's center of mass. We follow the rule of thumb [Liu 2009; Zheng and James 2010] and set the expansion order $N = \max(\frac{1}{4}kL, 4)$ ($N \leq 18$ in all our examples). S_n^m are singular Helmholtz basis functions, $S_n^m(\boldsymbol{r}) = h_n^{(2)}(kr) Y_n^m(\theta, \phi)$, where $\boldsymbol{r} = (r, \theta, \phi)$ is the spherical coordinate of $\boldsymbol{x} - \boldsymbol{x}_0$, $h_n^{(2)} \in \mathbb{C}$ are spherical Hankel functions of the second kind, and $Y_n^m \in \mathbb{C}$ are spherical harmonics. The expansion coefficients M_n^m depend on the modal vibration frequency ω, and are what we need to quickly update when ω is changed at runtime.

Previous methods of computing M_n^m (e.g. [Gumerov and Duraiswami 2004; Zheng and James 2010]) integrate the results of a boundary element (BE) solve of (1) over the entire object surface. Both the BE solve and surface integral are expensive. One might precompute a set of M_n^m using frequency values sampled in a frequency range, and use the interpolated M_n^m at runtime. However,

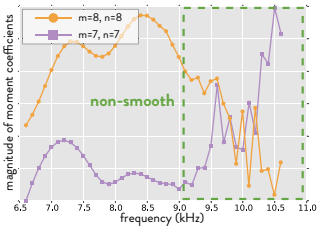


Fig. 4: **Non-smoothness of M_n^m:** The high-order M_n^m becomes non-smooth and fluctuates strongly at high frequencies, making direct interpolation difficult. We note that these orders (i.e., N=7,8) are necessary in the expansion for sufficient accuracy.

as shown in Figure 4, M_n^m at high orders fluctuates dramatically as the frequency value sweeps. Consequently, the frequency needs to be densely sampled to interpolate M_n^m, leading to a prohibitively long precomputation time.

Fast Least-Squares Approximation of M_n^m Inspired by the subspace construction for shape deformations [Meyer and Anderson 2007], we approximate M_n^m using a small-scale least-squares approximation. Here, the Helmholtz basis functions S_n^m construct a set of reduced-space bases of the Helmholtz solution. We estimate the basis coefficients M_n^m based on the transfer values at a set of key positions, and use the resulting M_n^m to compute transfer value $p_i(\boldsymbol{x})$ at any listening location $\boldsymbol{x}$.

First, we sample a set of *key positions* $\boldsymbol{x}_j, j = 1, ..., J$ outside of the object (see Figure 5). In the precomputation, we construct a frequency-sweeping transfer function $p_j(\omega)$ at $\boldsymbol{x}_j$. The representation of $p_j(\omega)$ is compact, adding very little memory overhead over the standard modal sound model (see §6). And its construction requires only coarse frequency samples. We defer the details of its construction until §5.1. Here we use the precomputed representation to evaluate $p_j(\omega)$ with user-specified ω at every key position $\boldsymbol{x}_j$ and stack them into a vector $\tilde{\boldsymbol{p}}$.

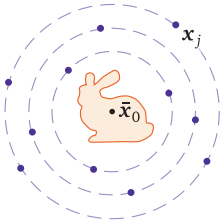


Fig. 5: Key Positions

The summation of (6) can be expressed in matrix form, $\tilde{\boldsymbol{p}} = \mathsf{A}\boldsymbol{m}$, where $\boldsymbol{m}$ stacks all the coefficients M_n^m, and A consists of the Helmholtz basis function values at all selected positions $\boldsymbol{x}_j$. Concretely, they have the form

$$\mathsf{A} = \begin{bmatrix} S_0^0(\boldsymbol{x}_1, \bar{\boldsymbol{x}}_0) & \dots & S_N^N(\boldsymbol{x}_1, \bar{\boldsymbol{x}}_0) \\ \vdots & \ddots & \vdots \\ S_0^0(\boldsymbol{x}_J, \bar{\boldsymbol{x}}_0) & \dots & S_N^N(\boldsymbol{x}_J, \bar{\boldsymbol{x}}_0) \end{bmatrix} \text{ and } \boldsymbol{m} = \begin{bmatrix} M_0^0 \\ \vdots \\ M_N^N \end{bmatrix}.$$

We then estimate the unknown coefficients M_n^m by solving the least squares problem

$$\mathsf{A}\boldsymbol{m} = \tilde{\boldsymbol{p}}. \tag{7}$$

As long as the number of key positions is larger than the number of columns of A, we have an over-constrained and complex-valued least-squares problem, and thus the solution is unique. Recall that the order of multipole expansion (6) is small (i.e., $N \leq 18$). The number of columns is also small (i.e., $N^2 \leq 324$), and thus the least-squares problem can be efficiently solved at runtime. Once M_n^m are computed, we substitute them into (6) to evaluate the transfer value at any location $\boldsymbol{x}$. In §6, we validate the accuracy and convergence of our transfer evaluation algorithm.

When sampling key positions, we need to cover the region where the listener will be located. We therefore select three spheres centered at the object's center of mass $\bar{\boldsymbol{x}}_0$ with radii of 1.6, 2.6, and 3.4 times the object's geometric size. We then uniformly sample positions over the spheres (see Figure 5). In §6, we validate the numerical accuracy and convergence of this scheme.

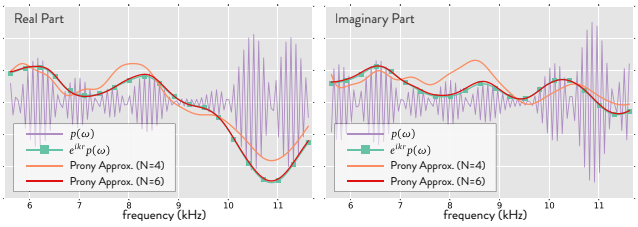


Fig. 6: **Frequency-Sweeping Transfers:** We choose one mode of the BUNNY model, evaluate $p(\omega)$ using BEM at a fixed point as frequency sweeps and plot both real and imaginary parts. $p(\omega)$ oscillates dramatically (purple); factoring out e^{-ikr} produces a much smoother curve (green); 4th-order Prony series gives a coarse interpolation curve (orange), while 6th-order series produces a curve (red) almost identical to the original function.

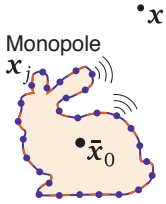
5. PRECOMPUTATION OF OUR METHOD

The core goal of our precomputation is to construct a representation of frequency-sweeping transfer function $p_j(\omega)$ for every key position $\boldsymbol{x}_j, j = 1, ..., J$ and every vibration mode. This representation is used at runtime to construct the right-hand-side vector $\tilde{\boldsymbol{p}}$ in (7). For a mode i with a natural vibration frequency ω_0, we allow the user to adjust its vibration frequency in the range $\mathcal{R} = [\omega_0 - \Delta\omega, \omega_0 + \Delta\omega]$. In practice, we allow the runtime frequency adjustment in a range of 5kHz (i.e., $\Delta\omega = 2.5\text{kHz} \cdot 2\pi$).

A simple approach is to compute $p_j(\omega_t)$ at a set of frequency samples $\omega_t, t = 1, \dots, T$ in $\mathcal{R}$, and interpolate to obtain $p_j(\omega)$. However, this approach requires a large number of frequency samples, since $p_j(\omega)$ oscillates at a high frequency as shown in Figure 6. And evaluation of transfer samples $p_j(\omega_t)$ at different frequencies requires expensive individual Helmholtz solves. Therefore, we seek to sample $\mathcal{R}$ using a sparse set of ω_t. Our algorithm strives to (i) avoid as many Helmholtz solves as possible (in §5.2) and (ii) improve the solving performance (in §5.3). We start by first presenting an efficient representation of $p_j(\omega)$ using sparse transfer samples $p_j(\omega_t)$. For simplicity, we will drop the subscript j, because we will consider transfer values at a single key position $\boldsymbol{x}_j$.

5.1 Frequency-Sweeping Transfer Representation

To reveal why $p(\omega)$ is oscillatory, consider a first-order approximation of sound radiation. As commonly used in the *Rayleigh method* [Cremer et al. 2005], we can discretize a vibrating surface into small elements, and estimate its radiation (up to first order) by placing a monopole on every element and summing up their contributions. Namely, the acoustic transfer at $\boldsymbol{x}$ is estimated as



$$p(\boldsymbol{x}) \approx \sum_{j=1}^{N} C_j e^{-ikr_j} = \sum_{j=1}^{N} C_j e^{-i\frac{\omega}{c}r_j}, \tag{8}$$

where C_j is the weight of a monopole on element j and r_j is the distance from $\boldsymbol{x}$ to the position of j-th monopole (i.e., $r_j = |\boldsymbol{x} - \boldsymbol{x}_j|$). This expression clearly shows that as ω sweeps in a range

$\mathcal{R}$, $p(\boldsymbol{x})$ oscillates because $e^{-i\frac{\omega}{c}r_j}$ is harmonic with respect to ω. In fact, for a far-field listener, r_j is large, and thus $p(\boldsymbol{x})$ oscillates strongly.

Interpolation of $p(\omega)$ We propose the following scheme to interpolate $p(\omega)$. For every acoustic transfer sample $p(\omega_t)$, we compute $\tilde{p}(\omega_t) = e^{ik_t r}p(\omega_t)$, where k_t is the wavenumber ω_t/c, and $r = |\boldsymbol{x}_j - \bar{\boldsymbol{x}}_0|$ is the distance from a key position $\boldsymbol{x}_j$ to the object's center of mass. We claim that $\tilde{p}(\omega)$ is much smoother than $p(\omega)$ (see Figure 6), and thus we can easily construct an interpolation of $\tilde{p}(\omega_t)$ in $\mathcal{R}$. To understand the reason, we again look to the approximation (8), and have $e^{ikr}p(\omega) \approx \sum_j C_j e^{-ik(r_j - r)}$. For a far-field listening location, we have $|r_j - r| \ll |r|$. Therefore, $e^{-ik(r_j-r)}$ and hence $e^{ikr}p(\omega)$ are much less oscillatory. Lastly, given a frequency ω, we interpolate $\tilde{p}(\omega)$ and compute the transfer value using $p(\omega) = e^{-ikr}\tilde{p}(\omega)$.

Representation of $\tilde{p}(\omega_t)$ using Prony's Method The remaining question is how to represent $\tilde{p}(\omega_t)$. Since we will create a representation of $\tilde{p}(\omega)$ for every key position and every vibration mode, the representation needs to be compact. Note $\tilde{p}(\omega)$ still oscillates with respect to ω, albeit smoothly, because of the term $e^{ik(r_j-r)}$. This suggests that we should use a small number of harmonic basis functions to interpolate $\tilde{p}(\omega_t)$. Fourier basis functions were considered; they are efficient for representing periodic signals. However, in our case, the amplitude of $p(\omega_t)$ generally tends to decrease as ω increases, because the high-frequency waves dissipate energy faster than low-frequency waves do. These damped signals require more Fourier bases for a plausible representation (see Figure 16 in Appendix F). Instead, we propose to use *Prony's method* [Hauer et al. 1990; Lobos et al. 2003], which approximates a uniformly sampled signal using a series of weighted complex exponentials. In our case, it approximates $\tilde{p}(\omega)$ as

$$\tilde{p}(\omega) \approx \sum_{i=1}^{N} c_i e^{\mu_i \omega} \tag{9}$$

where c_i and μ_i are complex values determined using the transfer samples $\tilde{p}(\omega_t)$. For readers not familiar with this method, we list the details in Appendix A. Here we highlight its advantages in our problems. (i) Prony series has been known for its efficiency on estimating damping coefficients apart from frequency, phase, and amplitude [Lobos et al. 2003]. It requires only sparse samples to represent the signals (i.e., $2N$ samples for N harmonic components). (ii) It offers a compact representation of $\tilde{p}(\omega)$, allowing fast runtime evaluation of $p(\omega)$. With precomputed c_i and μ_i, we use the Prony series (9) to construct the right-hand-side vector in the least-squares problem (7) for estimating the multipole coefficients M_n^m as described in § 4.2. Figure 6 shows that $N = 6$ is sufficient for a close approximation in our experiments. Table I lists the storage needed for runtime use, less than 25MB per model. (iii) Computing the parameters c_i and μ_i is fast, involving only two small least-squares solves and a polynomial root-finding (see Appendix A).

5.2 Adaptive Frequency Sweep

Creating a Prony's representation of $\tilde{p}(\omega)$ takes as input a set of transfer values $p(\omega_t)$ at uniformly sampled $\omega_t \in \mathcal{R}, t = 1, .., T$. Straightforward evaluation of $p(\omega_t)$ needs to solve the Helmholtz equation from scratch, which is expensive. Thus, we wish to bypass those solves as many as possible. To this end, we build our algorithm upon the method of *Asymptotic Waveform Evaluation* (AWE) [Pillage and Rohrer 1990; Gallivan et al. 1994] and perform adaptive Helmholtz solves.

Algorithm 2 Frequency Sweep Precomputation for Mode i

Require: the damped natural frequency ω_i of mode i
procedure ADAPTIVEFREQUENCYSWEEP(ω_i)
  Frequency range $\mathcal{R} \leftarrow [\omega_i - \Delta\omega, \omega_i + \Delta\omega]$
  Uniformly sample $\mathcal{R}$ with ω_t s.t. $\omega_1 > \omega_2 > \ldots > \omega_T$
  $\omega^* = \omega_1$
  Construct AWE coefficients at ω^* (i.e., $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ in (13))
  for all ω_t in descending order **do**
    if ω_t not in the convg. radius of ω^* (using (14)) **then**
      $\omega^* \leftarrow \omega_t - (\omega^* - \omega_{t+1})$
      Construct AWE coefficients at ω^*
    end if
    Compute $p_j(\omega_t)$ at key positions using (13) and (21)
  end for
  for all key position j **do**
    Build Prony series representation (9)
  end for
end procedure

Our key idea is to sweep the frequency range $\mathcal{R}$ with multiple steps. At each step, we choose a reference frequency ω_0 and build a local asymptotic expansion of the frequency-varying Helmholtz solution. At the next step, we choose a new reference frequency that cannot be covered by the estimated convergence radius of the expansion at ω_0 in the previous solve. We repeat the step until the entire $\mathcal{R}$ is covered by the convergence ranges of all expansions (see an outline in Algorithm 2).

Boundary Element Solve We use BEM to solve the Helmholtz equation at every reference frequency. For the exterior Helmholtz radiation problem as in our case, The conventional boundary integral equation (CBIE) has non-unique solutions at certain *fictitious frequencies* [Matsumoto et al. 2010]. This will cause serious problems as we need to sweep through a wide frequency range, which likely covers those fictitious frequency values. Instead, we follow the Burton-Miller method [Burton and Miller 1971], which solves a linear combination of CBIE and a hypersingular boundary integral equation (HBIE) to overcome the non-uniqueness (see Figure 13 for numerical validation). We refer the reader to Appendix B for our implementation details. Ultimately, we solve a dense linear system

$$\mathsf{A}(\omega)\boldsymbol{\phi}(\omega) = \boldsymbol{b}(\omega). \tag{10}$$

Here we explicitly express the system with a frequency parameter ω to emphasize its dependence on the frequency value that we are sweeping in $\mathcal{R}$. The solution $\boldsymbol{\phi}(\omega)$ is a vector stacking the acoustic transfer value on object surface elements. With this solution, the transfer value $p(\boldsymbol{x}_j)$ at a key position $\boldsymbol{x}_j$ is computed using the *Kirchhoff integral formula* detailed in (21) of Appendix B.

Asymptotic Waveform Evaluation After a BE solve at a frequency ω_0, we have $\boldsymbol{\phi}(\omega_0)$ that satisfies the linear system $\mathsf{A}(\omega_0)\boldsymbol{\phi}(\omega_0) = \boldsymbol{b}(\omega_0)$. Then, a polynomial asymptotic expansion of $\boldsymbol{\phi}(\omega)$ can be built in a local region centered at ω_0,

$$\boldsymbol{\phi}(\omega) = \sum_{i=0}^{N} \boldsymbol{\phi}_i(\omega - \omega_0)^i, \tag{11}$$

where $\boldsymbol{\phi}_0 = \boldsymbol{\phi}(\omega_0)$ and $\boldsymbol{\phi}_i, i = 1, \ldots, N$ are coefficients to be determined. To compute $\boldsymbol{\phi}_i$, we take the derivatives of both (10)

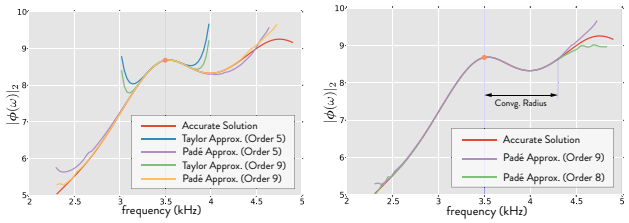


Fig. 7: **Asymptotic Waveform Evaluation:** Using the BUNNY model, we sweep a frequency range and evaluate $|\boldsymbol{\phi}(\omega)|_2$ in (10) using different expansions and accurate Helmholtz BE solves. (left) We compare the convergence radius of polynomial expansions against Padé approximant. Accurate BE solution is plotted in red. Both 5th-order and 9th-order polynomial expansion diverge from the accurate solution faster than their Padé counterparts. (right) 8th-order Padé approximant agrees with the 9th-order one closely until the convergence radius is reached.

and (11) with respect to ω and form a linear system of $\boldsymbol{\phi}_1$,

$$\mathsf{A}(\omega_0)\boldsymbol{\phi}_1 = \boldsymbol{b}'(\omega_0) - \mathsf{A}'(\omega_0)\boldsymbol{\phi}(\omega_0).$$

Here $\mathsf{A}'(\omega_0)$ and $\boldsymbol{b}'(\omega_0)$ can be computed analytically (see Appendix D for details), and we have factorized $\mathsf{A}(\omega_0)$ when solving $\mathsf{A}(\omega_0)\boldsymbol{\phi}(\omega_0) = \boldsymbol{b}(\omega_0)$. Therefore only a fast back-substitution is needed here. Higher-order coefficients $\boldsymbol{\phi}_i$ can be solved in a similar manner using higher-order derivatives of (10) and (11). We defer the detailed derivation in Appendix C. In short, we have the following equation to solve for $\boldsymbol{\phi}_i$,

$$n!\mathsf{A}(\omega_0)\boldsymbol{\phi}_n + \sum_{i=1}^{n}(n-i)!C_n^i\mathsf{A}^{(i)}(\omega_0)\boldsymbol{\phi}_{n-i} = \boldsymbol{b}^{(n)}(\omega_0). \quad (12)$$

where $C_n^i = \frac{n!}{i!(n-i)!}$ are the binomial coefficients. This is a linear system of the form $\mathsf{A}(\omega_0)\boldsymbol{\phi}_n = \boldsymbol{c}$, since all the $\boldsymbol{\phi}_i, i = 0, \ldots, n-1$ are known from previous computation. And again we quickly solve this system by reusing the factorization of $\mathsf{A}(\omega_0)$. After all $\boldsymbol{\phi}_i, i = 0, \ldots, N$ are solved, we can quickly compute the Helmholtz solution at a frequency ω using (11).

Extending Convergence Radius with Padé Approximant One drawback of the straightforward polynomial expansion (11) is that it tends to have a limited convergence radius (see Figure 7). Consequently, we need many AWE solves to cover the frequency range $\mathcal{R}$. To alleviate this problem, we propose to build a Padé approximant, which is known to provide a larger convergence radius than a polynomial expansion although it is derived from polynomial coefficients [Karlsson 1976]. In particular, provided a set of solved polynomial expansion coefficients $\boldsymbol{\phi}_i, i = 0, \ldots, N$, we match the polynomial expansion (11) with a rational polynomial,

$$\sum_{i=0}^{L+M+1}\boldsymbol{\phi}_i(\omega-\omega_0)^i = \frac{P_L(\omega-\omega_0)}{Q_M(\omega-\omega_0)} = \frac{\sum_{i=0}^{L}\boldsymbol{\alpha}_i(\omega-\omega_0)^i}{1+\sum_{j=1}^{M}\boldsymbol{\beta}_j(\omega-\omega_0)^j} \quad (13)$$

where both $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are vectors with the same length as $\boldsymbol{\phi}_i$; the quotient is computed using a component-wise division. In practice, we set the rational polynomial orders, $M = \lfloor N/2 \rfloor$ and $L = N - M$, so this Padé approximant has the same complexity as the polynomial expansion (11). We solve $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ by multiplying both sides of (13) by $Q_M(\omega-\omega_0)$ and matching the coefficients for all orders of terms. This amounts to solving

$$\begin{bmatrix} \boldsymbol{\phi}_L & \boldsymbol{\phi}_{L-1} & \cdots & \boldsymbol{\phi}_{L-M+1} \\ \boldsymbol{\phi}_{L+1} & \boldsymbol{\phi}_L & \cdots & \boldsymbol{\phi}_{L-M+2} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_{L+M-1} & \boldsymbol{\phi}_{L+M-2} & \cdots & \boldsymbol{\phi}_L \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_M \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\phi}_{L+1} \\ \boldsymbol{\phi}_{L+2} \\ \vdots \\ \boldsymbol{\phi}_{L+M} \end{bmatrix}.$$

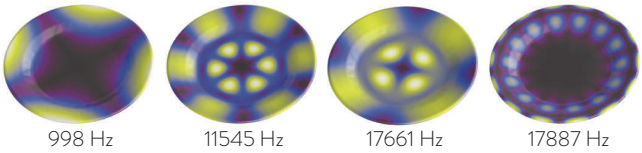


Fig. 8: **Smooth Modal Shapes:** Color encodes the modal displacement amplitude of the PLATE model; modal frequencies are listed below each subfigure. Even for high frequency modes, their modal displacement varies smoothly on the surface, making it possible to perform mesh simplification.

Let D denote the number of boundary elements and also the length of $\boldsymbol{\phi}_i$. The above equation describes D independent linear systems, each corresponding to a single component of $\boldsymbol{\phi}_i$ and $\boldsymbol{\beta}_i$; therefore we can solve all D linear systems in parallel. Once $\boldsymbol{\beta}_i$ is obtained, we compute $\boldsymbol{\alpha}_i$ using $\boldsymbol{\alpha}_i = \sum_{j=0}^{i}\boldsymbol{\beta}_j\boldsymbol{\phi}_{i-j}$. Again the product $\boldsymbol{\beta}_j\boldsymbol{\phi}_{i-j}$ indicates a component-wise multiplication. Figure 7 illustrates the improvement of the convergence radius.

Adaptive Helmholtz Solves With a depiction of our AWE solver in place, we now present our algorithm to sweep through $\mathcal{R}$ and adaptively perform AWE solves. Let $\omega_{t\ t=1,\ldots,T}$ denote our uniform frequency samples sorted in descending order. Our goal is to evaluate $p(\omega_t)$ at all key positions. We start from the highest frequency sample ω_T and build an AWE expansion series at ω_T. Then we move on to ω_{T-1} and check if ω_{T-1} is within the convergence radius of the series at ω_T. If it is, we directly evaluate the series to compute $p(\omega_{T-1})$ and continue to the next sample ω_{T-2}. At some point, a sample ω_i is out of the convergence radius, and thus the obtained AWE series becomes invalid. At this point, we know a lower bound of the convergence radius of the series at ω_T is $\omega_T - \omega_{i+1}$. We also observe that the convergence radius of the AWE series increases as the expansion frequency decreases. Therefore, we build a new AWE series at the frequency sample $\omega_j = \omega_i - (\omega_T - \omega_{i+1})$. Since ω_j is smaller than ω_T, we guarantee that ω_i is now within the convergence radius of the series at ω_j. From there we switch to the new series at ω_j and continue our transfer evaluation. We repeat these steps until the entire frequency samples $\omega_t, t = 1, \ldots, T$ are evaluated (see Algorithm 2). The main advantage of this process is that we solve the AWE only at a few automatically selected frequency samples while relying on the expansion form to quickly evaluate transfer values for all the samples.

A simple approach to check if a frequency ω is within the convergence radius is to evaluate the series at ω and substitute it back into (10) to compute the residual. However, this approach needs to construct the dense matrix $\mathsf{A}(\omega)$ for every check. Instead, we propose a faster algorithm by exploiting a mathematical insight of Padé approximant: two consecutive orders of Padé solutions are very close inside the convergence radius, but they diverge rapidly when this radius is reached. To harness this insight, we compute

$$\hat{\boldsymbol{\phi}}(\omega) = \frac{P_{L-1}(\omega-\omega_0)}{Q_{M-1}(\omega-\omega_0)} \text{ and } \boldsymbol{\phi}(\omega) = \frac{P_L(\omega-\omega_0)}{Q_M(\omega-\omega_0)}, \quad (14)$$

and require $\|\hat{\boldsymbol{\phi}}(\omega) - \boldsymbol{\phi}(\omega)\| \leq \epsilon$. For all our examples, we use $L = 6$, $M = 5$ and $\epsilon = 10^{-4}$. On average we only need about 5 AWE solves to cover a frequency range of 5kHz to achieve this error tolerance (See Figure 7).

5.3 Frequency-Adaptive Mesh Simplification

To further speed up the precomputation, we accelerate each Helmholtz solves by adaptively simplifying object surface meshes. It is well known that the complexity of BEMs depend on the number of surface elements N. The smaller N is, the faster computation can be. For Helmholtz solves, it is also found that the element size should be bounded by the wavelength [Jerri 2005]. Corresponding to human hearing range from 20Hz to 20kHz, the wavelength varies from 17 meters to 1.72 centimeters. Thus we can use fewer elements while retaining the accuracy for lower frequencies. Furthermore, we observe that the scale of spatial variance of an object's mode shapes is often much larger than the size of the mesh elements (see Figure 8). As a result, the modal displacement data can be well resolved even with a coarse surface mesh.

We therefore use frequency-adaptive surface discretization in our BE solves. We start from a fine surface mesh sufficient for the highest frequency (e.g., 20kHz). For each mode i, surface vertices are also associated with their modal displacement vectors $\tilde{\boldsymbol{u}}_i$ extracted from the shape matrix U. Next, we divide our interested frequency range $\mathcal{R} = [\omega_0 - \Delta\omega, \omega_0 + \Delta\omega]$ into multiple intervals, each with a fixed frequency band. In practice, we use a 2kHz frequency band and hence up to 3 intervals for $\mathcal{R}$. For each frequency interval and each mode i, we construct a simplified surface mesh as well as corresponding modal displacement data. We perform the mesh simplification at the beginning of our precomputation stage. During our Helmholtz solves performed in §5.2, we adaptively choose the mesh resolution and modal displacement data based on the target frequency range and mode.

Edge Collapse Algorithm We build our mesh simplification algorithm based on the edge collapse algorithm of Hoppe [1999], and follow their notations therein. Each mesh vertex v has a 6D vector $\boldsymbol{v} = [\boldsymbol{p}^T \ \boldsymbol{u}^T]^T$, where $\boldsymbol{p}$ is the vertex position, and the modal displacement vector $\boldsymbol{u}$ is used as a vertex attribute. The quadric error function for collapsing an edge is defined as

$$Q^v(\boldsymbol{v}) = \sum_{f \in \mathcal{N}(v)} A(f) \left(Q_p^f(\boldsymbol{v}) + Q_u^f(\boldsymbol{v}) \right)$$

where $A(f)$ is the area of a triangle f adjacent to $\boldsymbol{v}$, $Q_p^f(\boldsymbol{v})$ measures the distance of $\boldsymbol{p}$ to the plane containing f, and $Q_u^f(\boldsymbol{v})$ measures the deviation of $\boldsymbol{u}$ from a linearly changing modal displacement field on the triangle f. Both terms are simply zero-extended versions of those in [Hoppe 1999]. We therefore refer the reader to that paper for details. In short, $Q(\boldsymbol{v})$ has a 6D quadratic form, $Q^v(\boldsymbol{v}) = \boldsymbol{v}^T \mathsf{A} \boldsymbol{v} + 2\boldsymbol{b}^T \boldsymbol{v} + \boldsymbol{c}$. When collapsing an edge connecting v_1 and v_2 into a new vertex v_{new}, we solve for its position $\boldsymbol{p}_{new}$ and modal vibration vector $\boldsymbol{u}_{new}$ by minimizing

$$\begin{aligned} &\boldsymbol{v}_{new} = \arg\min_{\boldsymbol{v}} Q^{v_1}(\boldsymbol{v}) + Q^{v_2}(\boldsymbol{v}) \\ &\text{s.t.} \quad \boldsymbol{g}_{vol}^T \boldsymbol{p} + d_{vol} = 0. \end{aligned} \tag{15}$$

Here $\boldsymbol{g}_{vol}$ and d_{vol} are respectively a 3D vector and scalar determined by the 1-ring local geometry of the collapsing edge. This linear constraint is to ensure volume preservation.

Volume Velocity Preservation While previous methods can achieve volume preservation, they shrink the modal amplitude in the process of edge collapse, resulting in a loss of sound power (see Figure 9). We address this problem by introducing a constraint on the object's *volume-velocity*. Given a modal displacement vector $\boldsymbol{u}$ at a frequency ω, the object's volume velocity is defined as

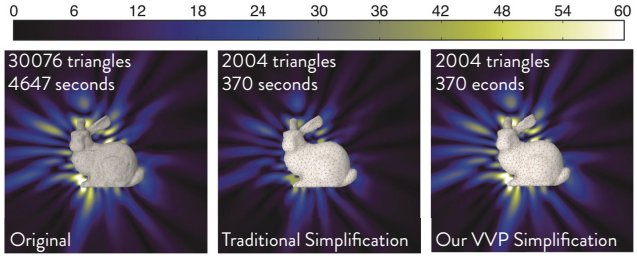


Fig. 9: **Volume-Velocity-Preserving Mesh Simplification:** We solve the Helmholtz equation using the original high-resolution mesh (left). We then simplify the mesh without volume-velocity preservation (middle) and with volume-velocity preservation (right). For both meshes, the Helmholtz solve is 12.6× faster than the original Helmholtz solve. Without volume-velocity preservation (middle), the acoustic transfer field loses radiation power, while the volume-velocity-preserving mesh simplification (right) results in almost identical pressure field to the original high-resolution solve.

$$\int_S (\boldsymbol{u} \cdot \boldsymbol{n}) \omega e^{i\omega t} dS = \omega e^{i\omega t} \int_S \boldsymbol{u} \cdot \boldsymbol{n} dS.$$

This quantity has been previously used as a far-field approximation of sound power [Johnson and Elliott 1995]. Our goal is to ensure its preservation during our mesh simplification. When an edge collapses into a vertex v, only the volume-velocity contributed by its 1-ring triangle fan can be changed. Ignoring the unchanged time-harmonic part and assuming a piece-wise constant modal vibration, we have the volume-velocity constraint,

$$\frac{1}{6} \sum_{f \in \mathcal{N}(v)} \left[(\boldsymbol{p} - \boldsymbol{p}_{f1}) \times (\boldsymbol{p} - \boldsymbol{p}_{f2}) \right]^T (\boldsymbol{u} + \boldsymbol{u}_{f1} + \boldsymbol{u}_{f2}) = C_v \tag{16}$$

where f denote a triangle adjacent to v, $(\boldsymbol{p}, \boldsymbol{p}_{f1}, \boldsymbol{p}_{f2})$ are f's vertex positions, $(\boldsymbol{u}, \boldsymbol{u}_{f1}, \boldsymbol{u}_{f2})$ are the modal displacement vectors on those vertices, and C_v is a constant volume velocity value computed from the corresponding triangle region before the edge collapse. Now we need to minimize $\bar{Q}^v(\boldsymbol{v}) = Q^{v_1}(\boldsymbol{v}) + Q^{v_2}(\boldsymbol{v})$ with two constraints when collapsing an edge,

$$\boldsymbol{v} = \arg\min_{\boldsymbol{v}} \bar{Q}^v(\boldsymbol{v}) \quad \text{subject to constraint (15) and (16).}$$

This is a quadratically constrained quadratic programming (QCQP) problem, generally considered to be NP-hard. Fortunately, in this particular QCQP problem, because the quadratic constraints involve only $\boldsymbol{p}^T \boldsymbol{u}$ but not $\boldsymbol{p}^T \boldsymbol{p}$ or $\boldsymbol{u}^T \boldsymbol{u}$, we are able to solve it using iterations of linearly constrained problems: we start from an initial guess of $\boldsymbol{v}$ by minimizing $Q^v(\boldsymbol{v})$ without any constraints, and this amounts to solving a 6D linear system, $\mathsf{A}\boldsymbol{v} = \boldsymbol{b}$. Then we iteratively apply a staggered sequence of two quadratic optimization solves

$$\boldsymbol{u} = \arg\min_{\boldsymbol{u}} Q^v([\boldsymbol{p}^T \ \boldsymbol{u}^T]^T) \quad \text{subject to (16) only,} \tag{17}$$

$$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} Q^v([\boldsymbol{p}^T \ \boldsymbol{u}^T]^T) \quad \text{subject to (15) and (16).} \tag{18}$$

In the first solve (17), we use vertex positions $\boldsymbol{p}$ from previous iterations and compute $\boldsymbol{u}$. In the second solve (18), we fix displacement vector $\boldsymbol{u}$ using values resulting from (17) and compute vertex positions. Both solves minimize a quadratic form with linear equality constraints. We solve them using the method of Lagrange Multipliers: problem (17) becomes a 4D linear system, while problem (18)

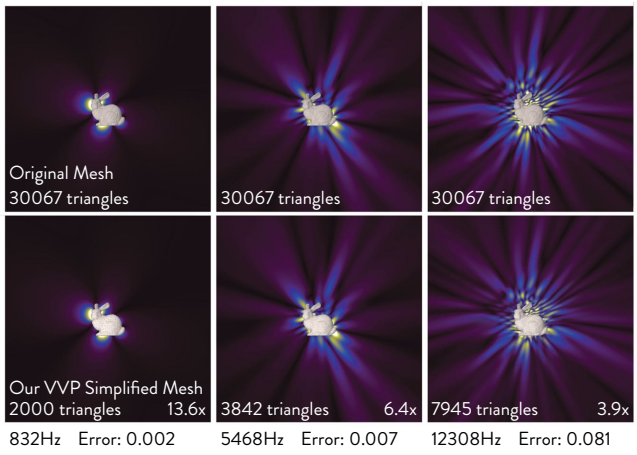


Fig. 10: **Transfer Values with Frequency-Adaptive Remeshing:** We compare transfer values computed using the original high-resolution mesh (top) with the values using simplified meshes (bottom) for three modes with low, medium and high frequency values. Even for high-frequency modes which require a relatively high-resolution mesh, our method achieves nearly $4\times$ speedup, while retaining a low L_2 error ($< 8.2\%$).

amounts to a 5D linear solve (see Appendix E). In practice, only tens of iterations are needed for convergence. As demonstrated in Figure 10, using the adaptively simplified meshes greatly speeds up the boundary element solves (in §5.2), while introducing very little numerical error.

6. VALIDATION

Performance We profiled the performance of each step of our algorithm, and summarized speedups of each step over the straightforward approaches, as well as the runtime speedups. Table I lists the statistics of our examples. The precomputation timings were measured on a 20-core Intel Xeon E5 cluster, and the runtime profiling was performed on a desktop with a quad-core Intel Xeon E5 (3.4GHz) CPU. Due diligence has been taken to exploit multi-core parallelization for both the precomputation and runtime sound synthesis. On average, our adaptive mesh simplification achieved $5\times$ speedups for Helmholtz solves; our adaptive frequency sweep led to at least $10\times$ speedups; and at runtime, given user-specified parameters, we are able to synthesize sound with more than $300\times$ speedups over the traditional approach which needs to recompute the Helmholtz solutions. We note that it is possible to further boost runtime performance for interactive parameter editing: for example, at runtime one can start a background thread performing the least-squares solves of multipole coefficients at more densely sampled frequencies while the user is adjusting parameters, and cache the computed transfer coefficients for later reuse.

The additional memory size needed for runtime use of our Prony representation for all key positions is less than 25MB per model. We also note that the major memory bottleneck of a modal sound model is the storage of shape modal matrix U that can take hundreds of megabytes of memory, depending on the mesh resolution. Our frequency-sweeping transfer representation is resolution-independent, and adds little memory overhead. We note a recent method [Langlois et al. 2014] that compresses the modal matrix U and complements to our approach.

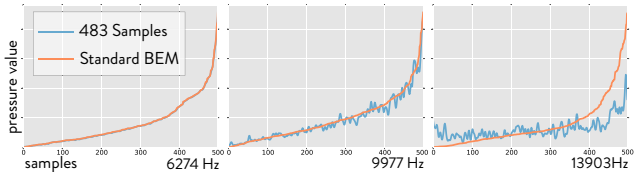


Fig. 11: **Accuracy of Least-Squares Approximation:** we sample 483 key positions for the least-squares estimation of M_n^m. We estimate M_n^m with three frequency values and use them to evaluate acoustic transfer values at 500 randomly selected locations (blue). Meanwhile, we compute the accurate Helmholtz solution at the same locations (orange). For better visualization, we sort the locations based on their accurate transfer values. As frequency increases, the accuracy of our approximated transfer values degrades gracefully.

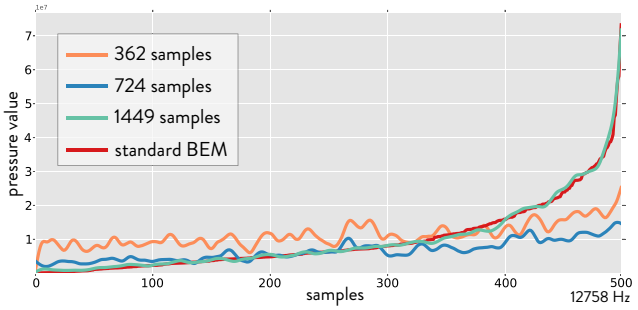


Fig. 12: **Convergence of Least-Squares Approximation:** Fixing the frequency value $f = 12758Hz$, we estimate M_n^m using an increasing number of key positions. We use the estimated M_n^m to evaluate acoustic transfer values at 500 randomly selected locations (orange, blue, and green curves), and compare them against accurate transfer values solved using conventional BEM (red). As the number of key positions increases, we get higher fidelity for the estimated transfer values.

Comparisons To demonstrate the effects of transfer values on final sounds and the accuracy of our transfer evaluation, we compared the sounds computed without transfer, with transfer at a fixed frequency, with transfer using our model, and the exact transfer using BE solves. For this comparison, we also chose different modal sound parameters to generate different sound effects. We observed that the resulting sounds from our model are very close to the sounds using brute-force transfer evaluation, while the sounds without transfer and with constant transfer both show audible differences from the ground-truth sounds. Please see the accompanying video for animations and sound comparisons.

Numerical Validation We further validated our models numerically. Our runtime transfer evaluation are approximated by least-squares problems formulated using key-position transfers. In Figure 11, we validate its accuracy by comparing with the results from full BE solves. For low-frequency Helmholtz solves (Figure 11 left), our results agree with the brute-force solution very

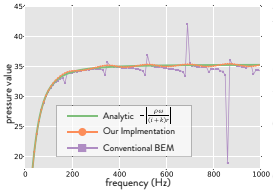


Fig. 13: **BEM Comparison:** Using a pulsating sphere with known analytic solution, our BE implementation (orange) agrees with the analytic solution (green) as frequency sweeps, whereas the CBIE solver (purple) has large error at fictitious frequencies.

Example	(i) Complexity		(ii) Mesh Simplification						(iii) Adaptive Freq. Sweep			(iv) Runtime Evaluation				
	# tet.	# modes	before (avg.) # tri.	before (avg.) BE Solve	after (avg.) # tri.	after (avg.) BE Solve	simp. time	speedup	before # solves	after # solves	speedup	before size	before time	after size	after time	speedup
PLATE	404k	59	100k	19m	5750	4.2m	16.8m	4.2×	4740	253	17.2×	8.1MB	59m	5.1MB	12.9s	274×
MUG	292k	61	68k	35m	7255	6.1m	14.7m	5.5×	4492	379	11.3×	8.7MB	96m	5.4MB	13.6s	424×
BUNNY	130k	56	30k	52m	4297	4.6m	10.2m	10.1×	3360	198	14.5×	7.7MB	132m	4.8MB	22.2s	356×
BOTTLE (solid)	160k	197	35k	21m	4139	4.1m	30.6m	4.9×	13396	1068	10.4×	30.2MB	237m	22.1MB	28.9s	492×
IJUMP	62k	70	14k	14m	3123	3.8m	6.5m	3.7×	5075	267	17.6×	9.2MB	96m	6.0MB	24.8s	232×
STAIRS	78k	49	12k	14m	5425	5.8m	21.2m	2.9×	3626	221	13.2×	6.7MB	38m	4.2MB	11.6s	197×
OLOID (shell)	-	300	32k	29m	7841	5.6m	28.4m	4.9×	12623	715	17.1×	62.4MB	258m	26.1MB	12.2s	1270×
COW (shell)	-	300	65k	42m	6406	5.1m	40.3m	7.8×	14131	624	22.2×	61.2MB	312m	25.7MB	23.8s	785×
BOTTLE (shell)	-	200	35k	23m	5364	4.7m	36.6m	4.4×	9246	436	20.5×	42.7MB	186m	19.9MB	19.4s	575×

Table I. : **Statistics and Timings**: (i) the size of tetrahedral meshes and modes; (ii) the averaged number of triangles before and after mesh simplification, the averaged BE solve time with and without simplification, the mesh simplification time, and the speedup to compute transfers of all modes, (iii) the total number of Helmholtz solves without and with adaptive frequency sweep, the speedup achieved using adaptive frequency sweep with simplified meshes. (iv) the memory overhead for transfer evaluation without and with key-position least-squares solves, the timings of transfer update using standard BE solves on a 20-core cluster, the timings of transfer evaluation using our approach on a quad-core desktop, and the computational speedup. Note: the memory without key-position least-squares solves only represents the storage on a single frequency. This storage increases as we sweep through the frequency range $\mathcal{R}$, whereas our model uses a fixed memory.

well. As expected, the high-frequency solves (Figure 11 right) are numerically more challenging, and our approximation degrades. However, Figure 12 shows the convergence of our approximation as the number of key positions increases. Therefore, we can always increase the accuracy of our runtime approximation by adding more key points. This feature provides the user easy control of the performance-accuracy tradeoffs for specific applications.

Lastly, Figure 13 validates our BEM implementation with the conventional CBIE approach. As shown, our implementation based on the Burton-Miller method [Burton and Miller 1971] is more robust and agrees with the analytical solution very closely.

7. RESULTS

7.1 Sound Editing Examples

Our interactive transfer estimation enables flexible and efficient approaches to tweak modal sound parameters, explore different sound characteristics, and achieve desirable sound effects. We now demonstrate with three applications. All the animations are simulated using [Kaufman et al. 2008] except IJUMP is from [Tan et al. 2012]. Please see the accompanying video for full results.

Fast Parameter Editing Modal sound models are often used to synthesize sounds automatically synchronized with simulated animations. To achieve certain sound characteristics, the user might start with physical parameters of target materials. However, even for a single material, its material parameters are given in a range. For instance, polyethylene, a common plastic, has a Young's modulus in a range from 0.11GPa to 0.45GPa, which doubles the modal frequencies when changed from the lower end to the upper end. In addition, there are no mechanically well-defined damping parameters [Adhikari and Woodhouse 2001], although the damping can significantly affect the sound perception [Klatzky et al. 2000]. Consequently, one has to rely on a trial-and-error approach to tune the parameters. It is therefore desirable to have a fast sound synthesis method to shorten the tuning cycle. In our examples, we take a rigid-body simulation as input, and edit Young's modulus and damping ratio to synthesize sounds produced by different materials ranging from wood, plastic, porcelain to metal (see Figure 14.**h**). Our runtime synthesis time is always less than 30 seconds.

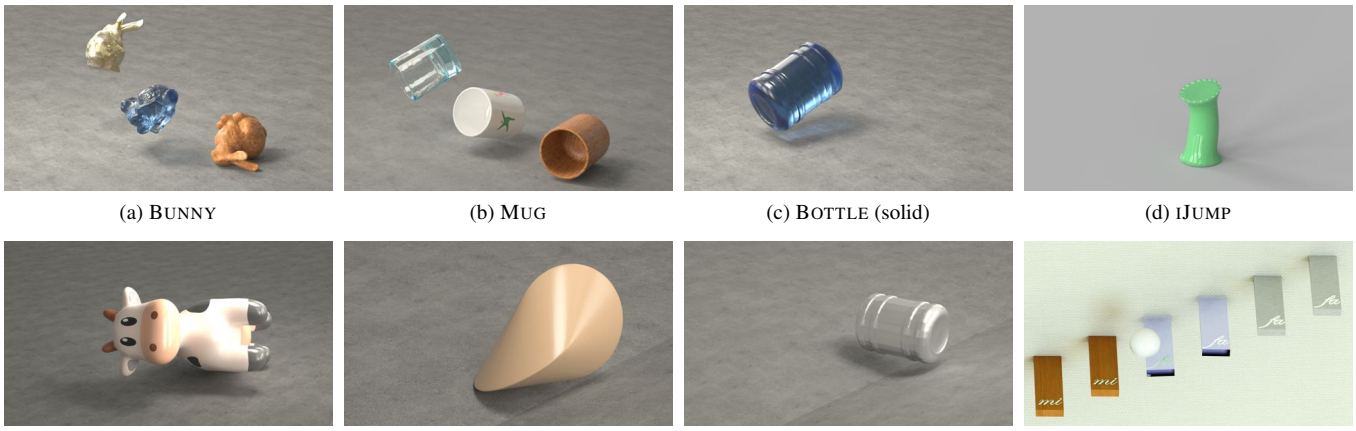
Parameter Space Exploration Our method allows the user to continuously explore the parameter space. In our implementation, we present the user with a 2D parameter space whose two axes are damping scale and stiffness scale respectively (see the video). When the user clicks a point in the coordinate system, we immediately synthesize the sounds with corresponding stiffness and damping values and present to the user. Take STAIRS as a demonstration. With a single pass of precomputation, we explore the parameter space, and identify a set of parameters that produces different pitches corresponding to a set of music notes with different timbres. After we are satisfied with the resulting sound characteristics, we use the parameters to generate sounds of more complex animations. In STAIRS, we choose three different materials and produce sounds that match the melody "Song of the Wind" (see Figure 14.**h**).

Thin-Shell Modal Models Our method is not limited to editing solid modal sound models. We also apply our method to edit thin-shell modal sound models. In the precomputation, we compute thin-shell modal matrices and vibration frequencies following the method proposed by Chadwick et al. [2009]. The rest of the pipeline is exactly the same as the solid modal sound model. In the accompanying video, we demonstrate different thin-shell sound effects edited using our method (see Figure 14.**e**, **f** and **g**).

Extension: Time-varying Frequency Effects Finally, we extend our method to allow the user to specify time-varying parameters, we can thus approximate sound effects with frequency shift, which is usually caused by nonlinear modal vibrations. This extension is straightforward: with user-guided time-varying parameters, we sample the values across the temporal domain and evaluate transfers for smooth interpolation. The modal vibration equation (3) with time-varying coefficients is still integrated using Runge-Kutta method as presented in §4.1. In our examples, we used the time-varying stiffness scale to mimic the nonlinear pitch changes, such as pitch gliding [Penttinen et al. 2006] (see Figure 14.**c**). We also explored an example in which the user specifies nonphysical time-varying frequencies to produce interesting artistic effects such as the one in IJUMP (see Figure 14.**d**).

7.2 Preliminary User Studies

Experiment Setup We perform four user studies to evaluate the perceptual quality of different levels of transfer approximation accuracy in our method.



(a) BUNNY (b) MUG (c) BOTTLE (solid) (d) IJUMP

(e) COW (shell) (f) OLOID (shell) (g) BOTTLE (shell) (h) STAIRS

Fig. 14: (**a** and **b**) We edit the Young's modulus and damping values at runtime, and produce sound effects corresponding to various materials. (**c** and **d**) We explore the examples that allow the user to change individual frequency values in a time-varying way, producing nonlinear artistic effects while retaining physical realism. (**e, f** and **g**) We apply our method to edit sound effects of thin shells. (**h**) We explore modal sound parameters so that each stair makes sounds corresponding to a music note, and the entire ball bouncing sequence produces a melody.

(1) We generate sounds using four different sets of hit locations and impact forces. For each setting, we compute three versions of sounds, denoted as A, B, and C, using different approximation accuracies. The accuracy is measured as the normalized least-squares residual as used in §6 (Table II). We then perform the Two-alternative forced choice (2AFC) tests: we generate three pairs of sounds, AB, BC and AC, for every set of sounds, and present the human subjects with each pair of sounds side by side, along with a reference sound generated without transfer approximation. The subjects are asked which sound in each pair is closer to the reference.

(2) We ask the subjects to rank the similarity of pairs of sounds, one approximated sound from above and one reference audio, on a Likert scale (i.e., choosing from "very similar", "similar", "neutral", "different", and "very different"). We repeat this experiment for all sound settings computed in previous study. This study is to examine preliminarily the correlation between the numerical errors and the perceived difference.

(3) Then, for each sound generation setting, we choose different number of samples to estimate transfer values. This results in sounds with different accuracies (measured by the normalized L_2 error in Table III). We present the subjects with each of these sounds and a fully simulated sound, along with the reference sound. We then ask them which sound (the approximated sound or the fully simulated sound) is closer to the reference.

(4) Lastly, we validate our assumption of using fixed modal shapes. For each test example, we computed two sounds, one with varying modal shapes and one with fixed modal shapes. We use the BUNNY example with three materials (wood, porcelain and metal). When building the modal sound model with fixed modal shapes, we use our method to simplify meshes. While the mesh simplification is frequency dependent, on average we observed more than $20\times$ reduction of the number of triangles. We then ask the subjects to rate the similarity of the two sounds on a Likert scale. We present these sounds in a random order to avoid possible biase from ordering.

sound	A	B	C
# samples	100%	50%	25%
averaged error	10^{-9}	0.299	0.641
winning percentage	86.6%	47.8%	15.6%

Table II. : **Statistics of the first user study.** The error is measured as the normalized least-squares residual. As the error increases, fewer and fewer subjects consider the approximated sounds to be similar to the reference.

# samples	100%	80%	60%	40%	25%
error	10^{-9}	0.08	0.19	0.37	0.53
winning percentage	47.5%	32.5%	5%	2.5%	2.5%

Table III. : **Statistics of the third user study.** As the error increases, more and more subjects can perceive the difference between the reference sound and the approximated sound.

Analysis of the Results We conducted the experiments with 40 subjects. In the first pairwise comparisons, we aggregated the results from all examples. Overall, 86.6% of the subjects thought the full-sample sound was more similar to the reference sound among all three sounds. Using half and a quarter of the samples won 47.8% and 15.6% of user selections, respectively. (see Table II).

Figure 15 visualizes the results of the second experiment. For each sound (of A, B and C), we plot its frequencies of being classified on each Likert scale category. We found that as the approximation error increases, it becomes easier for the subjects to notice the difference between the approximated sounds and the reference.

The third study shows that when the error is very small, the approximated sound and the reference sound were indistinguishable. As the L_2 error is slightly increased to 0.08, 32.5% of the subjects perceived the approximated sound to be similar to the reference sound. As the number of samples drops below 60%, we observed a clear decline in the perceived similarity. In other words, subjects were able to discern the difference once the errors were above 0.08 (see Table III).

Lastly, in the fourth user study, 82.5% of the subjects considered two sounds with fixed and varying modal shapes to be "Very simi-

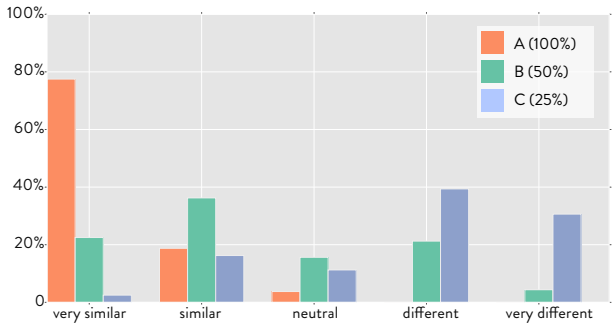


Fig. 15: **Statistics of the second user study.**

lar", and 17.5% of them chose "Similar". This suggests that using fixed modal shapes for fast transfer approximation is indeed plausible.

8. CONCLUSION

We have presented the method for fast runtime modal sound synthesis via efficient and more general precomputation. It greatly eases parameter tuning for desirable sound effects, and has the ability to generate various sound effects even using a single model. Our efficient runtime synthesis is realized by solving small-scale least-squares problems to estimate multipole coefficients of transfer functions. The least-squares formulation relies on precomputation. To improve its efficiency, we utilize Padé approximant to sample key frequencies adaptively and propose a volume-velocity-preserving mesh simplification algorithm to speed up individual Helmholtz solve. With numerical comparisons and user studies, we demonstrate its use in sound synthesis applications such as fast parameter tuning for various sound effects, and extend it to support the creation of time-varying sound effects. We augment and leverage several numerical techniques throughout, such as Prony's method and Padé approximant, hoping that these tools can be useful in other graphics research areas as well.

Limitations and Future Work Although we have shown that our approximation is comparable to the ground-truth results both numerically and qualitatively, it remains unexplored how far we can go to further speed up the computation. For example, can we take even coarser samples and solve Helmholtz on even simpler meshes while maintaining the perceptual plausibility? In addition, it is well-known that the Helmholtz problem at higher frequencies tends to be more ill-conditioned and thus numerically more challenging. This difficulty is also observed in our experiments, as our least-squares solves in §6 can not perfectly agree with the accurate solutions for frequencies higher than 12kHz, and the numerical error of transfer solves (shown in Figure 10) becomes larger as the frequency increases. For modal vibration sound, the high-frequency modes have large damping coefficients, and therefore this inaccuracy is hardly noticeable. However, when extending this method for editing other sound models such as fluid sounds, we hope to have a more accurate high-frequency approximation. In the proposed model we are able to generate different sound effects such as wood, porcelain, metal, etc.. With different input models, the results of linear model analysis, mostly the modal frequencies, are very distinct. As a result, it may require different parameters to achieve similar sound effects. One possible extension is to build a geometry-invariant measure such that a set of parameters can produce similar sound effects regardless of the input model geometry. Moreover, as observed in the OLOID (shell) example, different transfer approaches may produce similar sounds that the users cannot distinguish. We would like to better understand the reason that causes this ambiguity, which might in turn suggest a way to exploit this ambiguity. One common feedback from users is that the stiffness and damping parameters are not very intuitive at the beginning; they only started realizing their different effects during the second or even third trial. Therefore, one possible future work is to identify more intuitive sound model parameters for user adjustment. Finally, another interesting direction is to investigate a combination of our method and traditional Foley sound tools based on sound recording and granular synthesis to circumvent the numerical difficulties at high frequencies.

Acknowledgements

We thank the anonymous reviewers for their feedback. We also thank Jeff Chadwick for sharing the code of Harmonic Shells, Jie Tan for sharing the IJUMP animation data, Timothy Sun for adding the voice in the video, Breannan Smith for sharing the RoSI code, and Henrique Maia for helping revise an early draft. This research was supported in part by the National Science Foundation (CAREER-1453101) and generous donations from Intel. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funding agencies or others.

REFERENCES

ADHIKARI, S. AND WOODHOUSE, J. 2001. Identification of damping: Part 1, viscous damping. *Journal of Sound and Vibration 243,* 1, 43 – 61.

BONNEEL, N., DRETTAKIS, G., TSINGOS, N., VIAUD-DELMON, I., AND JAMES, D. 2008. Fast modal sounds with scalable frequency-domain synthesis. *ACM Trans. on Graph. 27,* 3 (Aug.).

BURTON, A. AND MILLER, G. 1971. The application of integral equation methods to the numerical solution of some exterior boundary-value problems. In *Proceedings of the Royal Society of London*. Series A. Math Phys Sci. 201–10.

CHADWICK, J. N., AN, S. S., AND JAMES, D. L. 2009. Harmonic Shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Trans. on Graph. 28,* 5, 1–10.

CHOWNING, J. 1973. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, J. Audio Eng. Soc. 21 (7), 526–534.

CISCOWSKI, R. AND BREBBIA, C. 1991. *Boundary Element methods in acoustics*. Computational Mechanics Publications and Elsevier Applied Science, Southampton.

COOK, P. R. 2002. *Real Sound Synthesis for Interactive Applications*. A. K. Peters, Ltd., Natick, MA, USA.

CORBETT, R., VAN DEN DOEL, K., LLOYD, J. E., AND HEIDRICH, W. 2007. Timbrefields: 3d interactive sound models for real-time audio. *Presence 16,* 6, 643–654.

CREMER, L., HECKL, M., AND PETERSSON, B. 2005. *Structure-Borne Sound: Structural Vibrations and Sound Radiation at Audio Frequencies*. Springer.

FUNKHOUSER, T. A., MIN, P., AND CARLBOM, I. 1999. Real-time acoustic modeling for distributed virtual environments. In *SIGGRAPH*. 365–374.

GALLIVAN, K., GRIMME, E., AND DOOREN, P. V. 1994. Asymptotic waveform evaluation via a lanczos method. *Appl. Math. Lett. 7,* 5, 75–80.

GARLAND, M. AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *SIGGRAPH*. 209–216.

GUMEROV, N. A. AND DURAISWAMI, R. 2004. *Fast Multipole Methods for the Helmholtz Equation in Three Dimensions*, first ed. Elsevier Science.

HAMMING, R. W. 1983. *Digital Filters*. Prentice-Hall.

HAUER, J., DEMEURE, C., AND SCHARF, L. 1990. Initial results in prony analysis of power system response signals. *Power Systems, IEEE Transactions on 5,* 1 (Feb), 80–89.

HOPPE, H. 1999. New quadric metric for simplifying meshes with appearance attributes. In *IEEE Visualization*. 59–66.

JAMES, D. L., BARBIC, J., AND PAI, D. K. 2006. Precomputed Acoustic Transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. on Graph. 25,* 3 (July), 987–995.

JAMES, D. L. AND PAI, D. K. 2002. Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. on Graph. 21,* 3 (July).

JERRI, A. J. 2005. The Shannon sampling theorem – Its various extensions and applications: A tutorial review. *Proceedings of the IEEE 65,* 11 (June), 1565–1596.

JOHNSON, M. E. AND ELLIOTT, S. J. 1995. Active control of sound radiation using volume velocity cancellation. *Journal of the Acoustical Society of America 4,* 98 (May), 2174–2186.

KARLSSON, J. 1976. Rational interpolation and best rational approximation. *Journal of Mathematical Analysis and Applications 53,* 1, 38–52.

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. *ACM Trans. on Graph. 27,* 5 (Dec.), 164:1–164:11.

KLATZKY, R. L., PAI, D. K., AND KROTKOV, E. P. 2000. Perception of material from contact sounds. *Presence: Teleop. Virtual Environ. 9,* 4 (Aug.), 399–410.

LANGLOIS, T. R., AN, S. S., JIN, K. K., AND JAMES, D. L. 2014. Eigenmode compression for modal sound models. *ACM Trans. on Graph. 33,* 4.

LENZI, M. S., LEFTERIU, S., BERIOT, H., AND DESMET, W. 2013. A fast frequency sweep approach using padé approximations for solving helmholtz finite element models. *Journal of Sound and Vibration 332*, 1897–1917.

LINDSTROM, P. AND TURK, G. 1998. Fast and memory efficient polygonal simplification. In *IEEE Visualization*. 279–286.

LIU, Y. J. 2009. *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*. Cambridge University Press.

LLOYD, B., RAGHUVANSHI, N., AND GOVINDARAJU, N. K. 2011. Sound synthesis for impact sounds in video games. In *Symposium on Interactive 3D Graphics and Games*.

LOBOS, T., REZMER, J., AND SCHEGNER, P. 2003. Parameter estimation of distorted signals using prony method. In *Power Tech Conference Proceedings, 2003 IEEE Bologna*. Vol. 4. 5 pp.

LUEBKE, D. P. 2001. A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl. 21,* 3, 24–35.

MATSUMOTO, T., ZHENG, C., HARADA, S., AND TAKAHASHI, T. 2010. Explicit evaluation of hypersingular boundary integral equation for 3-d helmholtz equation discretized with constant triangular element. *J Comput Sci Technol 4,* 3, 194–206.

MEHRA, R., RAGHUVANSHI, N., ANTANI, L., CHANDAK, A., CURTIS, S., AND MANOCHA, D. 2013. Wave-based sound propagation in large open scenes using an equivalent source formulation. *ACM Trans. on Graph. 32,* 2 (Apr.), 19:1–19:13.

MEYER, M. AND ANDERSON, J. 2007. Key point subspace acceleration and soft caching. *ACM Trans. on Graph. 26,* 3 (July).

O'BRIEN, J. F., COOK, P. R., AND ESSL, G. 2001. Synthesizing sounds from physically based motion. In *Proceedings of ACM SIGGRAPH 2001*. Computer Graphics Proceedings, Annual Conference Series. 529–536.

O'BRIEN, J. F., SHEN, C., AND GATCHALIAN, C. M. 2002. Synthesizing sounds from rigid-body simulations. In *The ACM SIGGRAPH 2002 Symposium on Computer Animation*. 175–181.

PAI, D. K., VAN DEN DOEL, K., JAMES, D. L., LANG, J., LLOYD, J. E., RICHMOND, J. L., AND YAU, S. H. 2001. Scanning physical interaction behavior of 3d objects. In *SIGGRAPH*. 87–96.

PELZER, S. AND VORLÄNDER, M. 2010. Frequency-and time-dependent geometry for real-time auralizations. In *Proceedings of 20th International Congress on Acoustics (ICA)*.

PENTLAND, A. AND WILLIAMS, J. 1989. Good vibrations: model dynamics for graphics and animation. In *SIGGRAPH*. Vol. 23. 215–222.

PENTTINEN, H., PAKARINEN, J., VÄLIMÄKI, V., LAURSON, M., LI, H., AND LEMAN, M. 2006. Model-based sound synthesis of the guqin. *J. of the Acoustical Society of America 120,* 6.

PILLAGE, L. T. AND ROHRER, R. A. 1990. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. Computer-Aided Design 9*, 352–366.

PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. 2007. *Numerical Recipes: The art of scientific computing*. Cambridge University Press.

RAGHUVANSHI, N. AND LIN, M. C. 2006. Interactive sound synthesis for large scale environments. In *SI3D*. 101–108.

RAGHUVANSHI, N., SNYDER, J., MEHRA, R., LIN, M., AND GOVINDARAJU, N. 2010. Precomputed wave simulation for real-time sound propagation of dynamic sources in complex scenes. *ACM Trans. on Graph. 29,* 4 (July), 68:1–68:11.

REN, Z., YEH, H., AND LIN, M. 2010. Synthesizing contact sounds between textured objects. In *IEEE Virtual Reality*.

REN, Z., YEH, H., AND LIN, M. C. 2013. Example-guided physically based modal sound synthesis. *ACM Trans. on Graph. 32,* 1, 1:1–1:16.

SHABANA, A. A. 1991. *Theory of Vibration, Volume II: Discrete and Continuous Systems*, first ed. Springer-Verlag, New York.

SILTANEN, S., LOKKI, T., SAVIOJA, L., AND LYNGE CHRISTENSEN, C. 2008. Geometry reduction in room acoustics modeling. *Acta Acustica united with Acustica 94,* 3, 410–418.

SMITH, J. O. 1985. A new approach to digital reverberation using closed waveguide networks. *International Computer Music Conference* STAN-M-31, 47–53.

STRAWN, J. 1987. Editing time-varying spectra. *J. Audio Eng. Soc 35,* 5, 337–352.

TAKALA, T. AND HAHN, J. K. 1992. Sound rendering. In *SIGGRAPH*. 211–220.

TAN, J., TURK, G., AND LIU, C. K. 2012. Soft body locomotion. *ACM Trans. on Graph. 31,* 4 (July), 26:1–26:11.

TSINGOS, N., CARLBOM, I., ELBO, G., KUBLI, R., AND FUNKHOUSER, T. 2002. Validation of acoustical simulations in the "Bell Labs Box". *IEEE Computer Graphics and Applications 22,* 4 (June), 28–37.

TSINGOS, N., DACHSBACHER, C., LEFEBVRE, S., AND DELLEPIANE, M. 2007. Instant sound scattering. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*. 111–120.

TSINGOS, N., FUNKHOUSER, T. A., NGAN, A., AND CARLBOM, I. 2001. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *SIGGRAPH*. 545–552.

VAN DEN DOEL, K., KRY, P. G., AND PAI, D. K. 2001. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *SIGGRAPH*. 537–544.

VAN DEN DOEL, K. AND PAI, D. K. 1996. Synthesis of shape dependent sounds with physical modeling. In *Intl Conf. on Auditory Display*. Xerox PARC, Palo Alto.

ZHENG, C. AND JAMES, D. L. 2010. Rigid-body fracture sound with precomputed soundbanks. *ACM Trans. on Graph. 29,* 4 (July).

ZHENG, C. AND JAMES, D. L. 2011. Toward high-quality modal contact sound. *ACM Trans. on Graph. 30,* 4 (Aug.).

APPENDIX

A. PRONY'S METHOD FOR TRANSFER COMPUTATION

Consider M complex-valued transfer samples $p(\omega_t), t = 0, \ldots, M-1$ that are uniformly sampled in a frequency range $\mathcal{R}$. We seek a N-th order Prony's series to approximate it,

$$p(\omega_t) \approx \sum_{i=1}^{N} c_i e^{\mu_i \omega_t} = \sum_{i=1}^{N} c_i \lambda_i^{\omega_t}. \quad (19)$$

where $\lambda_i = e^{\mu_i}$. First, we define a N-th order polynomial

$$\psi(z) = \prod_{i=1}^{N}(z-\lambda_i) = z^N + a_1 z^{N-1} + \ldots + a_{N-1} z + a_N, \quad (20)$$

which has N roots, $\lambda_i, i = 1, \ldots, N$. And thus $\lambda_i^k \psi(\lambda_i) = 0, \forall k \geq 0$. Next, notice the equality relationship,

$$\sum_{i=1}^{N} \lambda_i^k c_i \psi(\lambda_i) = p(\omega_{N+k}) + a_1 p(\omega_{N+k-1}) + \ldots + a_N p(\omega_k) = 0.$$

This is a linear equation of $a_i, i = 1, \ldots, N$. Let k go from $[0, \ldots, M-N-1]$. We form a linear system

$$\begin{bmatrix} p(\omega_{N-1}) & p(\omega_{N-2}) & \ldots & p(\omega_0) \\ p(\omega_N) & p(\omega_{N-1}) & \ldots & p(\omega_1) \\ \vdots & \vdots & \ddots & \vdots \\ p(\omega_{M-2}) & p(\omega_{M-3}) & \ldots & p(\omega_{M-N-1}) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} p(\omega_N) \\ p(\omega_{N+1}) \\ \vdots \\ p(\omega_{M-1}) \end{bmatrix}.$$

As long as we choose $N \leq M/2$, this is an over-constrained least-squares system with a unique solution a_i. Knowing the polynomial coefficients a_i of (20) allows us to find all its roots λ_i, and thus compute μ_i. Finally, after substituting λ_i into (19), we form another least-square system to solve c_i.

B. HELMHOLTZ BOUNDARY ELEMENT SOLVE

Our adaptive frequency sweeping algorithm in §5.2 samples frequency points and solves the Helmholtz equation. Our basic Helmholtz solver uses a BEM introduced in [Matsumoto et al. 2010]. Here we sketch out the important formulas to make the paper self-contained.

The BE solver is built upon the Kirchhoff integral formula, which is also used in [Tsingos et al. 2007].

$$p(\boldsymbol{x}) = \int_S \left[G(\boldsymbol{x};\boldsymbol{y}) \frac{\partial \phi}{\partial \boldsymbol{n}}(\boldsymbol{y}) - \frac{\partial G}{\partial \boldsymbol{n}}(\boldsymbol{x};\boldsymbol{y}) \phi(\boldsymbol{y}) \right] dS(\boldsymbol{y}), \quad (21)$$

where S denote the entire object surface; $G(\boldsymbol{x};\boldsymbol{y}) = \frac{e^{ik\|\boldsymbol{x}-\boldsymbol{y}\|}}{4\pi\|\boldsymbol{x}-\boldsymbol{y}\|}$ is the free-space Helmholtz Green's function; ϕ is the surface transfer value resulting from the BE solve (10); and $\frac{\partial \phi}{\partial \boldsymbol{n}}$ is the surface normal derivative of the acoustic transfer, as specified in the Helmholtz Neumann boundary condition (4). Once we have known the acoustic transfer $\phi(\boldsymbol{y})$ and its normal derivative $\frac{\partial \phi}{\partial \boldsymbol{n}}$ on object surface, we can use this integral formula to evaluate the transfer function at any location $\boldsymbol{x}$.

When the evaluation point is on the surface, i.e., $\boldsymbol{x} \in S$, we have the conventional boundary integral equation (CBIE),

$$\frac{1}{2}\phi(\boldsymbol{x}) = \oint\!\!\!\!\!\!-_S [G(\boldsymbol{x},\boldsymbol{y}) \partial_{\boldsymbol{n}} \phi(\boldsymbol{y}) - \phi(\boldsymbol{y}) \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})}] dS(\boldsymbol{y}). \quad (22)$$

Here we use $\oint\!\!\!\!\!\!-_S$ to indicate a Cauchy principal value at point $\boldsymbol{x}$ over the surface S. It is known that for the exterior Helmholtz problem, directly discretizing this equation using boundary elements fails to produce a unique solution at certain fictitious frequency values. Fictitious frequency results from numerical procedures and is related to the eigenfrequencies of the associated interior problem. The Burton-Miller method [Burton and Miller 1971] takes the directional derivation of (22) to get a hypersingular boundary integral equation (HBIE),

$$\frac{1}{2}\partial_{\boldsymbol{n}}\phi(\boldsymbol{x}) = \oint\!\!\!\!\!\!-_S \left[\frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})} \partial_{\boldsymbol{n}} \phi(\boldsymbol{y}) - \phi(\boldsymbol{y}) \frac{\partial^2 G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x}) \partial \boldsymbol{n}(\boldsymbol{y})} \right] dS(\boldsymbol{y}),$$

and uses a linear combination of CBIE and HBIE in boundary element discretization. Formally, it solves

$$\frac{1}{2}\phi(\boldsymbol{x}) + \mathsf{D}[\phi(\boldsymbol{y})] + \beta \mathsf{H}[\phi(\boldsymbol{y})] = \mathsf{S}[\partial_{\boldsymbol{n}}\phi(\boldsymbol{y})] + \beta \mathsf{M}[\partial_{\boldsymbol{n}}\phi(\boldsymbol{y})] - \frac{\beta}{2}\partial_{\boldsymbol{n}}\phi(\boldsymbol{y}), \quad (23)$$

where the integral operators D, H, S and M are respectively

$$\mathsf{D}[\phi(\boldsymbol{y})] = \oint\!\!\!\!\!\!-_S \phi(\boldsymbol{y}) \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y})$$
$$\mathsf{H}[\phi(\boldsymbol{y})] = \oint\!\!\!\!\!\!-_S \phi(\boldsymbol{y}) \frac{\partial^2 G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x}) \partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y})$$
$$\mathsf{S}[\phi(\boldsymbol{y})] = \oint\!\!\!\!\!\!-_S \partial_{\boldsymbol{n}}\phi(\boldsymbol{y}) G(\boldsymbol{x},\boldsymbol{y}) dS(\boldsymbol{y})$$
$$\mathsf{M}[\phi(\boldsymbol{y})] = \oint\!\!\!\!\!\!-_S \partial_{\boldsymbol{n}}\phi(\boldsymbol{y}) \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})} dS(\boldsymbol{y})$$

As long as the coefficient β has a nonzero imaginary part, this linear combination has a unique solution. A common practice is to choose $\beta = i/k$. This equation is then discretized and forms a dense linear system (10). In the integral equation (23), the surface transfer value $\phi(\boldsymbol{y})$ is unknown. Discretizing the equation (23) using the object's surface mesh yields a dense linear system (10) to solve for $\phi(\boldsymbol{y})$ on the surface.

C. DERIVATION OF (12)

To compute the nth order expansion coefficients $\boldsymbol{\phi}_i$ in (11), we take the n-th order derivative of (10) at ω_0, i.e.,

$$\sum_{i=0}^{n} C_n^i \mathsf{A}^{(i)}(\omega_0) \boldsymbol{\phi}^{(n-i)}(\omega_0) = \boldsymbol{b}^{(n)}(\omega_0), \quad (24)$$

where $C_n^i = \frac{n!}{i!(n-i)!}$ are the binomial coefficients. Noticing the n-th order derivative of the expansion (11) is $\boldsymbol{\phi}^{(n)}(\omega_0) = n!\boldsymbol{\phi}_n$, we substitute it into (24) and arrive (12) to solve for $\boldsymbol{\phi}_i$.

D. FREQUENCY DERIVATIVE OF LINEAR SYSTEM (10)

Our asymptotic waveform evaluation method involves the frequency derivative of (10) as derived in Appendix C. In particular, we need to compute

$$\frac{\partial^n \mathsf{A}(\omega)}{\partial \omega^n} \text{ and } \frac{\partial^n \boldsymbol{b}(\omega)}{\partial \omega^n}.$$

Their analytic forms can be computed by taking the derivatives of (23). In our implementation, we use piecewise constant boundary element. Therefore, $\partial_{\boldsymbol{n}}\phi(\boldsymbol{y})$ and $\phi(\boldsymbol{y})$ in (23) can be moved outside of the integral (i.e., the Cauchy principle value). For example, we discretize the first term of (23) as

$$\oint_S \phi(\boldsymbol{y}) \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y}) \approx \sum_{\text{triangle } i} \phi(\boldsymbol{y}_i) \oint_{\Delta_i} \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y}),$$

where Δ_i denotes the surface of the i-th triangle element, and $\phi(\boldsymbol{y}_i)$ is the constant sound pressure value at Δ_i. We evaluate the integral on Δ_i using a Gaussian quadrature scheme,

$$\oint_{\Delta_i} \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y}) \approx \sum_{j \text{ on } \Delta_i} w_j \frac{\partial G(\boldsymbol{x},\boldsymbol{y}_j)}{\partial \boldsymbol{n}(\boldsymbol{y}_j)},$$

for a set of Gaussian quadrature points $\boldsymbol{y}_j$ on Δ_i. This expression also shows that to compute the frequency derivative of (10) analytically we need to compute

$$\frac{\partial^n}{\partial \omega^n}\left(\frac{\partial G(\boldsymbol{x},\boldsymbol{y}_j)}{\partial \boldsymbol{n}(\boldsymbol{y}_j)}\right).$$

And similarly for other integral terms in (23), we need to compute

$$\frac{\partial^n}{\partial \omega^n}\left(\frac{\partial G(\boldsymbol{x},\boldsymbol{y}_j)}{\partial \boldsymbol{n}(\boldsymbol{x})}\right), \frac{\partial^n}{\partial \omega^n} G(\boldsymbol{x},\boldsymbol{y}_j) \text{ and } \frac{\partial^n}{\partial \omega^n}\left(\frac{\partial^2 G(\boldsymbol{x},\boldsymbol{y}_j)}{\partial \boldsymbol{n}(\boldsymbol{x})\partial \boldsymbol{n}(\boldsymbol{y}_j)}\right).$$

For the infinite-space Green's function $G(\boldsymbol{x},\boldsymbol{y}) = \frac{e^{ikr}}{4\pi r}$, where $r = \|\boldsymbol{x}-\boldsymbol{y}\|_2$, the analytic normal derivatives are

$$\begin{aligned}
\frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} &= -\frac{e^{ikr}}{4\pi r^2}(1-ikr)\frac{\partial r}{\partial \boldsymbol{n}(\boldsymbol{y})} \\
\frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})} &= -\frac{e^{ikr}}{4\pi r^2}(1-ikr)\frac{\partial r}{\partial \boldsymbol{n}(\boldsymbol{x})} \\
\frac{\partial^2 G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})\partial \boldsymbol{n}(\boldsymbol{y})} &= \frac{e^{ikr}}{4\pi r^3}[(-3+3ikr+k^2r^2)\frac{\partial r}{\partial \boldsymbol{n}(\boldsymbol{x})}\frac{\partial r}{\partial \boldsymbol{n}(\boldsymbol{y})} + \\
&\quad (1-ikr)\boldsymbol{n}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{y})]
\end{aligned}$$

The n-th order frequency derivative of these terms are polynomials with respect to k, because $k = \omega/c$ is linear in ω and appears only in e^{ikr} and the polynomials of k in these formulas. Finally, since $\beta = i/k$ depends on ω, we compute

$$\frac{\partial^n}{\partial \omega^n}\beta = (-1)^n n! i c \omega^{-n-1},$$

where c is the speed of sound.

E. LINEAR SOLVES FOR MESH SIMPLIFICATION

The quadric error function for collapsing an edge is

$$Q^v(\boldsymbol{v}) = \frac{1}{2}\boldsymbol{p}^T\mathsf{A}\boldsymbol{p} + \frac{1}{2}\boldsymbol{u}^T\mathsf{C}\boldsymbol{u} + \boldsymbol{p}^T\mathsf{G}\boldsymbol{u} + \boldsymbol{a}^T\boldsymbol{p} + \boldsymbol{b}^T\boldsymbol{u} + c_0,$$

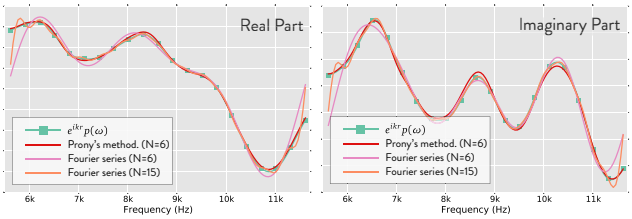


Fig. 16: **Prony Series vs Fourier Series:** We compare the approximation of frequency-dependent pressure curves using Prony series and Fourier series. The ground-truth pressure curves are the same as the ones in Figure 6. We plot the Prony approximation using 6 terms, and Fourier approximation using 6 and 15 Fourier basis functions respectively.

where A, G, and C are 3×3 matrices, $\boldsymbol{a}$ and $\boldsymbol{b}$ are 3D constant vectors, and c_0 is a constant scalar. We refer the reader to [Hoppe 1999] for their formulas. Initially, we minimize $Q^v(\boldsymbol{v})$ without constraints by solving

$$\begin{bmatrix} \mathsf{A} & \mathsf{G} \\ \mathsf{G} & \mathsf{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{u} \end{bmatrix} = -\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}.$$

Next, we iteratively solve the linearly constrained quadratic programming (LCQP) problems, (17) and (18). When we solve (17), both $\boldsymbol{p}^T\mathsf{A}\boldsymbol{p}$ and $\boldsymbol{a}^T\boldsymbol{p}$ are constant values, and the constraint (16) is linear to $\boldsymbol{u}$ with a form $\boldsymbol{t}^T\boldsymbol{u} + s = 0$, where $\boldsymbol{t} = \frac{1}{6}\sum_{f\in\mathcal{N}(v)}(\boldsymbol{p} - \boldsymbol{p}_{f1}) \times (\boldsymbol{p} - \boldsymbol{p}_{f2})$ following the notations in (16) and $s = \frac{1}{6}\sum_{f\in\mathcal{N}(v)}\left[(\boldsymbol{p} - \boldsymbol{p}_{f1}) \times (\boldsymbol{p} - \boldsymbol{p}_{f2})\right]^T(\boldsymbol{u}_{f1} + \boldsymbol{u}_{f2}) - C_v$. Using the method of Lagrange Multipliers, we solve this LCQP problem using a 4D linear system,

$$\begin{bmatrix} \mathsf{A} & \boldsymbol{t} \\ \boldsymbol{t}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \lambda \end{bmatrix} = -\begin{bmatrix} \boldsymbol{a} \\ s \end{bmatrix}.$$

Next, we fix $\boldsymbol{u}$ and solve the vertex position $\boldsymbol{p}$. This is an LCQP problem (18) with two linear equality constraints, including the volume preservation constraint $\boldsymbol{g}_{VOL}^T\boldsymbol{p} + d_{VOL} = 0$ and the volume-velocity constraint which is of a form $\boldsymbol{h}^T\boldsymbol{p} + n = 0$. Using Lagrange Multipliers, we solve a 5D linear system,

$$\begin{bmatrix} \mathsf{A} & \boldsymbol{g}_{VOL} & \boldsymbol{h} \\ \boldsymbol{g}_{VOL}^T & 0 & 0 \\ \boldsymbol{h}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = -\begin{bmatrix} \boldsymbol{a} \\ d_{VOL} \\ n \end{bmatrix}.$$

F. PRONY SERIES VS FOURIER SERIES

Here we compare the approximations using Prony series and Fourier series. As shown in Figure 16, Prony series with just 6 terms approximate the frequency-varying pressure curves (also shown in Figure 6) very closely. However, the Fourier series with only 6 basis oscillate dramatically at the beginning and the ending part of the frequency window. Increasing the number of Fourier basis (even using 15 terms) still cannot completely eliminate the oscillation.