NeuralSound: Learning-based Modal Sound Synthesis with Acoustic Transfer

XUTONG JIN, School of Computer Science, Peking University, China
SHENG LI*, School of Computer Science, Peking University, China
GUOPING WANG, School of Computer Science, Peking University, China
DINESH MANOCHA, University of Maryland at College Park, U.S.A

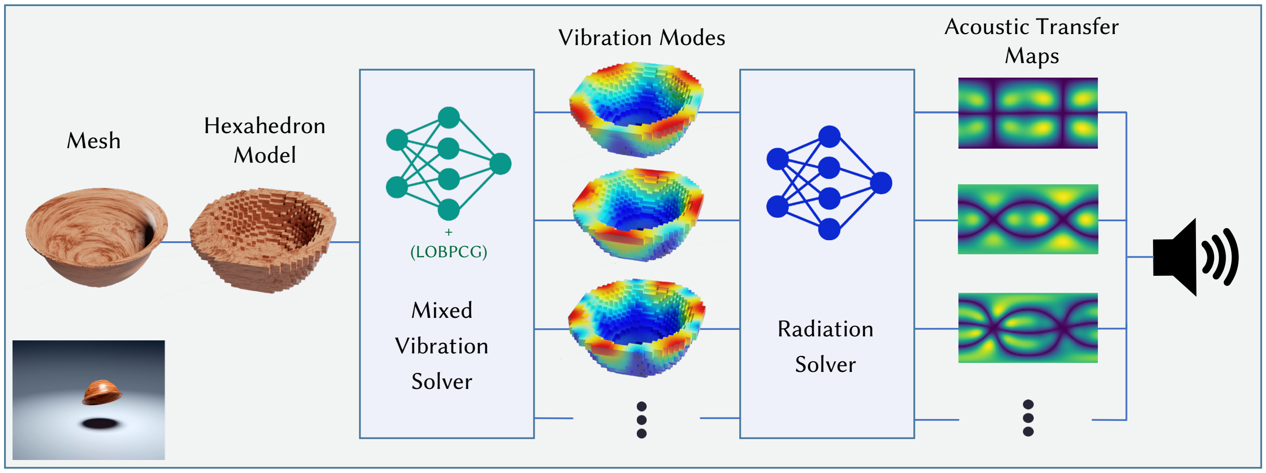


Fig. 1. Learning-based approach for modal sound synthesis: We use neural networks to accelerate both modal analysis and acoustic transfer precomputation, and evaluate the performance on many new and unseen objects. Our approach can solve both vibration and radiation for plausible sound effects within one second per object on a GeForce RTX 3080 Ti GPU.

We present a novel learning-based modal sound synthesis approach that includes a mixed vibration solver for modal analysis and a radiation network for acoustic transfer. Our mixed vibration solver consists of a 3D sparse convolution network and a Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) module for iterative optimization. Moreover, we highlight the correlation between a standard numerical vibration solver and our network architecture. Our radiation network predicts the Far-Field Acoustic Transfer maps (FFAT Maps) from the surface vibration of the object. The overall running time of our learning-based approach for most new objects is less than one second on a RTX 3080 Ti GPU while maintaining a high sound quality close to the ground truth solved by standard numerical methods. We also evaluate the numerical and perceptual accuracy of our approach on different objects with various shapes and materials.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**; • **Applied computing** → *Sound and music computing*.

Additional Key Words and Phrases: sound synthesis, deep learning, modal analysis, vibration, convolution networks, acoustic transfer, sound radiation

ACM Reference Format:
Xutong Jin, Sheng Li, Guoping Wang, and Dinesh Manocha. 2022. NeuralSound: Learning-based Modal Sound Synthesis with Acoustic Transfer. *ACM Trans. Graph.* 41, 4, Article 121 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530184

*corresponding author
Project URL: hellojxt.github.io/NeuralSound

Authors' addresses: Xutong Jin, jinxutong@pku.edu.cn, School of Computer Science, Peking University, China; Sheng Li, lisheng@pku.edu.cn, School of Computer Science, Peking University, China; Guoping Wang, wgp@pku.edu.cn, School of Computer Science, Peking University, China; Dinesh Manocha, dmanocha@umd.edu, University of Maryland at College Park, U.S.A.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
0730-0301/2022/7-ART121 $15.00
https://doi.org/10.1145/3528223.3530184

1 INTRODUCTION

3D modal sound models are used for synthesizing physically-based rigid-body sounds in animation and virtual environments [Bonneel et al. 2008; James et al. 2006; Liu and Manocha 2020; O'Brien et al. 2002; Raghuvanshi and Lin 2006; van den Doel et al. 2001]. The standard pipeline generally requires expensive precomputation for sound vibration and radiation for each object in the scene. The sound vibration data is obtained by modal analysis [van de Doel and Pai 1996; van den Doel et al. 2001], which consists of an eigenmode matrix and eigenfrequencies that correspond to the surface motions and frequencies of some vibration modes. The sound radiation data corresponds to the acoustic transfer function [James et al. 2006], which describes the radiation efficiency of each vibration mode at different spatial positions.

The standard numerical methods used for modal sound synthesis can be expensive, especially in terms of computing the sound radiation data [James et al. 2006; Li et al. 2015]. For a given object, a numerical vibration solver [Lanczos 1950] usually takes a few minutes [James et al. 2006; Jin et al. 2020; Zheng and James 2010], and a numerical radiation solver such as Boundary Element Method (BEM) [Kirkup 2019] can take several minutes to a few hours [James et al. 2006; Wang and James 2019; Zheng and James 2010]. This makes it hard to use these methods in interactive scenarios with dynamically generated objects corresponding to shape modification or fractured objects.

Main Results: We propose NeuralSound, a novel learning-based modal sound synthesis approach. It consists of a *mixed vibration solver* and a *radiation solver*. Our mixed vibration solver is a 3D linear sparse convolution network used to predict the eigenvectors of an object and an iterative optimizer that uses Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) [Duersch et al. 2018; Knyazev 2001]. We highlight the connection between a 3D sparse convolution network and the matrix calculations of standard numerical solvers for modal analysis to motivate the design of our network. We also present a neural network architecture based on the properties of BEM for acoustic transfer. Thus, our end-to-end radiation network encodes the surface displacement and frequency of a vibration mode and decodes it into the corresponding scalar-valued Far-Field Acoustic Transfer maps (FFAT Maps) [Chadwick et al. 2009; Wang and James 2019], which compress the acoustic transfer function at the level-of-accuracy suitable for sound rendering. The total computation for sound vibration and radiation of an object can be completed within one second on a commodity GPU (RTX 3080Ti). Moreover, these two learning-based solvers can either be combined or used independently. In other words, a numerical vibration solver can be combined with our learning-based radiation solver, or a numerical radiation solver can be combined with our learning-based vibration solver to design a sound synthesis system.

Our dataset is based on the ABC Dataset [Koch et al. 2019] and uses ~100K 3D models. We compare our mixed vibration solver with the standard numerical solution using different metrics. The result of a LOBPCG solver with a tight convergence tolerance is treated as the ground truth. Our mixed vibration solver obtains high accuracy (convergence error $\varepsilon < 0.01$) using only 0.03s and always outperforms other standard numerical solvers within the same time budget. We compare our radiation solver with the standard numerical solution (BEM). Our approach achieves high accuracy (MSE of normalized FFAT Map $\sim$ 0.06, and MSE of Log Norm $\sim$ 0.07) using only 0.04s. We observe ~2000× speedup using our radiation solver over BEM in terms of synthesizing plausible acoustic transfer effect.

We conduct a user study to evaluate our sound quality compared to two baselines in five different scenes. The statistical results (mean value and standard deviation) indicate that our approach, including both the mixed vibration solver and the radiation solver, shows very high fidelity to the ground truth.

2 RELATED WORKS

Modal Sound Synthesis. Modal sound synthesis is a technique that has been used to synthesize sounds of rigid bodies [Cook 1995; O'Brien et al. 2002; Raghuvanshi and Lin 2006; van den Doel et al. 2001]. These methods compute the vibration modes of a 3D object using eigendecomposition as a preprocessing. Many methods use 3D objects' pre-computed eigendata to render runtime sound, e.g., reducing the computational complexity by approximations [Bonneel et al. 2008]. Other methods use complex modal sound synthesis models to simulate sounds such as knocking, sliding, and friction models [van den Doel et al. 2001], acceleration noise synthesis models [Chadwick et al. 2012], accurate damping models [Sterling et al. 2019], contact models [Zheng and James 2011] or data-driven approximations [Pai et al. 2001; Ren et al. 2013].

Acoustic Transfer. Directly adding the vibration modes does not result in high-fidelity sound effects, as that process lacks the modal amplitude variations and spatialization effects due to acoustic wave radiation [Wang and James 2019]. In addition to BEM [Kirkup 2019], precomputed acoustic transfer [James et al. 2006] methods are used to more accurately model the sound pressure at the listener's position. Other techniques use a single-point multipole expansion with higher-order sources [Langlois et al. 2014; Rungta et al. 2016; Zheng and James 2010, 2011], inter-object transfer functions [Mehra et al. 2013], or Far-Field Acoustic Transfer (FFAT) Maps [Chadwick et al. 2009; Wang and James 2019]. Since the FFAT map is a simple, efficient representation of the transfer function, we also use scalar-valued FFAT Maps defined in KleinPAT [Wang and James 2019] in our approach.

Precomputation Speedup. Some methods focus on reducing the time-consuming precomputation of modal sound synthesis and acoustic transfer functions. Li et al. [2015] proposed a method to enable interactive acoustic transfer evaluation without recalculating the acoustic transfer when changing an object's material. KleinPAT [Wang and James 2019] can accelerate the precomputation of acoustic transfer on the GPU by packing-depacking modes and using a time-domain method. A deep neural network [Jin et al. 2020] was proposed to synthesize the sound of an arbitrary rigid body being hit at runtime and it predicts the amplitudes and frequencies of sound directly. However, this approach cannot model acoustic transfer. Moreover, it may incur a severe loss of accuracy because it treats all the modes within a frequency interval as one mode.

Learning-based Approaches. The computation of modal vibration and radiation is highly dependent on the geometric shape of the object, where solving for shape-related features is the key issue. Many methods have been proposed to utilize three-dimensional geometric features for learning, including perspective-based methods, which regard the combination of 2D image features from different viewing perspectives as the 3D geometry feature [Su et al. 2015], voxel-based methods [Wu et al. 2015], which regard an object as a 3D image by hexahedral meshing, and point cloud-based methods [Meng et al. 2021; Qi et al. 2017], which represent the model as a point cloud.

3 BACKGROUND

We first provide the background of the 3D physically-based modal sound synthesis as well as the acoustic transfer function for high-quality sound effect.

3.1 Vibration Solver

We begin with the linear deformation equation for a 3D linear elastic dynamics model [Shabana 1991] that is commonly used in rigid-body sound synthesis [James et al. 2006; O'Brien et al. 2002; Zheng and James 2010]:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}(t)\ , \tag{1}$$

where $\mathbf{u}$ is nodal displacements, and $\mathbf{M}$, $\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}$, and $\mathbf{K}$ represent the mass, Rayleigh damping, and stiffness matrices, respectively. α, β are Rayleigh damping coefficients. $\mathbf{F}(t)$ represents the external nodal forces. The generalized eigenvalue decomposition as

$$\mathbf{KU} = \mathbf{MU\Lambda} \tag{2}$$

is required first to solve the linear system, where $\mathbf{U}$ is the eigenmode matrix (consists of eigenvectors) and Λ is the diagonal matrix of eigenvalues. Then the system can be decoupled as:

$$\ddot{\mathbf{q}} + (\alpha + \beta\mathbf{\Lambda})\dot{\mathbf{q}} + \mathbf{\Lambda}\mathbf{q} = \mathbf{U}^T\mathbf{F}(t)\ , \tag{3}$$

where $\mathbf{q}$ satisfies $\mathbf{u} = \mathbf{Uq}$. The solution to the equation is a bank of damped sinusoidal waves corresponding to each mode. The generalized eigenvalue decomposition $\mathbf{KU} = \mathbf{MU\Lambda}$ is the core of modal analysis. Therefore, a generalized eigenvalue decomposition solver is required (e.g., Lanczos method [Lanczos 1950], LOBPCG [Duersch et al. 2018; Knyazev 2001]).

3.2 Acoustic Transfer Solve

Acoustic transfer function $p_i(\mathbf{x})$ describes the sound pressure in space (at position $\mathbf{x}$) generated by the ith mode with unit amplitude. A radiation solver is required to solve the acoustic transfer function from the surface displacement and frequency of a mode, where the surface displacement is computed from the eigenmode matrix $\mathbf{U}$ solved in Sec. 3.1, and the frequencies are computed from the eigenvalues Λ. BEM is a standard method used to solve this acoustic transfer problem. After solving the acoustic transfer function, compression methods are needed to represent the function for runtime sound rendering. These methods include Equivalent-Source Representations [James et al. 2006; Mehra et al. 2013], Multipole Source Representation [Zheng and James 2010], Far-Field Acoustic Transfer (FFAT) Maps [Chadwick et al. 2009], etc. In this paper, we choose FFAT Maps as our compression method.

3.3 Sound Synthesis

To synthesize modal sound at runtime, the eigenvalue matrix $\mathbf{\Lambda} = diag\{\lambda_1, \lambda_2, ..., \lambda_k\}$, eigenmode matrix $\mathbf{U}$, and acoustic transfer function $\mathbf{P}(\boldsymbol{x}) = \{p_i(\boldsymbol{x}) | i = 1, ..., k\}$ are precomputed and stored [Langlois et al. 2014]. The external nodal force $\mathbf{F}(t)$ is projected into the modal subspace by $\mathbf{U}^T\mathbf{F}(t) = \{f_0(t), f_1(t), ..., f_k(t)\}^T$, then the sound waveform $w(\boldsymbol{x}, t)$, i.e., the sound pressure at position $\boldsymbol{x}$ and time t, is solved as [James 2016]:

$$w(\boldsymbol{x}, t) = \sum_{i=1}^{k} p_i(\boldsymbol{x}) \int_0^t \frac{f_i(\tau)}{\omega_i'} e^{-\xi_i\omega_i(t-\tau)} \sin\left(\omega_i'(t-\tau)\right) \mathrm{d}\tau,$$

where $\omega_i = \sqrt{\lambda_i}$ is the undamped natural frequency of ith mode, $\xi_i = \frac{\alpha+\beta\lambda_i}{2\omega_i}$ is the dimensionless modal damping factor of ith mode, and $\omega_i' = \omega_i\sqrt{1-\xi_i^2}$ is the damped natural frequency of ith mode.

3.4 Network Architectures for Modal Sound Synthesis

Modal analysis and acoustic transfer precomputations can be expensive, making the current numerical solvers too slow for interactive applications. In contrast to these solvers, the inference process of a neural network can be completed in a very short time (even milliseconds) on current GPUs. Therefore, a learning-based approach to resolving modal sound synthesis can inherit the advantage of high efficiency.

Our approach exploits the intrinsic correspondence between the convolution neural network and physically-based sound synthesis methods. Specifically, local connections and shift-invariance are two characteristics of convolution neural networks (CNNs). We also observe similar characteristics for (i) the assembled matrix (i.e., stiffness matrix) in modal analysis and (ii) the interaction/convolution matrix in the BEM for acoustic transfer [Kirkup 2019]. Our approach is also inspired by the fact that multi-scale structure in convolution neural networks (e.g., in ResNet [He et al. 2016] and U-Net [Ronneberger et al. 2015]) has also been applied to (i) the multigrid method in linear solvers and (ii) the hierarchical strategy in the fast multipole method (FMM) [Liu 2009].

We design two neural networks that coincide with the fundamentals of numerical solvers for modal analysis and acoustic transfer but can predict approximate results much faster. This is the motivation behind our learning-based approach. Specifically, our approach includes (i) a mixed vibration solver for modal analysis and (ii) a radiation solver for acoustic transfer (see Figure 1). For our mixed vibration solver, we highlight its correspondence with the numerical solvers and present its details in Sec. 4. The correspondence with BEM and the details of our radiation solver are given in Sec. 5.

4 VIBRATION SOLVER: LEARNING EIGENVECTOR APPROXIMATION

In this section, we propose a self-supervised learning-based solver to resolve vibration problems introduced in Sec. 3.1, i.e., to resolve the core generalized eigenvalue decomposition (Equation 2). We explain the design of our vibration solver, present the architecture, and describe training our 3D sparse network by reducing residual-based error instead of using data generated from a numerical solver.

4.1 Network and Matrix Computations

The assembled matrix and its inverse are key components in numerical solvers for generalized eigenvalue decomposition in modal analysis. We first analyze the intrinsic connections between the assembled matrix and the 3D sparse convolution as well as the inverse assembled matrix and the 3D sparse U-Net.

4.1.1 Assembled Matrix and 3D Sparse Convolution. We use 3D sparse convolutions [Choy et al. 2019; Graham et al. 2018; Graham and van der Maaten 2017] as the basic components of our network. 3D sparse convolution can be applied to sparse tensors of any shape due to the shift-invariance of convolution kernel. In a $3 \times 3 \times 3$

sparse convolution, the convolution kernel defines the linear relationship between the output feature of each voxel and the input features of neighboring voxels (including itself). In an assembled matrix, the element matrix defines the linear relationship between the output displacement of each vertex and the input displacements of neighboring vertices (including itself) in a voxel. So, there is an intuitive connection between 3D sparse convolutions and assembled matrices. Assuming that the element matrix is fixed (voxel size and material are fixed), the assembled matrix-vector multiplication can be represented by a corresponding $3 \times 3 \times 3$ sparse convolution (see bottom-right of Figure 2). We provide detailed analysis and experimental validation in Appendix A.1.

4.1.2 Inverse Assembled Matrix and Sparse U-Net. Using a multigrid method [Briggs et al. 2000] can accelerate solving inverse matrix-vector multiplication through a hierarchy of discretizations, which refine the solution first from a fine to coarse model and then from coarse to fine. Each correction operation corresponds to matrix-vector multiplication and addition, equivalent to a 3D sparse convolution with bias if the involved matrix is an assembled matrix. Therefore, there is an intuitive connection between a 3D sparse linear U-Net [Ronneberger et al. 2015] and the inverse of an assembled matrix. We provide experimental validation in Appendix A.1 to show that 3D sparse linear U-Net can be a good approximation of an assembled matrix inverse.

4.1.3 Sparse Linear U-Net for Eigenvector Approximation. The matrix $\mathbf{K}^{-1}\mathbf{M}$ can be used to span the standard Krylov space where the Rayleigh–Ritz method [Knyazev 1997, 1998, 2001] is applied to resolve approximate eigenvectors within this space. We explain the feasibility and rationality of such a process used for modal analysis with example in Appendix A.2. In principle, a linear neural network with certain parameters can represent a linear mapping. Based on the above analysis, a 3D sparse linear U-Net can represent a linear mapping that approximates an assembled matrix (or inverse matrix). Therefore, we conjecture that the U-Net can be used to approximate a polynomial of an assembled matrix (or inverse matrix) when the network with multiple layers is deep enough. A 3D sparse linear U-Net with different parameters can represent various linear mappings. All these possible linear mappings make up the feasible domain of this U-Net. Like Krylov space, U-Net representation of the linear mappings can span a subspace and employ Rayleigh-Ritz method for eigenvector approximations. Based on the large feasible domain of deep neural networks, we can train the U-Net to search for the best possible linear mapping in its feasible domain rather than to simulate a specific one like $\mathbf{K}^{-1}\mathbf{M}$ of the krylov space.

4.2 Vibration Network Architecture

As discussed above, we design a 3D sparse linear U-Net to transform some random initial vectors into output vectors. The output vectors span a subspace, and the Rayleigh–Ritz method is used to compute the approximate eigenvectors and eigenvalues in this subspace. We use a self-supervised training strategy without any labels or groundtruth data. The convergence error of the approximate eigenvectors solved by Rayleigh–Ritz method is used as the loss function.

The overall pipeline is shown in Figure 2. Our U-Net consists of several linear residual blocks. The down-sampling layer in the U-Net is a 3D sparse convolution with a stride of 2. The upsampling layer in the U-Net is a 3D sparse transposed convolution with a stride of 2.

4.2.1 Projection between Vertex & Voxel. First, we introduce the mechanism to ensure the accurate correspondence between 3D sparse tensors and eigenvectors. Voxels in the network correspond to hexahedrons in the finite element model. An eigenvector represents the displacement of vertices in the direction of x, y, z caused by a unit vibration mode. A 3D sparse tensor in the neural network represents the feature of voxels. We design a projection from vertex representation to voxel representation and vice versa. The vertex to voxel projection concatenates all the features (displacement in x, y, z) of the current voxel's eight vertices. Therefore, the feature number of each voxel after projection is 24. For each vertex, 3 of the 24 features of adjacent voxels belong to this vertex. The voxel to vertex projection averages the corresponding three features of all adjacent voxels.

4.2.2 Training Process. We denote our neural network as g and its optimizable parameters as θ. There are N voxel models in the training dataset, each consisting of k randomly initialized vectors $\mathbf{x}^i = \left(\mathbf{x}_1^i, \mathbf{x}_2^i, ..., \mathbf{x}_k^i\right)$, which represent the randomly initialized nodal displacements. The goal is to find an optimal set of network parameters $\hat{\theta}$ to minimize the average loss function:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \ell \left(\text{Rayleigh-Ritz}\left(g(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{x}^i\right)\right) , \qquad (4)$$

where $g(\mathbf{x}^i; \boldsymbol{\theta}) = \left(g(\mathbf{x}_1^i; \boldsymbol{\theta}), g(\mathbf{x}_2^i; \boldsymbol{\theta}), ..., g(\mathbf{x}_k^i; \boldsymbol{\theta})\right)$ represents k output vectors from the network, and ℓ is the loss function. All computation are differentiable, so the common gradient optimization method can be employed. The loss function and the Rayleigh-Ritz method will be described in detail in the next subsection.

4.2.3 Rayleigh-Ritz Method & Loss Function. The Rayleigh-Ritz method is used to find the approximate eigenvectors and eigenvalues in a subspace:

$$(\mathbf{S}^T\mathbf{K}\mathbf{S})\hat{\mathbf{U}} = (\mathbf{S}^T\mathbf{M}\mathbf{S})\hat{\mathbf{U}}\hat{\mathbf{\Lambda}} , \qquad (5)$$

where S is a set of basis of the linear subspace $\mathcal{S} = \text{span}\{g(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{x}^i\}$ and $\mathbf{K}, \mathbf{M}$ are stiffness matrix and mass matrix, respectively. The Ritz pairs $(\hat{\mathbf{\Lambda}}, \mathbf{S}\hat{\mathbf{U}})$ are approximations to the eigenvalues and eigenvectors in the original problems. It turns out that $k \ll 3N$, where N is the number of vertices. Therefore, a solution to the generalized eigenvalue problem in subspace (Equation 5) is much faster than directly solving the original generalized eigenvalue problem in modal analysis (Equation 2).

The vectors in $g(\mathbf{x}^i; \boldsymbol{\theta})$ may be linearly dependent. In this case, the dimension of the spanned subspace $\mathcal{S} = \text{span}\{g(\mathbf{x}^i; \boldsymbol{\theta})\}$ is less than k and the number of eigenvectors that can be solved in the Rayleigh-Ritz process is also less than k. To ensure k eigenvectors, we supplement random vectors $\mathbf{x}^i$ into subspace $\mathcal{S}$ as $\mathcal{S} = \text{span}\{g(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{x}^i\}$ because $\mathbf{x}^i$ are linearly independent.

A numerical instability issue may occur in the Rayleigh-Ritz process in Equation 5. This is due to the fact that the projection $\mathbf{S}^T\mathbf{M}\mathbf{S}$

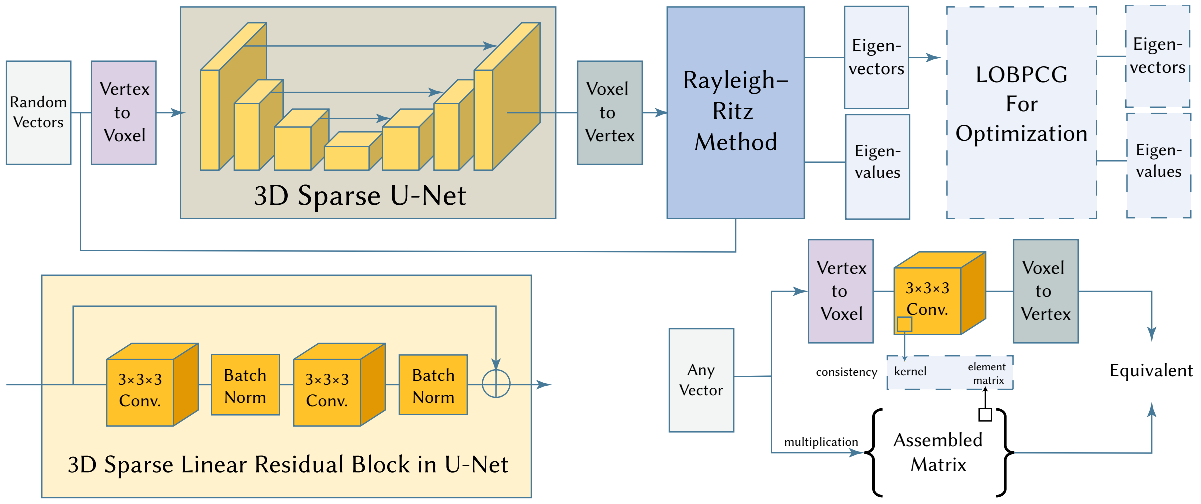


Fig. 2. (**Top**) Architecture of our vibration solver. k random initial vectors are fed into the 3D sparse U-Net. The output vectors are concatenated with random vectors and converted into the approximate eigenvectors and eigenvalues using the Rayleigh-Ritz method. LOBPCG will optimize The approximate eigenvectors to further reduce errors. (**Bottom left**) Linear residual block in our U-Net. (**Bottom right**) The connection behind our network design: a $3 \times 3 \times 3$ 3D sparse convolution is equivalent to an assembled matrix with a corresponding element matrix.

can be ill-conditioned or rank deficient [Duersch et al. 2018]. We use the SVQB algorithm [Duersch et al. 2018] to convert the generalized eigenvalue problem of Equation 5 into a standard eigenvalue problem, which can be solved by numerically stable method.

Since Rayleigh-Ritz method is critical to our approach, we provide the validation experiments in Appendix A.3.

Loss function is also a critical part of the self-supervised training strategy. When the Ritz pairs $(\hat{\mathbf{\Lambda}}, \mathbf{S}\hat{\mathbf{U}})$ is solved, we denote the approximate eigenvalues as $\hat{\mathbf{\Lambda}} = diag\{\hat{\lambda}_1, \hat{\lambda}_2, ..., \hat{\lambda}_k\}$ and the approximate eigenvectors as $\mathbf{S}\hat{\mathbf{U}} = [\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, ..., \hat{\mathbf{v}}_\mathbf{k}]$. The loss function is defined as:

$$\ell(\hat{\mathbf{\Lambda}}, \mathbf{S}\hat{\mathbf{U}}) = \frac{1}{k}\sum_{i=1}^{k} \frac{||\mathbf{K}\hat{\mathbf{v}}_\mathbf{i} - \hat{\lambda}_i \mathbf{M}\hat{\mathbf{v}}_\mathbf{i}||_2}{(||\mathbf{K}||_2 + |\hat{\lambda}_i| ||\mathbf{M}||_2)||\hat{\mathbf{v}}_\mathbf{i}||_2} + \gamma \frac{1}{k}\sum_{i=1}^{k} |\hat{\lambda}_i| , \tag{6}$$

where γ is a manually set hyper-parameter. The left item of Equation 6 is the residual-based error defined in the convergence criterion of LOBPCG [Duersch et al. 2018]. Reducing this residual-based error increases the accuracy of the solved eigenvectors. The right item of Equation 6 is the average of the solved eigenvalues. Reducing the average eigenvalue can facilitate resolving the first k smallest eigenvalues for modal analysis.

Our U-Net is trained with a fixed material and voxel size. The eigenvectors and eigenvalues of an object with different materials or voxel sizes can be solved by linearly scaling the results obtained by our vibration solver. For more details about the scaling, please refer to Jin et al. [2020]; Zheng and James [2010].

4.2.4 Warm starting to LOBPCG. In principle, a neural network-based method does not have the characteristic of convergence through iterations like traditional numerical solutions. Therefore, the predicted values from our vibration solver will inevitably produce some errors. To reduce the accuracy loss and further improve sound quality, we design a mixed vibration solver that uses the results of our learning-based module to warm-start an LOBPCG solver [Duersch et al. 2018; Knyazev 2001]. These two modules can be integrated naturally because the output of our network is consistent with the representation of standard finite element model (see Sec. 4.1 and 4.2.1).

Since our network can yield outputs close to the actual eigenvectors, the warm-started LOBPCG solver tends to converge quickly. In other words, our mixed vibration solver can obtain more accurate results quickly, as shown in Sec. 7.

5 RADIATION SOLVER: LEARNING FFAT MAP

We propose a learning-based radiation solver for the acoustic transfer problem introduced in Sec. 3.2. Our radiation network predicts the acoustic transfer function from the surface displacement and frequency of a vibration mode. We select the scalar-valued FFAT Map as the compressed representation of the acoustic transfer function. The scalar-valued FFAT Map compresses the acoustic transfer function of a mode into a 2D image as [Chadwick et al. 2009] :

$$\frac{\psi_i(\theta, \phi)}{r} \approx |p_i(\mathbf{x})| , \tag{7}$$

where θ, ϕ are spherical coordinates with the center of the object as the origin, and r is the distance from $\mathbf{x}$ to the origin. The pixel value of (θ, ϕ) position in the 2D image is $\psi_i(\theta, \phi)$. We provide the details of the scalar-valued FFAT Map in Appendix A.4. In the following subsections, we first highlight the connection between the convolution neural network and a standard numerical method, i.e., BEM. Next, we introduce our end-to-end radiation network for fast precomputation of the acoustic transfer function.

5.1 Network and Radiation Solver

BEM is a standard method used to compute the acoustic transfer function. We first analyze the intrinsic connections between the

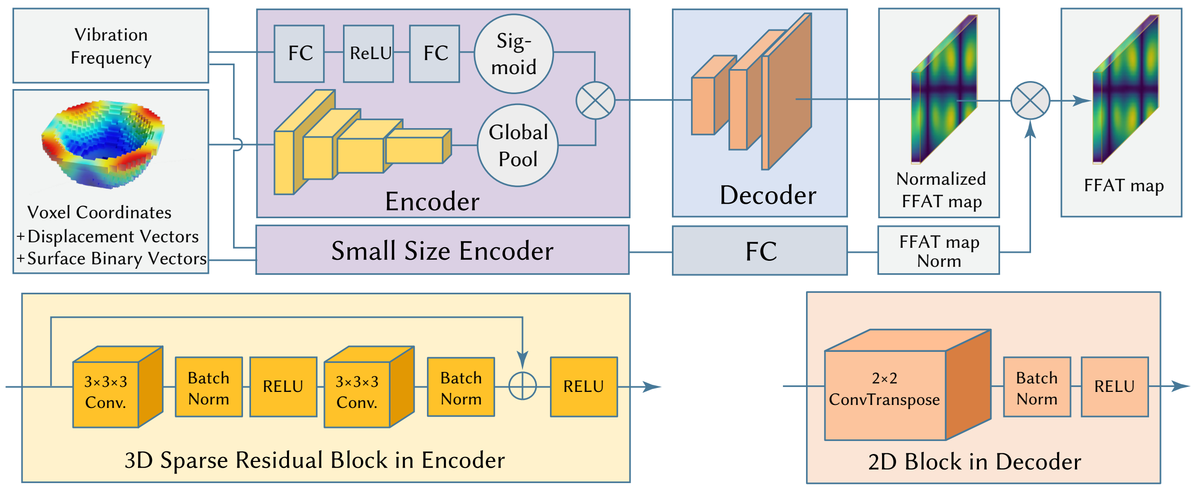


Fig. 3. (**Top**) Architecture of our radiation solver. It consists of two branches for normalized FFAT Map and FFAT Map norm. In each branch, the encoder consists of a 3D sparse ResNet for encoding the features of surface voxels and two fully connected layers for encoding vibration frequency. Two encoder parts fuse by multiplying their results. The ResNet consists of several residual blocks (**Bottom left**). The decoder for normalized FFAT Map consists of several 2 × 2 transpose convolutional blocks (**Bottom right**). In addition, the decoder for FFAT Map norm is a fully connected layer.

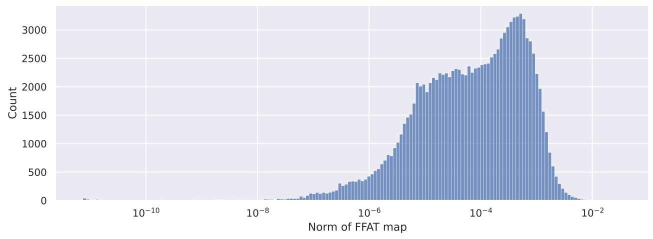


Fig. 4. Distribution of FFAT Map norms in our dataset. The surface vibration vectors are normalized to a unit norm before being fed to BEM to solve FFAT Maps. We trained our network to predict the norm in log scale due to its distribution's smoothness.

neural network and the BEM, and use that to design our radiation network.

Assume that the surface acoustic transfer value $\mathbf{p}_{surf}$ and its normal derivative $\partial_{\boldsymbol{n}}\mathbf{p}_{surf}$ on a surface are known. The acoustic transfer values $\mathbf{p}_{outer}$ on an outer sphere is needed for FFAT Map (see Figure 11 in Appendix) and can be computed by the potential operators [Betcke and Scroggs 2021; Liu 2009]:

$$\mathbf{p}_{outer} = \mathcal{V}\partial_{\boldsymbol{n}}\mathbf{p}_{surf} - \mathcal{K}\mathbf{p}_{surf}\,, \tag{8}$$

where $\mathcal{V}, \mathcal{K}$ are the single layer potential operator and the double layer potential operator, respectively. Fast Multipole Method (FMM) [Liu 2009] can be applied to compute the projection ($\mathcal{V}$ and $\mathcal{K}$) from the object surface to the outer sphere. The element-to-element interactions in the conventional BEM can be analogized to cell-to-cell interactions within a hierarchical tree structure of cells containing groups of elements. Therefore, there is an intuitive connection between a neural network with a downsampling-upsampling architecture and this sound radiation process.

Specifically, the downsampling part of our network is inspired by the particles to multipole (P2M) and multipole to multipole (M2M) processes in FMM. We observe two important features of FMM: (i) shift-invariance, since interactions depend on the relative position between elements; (ii) spatial locality, since nearby elements are aggregated into cells. They are also features of 3D sparse convolution network, as described in [Choy et al. 2019]. Therefore, we apply 3D sparse convolution as the downsampling part of our network.

Similarly, the upsampling part is inspired by the local to local (L2L) and local to particles (L2P) computations in FMM. Shift-invariance and spatial locality are also applicable. Therefore, we apply 2D transpose convolution as the upsampling part of our network.

It turns out that $\partial_{\boldsymbol{n}}\mathbf{p}_{surf}$ can be computed from the surface vibration, whereas $\mathbf{p}_{surf}$ is still unknown. Without considering the fictitious frequency [Li et al. 2015; Liu 2009], $\mathbf{p}_{surf}$ can be solved from the linear equations of the conventional boundary integral equation (CBIE) [Betcke and Scroggs 2021; Liu 2009]:

$$(\frac{1}{2}\mathbf{I} + \mathbf{K})\mathbf{p}_{surf} = \mathbf{V}\partial_{\boldsymbol{n}}\mathbf{p}_{surf}\,, \tag{9}$$

where $\mathbf{I}, \mathbf{V}, \mathbf{K}$ are identity operator, single layer boundary operator, and double layer boundary operator, respectively. $\mathbf{V}$ is derived from Green's function $G(\boldsymbol{x};\boldsymbol{y}) = \frac{e^{ik\|\boldsymbol{x}-\boldsymbol{y}\|}}{4\pi\|\boldsymbol{x}-\boldsymbol{y}\|}$, and $\mathbf{K}$ is derived from $\frac{\partial G(\boldsymbol{x};\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})}$. $G(\boldsymbol{x};\boldsymbol{y})$ and $\frac{\partial G(\boldsymbol{x};\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})}$ are inversely proportional to the relative distance from $\boldsymbol{x}$ to $\boldsymbol{y}$, which shows the spatial locality and shift-invariance between the elements of $\partial_{\boldsymbol{n}}\mathbf{p}_{surf}$ and the elements of $\mathbf{p}_{surf}$. We infer that 3D sparse convolution can also handle the translation from $\partial_{\boldsymbol{n}}\mathbf{p}_{surf}$ to $\mathbf{p}_{surf}$.

5.2 Radiation Network Architecture

As analyzed above, we design a 3D sparse ResNet [He et al. 2016] as the encoder and transposed convolutions as the decoder to transform the surface vibration information to the scalar-valued FFAT Map. The overall architecture is illustrated in Figure 3.

5.2.1 Inputs & Outputs. The input consists of (i) features of surface voxels and (ii) vibration frequency. The features of each surface voxel are concatenated by three components: normalized coordinate, vibration displacement of a voxel (averaging the displacement of 8 vertices), and a binary vector (6 elements). The binary vector indicates whether each face of a voxel is exposed to the outer space. The vibration frequency is normalized by min-max scaling in Mel scale (corresponding to 20Hz - 20000Hz).

We also design the output formulation to improve the quality of predicted scalar-valued FFAT Map. Note that the overall radiation efficiencies of different modes are different [Cremer and Heckl 2013]. As mentioned in James et al. [2006], some modes are thousands of times more radiative than others. Therefore, the matrix norm of the FFAT Map (represented as a matrix) corresponding to different modes is different, as shown in Figure 4. This large difference can lead to difficulties in network training. Therefore, we train the network to predict the matrix norm in the log scale (i.e., FFAT Map norm) and the normalized FFAT Map separately. In other words, the network is trained to predict the overall radiation efficiency and directional difference separately.

5.2.2 Network Architecture & Loss Function. There are two branches in our radiation network to predict the normalized FFAT Map and the FFAT Map norm (see Figure 3). Each branch is an encoder-decoder-based architecture. The encoders in both branches have the same architecture, which mainly consists of a 3D sparse ResNet and two fully connected layers with activation functions. The ResNet encodes features of surface voxels and the fully connected layers encode vibration frequency. These two information codes are fused by multiplication. The encoder in FFAT Map norm branch is designed with fewer parameters (similar structure, fewer layers and channels) to prevent over-fitting. The decoder in normalized FFAT Map branch mainly consists of some 2D transpose convolutions, and the decoder in FFAT Map norm branch is a fully connected layer. The loss function used to train the network is Mean Squared Error (MSE). The MSE loss of normalized FFAT Map and the MSE loss of FAT Map norm are summed up to compute the final loss.

5.2.3 Post-processing. The scalar-valued FFAT Map is reconstructed by multiplying the FFAT Map norm with the normalized FFAT Map. Note that the radiation network is trained with a fixed voxel size. Nonetheless, our approach is applicable to any voxel size because the changes in acoustic transfer function caused by variation of voxel sizes can be handled in the same manner as Zheng and James [2010]. Specifically, if the geometry of an object is scaled by γ, the multipole coefficient M_n^m for acoustic transfer will be scaled as $M_n^m \rightarrow \gamma^{-5/2} M_n^m$. Similarly, our FFAT Map is scaled as $\mathbf{\Psi} \rightarrow \gamma^{-5/2}\mathbf{\Psi}$.

6 NETWORK IMPLEMENTATION

6.1 Dataset

The 3D models we use for training and evaluation come from the ABC Dataset [Koch et al. 2019], which is a large CAD model dataset for geometric deep learning. We use ~100K 3D models and voxelize these models at a resolution of $32 \times 32 \times 32$, so that the training time and GPU memory occupied are feasible.

For the vibration solver, we use the ceramic material as representative and fix the unit voxel size at 0.5cm. For each 3D model, we precompute the stiffness matrix and mass matrix as well as voxel-to-vertex projection matrix and vertex-to-voxel projection matrix. These four matrices of the ~100K model are saved as the dataset for our vibration solver. We classify the dataset into a training set, a test set, and a validation set with a ratio of 4:1:1.

For the radiation network solver, we fix the resolution of the FFAT Map at 64×32. For each 3D model, we use the BEM solver Bempp-cl [Betcke and Scroggs 2021] to compute FFAT Maps of five randomly selected vibration modes. The surface vibration vectors are normalized before the BEM solution. To handle more general cases, we assign each 3D model a random material (one each for ceramic, glass, wood, plastic, iron, polycarbonate, steel, and tin). The input information and FFAT Maps of ~100K models are saved as the dataset for our radiation solver. We classify the dataset into a training set, a test set, and a validation set with a ratio of 4:1:1.

6.2 Network Training

We implement our networks on Minkowski Engine [Choy et al. 2019] and Pytorch. We optimize the networks using ADAM [Kingma and Ba 2015]. The learning rates of our networks are all $1e$-4, and the learning rate is reduced to 0.5 times every 20 epochs. For the vibration solver, $k = 20$ random starting vectors of each 3D object are concatenated into a mini-batch (i.e., the batch size is 20, and one object constitutes each mini-batch). To reduce training time cost, we generate a random subset (1/20 of the full dataset) for training at each epoch. For the radiation solver, we set the batch size to 16. We train 100 epochs for both networks and finally store the network parameters with the smallest loss in the validation set.

7 RESULTS

We highlight our experimental results along with those of the standard numerical solver and other comparable methods to evaluate the accuracy of our approach. The rendered sounds are shown in the video.

7.1 Sound Vibration Solver

Table 1. Performance evaluation (average across the test dataset) of Lanczos, LOBPCG, and our solver. The first 20 modes are solved for each object.

Method	Conv. Error	Freq. Error	Time
Lanczos (CPU)	1.7e-6	1.7e-6	15.32s
LOBPCG (GPU)	1.4e-7	0	2.71s
Lanczos (CPU)	N/A	N/A	0.03s
LOBPCG (Reduced)	0.165	0.90	0.03s
Ours	0.008	0.10	0.03s

We compare different vibration solvers on a randomly-selected subset (100 models) from the test dataset. Two classic numerical methods include: (i) standard Lanczos methods [Lanczos 1950; Lehoucq et al. 1997], which is implemented in ARPACK on a CPU (Intel i7-8700k) [Lehoucq et al. 1997]; (ii) standard LOBPCG [Duersch et al. 2018], which is a iterative solver and implemented using Pytorch on

Table 2. Performance evaluation (average across the test dataset) of Lanczos, LOBPCG, Rayleigh-Ritz (R-R), and our mixed vibration solver in terms of different error bounds (ε).

Method	$\varepsilon < 0.01$	$\varepsilon < 0.005$	$\varepsilon < 0.001$
Lanczos (ILU)	10.75s	12.13s	13.52s
Lanczos (Iterative)	9.75s	12.78s	14.52s
R-R ($k = 1, J = 20$)	7.82s	8.14s	9.03s
R-R ($k = 20, J = 1$)	7.56s	7.77s	8.66s
R-R ($k = 20, J = 20$)	8.05s	8.31s	9.13s
LOBPCG (Reduced)	0.20s	0.30s	0.50s
Our Mixed Solver	0.03s	0.12s	0.29s

a GPU (Nvidia RTX 3080Ti). Three metrics are used to measure the performance: residual-based convergence error (left term of Equation 6), Mean Square Error of frequency in Mel scale (divided by the square of the Mel scale length corresponding to 20Hz-20000Hz), and the running time cost. The mean value of objects' first 20 modes on these metrics is evaluated for our test set.

We first show two numerical algorithms' performance on convergence in the upper part of Table 1. Both Lanczos and LOBPCG can converge, while LOBPCG is much faster, and this result also confirms the conclusion of Arbenz et al. [2005]. The converged numerical results are used as the ground truth, we then evaluate the performance of our learning-based solver in the lower part of Table 1. Our solver completes network inference within 0.03s to resolve the vibration with high accuracy in terms of both convergence error and frequency error. Using the same time budget, LOBPCG (Reduced) obtains far inaccurate results with a few iterations, while Lanczos can not produce any output.

As a warm-start initialization, our learning-based solver is integrated with LOBPCG (called mixed solver), we made further validation and highlighted the performance of our mixed solver in Table 2. We compared our mixed solver with others in terms of time cost to reach different level of accuracy, i.e., different error bounds.

Standard Lanczos generally takes most of the time to perform the LU decomposition of the stiffness matrix, and it is not suitable for iterative refinement. Therefore, we provide two generally-used alternatives: (i) replacing the LU decomposition with incomplete LU decomposition (ILU), denoted as Lanczos (ILU), wherein the expected fill ratio upper bound of ILU decomposition is fine-tuned for different error bounds; (ii) computing the matrix inverse by an iterative solver instead of direct sparse solver, denoted as Lanczos (Iterative), wherein the number of Lanczos iterations, and number of iterative solver iterations, is fine-tuned for different error bounds.

In addition to LOBPCG and Lanczos, Rayleigh-Ritz algorithm (abbr. R-R) is also used as a baseline: starting with k random vectors $\mathbf{x_1}, ..., \mathbf{x_k}$, the standard krylov space is spanned by vectors $(\mathbf{K}^{-1}\mathbf{M})^j\mathbf{x_i}$, where $i = 1, ..., k$ and $j = 1, ..., J$. Then the Rayleigh-Ritz algorithm is applied in this space to solve approximate eigenvectors. As the first 20 modes should be figured out in our experiments, we set three group settings for test: $(k = 1, J = 20)$, $(k = 20, J = 1)$, and $(k = 20, J = 20)$. The Rayleigh-Ritz algorithm is GPU-accelerated except for the LU decomposition of $\mathbf{K}$. Like our mixed solver, the Rayleigh-Ritz algorithm works as a warm-start and then is further optimized by LOBPCG.

As can be seen from Table 2, our mixed solver consistently shows superior performance over other approaches (covering various settings) using different error bounds. Notwithstanding being inferior to our approach, LOBPCG still significantly outperforms the Lanczos and Rayleigh-Ritz algorithms. Our mixed solver's speedup over LOBPCG decreases as the tolerance tightens (to ~0.001) due to more iterations of LOBPCG being required. Based on the above results, we choose LOBPCG (a stronger baseline) to compare the performance in the following sections.

7.1.1 *Visualization of Results.*

We draw a scatter plot to illustrate that our mixed vibration solver results in more accurate frequencies than standard LOBPCG in the same time budget, especially for low-frequency modes (see Figure 5). Generally, a user is perceptually sensitive to the low-frequency modes with small damping coefficients than that of high-frequency. As a result, our mixed vibration solver can significantly improve the quality of sound synthesized within a limited time budget.

We denote the accurate eigenvectors as $\mathbf{V}$ and the predicted eigenvectors as $\hat{\mathbf{V}}$ and plot their relationship matrix $\mathbf{V}^T\mathbf{M}\hat{\mathbf{V}}$ for different numbers of iterations. Figure 6 shows the relationship matrices within the time budget and convergence curves of our mixed vibration solver and standard LOBPCG. The results demonstrate that our mixed solver can obtain higher accuracy and result in better convergence than LOBPCG.

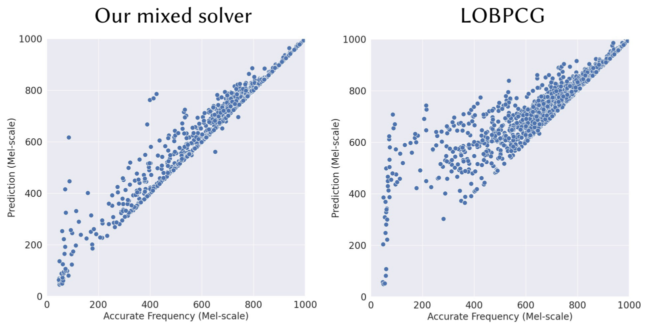


Fig. 5. Visualization of mode frequencies computed by our mixed vibration solver vs. LOBPCG using same time budget (0.4s). The horizontal and vertical axes represent actual and predicted frequency in Mel scale, respectively.

Table 3. Performance evaluation (average across the test dataset) of our vibration solver and standard LOBPCG with 80 modes for each object. Our approach results in significantly lower error than LOBPCG (Reduced) within a similar time budget.

Method	Conv. Error	Freq. Error	Time
LOBPCG (GPU)	$1.9e$-8	0	6.52s
LOBPCG (Reduced)	0.210	0.33	0.11s
Ours	0.024	0.04	0.10s

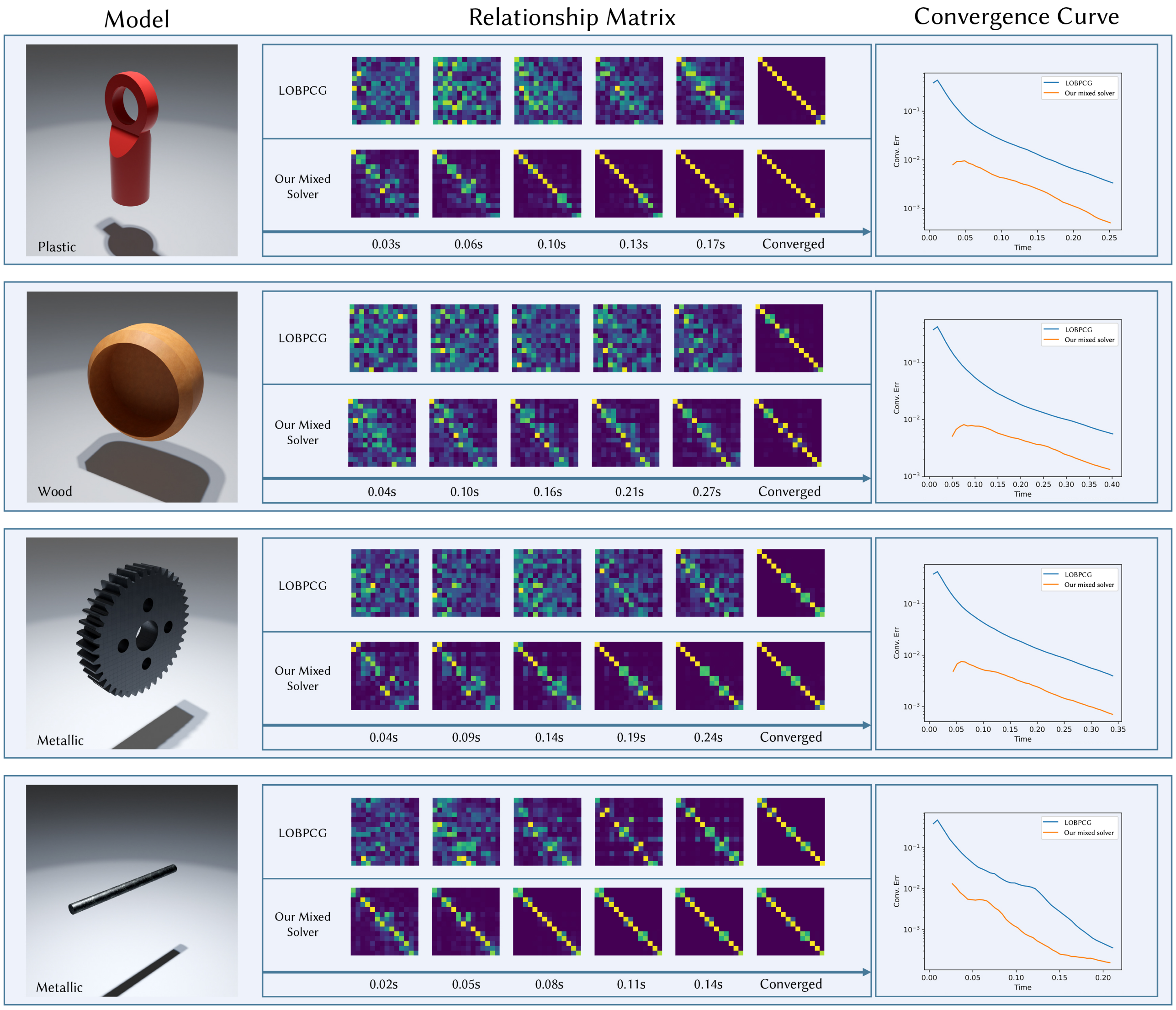


Fig. 6. Performance evaluation of our mixed vibration solver and LOBPCG, visualized using relationship matrices and convergence curves of different objects with different materials. Our mixed solver results in a significant improvement in terms of convergence.

7.1.2 Varying Number of Modes or Different Resolution. In general, a larger number of modes can enrich the sound quality. Our vibration solver also works with different numbers of modes by adjusting the number of initial vectors for each 3D object. We conduct an experiment to measure the efficiency and accuracy of our vibration solver with a larger number of modes. Table 3 shows the results of our solver and standard LOBPCG with 80 modes for each object. This vibration solver is retrained with 80 initial vectors and still obtains higher accuracy than LOBPCG (Reduced) with fewer iterations using the same time budget. As compared to the performance obtained with 20 modes, as shown in Table 1, one network inference to resolve 80 modes can improve the mean frequency accuracy (error 0.10 $\rightarrow$ 0.04). This may be largely because high-frequency modes are less discriminatively perceptible at Mel scale.

Rather than using a resolution of $32 \times 32 \times 32$, our vibration solver can also be used for higher resolution models without retraining. This is a benefit of the shift-invariance of the convolutional network and the consistency between a finite element model and a network, as highlighted in Sec. 4.1. Table 4 shows the results of our vibration solver and LOBPCG using the resolution of $64 \times 64 \times 64$ within same time budget. Our vibration solver still demonstrates superior performance over the LOBPCG, regardless of the resolution. However, higher resolution data should take more time to complete one network inference, as shown in Table 4 and Table 1.

Table 4. Performance evaluation (average across the test dataset) of our vibration solver and LOBPCG with a resolution of $64 \times 64 \times 64$.

Method	Conv. Error	Freq. Error	Time
LOBPCG (GPU)	8.8*e*-7	0	12.71s
LOBPCG (Reduced)	0.134	1.26	0.12s
Ours	0.004	0.30	0.11s

7.2 Sound Radiation Solver

We compare the performance of the radiation solver used in our approach with (i) BEM and (ii) a random selection method. This comparison is also made on a subset (100 models) randomly selected from the test dataset. Note that BEM's performance is less sensitive to the number of iterations. In most cases, the construction of the boundary integrals and the evaluation of FFAT Map dominate the costs [Wang and James 2019]. Furthermore, KleinPAT [Wang and James 2019] is not an effective solution for our cases because the spatial range of interest is generally large ($3\times \sim 27\times$ object size), and the time-domain method requires high-resolution voxel grids, which can increase the time-stepping costs significantly. As far as we know, the only method with a comparable speed is the random selection method, i.e., a scalar-valued FFAT Map is randomly selected from the training dataset. We use the latest Bempp-cl library [Betcke and Scroggs 2021] as the underlying BEM solver, and various operators run on a GPU to achieve the best performance. Our radiation network runs on the same GPU.

Three metrics are used to measure the performance: MSE of the normalized FFAT Map, MSE of the log FFAT Map norm, and the running time cost. Table 5 shows that our radiation solver can achieve high accuracy quickly with approximately $\sim$2200$\times$ speedup over a numerical solver to solve FFAT Maps of 20 modes, i.e., 0.04s compared to 88s of BEM. The user study in Sec. 8 also shows that the sound quality obtained by our radiation solver is close to the ground truth. Our extensive comparison (near spatial range radiation) can be found in the Appendix A.5.

Table 5. Performance evaluation (average across the test dataset) of BEM, random selection method, and our radiation solver. FFAT Maps of the first 20 modes are solved for each object.

Method	Normalized FFAT Map MSE	Log Norm MSE	Time
BEM	0	0	88s
Random Selection	0.63	4.76	0s
Ours	0.06	0.07	0.04s

7.2.1 *Visualization of Results.*

We similarly visualize the normalized FFAT Map as KleinPAT [Wang and James 2019] and the distribution of FFAT Map norm (in log scale) for several objects, to compare our radiation network solver with BEM. Our results are similar to the results of BEM (see Figure 7). Overall, our radiation solver works better at low frequencies than at high frequencies. Furthermore, we observe that the FFAT Map Norm distribution can significantly affect the pitch of a sound, which humans are sensitive to, so its accuracy is critical. Our radiation solver can accurately predict the FFAT Map Norm distribution.

7.3 Extensive Cases

Our learning-based sound synthesis approach enables the efficient synthesis of different desirable sound effects. We demonstrate some results with comparisons and applications. All the animations are generated using the physically-based simulator Pybullet [Coumans and Bai 2016] except the Jumping Jelly, which is an artistically designed animation. The first three 3D models are used in Wang and James [2019], and the Tin Bell model is from James et al. [2006]. Other models are designed using Blender software.

Fast Editable Shapes: To achieve certain sound characteristics, the user might rely on a trial-and-error approach to tune the material parameters [Li et al. 2015], and a fast sound synthesis method can shorten the tuning cycle. Our learning-based sound synthesis approach can not only shorten the material tuning cycle but also the model shape tuning cycle. Each adjust-synthesis cycle time is less than 2 seconds in the "Little Star" benchmark (see Figure 8f).

Fast Precomputation: Precomputation will needs more time to handle more objects. Our learning-based sound synthesis approach can significantly accelerate the computation. It takes less than one minute to precompute all the sound vibration and radiation data by our approach for a scene with 36 letter objects (see Figure 8h). On the other hand, numerical solvers (standard LOBPCG + BEM) takes about one hour for sound synthesis methods.

Artistic Effect Design: In Li et al. [2015], user-specific non-physical time-varying frequencies are used to generate interesting artistic effects in soft-body animation. By approximating the soft body model in each frame as a different object, our learning-based sound synthesis approach can quickly generate interesting sound effects synchronized with the animations (see Figure 8g).

8 PERCEPTUAL EVALUATION

We conducted a preliminary user study to evaluate the sound quality generated by our learning-based sound synthesis approach. Our goal was to verify that our approach generates plausible sounds similar to the reference sounds. A total of 106 subjects (each with normal hearing) were enrolled in this experiment, and each subject used a pair of headphones for sound playback. Each user performed two tests (vibration test and radiation test):

Vibration test: Five videos, as shown in **a-e** of Figure 8, cover a variety of materials, including a ceramic plate (20 modes), a wood bowl (40 modes), a plastic bunny (20 modes), a tin bell (80 modes), and a glass vase (17 fractures, 20 modes each fracture). In each video clip, three sound vibration solvers are applied to generate sound consistent with the animation: (i) LOBPCG (ground truth as the *reference*), (ii) our mixed vibration solver, and (iii) LOBPCG (reduced) using the same time budget, with (ii) as the *baseline.* During this test, the reference sound (i) was played first, then (ii) and (iii) were played in a random order, and a counter balance was used to control the order effect. The subjects were asked to measure the similarity between (ii) and (i) as well as between (iii) and (i) using a 7-point

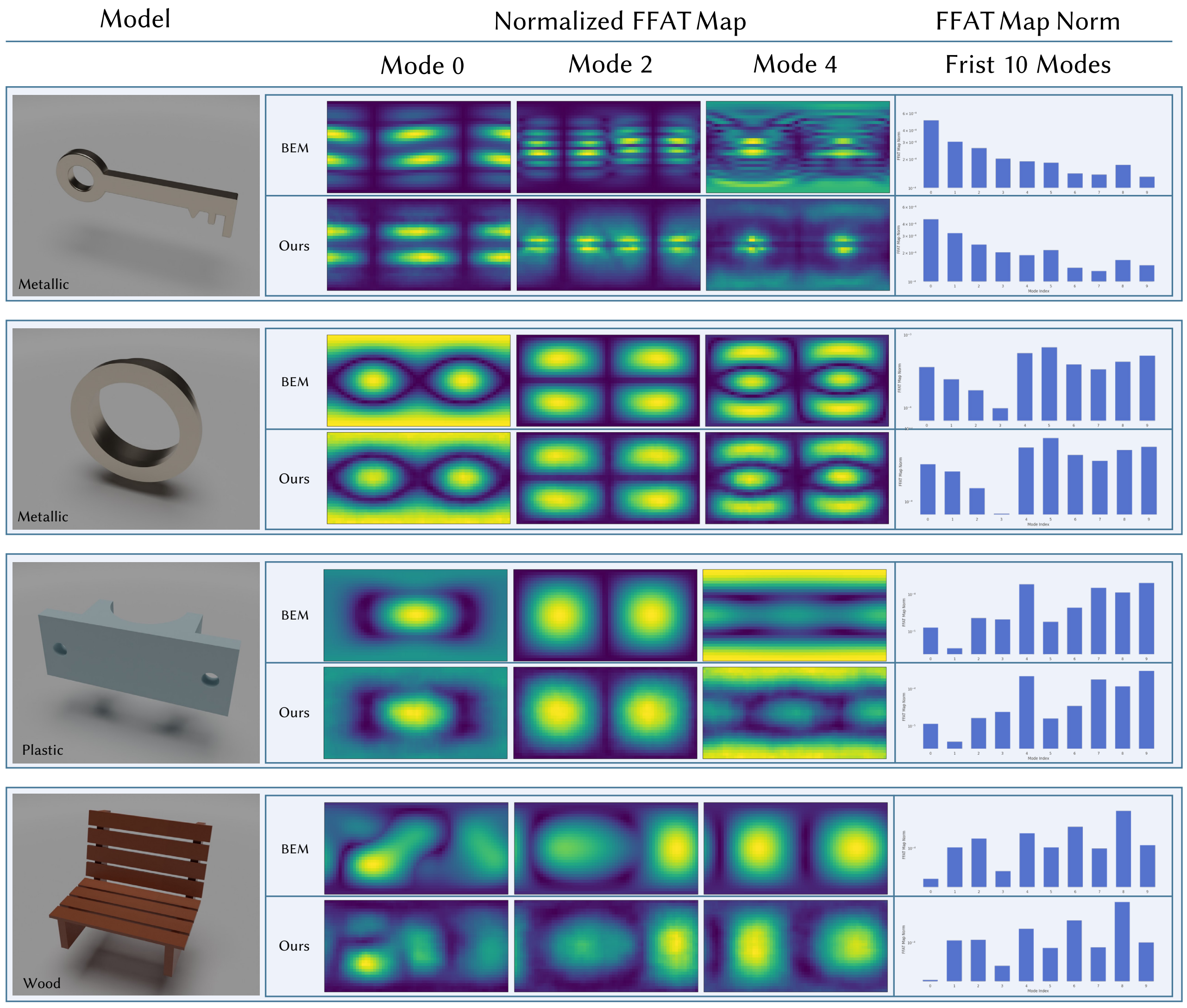


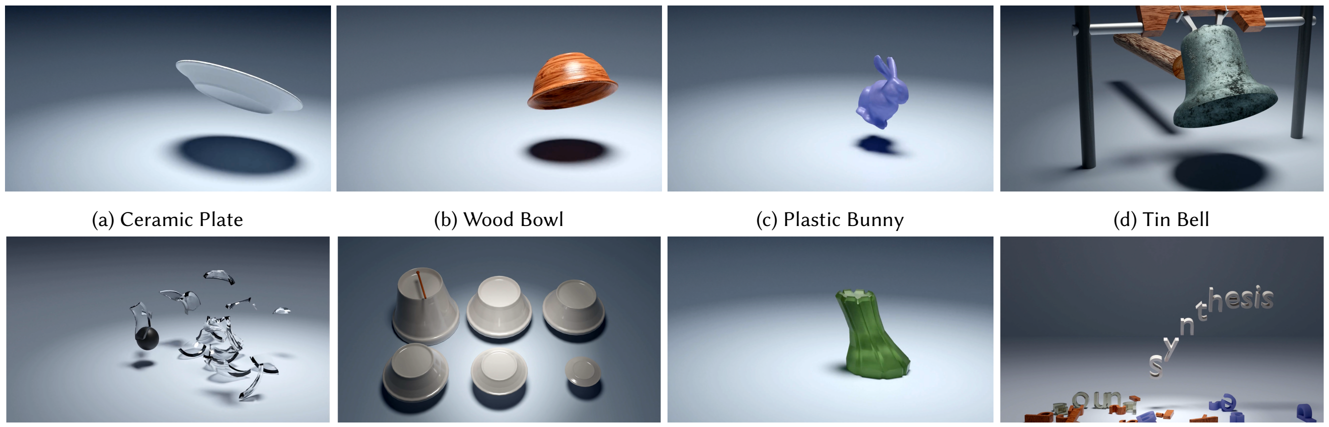
Fig. 7. Results of our radiation solver and BEM show very similar structures. Normalized FFAT Map of mode 0, 2, 4 and the FFAT Map norm of the first 10 modes in log scale are plotted for comparison.

Likert scale ranging from 1 (no similarity at all) to 7 (no difference at all). Note that the subject did not have any prior knowledge about which one from (ii) and (iii) was produced by our approach.

Radiation test: Sound benchmarks from the vibration test were also used to evaluate the radiation solver. In each video clip, three sound radiation solvers were applied to generate sound consistent with the animation: (i) BEM (ground truth as the *reference*), (ii) our radiation solver, and (iii) the random selection method described in Sec. 7.2 as the *baseline*. During this test, the reference sound (i) was played first, then (ii) and (iii) were played at random, and a counter balance was used to control the order effect. The subjects were asked to measure the similarity between (ii) and (i) as well as (iii) and (i) using the same Likert scale used in the vibration test. Note that the subject did not have any prior knowledge about which one from (ii) and (iii) was produced by our approach.

The descriptive statistical analysis for different methods and scenes is shown in Table 6. We show the mean value with standard deviation of similarity obtained from the *vibration test* and the *radiation test*. Our approach shows very high fidelity to the ground truth (averaging 6.29 for sound vibration and 6.31 for sound radiation, respectively) and significantly outperforms the baseline in both the vibration and radiation tests.

As to the performance differences across all five scenes, our vibration solver obtain a significantly lower score on scene **e** (glass fracture) than other scenes ($p < .001$ when compared with scenes **a**



(a) Ceramic Plate (b) Wood Bowl (c) Plastic Bunny (d) Tin Bell

(e) Glass Vase Fracture (f) Little Star (g) Jumping Jelly (h) Letters

Fig. 8. (**a**, **b**, **c**, **d**, **e**) We compare our approach with other baselines on 3 drop scenes, a ceramic plate (20 modes), a wood bowl (40 modes), and a plastic bunny (20 modes), as well as a ringing bell scene (80 modes) and a glass fracture scene (17 fractures, each 20 modes). (**f**) We play the melody "Little Star" by tapping bowls of different shapes and sizes. The vibration and radiation data after editing the model are solved by our approach. (**g**) Our approach can generate artistic sound effects with time-varying frequencies and acoustic transfer in a soft body animation. (**h**) Our approach enables fast sound synthesis for scenes with multiple moving objects, where we use 20 modes for each object.

Table 6. Mean value with standard deviation obtained from the *vibration test* and the *radiation test*, respectively. Each scene and test had all 106 subjects. Our approach shows very high fidelity to the ground truth and significantly outperforms the baseline in both the vibration and radiation tests.

Scene	Vibration		Radiation	
	Ours	Baseline	Ours	Baseline
a	6.55 ± 0.69	1.95 ± 1.11	6.65 ± 0.66	2.53 ± 1.49
b	6.49 ± 0.73	2.53 ± 1.42	6.15 ± 1.01	5.02 ± 1.71
c	6.30 ± 0.87	4.45 ± 1.75	6.58 ± 0.74	3.75 ± 1.68
d	6.46 ± 0.79	3.53 ± 1.85	5.77 ± 1.15	3.45 ± 1.80
e	5.67 ± 1.45	4.03 ± 1.73	6.39 ± 0.84	5.03 ± 1.61
ALL	6.29 ± 1.00	3.30 ± 1.85	6.31 ± 0.95	3.96 ± 1.92

- **d**), and our radiation solver obtain a lower score on scene **d** (tin bell) than others scenes (p = .055 when compared with scene **b**, and $p < .001$ when compared with scenes **a**, **c**, **e**).

To further evaluate the difference between our approach and the baselines, we employed two-way repeated measures ANOVAs (Analysis of Variance) with the within-subjects factor **method** (ours, baselines) and **scene** (**a** - **e** of Figure 8) for vibration and radiation individually.

Vibration: There is a significant main effect of **method** ($F(1, 105)$ = 442.432, $p < .001$), a significant main effect of **scene** ($F(4, 420)$ = 42.154, $p < .001$), and a significant interaction between **method** and **scene** ($F(4, 420)$ = 101.731, $p < .001$) on participants' scores. As shown in the left side of Table 7, Bonferroni-adjusted comparisons indicate that our vibration solver outperforms the baseline, i.e., LOBPCG (reduced), significantly in all five scenes.

Radiation: There is a significant main effect of **method** ($F(1, 105)$ = 365.844, $p < .001$), a significant main effect of **scene** ($F(4, 420)$ = 71.225, $p < .001$), and a significant interaction between **method** and **scene** ($F(4, 420)$ = 68.180, $p < .001$) on participants' scores. As shown in the right side of Table 7, Bonferroni-adjusted comparisons indicate that our radiation solver outperforms the baseline, i.e. the random selection method, significantly in all five scenes.

Table 7. Simple effect of methods (ours vs. baseline) on scores in each test scene (Bonferroni correction applied). Our approach achieves significantly higher scores than baselines in both the vibration and radiation tests.

Scene	Vibration		Radiation	
	Mean Diff.	p-value	Mean Diff.	p-value
a	4.594	<.001	4.123	<.001
b	3.962	<.001	1.132	<.001
c	1.849	<.001	2.821	<.001
d	2.934	<.001	2.321	<.001
e	1.642	<.001	1.358	<.001

Overall, the results show that the sounds synthesized by our approach are much closer to the ground truth than the sounds synthesized by the baselines.

9 CONCLUSION, LIMITATIONS, AND FUTURE RESEARCH

We present a novel learning-based sound synthesis approach. We design our vibration solver based on the connection between a 3D sparse U-Net and the numerical solver based on matrix computations. Our vibration solver can compute approximate eigenvectors and eigenvalues quickly. The accuracy of the results is further optimized by an optional LOBPCG solver (mixed vibration solver). We design our radiation solver as an end-to-end network based on the connection between convolution neural networks and BEM. We evaluate the accuracy and speed on many benchmarks and highlight the benefits of our method in terms of performance. Our approach has some limitations. The hyperparameters used in our network have not been fine-tuned. The vibration solver is not much faster than the standard LOBPCG, especially under highly tight-tolerance conditions. Despite good performance on some objects, our radiation solver does not predict with good accuracy in high-frequency.

There are many avenues for future work. In addition to overcoming these limitations, developing a neural network that can be equivalent to the traditional iterative solution with fast convergence is an interesting area of research. We need to evaluate our approach on other benchmarks and complex scenarios, e.g., multi-contact of multiple objects [Zhang et al. 2015] or frictional scenarios. A better predictor in radiation for high frequencies can further improve the quality of sound acoustic transfer. Finally, we would expect to integrate and evaluate our method with learning-based sound propagation algorithms [Ratnarajah et al. 2021; Tang et al. 2022] and use these methods for interactive applications, including games and VR.

ACKNOWLEDGMENTS

We thank Mr. Xiang Gu for his helpful suggestions on the user study. This work is supported by the National Key R&D Program of China (No. 2021YFF0500901) and NSFC of China (No. 62172013).

REFERENCES

Peter Arbenz, Ulrich L Hetmaniuk, Richard B Lehoucq, and Raymond S Tuminaro. 2005. A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods. *Internat. J. Numer. Methods Engrg.* 64, 2 (2005), 204–236.

Timo Betcke and Matthew W Scroggs. 2021. Bempp-cl: A fast Python based just-in-time compiling boundary element library. *Journal of Open Source Software* 6, 59 (2021), 2879.

Nicolas Bonneel, George Drettakis, Nicolas Tsingos, Isabelle Viaud-Delmon, and Doug James. 2008. Fast modal sounds with scalable frequency-domain synthesis. In *ACM SIGGRAPH 2008 papers*. 1–9.

William L Briggs, Van Emden Henson, and Steve F McCormick. 2000. *A multigrid tutorial*. SIAM.

Jeffrey N Chadwick, Steven S An, and Doug L James. 2009. Harmonic shells: a practical nonlinear sound model for near-rigid thin shells. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009)* 28, 5 (2009), 1–10.

Jeffrey N. Chadwick, Changxi Zheng, and Doug L. James. 2012. Precomputed Acceleration Noise for Improved Rigid-Body Sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)* 31, 4 (Aug. 2012).

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 2019. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3075–3084.

Perry R. Cook. 1995. Integration of Physical Modeling for Synthesis and Animation. In *Proceedings of the 1995 International Computer Music Conference, ICMC 1995, Banff, AB, Canada, September 3-7, 1995*. Michigan Publishing.

Erwin Coumans and Yunfei Bai. 2016. Pybullet, a python module for physics simulation for games, robotics and machine learning. (2016).

Lothar Cremer and Manfred Heckl. 2013. *Structure-borne sound: structural vibrations and sound radiation at audio frequencies*. Springer Science & Business Media.

Jed A. Duersch, Meiyue Shao, Chao Yang, and Ming Gu. 2018. A Robust and Efficient Implementation of LOBPCG. *SIAM Journal on Scientific Computing* 40, 5 (2018), C655–C676.

Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 2018. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9224–9232.

Benjamin Graham and Laurens van der Maaten. 2017. Submanifold Sparse Convolutional Networks. *arXiv preprint arXiv:1706.01307* (2017).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Doug L. James. 2016. Physically Based Sound for Computer Animation and Virtual Environments. In *ACM SIGGRAPH 2016 Courses* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 22, 8 pages.

Doug L James, Jernej Barbič, and Dinesh K Pai. 2006. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)* 25, 3 (2006), 987–995.

Xutong Jin, Sheng Li, Tianshu Qu, Dinesh Manocha, and Guoping Wang. 2020. Deep-Modal: Real-Time Impact Sound Synthesis for Arbitrary Shapes. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 1171–1179.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

Stephen Kirkup. 2019. The boundary element method in acoustics: A survey. *Applied Sciences* 9, 8 (2019), 1642.

Andrew Knyazev. 1997. New estimates for Ritz vectors. *Mathematics of computation* 66, 219 (1997), 985–995.

Andrew V Knyazev. 1998. Preconditioned eigensolvers—an oxymoron. *Electron. Trans. Numer. Anal* 7 (1998), 104–123.

Andrew V Knyazev. 2001. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing* 23, 2 (2001), 517–541.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset For Geometric Deep Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cornelius Lanczos. 1950. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA.

Timothy R. Langlois, Steven S. An, Kelvin K. Jin, and Doug L. James. 2014. Eigenmode Compression for Modal Sound Models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)* 33, 4 (Aug. 2014).

R. B. Lehoucq, D. C. Sorensen, and C. Yang. 1997. ARPACK Users Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods.

Dingzeyu Li, Yun Fei, and Changxi Zheng. 2015. Interactive acoustic transfer approximation for modal sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2015)* 35, 1 (2015), 1–16.

Shiguang Liu and Dinesh Manocha. 2020. Sound Synthesis, Propagation, and Rendering: A Survey. *arXiv preprint arXiv:2011.05538* (2020).

Yijun Liu. 2009. *Fast multipole boundary element method: theory and applications in engineering*. Cambridge university press.

Ravish Mehra, Nikunj Raghuvanshi, Lakulish Antani, Anish Chandak, Sean Curtis, and Dinesh Manocha. 2013. Wave-based sound propagation in large open scenes using an equivalent source formulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)* 32, 2 (2013), 1–13.

Hsien-Yu Meng, Zhenyu Tang, and Dinesh Manocha. 2021. Point-based Acoustic Scattering for Interactive Sound Propagation via Surface Encoding. *CoRR* abs/2105.08177 (2021).

James F. O'Brien, Chen Shen, and Christine M. Gatchalian. 2002. Synthesizing Sounds from Rigid-Body Simulations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. Association for Computing Machinery, New York, NY, USA, 175–181.

Dinesh K Pai, Kees van den Doel, Doug L James, Jochen Lang, John E Lloyd, Joshua L Richmond, and Som H Yau. 2001. Scanning physical interaction behavior of 3D objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 87–96.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*. 5099–5108.

Nikunj Raghuvanshi and Ming C Lin. 2006. Interactive sound synthesis for large scale environments. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. 101–108.

Anton Ratnarajah, Shi-Xiong Zhang, Meng Yu, Zhenyu Tang, Dinesh Manocha, and Dong Yu. 2021. FAST-RIR: Fast neural diffuse room impulse response generator. https://doi.org/10.48550/ARXIV.2110.04057

Zhimin Ren, Hengchin Yeh, and Ming C Lin. 2013. Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)* 32, 1 (2013), 1–16.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Atul Rungta, Carl Schissler, Ravish Mehra, Chris Malloy, Ming Lin, and Dinesh Manocha. 2016. SynCoPation: Interactive synthesis-coupled sound propagation. *IEEE transactions on visualization and computer graphics* 22, 4 (2016), 1346–1355.

Ahmed A Shabana. 1991. *Theory of vibration*. Vol. 2. Springer.

Auston Sterling, Nicholas Rewkowski, Roberta L Klatzky, and Ming C Lin. 2019. Audio-material reconstruction for virtualized reality using a probabilistic damping model. *IEEE transactions on visualization and computer graphics* 25, 5 (2019), 1855–1864.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*. 945–953.

Zhenyu Tang, Rohith Aralikatti, Anton Ratnarajah, and Dinesh Manocha. 2022. GWA: A Large High-Quality Acoustic Dataset for Audio Processing. https://doi.org/10.48550/ARXIV.2204.01787

Kees van de Doel and Dinesh K Pai. 1996. Synthesis of shape dependent sounds with physical modeling. Georgia Institute of Technology.

Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. 2001. FoleyAutomatic: Physically-Based Sound Effects for Interactive Simulation and Animation *(SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA.

Jui-Hsien Wang and Doug L. James. 2019. KleinPAT: Optimal Mode Conflation for Time-domain Precomputation of Acoustic Transfer. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2019)* 38, 4, Article 122 (July 2019), 12 pages.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

Tianxiang Zhang, Sheng Li, Dinesh Manocha, Guoping Wang, and Hanqiu Sun. 2015. Quadratic Contact Energy Model for Multi-impact Simulation. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 133–144.

Changxi Zheng and Doug L James. 2010. Rigid-body fracture sound with precomputed soundbanks. In *ACM SIGGRAPH 2010 papers*. 1–13.

Changxi Zheng and Doug L James. 2011. Toward high-quality modal contact sound. In *ACM SIGGRAPH 2011 papers*. 1–12.

A APPENDIX

A.1 Analogy Between Network and Matrix Computation

A.1.1 Assembled Matrix and 3D Sparse Convolution. Assuming a finite element model with N hexahedrons and M vertices, an assembled matrix (e.g. mass matrix or stiffness matrix) $\mathbf{A} \in \mathbb{R}^{3M\times 3M}$ is built by assembling the element matrix $\mathbf{A_e} \in \mathbb{R}^{24\times 24}$ for all hexahedrons. A vector $\mathbf{x} \in \mathbb{R}^{3M}$ represents the displacements of all vertices (in x, y, z directions). A equivalent form $\mathbf{x'} \in \mathbb{R}^{24N}$ represents the displacements of all hexahedrons, where the displacement of each hexahedron consists of the displacements of all its vertices. The matrix-vector multiplication $\mathbf{Ax}$ has an equivalent form $(\mathbf{Ax})'$, which satisfies:

$$(\mathbf{Ax})'_{\mathbf{u}} = \sum_{\mathbf{i}\in\mathcal{N}(\mathbf{u})} \mathbf{W_{u,i}}\mathbf{x'}_{\mathbf{u+i}} , \tag{10}$$

where $\mathbf{x'}_{\mathbf{u}}$, $(\mathbf{Ax})'_{\mathbf{u}}$ represents the displacement of the hexahedron at $\mathbf{u}$ before and after multiplication, $\mathbf{u} \in \mathbb{Z}^3$ is the 3D coordinate of a hexahedron, $\mathcal{N}(\mathbf{u}) = \{(x, y, z)| -1 \le x, y, z \le 1 , x, y, z \in \mathbb{Z}\}$ is the set of coordinate offsets from the current hexahedron to the neighboring hexahedrons (including itself), and $\mathbf{W_{u,i}} \in \mathbb{R}^{24\times 24}$ is the transform matrix for the hexahedron at u and the coordinate offset $\mathbf{i}$. When $\mathbf{W_{u,i}}$ is independent of $\mathbf{u}$, Equation 10 can also be regarded as the definition of 3D sparse convolution [Choy et al. 2019]. Assuming the element matrix $\mathbf{A_e}$ is fixed, $\mathbf{W_{u,i}}$ is independent of $\mathbf{u}$ as:

$$\mathbf{W_i}(j,k) = \begin{cases} 0 & Vertex(j) \notin Voxel(\mathbf{u}+\mathbf{i}) \\ \mathbf{A_e}(j',k) & Vertex(j) \in Voxel(\mathbf{u}+\mathbf{i}) \end{cases} \tag{11}$$

where j is the index of a vertex $\mathbf{v}_j$ in the hexahedron at $\mathbf{u}$, j' is the index of the vertex $\mathbf{v}'_j$ in the neighbor hexahedron $\mathbf{u}+\mathbf{i}$. $\mathbf{v}_j$ and $\mathbf{v}'_j$ coincide in coordinates and k is the index of any vertex in a hexahedron. Therefore, the matrix-vector multiplication corresponds to a $3 \times 3 \times 3$ sparse convolution.

We also conduct an experiment to validate that a 3×3×3 sparse convolution is equivalent to an assembled matrix. We train a network $\mathbf{g}$ (with one 3×3×3 3D sparse convolution, see bottom right of Figure 2) with parameters θ on a dataset with N objects by reducing the mean relative error:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} \frac{||g(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{A}_i\mathbf{x}_i||_2}{||\mathbf{A}_i||_2||\mathbf{x}_i||_2} , \tag{12}$$

where $\mathbf{A}_i, \mathbf{x}_i$ are the assembled matrix and the random vector of the ith object, respectively. For each type of assembled matrix (e.g., stiffness matrix for ceramic objects), we retrain the network and list the mean relative error on the test set after 10 epochs in Table 8. The result shows that this sparse convolution can be trained to represent an assembled matrix with high accuracy and validates our analysis in Sec. 4.1.

Table 8. Mean relative error between an assembled matrix and a 3×3×3 sparse convolution in test set. We choose the assembled matrix as mass and stiffness matrices with different materials.

	ceramic	steel	plastic	glass
stiffness	7e-4	5e-4	6e-4	6e-4
mass	6e-4	7e-4	6e-4	7e-4

A.1.2 Inverse Assembled Matrix and Sparse U-Net. We conduct an experiment to validate that a 3D sparse U-Net can approximate the inverse of an assembled matrix. For a U-Net g with parameters θ, we train it on a dataset with N objects by reducing the mean relative error:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} \frac{||\mathbf{A}_i g(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{x}_i||_2}{||\mathbf{x}_i||_2} . \tag{13}$$

Where $\mathbf{A}_i, \mathbf{x}_i$ are the assembled matrix and the random vector of the ith object. We train two types of U-Net respectively, including a standard 3D sparse U-Net (with Relu activation) and a 3D sparse linear U-Net (without nonlinear activation). We plot their mean relative error in the test set for first 40 epochs, as shown in Figure 9. The standard nonlinear U-Net cannot converge, while the linear U-Net converges to a low error and validates our theoretical analysis in Sec. 4.1.

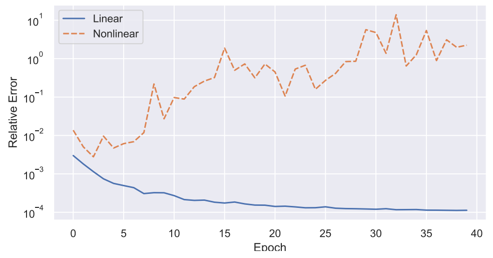


Fig. 9. Mean relative error of 3D sparse linear U-Net and nonlinear U-Net in test set.

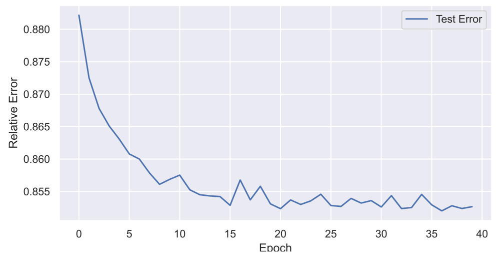


Fig. 10. Mean relative error in test set of 3D sparse linear U-Net trained without the Rayleigh-Ritz method.

A.2 Solving Eigenvectors in Krylov Subspace

In the generalized eigenvalue decomposition $\mathbf{KU} = \mathbf{MU\Lambda}$, the ith eigenvector $\mathbf{u}_i$ and eigenvalue λ_i ($\lambda_i > 0$) satisfy $\mathbf{Ku}_i = \lambda_i \mathbf{Mu}_i$. The initial vector, $\mathbf{b}$, can be written as a linear combination of all the eigenvectors (assuming there are n eigenvectors):

$$\mathbf{b} = \sum_{i=1}^{n} c_i \mathbf{u}_i \,, \tag{14}$$

where c_i is the linear coefficient. Multiplying the matrix $\mathbf{K}^{-1}\mathbf{M}$ to $\mathbf{b}$, then:

$$\mathbf{K}^{-1}\mathbf{Mb} = \sum_{i=1}^{n} c_i \mathbf{K}^{-1}\mathbf{Mu}_i = \sum_{i=1}^{n} \frac{c_i}{\lambda_i}\mathbf{u}_i \,. \tag{15}$$

In modal analysis, the first k smallest eigenvalues and corresponding eigenvectors with respect to the human audible perception range (usually 20HZ ~ 20000HZ) are used. From Equation 15, the weight of the eigenvector $\mathbf{u}_i$ is re-scaled from c_i to $\frac{c_i}{\lambda_i}$. Therefore, eigenvectors with larger eigenvalues will have smaller weights, and eigenvectors with smaller eigenvalues will dominate the result of this linear combination. Suppose there are k items from multiplying the matrix $\mathbf{K}^{-1}\mathbf{M}$ to different initial vectors: $\mathbf{K}^{-1}\mathbf{Mb}_0, \mathbf{K}^{-1}\mathbf{Mb}_1, ..., \mathbf{K}^{-1}\mathbf{Mb}_k$, then the Rayleigh–Ritz method [Knyazev 1997, 1998, 2001] can be used to compute the first k smallest approximate eigenvalues and corresponding eigenvectors in the subspace $S = span\{\mathbf{K}^{-1}\mathbf{Mb}_i\}, i = 1, 2, ..., k$. In addition to $\mathbf{K}^{-1}\mathbf{M}$, other matrices (e.g., $\left(\mathbf{K} - \hat{\lambda}\mathbf{M}\right)^{-1}\mathbf{M}$ for any $\hat{\lambda}$) can also generate a subspace to compute the approximate eigenvectors in a similar manner.

A.3 Effectiveness of Using Rayleigh–Ritz Method

We add an experiment to validate the function of Rayleigh–Ritz method by taking it out as an ablation study. Specifically, we precompute the first 20 eigenvectors of N object in our dataset. Then we train a 3D linear sparse U-Net g with parameters θ on this dataset by reducing the mean relative error between the accurate eigenvectors (ground-truth) and the predicted eigenvectors:

$$\hat{\theta} = \underset{\theta}{\text{argmin}} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{20} \sum_{j=1}^{20} \frac{||g(\mathbf{x}_{i,j}; \boldsymbol{\theta}) - \mathbf{y}_{i,j}||_2}{||\mathbf{y}_{i,j}||_2} \,, \tag{16}$$

where $\mathbf{y}_{i,j}, \mathbf{x}_{i,j}$ are the jth ground-truth eigenvector and the jth random vector of the ith object. We plot the mean relative error in test set for first 100 epochs in Figure 10. Without the Rayleigh–Ritz method, the 3D U-Net can only converge to a result (error ≈ 0.85) slightly better than a zero vector (error = 1).

Table 9. Performance evaluation of BEM, random selection method, and our radiation solver for small spatial range radiation.

Method	Normalized FFAT Map MSE	Log Norm MSE	Time
BEM	0	0	88s
Random Selection	0.68	4.68	0s
Ours	0.08	0.08	0.04s

A.4 Scalar-valued FFAT Map

FFAT Maps can perform fast transfer rendering [James 2016] and can be efficiently compressed [Wang and James 2019].

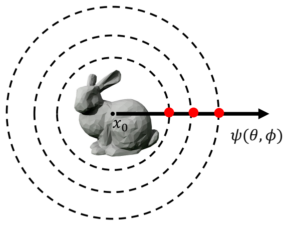


Fig. 11. Illustration of the scalar-valued FFAT Map. The values in the map are estimated by the least square method for the sample points.

Our scalar-valued FFAT Map is defined as follows. Acoustic transfer function $p_i(\mathbf{x})$ describes the sound pressure at position $\mathbf{x}$ generated by the vibration of the ith mode with unit amplitude. We use a function ψ_i to approximate the directionality information of $|p_i(\boldsymbol{x})|$, and we use $1/r$ to approximate the attenuation of the transfer amplitude as the radial distance grows, as shown in Equation 7. θ, ϕ are the coordinates in the spherical coordinate system. When we shoot a ray from x_0 at an angle (θ, ϕ), this ray will intersect with N_s spheres with different radii. The value of $\psi_i(\theta, \phi)$ can be determined by the least square method based on the values of these intersection points (see Figure 11). We set $N_s = 3$, and the radii of these three spheres are computed as $R_i = (3a)^i, i = 1, 2, 3$, where a is the longest side of the bounding box. The scalar-valued FFAT Map $\boldsymbol{\Psi}$ can be computed via a uniform sampling of (θ, ϕ) space and used as the ground-truth for our radiation network solver.

A.5 Near Spatial Range Radiation

As mentioned in Sec. A.4, our scalar-valued FFAT Maps are computed with three spheres of radii $R_i = (3a)^i, i = 1, 2, 3$, where a is the longest side of the bounding box. Our radiation solver can also be trained for FFAT Maps with different radii. To validate the generalization of our network, we generate the FFAT Map dataset with sphere radii $R_i = (1.25a)^i, i = 1, 2, 3$ and retrain our radiation solver. Table 9 shows the results of our solver, BEM, and a random selection method (baseline). Our approach also works well for relatively near range radiation and outperforms the baseline significantly in terms of accuracy. We also show comparable results with the ground truth in the accompanying video.