# Accelerated Wave-based Acoustics Simulation

Nikunj Raghuvanshi    Nico Galoppo    Ming C. Lin

{nikunj, nico, lin}@cs.unc.edu
Department of Computer Science
UNC Chapel Hill

## Abstract

We present an efficient technique to model sound propagation accurately in an arbitrary 3D scene by numerically integrating the wave equation. We show that by performing an offline modal analysis and using eigenvalues from a refined mesh, we can simulate sound propagation with reduced dispersion on a much coarser mesh, enabling accelerated computation. Since performing a modal analysis on the complete scene is usually not feasible, we present a domain decomposition approach to drastically shorten the pre-processing time. We introduce a simple, efficient and stable technique for handling the communication between the domain partitions. We validate the accuracy of our approach against cases with known analytical solutions. With our approach, we have observed up to an order of magnitude speedup compared to a standard finite-difference technique.

**CR Categories:** H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing—Modeling, Methodologies and techniques I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

**Keywords:** Acoustics, Wave Equation, Domain Decomposition, Modal Analysis, Finite Difference

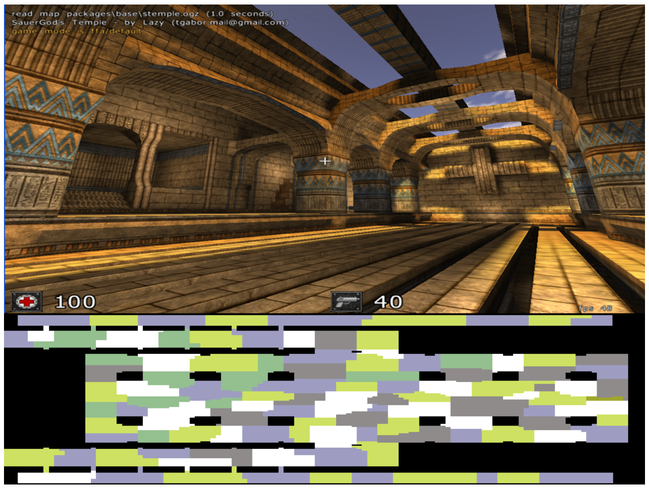

**Figure 1:** *Efficient sound propagation simulation in an architectural model.* We demonstrate our algorithm on a modestly complex 3D environment, enabling realistic interference, diffraction and reverberation sound effects. The volume of this scene is $\approx 500m^3$, which was partitioned into 290 arbitrary partitions with near-equal volume. The bottom of the figure shows the partitioning on a horizontal 2D slice of the scene, taken about 1 meter above ground. Black indicates walls and the free-space partitions are color-coded with different colors. We are able to generate one sample of audio on this scene in about 250 ms.

## 1 Introduction

One of the most important considerations in architectural design is room acoustics and noise control. Computer-aided tools are indispensable for evaluating the acoustic properties of a proposed auditorium layout or determining the noise level of any machinery on a factory floor. Furthermore, the acoustics of a virtual scene can also affect the sense of immersion in the simulated environment. Acoustic modeling can provide sound cues to a user's navigation, communication, and exploration of a virtual environment.

Humans have a very acute sense for reverberations and other acoustical effects and these tend to convey a sense of the geometric qualities of the scene to the listener. For example, delayed, unattenuated echoes tend to convey the impression of a big hall. Due to the sound propagation and the enhanced immersion achievable with correct acoustics, the problem of modeling sound propagation accurately and efficiently has traditionally been an active area of research. A few characteristics pertinent to sound waves make modeling sound propagation a challenging problem. First of all, audible sound has wavelength in the range of a few millimeters to a few meters, which is comparable to most objects' dimensions and thus capturing diffraction accurately is very important, especially for low frequencies. Secondly, sound waves are *coherent*, which means that two sound waves must be added taking their phases into account. Therefore, capturing interference is critical for modeling sound propagation accurately. Thirdly, a full time-domain solution is needed (as opposed to only a steady-state solution as in global illumination), at sampling rates in the Kilohertz (kHz) range for dynamic sound sources and listener.

Prior research in sound propagation modeling falls broadly into two categories: *geometric methods* based on a ray/beam approximation of sound and *numerical methods* that seek to solve the underlying wave-equation for sound propagation directly on a discretization of the domain. The problem with geometric methods is that they often assume little diffraction and capturing it accurately in such a framework is quite difficult. Also, capturing higher-order reflections using such schemes is difficult due to the large number of propagation paths that must be computed. Numerical methods, in contrast, elegantly capture all wave phenomena, including higher-order reflections in a single framework. However, they are computationally expensive, especially for high frequencies, as the number of elements depends cubically on the diameter of the scene being modeled, in 3D.

**Main Results:** In this paper, we propose a technique based on the Finite Difference Method (FDM) and modal analysis, which is ca-

SPM 2008, Stony Brook, New York, June 02–04, 2008.
© 2008 ACM 978-1-60558-106-4/08/0006 $5.00

pable of performing accurate sound propagation simulation for arbitrary static 3D scenes with frequencies up to 1 kHz. The sources and the listener may move arbitrarily through the scene. The main idea behind our approach is that the wave-equation on a domain can be integrated exactly in time once space has been discretized, by performing modal analysis, which involves computing an offline matrix diagonalization. However, as we describe later, the preprocessing for this basic approach is computationally intractable both in terms of time and memory consumption, especially in 3D for high-resolution meshes which are required for realistic simulation of frequencies in the kHz range.

In order to remedy this problem and improve the runtime efficiency, we introduce two acceleration techniques –

1. A domain decomposition technique in which the domain is divided into many non-overlapping partitions and modal analysis on each partition is performed separately. This makes pre-processing computationally tractable, as it is typically infeasible in 3D to carry out a modal analysis on the complete scene. At runtime, the sound is simulated within each partition using a modal technique described later and an explicit communication step is performed at the partition interfaces at each time-step. We discuss a simple second-order integration scheme for performing the communication, that is both efficient and stable.

2. An eigenvalue correction technique that utilizes corrected eigenvalues precomputed using modal analysis on a refined mesh and enables accelerated sound propagation simulation on a coarser mesh at runtime, without much loss of accuracy. This technique incurs no additional runtime overhead and simply offers a trade-off between accuracy at runtime and slightly increased pre-processing time. We discuss how to find the best trade-off for a given scene later in the paper. With this technique we are able to perform reasonably accurate acoustic simulations on a mesh with only about 2-4 spatial samples per the smallest simulated wavelength, instead of 8-10 samples, as usually used in numerical simulations [Alford et al. 1974].

As we describe later, together, these two techniques offer up to an order of magnitude improvement in terms of runtime efficiency over a standard numerical approach, enabling realistic sound simulation on moderately large scenes in 3D.

The main characteristics of our approach are:

- Automatic capturing of all wave phenomena, including diffraction and reflection, in one unified framework;
- Ability to perform simulations on very coarse meshes while retaining good accuracy, resulting in large reduction in computational requirements;
- Simple domain decomposition technique for handling large scenes.

**Organization:** The rest of the paper is organized as follows. We review related work in Section 2. We present the problem formulation for modeling sound propagation in 3D scenes along with the efficiency considerations in Section 3. We then describe our eigenvalue correction and domain decomposition techniques that enable efficient sound propagation, in Section 4. In Section 5, we discuss implementation issues and demonstrate the results of our system on moderately complex scenes, along with the validation of its accuracy and efficiency. Finally, we conclude with possible future research directions.

## 2 Previous Work

Room acoustics has been an active area of research for more than five decades [Sabine 1953]. Depending on the amount of compute required and achievable accuracy, methods for simulating sound propagation can be broadly classified into two categories: Geometric and Numerical methods. In the following, we will discuss these methods in detail. For a basic treatment of the theory and practice of room acoustics, the reader is referred to the classic texts [Kuttruff 2000; Kinsler et al. 1999]. For a recent survey of computational acoustics, please refer to [Lokki 2002].

### 2.1 Geometric Methods

Geometric methods assume that sound propagates in straight lines unless obstructed by an object, much like light. However, this approximation works well only when the wavelength of the sound is much smaller than the size of the objects in the scene, so that diffraction can be effectively neglected. Nevertheless, owing to their simplicity and efficiency, geometric techniques were the first to be formulated and implemented on a digital computer [Krockstadt 1968; Allen and Berkley 1979]. Over the decades, there has been a significant amount of research on geometric methods, which only differ from each other in the way sound propagation is approximated. Some salient examples are, the image source method [Allen and Berkley 1979], ray-tracing [Krockstadt 1968], beam-tracing [Min and Funkhouser 2000; Tsingos et al. 2001; Funkhouser et al. 2003; Funkhouser et al. 2004; Antonacci et al. 2004], or phonon-tracing [Bertram et al. 2005; Deines et al. 2006].

### 2.2 Numerical Methods

In contrast to geometric methods, numerical methods typically solve for the complete sound field by numerically evaluating the wave equation after suitable discretization. Since the complete field is resolved at each time-step, these methods typically scale poorly as the size of the scene is increased. However, the strength of numerical methods is that they are very accurate, especially for low frequencies in the range of a few hundred Hertz, which have wavelengths in the range of meters and exhibit noticeable diffraction. Geometric methods, by their basic assumption, do not capture such effects. Although there have been attempts to account for diffraction using the Geometric Theory for Diffraction [Tsingos et al. 2001], integrating accurate diffraction effects in a geometric solver still remains a challenging problem. Based on the choice of discretization, numerical methods for acoustics may be roughly classified into: Finite Element Method (FEM), Boundary Element Method (BEM), Digital Waveguide Mesh (DWM) and Finite Difference Time Domain (FDTD) Method.

FEM/BEM is typically used to find the frequency-domain (in other words, steady state) response of an environment only, as opposed to the full time-domain response [Kleiner et al. 1993]. DWM was introduced by Duyne [Van Duyne and Smith 1993] as a method to perform efficient time-domain simulation of wave propagation. The basic idea is to model the medium as a mesh of 1D waveguides, which carry waves along them as per the 1D wave equation, for which the analytical solution is known. The nodes in the mesh where these waveguides intersect are implemented as scattering junctions which ensure that the incoming and outgoing acoustic energy at each junction is distributed properly. Since its introduction, DWM has been an active area of research for room acoustics [Savioja et al. 1994; Savioja et al. 1995; Karjalainen and Erkut 2004]. For a brief survey of different methods for room acoustics and an introduction to DWM in this context, the reader is referred to [Savioja 1999]. And, for a very recent survey on the state of the art in DWM, please refer to [Murphy et al. 2007].

The FDTD method was first introduced to room acoustics in [Botteldooren 1994; Botteldooren 1995] and is most closely related to our work. A typical FDTD scheme begins by first subdividing the domain into a grid. In most cases, as in this paper, the grid is cartesian with uniform cell size. Next, the continuum spatial operators are approximated by their discrete representation. At each time-step, the whole sound field in each cell is updated based on the field values at the previous time-step(s). Different FDTD schemes differ in the way the spatial operator is approximated and the rules that are used to update the field values. FDTD for acoustics has been a very active area of research, especially for low-frequency acoustic simulation on small rooms [Sakamoto et al. 2004; Sakamoto et al. 2006]. Although DWM and FDTD appear quite different, they are actually equivalent [Van Duyne and Smith 1993]. In fact, not only are they equivalent, one can mix the two within the same domain [Karjalainen and Erkut 2004].

However, both DWM and FDTD face the same computational challenges: For medium-range frequencies in the range of a Kilohertz and/or large architectural scenes, the compute requirements grow super-linearly in higher dimensions and thus quickly become intractable, as both methods require a fine mesh to resolve the smallest wavelength. In this paper, we present a technique which is able to perform acoustic simulation on a very coarse mesh, thereby reducing the computational requirements, at the cost of a slight loss in accuracy. This is done by first dividing the domain into many non-overlapping partitions and then performing modal analysis on each separately.

Modal analysis is a very well-known technique and it would be nearly impossible to mention all the branches of science where it has been applied. However, an area of research close to this work where modal analysis has found a lot of application is physically-based sound synthesis [van den Doel et al. 2001; O'Brien et al. 2002; Raghuvanshi and Lin 2006]. For sound synthesis, modal analysis is typically performed to find the vibrational modes of an elastic object and a complex vibration expressed as a linear combination of these modes. Analogously for acoustics, modal analysis yields the resonant modes of pressure vibrations in a room, and the pressure field in the room is expressed as a linear combination of the resonant modes. The novelty of our work lies in the fact that we exploit some numerical properties of modal analysis which enables us to perform runtime simulation on a coarse mesh using corrected eigenvalues from a refined mesh. Also, we do not apply modal analysis on the complete scene because that would be computationally infeasible. The domain is first divided into many manageable partitions, and each one is analyzed separately.

Domain Decomposition is a very-well established area and has been widely studied in parallelizing various numerical techniques on many processors [DDM ]. Our main contribution in this work is the combination of a simple domain decomposition technique with eigenvalue correction, which offers tractable pre-processing times, efficient running times and improved simulation accuracy.

### 2.3 Interactive Techniques

Till now we have mainly discussed techniques which focus on the accuracy of the simulation. There has been a lot of research on interactive acoustics simulation, where the emphasis is placed on perceptual correctness, rather than numerical correctness of the simulation. As was mentioned earlier, most of these techniques are geometric methods [Funkhouser et al. 2003; Funkhouser et al. 2004; Antonacci et al. 2004]. Most room acoustics software use a combination of ray-tracing and image source method [Rindel ; Lokki 2002]. A notable exception in this category is [Tsingos et al. 2007], where the authors discuss an interactive GPU-based technique to approximate first-order scattering based on the wave equation. However, at present the technique does not handle higher order reflections and it is mainly applicable to open outdoor scenes where higher-order reflections are typically not important. Our work is complementary to these techniques, as we focus on efficiently capturing all types of important wave phenomena, including diffraction and higher-order reflections, while losing as little accuracy as possible.

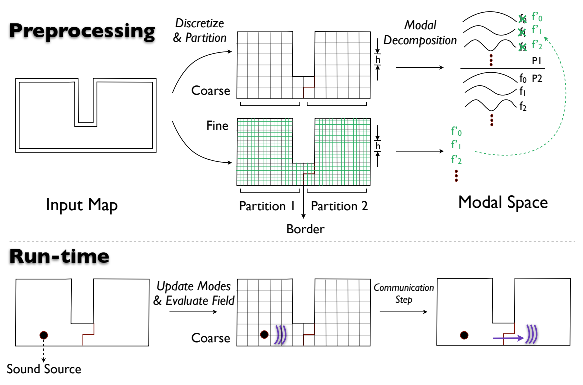

**Figure 2:** *A schematic diagram of our technique.* During preprocessing, the map volume is discretized into a uniform grid with spacing $h$ and then divided into many partitions (two in this case) such that the border area is minimized. Modal Analysis is performed on the partitions to retrieve their resonant modes of vibration and the corresponding eigenvalues. After that, the mesh is refined by a user specified amount based on the accuracy requirements and available compute and memory, and the corresponding, more accurate eigenvalues found. These eigenvalues are used to replace the ones found on the coarse mesh. As we describe in the paper, this improves the accuracy of the simulator. The runtime code runs on a coarse mesh. At each time step, the modes are evaluated in each partition to update the sound field within the partitions and an explicit communication step is performed at the borders to propagate the sound between partitions.

## 3 Methodology

In this section, we describe the mathematical formulation and give a description of our basic approach.

### 3.1 Mathematical Formulation

As described earlier, we use an FDM-based approach to directly solve the wave equation. The linear wave equation, on a domain $S$, is given by

$$\frac{\partial^2 p}{\partial t^2} - c^2 \nabla^2 p = F(\mathbf{x}, t), \tag{1}$$

where $\nabla^2$ is the Laplacian operator in 3D, $c$ is the speed of wave propagation, which is about 340 m/s for sound traveling in air at room temperature and $p$ is the pressure field to be computed[1]. It is interesting to note that except the sound speed, $c$, the exact physical phenomena resulting in the wave do not affect the above equation,

[1]From the wave equation, it can be shown directly that for a wave with constant frequency traveling in free space, $c = \nu\lambda$, where $\nu$ is the frequency and $\lambda$ the wavelength of the wave

so all the techniques we describe in this paper are also applicable to scalar wave propagation simulation in general, for example, seismic wave propagation. The term on the RHS, $F(\mathbf{x},t)$ is the forcing term and is non-zero wherever there is a sound source in the domain. This equation, along with a specification of the boundary conditions, constitute the complete problem we seek to solve. In this paper, we will only deal with the Neumann boundary condition, which corresponds to perfectly reflective walls and is imposed as $\frac{\partial p}{\partial \mathbf{n}} = 0$ on $\delta S$. It is possible to incorporate walls with arbitrary reflectivity in the formulation we present in this paper, but that would involve much more computation. This is because it is not possible in this framework to model arbitrary surface impedance completely in the spatial domain, so that they are incorporated in the discrete Laplacian operator. In contrast, as we describe later, this is easily done for the Neumann boundary condition. Thus, the surface interaction would have to be modeled explicitly at runtime and an extra forcing term applied at each surface cell, which would incur an extra computational cost proportional to the surface area of the scene.

Given the above equation, ideally, we seek a solution for the pressure field, $p$, in the continuum, which is an intractable mathematical problem, given the arbitrary geometry of the boundary. Traditionally, this problem is circumvented by discretizing the wave equation, for which there are many different methods. For this work, we have used an FDM-based formulation. We discretize the domain of interest into a Cartesian grid with uniform resolution. The resolution of the grid fixes the minimum wavelength and by implication, the maximum frequency that can be accurately simulated. Typically, the grid spacing should be at least 1/8 to 1/10 of the smallest wavelength. Since there is no a-priori knowledge of where a high frequency signal might travel within the domain during simulation, it is necessary that the grid have sufficient resolution everywhere to simulate the signal properly. Therefore, we use a uniform grid without employing any adaptive schemes. Once this discretization has been performed, the wave equation is transformed from a partial differential equation over space and time into a system of coupled ordinary differential equations in time alone:

$$\frac{\partial^2 P}{\partial t^2} + KP = F(t). \tag{2}$$

For a given discretization with say, $n$ cells, $P$ is a vector of length $n$ containing the pressure values at the cell centers. As a result of spatial discretization, the spatial operator, $-c^2\nabla^2$ is transformed into a symmetric matrix, $K$ of dimension $n \times n$. This is done by approximating the spatial differentiation by a difference of pressure values in neighboring cells, resulting in the standard 7-point stencil for the discrete Laplacian operator in 3D, which is second-order accurate in the cell size, $h$. It is important to note that in case of Neumann boundary condition, $\frac{\partial p}{\partial \mathbf{n}} = 0$ translates to a missing difference term in the Laplacian matrix, $K$. In this way, $K$ encodes the discrete Laplacian operator *along with the boundary conditions*.

The basic idea behind our approach is to perform a diagonalization of $K$, usually referred to as Modal Analysis:

$$K = E\Lambda E^T, \tag{3}$$

where $\Lambda$ is a diagonal matrix containing the eigenvalues and $E$ is the eigenvector matrix. The $i^{th}$ column of $E$ is the eigenvector corresponding to the $i^{th}$ eigenvalue, $\lambda_i$. Since $K$ is symmetric, $E^{-1} = E^T$. That is, the eigenvectors are mutually orthonormal. This leads to a large reduction in pre-computation time since the inverse need not be computed separately. Also, all the eigenvalues are real, and hence ordered, because $K$ is symmetric.

Plugging (3) into (2), multiplying by $E^T$ throughout and defining,

$$M = E^T P, \tag{4}$$

$$\widetilde{F} = E^T F, \tag{5}$$

we have the following:

$$\frac{\partial^2 M}{\partial t^2} + \Lambda M = \widetilde{F}(t). \tag{6}$$

Since $\Lambda$ is a diagonal matrix, the above equation is a system of $n$ *independent* ordinary differential equations in time for finding the mode vector, $M$. To further simplify analysis, we will assume that the forcing term, $F(t)$ is a piecewise constant function of time, so that it may be treated as a constant for a given period of time. Thus, the equation for the $i^{th}$ mode $m_i$ becomes

$$\frac{\partial^2 m_i}{\partial t^2} + \lambda_i m_i = \widetilde{F}_i. \tag{7}$$

Before we go on to describe the solution of this equation, we must address how we incorporate damping into the simulation. We introduce frequency-independent, mass-proportional damping to model viscous dissipation in air by modifying Equation (1) as follows:

$$\frac{\partial^2 p}{\partial t^2} + \alpha\frac{\partial p}{\partial t} - c^2\nabla^2 p = F(\mathbf{x},t), \tag{8}$$

where $\alpha$ is a constant pertaining to the amount of viscous damping. Although this is not the most physically accurate method for modeling damping, it is the most computationally inexpensive one and leads to acceptable results in practice. Performing a similar analysis on the above equation as in Eqns. (1) through (7), we get the following equation for the $i^{th}$ mode:

$$\frac{\partial^2 m_i}{\partial t^2} + \alpha\frac{\partial m_i}{\partial t} + \lambda m_i = \widetilde{F}_i(t). \tag{9}$$

This equation is the standard equation for forced vibration of a simple harmonic oscillator. We use the second-order in time leapfrog integrator to solve this equation. The update rule for the leapfrog scheme is given by:

$$m_i^+ = \frac{2 - \lambda dt^2}{1 + \frac{\alpha dt}{2}} m_i - \frac{1 - \frac{\alpha dt}{2}}{1 + \frac{\alpha dt}{2}} m_i^- + \frac{dt^2}{1 + \frac{\alpha dt}{2}} \widetilde{F}_i, \tag{10}$$

where $m_i^-$, $m_i$ and $m_i^+$ are the values of the $i_{th}$ mode at the previous, current and the next time-step respectively. Note that all the coefficients in the above equation can be pre-calculated during preprocessing time. Given this solution, the pressure field can be retrieved directly from (4):

$$M = E^T P \Rightarrow P = EM. \tag{11}$$

The intuition behind the mathematics of modal analysis is as follows: Any volume of air can be treated as a resonant cavity with a discrete set of resonant modes of vibration. Modal analysis, as expressed in Eqn. (3), seeks to extract these modes of vibration given the physical properties of the resonant cavity, as encoded by the Laplacian matrix, $K$. The resulting eigenvalues correspond to the inverse wavelengths, called *wavenumbers*, of the modes and the eigenvectors contained in the columns of $E$ encode the actual spatial pressure distribution of the modes. Moreover, the eigenvectors form a complete basis in the sense that any pressure distribution in the cavity can be expressed as a linear combination of the modes. As expressed in Eqn. (11), the eigenvector matrix $E$ can be interpreted as a basis transformation matrix, which takes the scaling factors for the modes encoded in the Mode Vector, $M$, and transforms them to the spatial pressure values, $P$. Similarly, $E^T$ performs the opposite transformation. In short, $E$ does a transformation from modal basis to spatial basis and $E^T$ does the inverse transform.

The basic approach based on modal analysis described above is computationally infeasible. The pre-processing time, for even a medium sized scenes that are a few tens of meters across, can easily run into days, requiring several Terabytes of memory. To illustrate why this is so and motivate the techniques to be presented later, we first discuss a basic implementation of the method outlined above and analyze the computational issues.

## 3.2 Basic Approach

The input to our algorithm is a closed, polygonal scene in 3D, along with the position of the listener, the sound sources and the forcing signals for the sound sources. We assume that the scene is static. However, the listener as well as the sound sources may move arbitrarily.

**Pre-processing**

1. Discretize the domain of interest into a uniform Cartesian grid with fixed spacing, $h$. The value of $h$ is decided based on the maximum frequency which will be simulated. As a rule of thumb, $h$ should be such that the maximum frequency, which corresponds to the smallest wavelength, is sampled at least about 8-10 times in each dimension.

2. Construct the Laplacian matrix, $K$, from the spatial discretization, assuming Neumann boundary condition on the whole boundary.

3. Diagonalize $K$, as described in Eqn. (3).

4. Store the resulting eigenvector matrix, $E$ and the corresponding eigenvalues, $diag\left(\Lambda\right)$.

**Runtime**

1. Based on the maximum frequency to be simulated, $\nu_{max}$, fix the grid spacing $h$ and set the simulation time step as $dt \leq \frac{h}{c\sqrt{3}}$, so that stability is ensured.

2. Initialize all modes, $m_i = 0$.

3. For each time step:

   (a) **Forcing Term**: Clear the forcing vector, $F$. For each sound source, use the source's location to find the cell it lies in and accumulate its contribution in the corresponding element of the forcing vector. Finally, use Eqn. (5) to transform the forcing term to modal basis, $\widetilde{F}$.

   (b) **Mode Update:** Use relation (10) to update each mode, taking the forcing term computed above into account.

   (c) **Pressure Evaluation:** Use the listener location to find the cell it is in, say $j$. Using relation (11), the pressure in the $j^{th}$ cell can be found as, $p = E_j M$, where $E_j$ is the $j^{th}$ row of $E$. This value of pressure is output as the sound signal value at the listener location at the current time.

A brute force implementation of the approach described above is intractable in both storage and computation for most interesting acoustic domains.

## 3.3 Computational Issues

Let us calculate the size of a typical domain. Consider an empty cubical room in 3D with dimensions $l \times l \times l$. Assume that the highest frequency we need to simulate is $\nu_{max}$, which corresponds to a wavelength of $\lambda_{min} = \frac{c}{\nu_{max}}$. The grid spacing, $h$, should be sufficient to capture this wavelength. Let us assume we wish to sample the smallest wavelength at least $k$ times in each direction, which implies $h = \frac{\lambda_{min}}{k}$. As stated earlier, typically $k = 8$ to 10. The number of grid cells can thus be computed as,

$$n = \left(\frac{l}{h}\right)^3 = \left(\frac{lk\nu_{max}}{c}\right)^3. \tag{12}$$

That is, $n$ increases cubically in both the maximum simulated frequency, $\nu_{max}$ and the linear dimension of the room, $l$. For example, for a medium-sized room, with $l = 10m$, $k = 6$ $and$ $\nu_{max} = 1000\,Hz$, $n \approx 5,600,000$. Even for this medium-sized scene, when we are simulating frequencies only up to $1\,kHz$ when the human audible range goes up to $22\,kHz$, the number of cells is more than a million! Also note that the dimensionality appears as the power in the above expression.

In light of the typical values of $n$ as described above, let us analyze the computational and space complexity of different steps of the basic approach. The pre-processing time is dominated by the diagonalization of the Laplacian matrix, $K$ (step 3). The size of $K$ is $n \times n$, and a brute-force dense matrix diagonalization would take $O\left(n^2\right)$ space and $O\left(n^3\right)$ time. Plugging in $n \approx 1M$, it is easy to see that the compute and memory requirements are in the range of $10^9$ GFLOPS and 1 Terabyte respectively. To make these values feasible on the machines available today, we need to reduce the pre-processing time as well as the memory requirement by at least a few orders of magnitude. We discuss how we achieve this using domain partitioning in the next section.

Next we discuss the runtime complexity of the basic approach, step by step. The bottleneck is, of course, step 3. A naive implementation of step 3(a) would take $\Theta\left(n^2\right)$ operations. However, observe that the number of sound sources in a scene, $s$, is usually much smaller than the number of cells, $n$, and hence the forcing vector $F$ contains only $s$ non-zero values. Performing multiplications with only non-zero values, the overall computation can be reduced to $\Theta\left(ns\right)$ operations. Both Step 3(b) and 3(c) obviously take $\Theta\left(n\right)$ operations. The important point to note from this analysis is that *the computation time is* $\Theta\left(ns\right)$ and the memory requirement is $\Theta(n^2)$, since the full matrix $E$ needs to be stored. Thus, the focus of the rest of this paper will be on reducing the pre-processing time and the resulting values of $n$ and $s$ at runtime.

## 4 Acceleration Techniques

In this section, we present our main techniques which enable efficient 3D sound simulation in scenes which would otherwise be intractable with the basic approach described in the previous section. First, we discuss domain partitioning to make pre-processing computationally tractable, so that it becomes feasible to handle moderately large scenes with hundreds of thousands of elements. To compensate for the resulting degradation in runtime performance due to domain partitioning, we discuss an eigenvalue correction technique to perform accurate simulations on a much coarser mesh at runtime, while using accurate eigenvalues from a finer version of the mesh.

### 4.1 Domain Partitioning

The basic approach described in Section 3.2 had a computational complexity of $\Theta\left(n^3\right)$ but this is still not enough for handling even medium-sized scenes which are a few ten meters in diameter. However, since the time complexity is super-linear in $n$, the number of elements in the scene, we can reduce the running time by making use of a divide and conquer approach. Suppose we were to partition the simulation domain into $R$, connected, non-overlapping partitions with roughly equal number of cells and pre-processed each one independently. The total time for pre-processing would be reduced to $T_{part} \approx R\left(\frac{n}{R}\right)^3 = \frac{n^3}{R^2}$ and the memory requirement reduced to $M_{part} \approx R\left(\frac{n}{R}\right)^2 = \frac{n^2}{R}$. Thus, the total pre-processing time scales down quadratically as we increase the number of partitions, $R$ and the memory requirement scales down linearly. For example, if we use $R = 300$, the pre-processing time decreases by $90,000$ times, and the memory requirement by 300 times. This is the chief motivation for Domain Partitioning, as it provides a scalable way to manage pre-processing resource requirements. However, this decrease in pre-processing time is accompanied by a decrease in performance at runtime. We address this problem in Section 4.4.

Next, we discuss how to perform sound simulation in the presence of Domain Partitioning. For clarity of presentation, we will discuss the theory for a 1D simulation with the number of partitions, $R = 2$. The mathematics for handling multiple partitions in 3D follows analogously from this description. Let us consider a 1D domain with grid spacing $h$ and number of cells $2n$. As described previously, the spatially discretized equation for wave propagation for this domain is Eqn. (2). Let us consider the equation in more detail:

$$\frac{\partial^2 P}{\partial t^2} - \frac{c^2}{h^2}\begin{bmatrix} \ddots & & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & 1 & -2 & 1 & & \\ & & & 1 & -2 & 1 & \\ & & & & & & \ddots \end{bmatrix} P = F(t)\,. \tag{13}$$

Suppose we wish to partition the domain into two equal partitions with $n$ cells each, so that each might be treated independently during pre-process. This would mean that $K$ must somehow be decoupled into a block diagonal form and the off-diagonal entries must be accounted for properly. Mathematically, this can be achieved as follows:

$$\frac{\partial^2 P}{\partial t^2} - \frac{c^2}{h^2}\left[\begin{array}{ccc|ccc} \ddots & & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -1 & & & \\ \hline & & & -1 & 1 & \\ & & & 1 & -2 & 1 \\ & & & & & \ddots \end{array}\right] P = F(t) + CP, \tag{14}$$

where the coupling matrix, $C$ is given by

$$C = \frac{c^2}{h^2}\left[\begin{array}{ccc|ccc} 0 & & & & & \\ & \ddots & & & & \\ & & -1 & 1 & & \\ \hline & & 1 & -1 & & \\ & & & & \ddots & \\ & & & & & 0 \end{array}\right]. \tag{15}$$

That is, the pressure coupling at the interface between two partitions is moved into the forcing term on the RHS in the form of the coupling matrix, $C$. Also, note that the action of $C$ is to compute the gradient of pressure on the interface, scaled by $(\frac{c^2}{h})$. Thus, in effect, the forcing term is explicitly updated at each time step depending on the gradient of pressure on the interface. This explicit update may be intuitively seen as a communication step to pass the sound between the two partitions. From this description, the extension to general partitioning in 3D is quite intuitive:

- The system on the LHS can be seen as two completely independent resonant cavities, each with its own Laplacian matrix. In general, with $R$ partitions, $K$ is transformed into a block diagonal form with $R$ blocks, and all the coupling terms are moved into the coupling matrix, $C$ on the RHS, which are computed at runtime. The number of coupling terms is equal to the total number of faces on the partition interfaces.

- Each partition is treated independently while pre-processing and Neumann boundary condition is applied on the interfaces when constructing the individual Laplacian Matrices.

- To evaluate the forcing term on the partition interfaces, an explicit pressure evaluation must be performed at runtime on all cells having a face on a partition interface, so that the gradient might be computed.

- Since the partitioning is based on splitting a second-order Laplacian operator, the interface handling is also second-order accurate in space.

## 4.2 Runtime Processing

To perform the communication step, we need to calculate the pressure values of all the cells on the interfaces of the $j^{th}$ partition and use that to somehow find the appropriate forcing terms for all the modes in the current partition. We use a simple technique for this: Use the relation $P_j = E_j M_j$, which is Eqn. (11) rewritten with the subscript denoting the partition number. That is, do a basis transformation from modal space to real space, to retrieve the pressure field on the interface. Once that is done, the forcing vector, $F_j$ on the right can be calculated easily for each partition, as described in Eqns. (14) and (15). Transform this forcing vector back to modal space as $\widetilde{F_j} = E_j^T F_j$. Use $\widetilde{F_j}$ for the next mode update loop, as given by equation 10.

## 4.3 Complexity Analysis

Let us consider the $j^{th}$ partition. The time complexity analysis is very similar to that presented in Section 3.2, except the presence of the additional pressure evaluation on the interface, as in Eqn. (10). Let the number of cells in the $j^{th}$ partition be $n_j$ and the number of cells which have a face on the partition interface be $b_j$ and the number of sound sources be $s_j$. A naive implementation for pressure evaluation would take $\Theta\left(n_j^2\right)$ time. This can be improved by observing that we only need to evaluate the pressure on $b_j$ cells – the ones that have a face on the partition interface. That means we only need to use the corresponding rows in $E_j$ and the time complexity reduces to $\Theta\left(n_j b_j\right)$ operations. Summing the running time over all partitions, we have

$$T_{runtime} = \Theta\left(\sum_j \left(n_j + n_j s_j + n_j b_j\right)\right).$$

Assuming the maximum number of cells over all partitions is $n_{max}$, and denoting $\Sigma b_j = B$, $\Sigma s_j = s$ and noting that $\Sigma n_j = n$, we get the following upper bound for the runtime performance

$$T_{runtime} = O\left(n + n_{max}\left(s + B\right)\right).$$

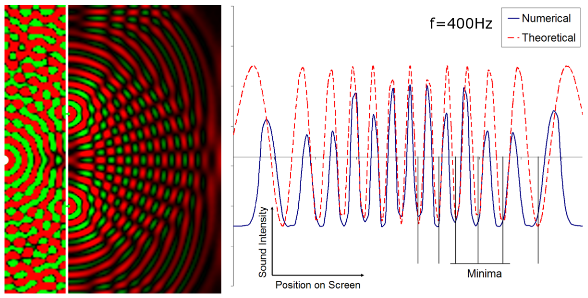

**Figure 3:** *Results of sound simulation for a double slit experiment.* The images show the color-coded pressure field (or equivalently, the sound) in a 2D domain. A sound source is placed at the center-left of the domain (shown as a white circle), emitting a sine wave with fixed frequency of 400 Hz. Note that the sound wave diffracts considerably at the slits, which act as secondary point sources. The sound waves from both the slits interfere to generate the classical interference pattern, with clearly visible maxima and minima. The graph on the right compares the theoretical and predicted sound intensity levels on a virtual screen placed to the right of the slits. Note the close agreement between the theoretical and predicted positions of the extrema.

This expression clearly shows that the runtime performance is sensitive to the total number of cells, $n$ and the total number of interface cells, $B$. The total number of sound sources, $s$ is usually many orders of magnitude smaller than $B$ and can be safely disregarded. We can easily handle $\sim$ 100 sound sources without changing the runtime performance much.

It is important to note that instead of computing the coupling in space, one may attempt to do so directly in modal space. However, this would be much more costly because every mode of one partition would interact with every other mode of a neighboring partition, which would mean that if the number of cells in the two partitions are $n_i$ and $n_j$ respectively, the complexity for interface handling would be $n_i n_j$. This is much worse than with spatial coupling, since only coupling at the interface cells needs to be computed. That is, if the two partitions share $B$ faces on the interface, the complexity would be $(n_i b_i + n_j b_j)$, which is much better, since $b_i << n_i$.

In order to ensure that the pre-processing time is uniformly distributed, the different values of $n_j$ should be nearly equal. There are packages for this task which usually give near-optimal results for most domains. In particular, we have used METIS [Karypis and Kumar 1999] for performing the domain partitioning. Based on the voxelization of the scene, we build an unweighted connectivity graph based on the 6-neighborhood of a voxel in 3D and run METIS on the resulting graph. METIS makes sure that the partitions are near-equal in size and also that the total interface area between partitions is minimized. However, these constraints are only meant for increasing efficiency and the techniques presented in this paper work for any valid partitioning of the domain.

## 4.4 Accurate Simulations on a Coarse Mesh

The Domain Decomposition technique described above renders the problem of pre-processing more tractable. At the same time, it naturally decomposes the problem we are attacking into two parts: Wave propagation within a partition and communication across the interface. Intuitively, the numerical behavior of sound traveling in the domain can be understood in terms of its behavior when it is within a partition and when it is crossing the interface between two partitions. As discussed above, the efficiency of our technique is mainly governed by the number of cells in the scene, $n$, and the total number of interface cells, $B$. Therefore, the efficiency is very sensitive to the mesh resolution. If a mesh is refined $r$ times in each dimension in 3D, the number of cells will become $nr^3$ and the number of interface cells will change to $Br^2$. Therefore, the runtime performance will reduce by a factor of $r^5$.

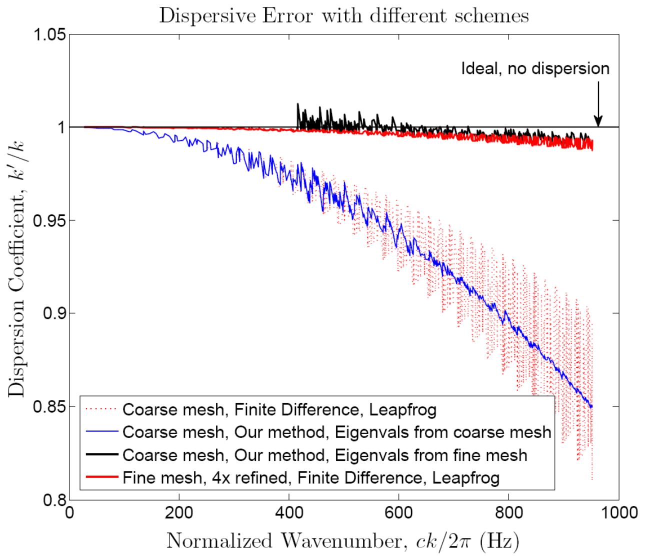

**Figure 4:** *Accuracy comparison with a Leapfrog finite difference solver.* The numerical dispersion with our technique on a coarse mesh (blue) follows a similar trend as that of the finite difference technique on the same mesh. However, with corrected eigenvalues taken from a $4\times$ refined mesh, our technique exhibits improved dispersion characteristics, similar to the finite difference technique on a refined mesh. Since the eigenvalues can be pre-calculated accurately on a refined mesh, we are able to achieve reduced dispersion on a coarse mesh.

Therefore, to increase efficiency, the simplest method would be to just use a very coarse mesh, which supports about 2 samples per the smallest wavelength. Any fewer number of samples are not allowed by the Nyquist criterion and the signal cannot possibly be reconstructed correctly on such a mesh. Compared to a mesh with say, 6 samples per wavelength, such an implementation would be $3^5 = 243$ times faster. However, working on a coarse mesh reduces the accuracy of the simulation. The main problem with a coarse mesh in context of sound simulation is numerical dispersion. Because we are using a second-order spatial discretization to approximate the continuum Laplacian operator, there is a third order residual term, the effect of which is that higher frequencies travel slower through the medium than lower frequencies. Of course, this is not physically observed and is a purely numerical artifact. The net effect of dispersion is that any signal traveling through the medium gradually loses its shape with time, as the phase relations between the different frequency components of the wave are lost. Perceptually, this means that eventually the sound degrades into a noisy signal with no coherent information. So, if we are willing to use a coarse mesh, some method must be found to reduce the resulting dispersion errors as sound propagates within a partition. Therefore, in the rest of the discussion in this section, we will restrict our attention to propagation within a single partition.

We will now discuss some important mathematical equivalences to motivate our technique of using corrected eigenvalues, as well as

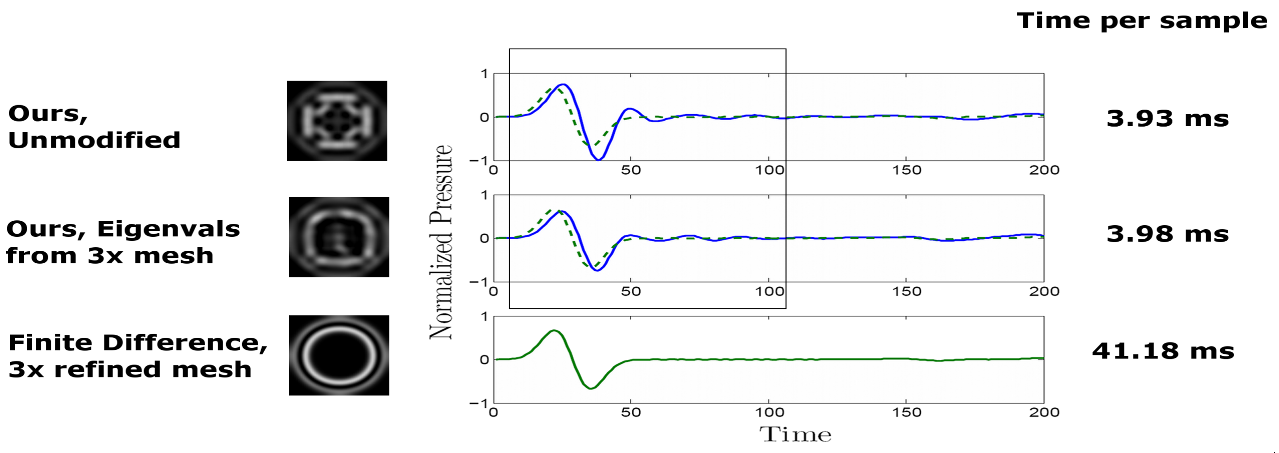

**Figure 5:** *Performance comparison with a Leapfrog Finite Difference solver.* Three different scenarios are shown. From top to bottom: Our unmodified technique on a coarse mesh, our technique with eigenvalues taken from a $3\times$ refined mesh with runtime computation on a coarse mesh and, a reference solution with the Leapfrog finite difference technique on a $3\times$ refined mesh. The scene is a $2m \times 2m \times 2m$ cube in 3D, which was divided into 10 partitions, with $h = 0.125$ and the time step set to 20,000 Hz. A 700 Hz pulse (corresponding to 3-4 cells per wavelength), is introduced at the center. The images show a snapshot of the intensity field on a 2D slice of the cube along Z. Note that our technique with corrected eigenvalues is able to propagate sound without much dispersion, even though the spatial reconstruction is coarse because of the coarse mesh it is working on. In comparison, our technique without corrected eigenvalues shows large amounts of dispersion – the circular shape of the signal is visibly lost. The plots to the right show the pressure at the center as a function of time. Our technique with corrected eigenvalues reduces the dispersion noticeably and matches the reference numerical solution much more closely, shown as the dotted line. Note that our technique is about $10\times$ faster than the finite difference implementation on a $3\times$ refined mesh, with slight loss in accuracy. Also note that the domain decomposition is arbitrary and there are no visible discontinuities in the pressure field due to domain partitioning.

give the reader an intuition of why it works. We present numerical results which corroborate our theoretical derivations in the next section. Consider the discretized equation governing undamped, unforced wave propagation, which one gets by neglecting the forcing term on the RHS of equation 2:

$$\frac{\partial^2 P}{\partial t^2} + KP = 0. \tag{16}$$

Since K is a second-order approximation of the continuum Laplacian operator, one can write the following:

$$\frac{\partial^2 P}{\partial t^2} - c^2\nabla^2 P \approx \epsilon h^2 \nabla^3 P. \tag{17}$$

That is, we can see the discrete system as if it was the continuum system with an extra dispersion term on the RHS. This truncation error causes any finite difference simulation on a mesh to suffer from numerical dispersion. In the context of our framework, we diagonalize the Laplacian matrix, $K$ and transform basis to work in modal space instead of real space, as given in equation 6. In light of the fact that we are diagonalizing an approximation of the continuum operator, the error being $O\left(h^2\right)$, we get the following relation in modal space:

$$\frac{\partial^2 M}{\partial t^2} + \Lambda M \approx O\left(h^2\right). \tag{18}$$

Therefore, the residual term in both cases is of the same order, and both equations correspond exactly to each other, assuming that the diagonalization is exact to floating point accuracy. The most important upshot of this is: *For a finite difference scheme, dispersive error depends on the mesh resolution, while for the modal scheme, dispersive error depends on the accuracy of the eigenvalues.* It should be obvious from the above that if the same mesh is used for a finite difference simulator and the modal scheme described, with the same time integration method, one would get similar dispersive errors in both simulations. However, consider what happens if we find the eigenvalues, $\Lambda$ on a much finer version of the mesh. Since all eigenvalues are real and ordered, the replacement procedure is trivial – the first, say, $i$ eigenvalues are replaced by the first $i$ eigenvalues on the finer mesh. Note that this works even in the presence of degenerate eigenvalues; switching two identical values doesn't change anything. It immediately follows that the dispersive error would correspond to that of the finer mesh, even though the eigenvectors correspond to the coarse mesh. Intuitively, this means that the signal will be reconstructed more approximately in space, but it will propagate with much less dispersive error, the error being similar to that on a fine mesh from which the eigenvalues are derived.

Here is where the main advantage of modal analysis becomes clear. Calculating the eigenvalues is a one-time operation. We can, for example, use eigenvalues from a 4x refined mesh (refined 4 times in each dimension) and get $\frac{1}{16}^{th}$ the dispersive error, since the error is second-order in cell size, $h$. Since only the eigenvalues have changed and the computation is just the same, there is absolutely no change in the runtime performance or the memory requirements. Contrast this with a finite difference scheme on a 4x refined mesh. The computation per time step immediately becomes 16 times in 2D, and 64 times in 3D. The finite difference approach, with its lack of pre-processing, would be more useful for short simulations. But for long time simulations, where the gain in runtime performance justifies the pre-processing time, the modal approach is a much better choice.

The gain in runtime performance described above is at the cost of pre-processing time. Consider a scene with $n$ cells, divided into $R$ partitions and each partition refined $r$ times in each dimension.

Also note that we only need the first $\frac{n}{R}$ eigenvalues of the refined mesh for each partition. Thus the time taken for each partition is proportional to $\frac{n}{R}\left(\frac{nr^3}{R}\right)^2$. Since this is repeated for $R$ partitions, the total time becomes $n\left(\frac{nr^3}{R}\right)^2$. Observing that a full diagonalization on the coarse mesh would take time proportional to $n^3$, the net gain in pre-processing performance becomes, $Gain = R^2/r^6$. Thus, given a particular scene, we select the value of $R$ based on the one which yields the best runtime performance, based on the complexity analysis given in 4.3 and then select a value of $r$ as high as possible, to get the best accuracy while keeping the above relation in mind, so that the pre-processing requirements are tractable. This way, we are able to trade-off between runtime performance, accuracy and pre-processing resources. As an implementation detail, in this work, we have used the ARPACK [Lehoucq et al. 1997] eigenvalue package for computing a selected initial part of the eigenvalue spectrum.

## 5 Results

In this section, we present the main results to demonstrate the accuracy and efficiency achievable with our approach. All the simulations were performed on an Intel Xeon Quad-Core 2.79GHz CPU with 2GB RAM.

### 5.1 Accuracy: Interference and Diffraction

We evaluated the accuracy of our simulator on the basis of how well it simulates the wave phenomena of reverberation, interference and diffraction. As mentioned in the introduction, these phenomena are critical for a physically correct simulation of sound in general environments. We first discuss our results for interference and diffraction on a 2D double-slit experiment. The setup is shown in Figure 3. The image on the left shows the color-coded pressure amplitude in the scene. A sound source is placed at the center-left of the domain (shown as a white circle), emitting a sine wave with fixed frequency of 400 Hz. The domain boundary is fully reflective. The grid spacing for the spatial discretization of the domain is $h = \frac{1}{8}m$, which means we can handle up to $\sim$ 900 Hz with about $k = 3$ elements per wavelength. An obstructing wall is placed in front of the sound source at a distance of $4m$. The wall has two symmetrically placed slits, each one cell wide ($\frac{1}{8}m$). A virtual screen is placed on the opposite side of the wall, at a distance of $2.5m$ from the slits.

Sound waves traveling from the source first hit the wall and only a small portion of it passes through the slits, which act as secondary point sources on the other side of the wall. This demonstrates that diffraction is properly captured by our technique. Next, sound from the two slits interferes throughout the domain, resulting in an interference pattern. The pressure field shows a near-uniform angular distribution of maxima and minima, depending on where sound waves interfere constructively and destructively, respectively. This result clearly demonstrates that interference is also simulated correctly with our technique. To quantify the accuracy of the resulting sound field, we compare the observed and theoretical sound intensity (square of the sound amplitude) along the length of the virtual screen. The right side of Figure 3 shows the graphs for the sound intensity against the position on the virtual screen. Note the close agreement between the positions of the expected minima and maxima and the numerically computed values.

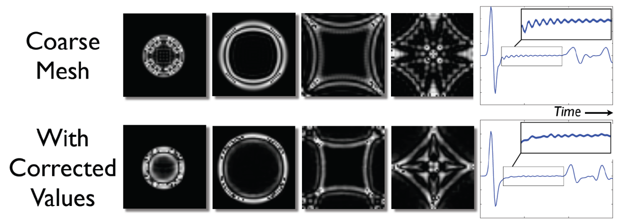

**Figure 6:** *Visualization of reduced numerical dispersion.* Comparison of numerical results from simulation on a coarse mesh with and without corrected eigenvalues from a $3\times$ refined mesh on a 2D scene. The images to the left show snapshots in time of the sound intensity in a square 2D domain with dimensions $6m \times 6m$ and h=0.125m. The domain was divided into 10 partitions. A 500Hz Gaussian pulse is triggered at the center and spreads out, reflects from the walls and interferes with itself at the center. Note that the coarse simulation has a lot more "ringing," as the signal shape is gradually lost over time, which corresponds to numerical dispersion. This is more obvious in the last image. The plots to the right show the pressure at the center of the domain as a function of time. As shown in the insets, after the initial impulse is fed, the coarse simulation has a high frequency tail, while for the simulation with corrected eigenvalues, this is much less so. This shows that with corrected eigenvalues, the dispersion is reduced.

### 5.2 Accuracy: Dispersion analysis and comparison with a finite difference solver

In order to clearly assess the benefit with our approach, it is important to assess how it compares in terms of accuracy to a standard finite difference implementation. In this section, we will briefly discuss a second-order accurate in space and second-order accurate in time method for solving the wave equation numerically, and compare it against our method. This is done by performing a dispersion analysis of both methods on a 2D square scene, for undamped, unforced wave propagation. This scene has been chosen because it is easier to perform dispersion analysis on rectangular domains.

We had theoretically motivated in Section 4.4 that the accuracy of our method with eigenvalues substituted from a refined mesh should be comparable to the accuracy of a finite difference method on the refined mesh. The finite difference method we consider in this work is the standard cell-centered second-order Leapfrog integrator. In particular, consider a square domain of size $[0,1]\times[0,1]$, discretized uniformly into $N \times N$ cells. The cell size is thus given by $h = \frac{1}{N}$. Lets denote the pressure at the cell with coordinate $(ih, jh)$ at time step $n$ by $P^n_{i,j}$. Then the finite difference integrator is given by –

$$\begin{aligned} P^{n+1}_{i,j} &= 2P^n_{i,j} - P^{n-1}_{i,j} \\ &+ \frac{c^2\Delta t^2}{h^2}\left(P^n_{i+1,j} + P^n_{i-1,j} + P^n_{i,j-1} + P^n_{i,j+1} - 4P^n_{i,j}\right) \end{aligned} \tag{19}$$

As a first step in the dispersion analysis, note that the wave equation can be solved analytically for the square domain with Neumann boundary conditions. This is done by expressing the pressure in the domain in the Cosine basis as –

$$P(x,y,t) = \sum_{k_x=0}^{N-1}\sum_{k_y=0}^{N-1} m(k_x,k_y,t)\, cos(\pi k_x x)\, cos(\pi k_y y), \tag{20}$$

where $k_x$ and $k_y$ are the wavenumbers (proportional to the inverse of the wavelength) in the X and Y directions respectively and $m(k_x,k_y,t)$ is the corresponding scaling coefficient. Plugging this into the wave equation (1) and assuming the forcing term on the RHS is absent, and solving w.r.t time, we get the important relation –

$$\begin{aligned} m(k_x,k_y,t) &= Ae^{ick(k_x,k_y)t} + Be^{-ick(k_x,k_y)t}, \\ k(k_x,k_y) &= \pi\sqrt{k_x^2+k_y^2}. \end{aligned} \tag{21}$$

The constants $A$ and $B$ depend on the initial conditions and are not relevant here. The important quantity is $k(k_x,k_y)$, which, gives the analytically computed wavenumber, given the spatial wavenumbers in the X and Y direction. Carrying out a similar procedure with equation (19), by plugging in equation (20) and denoting all corresponding variables with a prime, we get –

$$\begin{aligned} m'(k_x,k_y,t) &= Ae^{ick'(k_x,k_y)t} + Be^{-ick'(k_x,k_y)t}, \\ k'(k_x,k_y) &= \tfrac{2}{h}\sqrt{sin^2(\pi k_x h/2) + sin^2(\pi k_y h/2)}. \end{aligned} \tag{22}$$

Therefore, the numerical speed of propagation for different wavenumbers is not the same as the analytical expression. Note that for every wavenumber $(k_x,k_y)$, $Lim_{h\to 0}k' = k$, which is required for consistency. The amount of numerical dispersion for a given wavenumber, $k = \sqrt{k_x^2+k_y^2}$ can be quantified by $\frac{k'}{k}$, called the dispersion coefficient. Intuitively, the dispersion coefficient measures how slow or fast a wave with a particular wavenumber would travel on the mesh, compared to the speed of sound, $c$. Ideally, the dispersion coefficient should be 1 for all wavenumbers. Figure 4 shows the dispersion coefficient of the finite difference scheme described above for increasing wavenumbers, $k$. Note that when the cell size, $h$ is large, as on the coarse mesh, the dispersion is really large for higher wavenumbers. This improves considerably on a mesh which has been refined 4 times in both dimensions.

Till now we have discussed the Finite Difference technique we compare against and its dispersion analysis. The dispersion analysis of our method closely parallels the above discussion. Consider Equation (7) and assume the forcing term is 0. It is easy to see that one gets a solution very similar as (21), with $k_i'' = \frac{\sqrt{\Lambda_i}}{c}$. Again, one can obtain the dispersion coefficient by dividing by the analytical expression for $k$ given by (21). Figure 4 shows a comparison of the dispersion coefficients obtained for our method and the finite difference method for different cases. Firstly, note that as was theoretically discussed in Section 4.4, the dispersion coefficients on a coarse mesh for our technique and the finite difference technique follow the same trend, exhibit large amounts of dispersion for higher wavenumbers. The most important fact to note, however, is that the dispersion coefficients for the finite difference method on a 4x refined mesh are very similar to that of our technique on a coarse mesh with eigenvalues derived from a 4x refined mesh. This corroborates the theoretical claims made in Section 4.4 that the accuracy of the eigenvalues determines the dispersion of our technique and demonstrates that substituting eigenvalues from a refined mesh does reduce the dispersion error with our technique without incurring extra runtime cost. At this point we emphasize again that this is the chief advantage and main motivation for working in the modal basis instead of real space.

## 5.3 Efficiency

To compare the performance and accuracy of our approach with the finite difference scheme described above, we implemented both in a common code-base. For the test scene, we considered a $2m \times 2m \times 2m$ cube in 3D, with $h = 0.125m$ and divided the domain into 10 arbitrary partitions and initiated a 700 Hz pulse at the center. We would like to emphasize here that the partitioning need not have any specific structure. Figure (5) illustrates the results we obtained. There are three cases we have considered: the bottom row shows the results obtained with the finite difference scheme on a 3x refined mesh, and can be treated as a reference solution, the top row shows our technique on a coarse mesh without using corrected eigenvalues, which can be considered a basic implementation and the middle row shows our technique with corrected eigenvalues from a 3x refined mesh. Note that the accuracy of our technique is intermediate between the two. Although dispersion is still present, visible as the wavy tail after the initial signal, it is much smaller compared to the basic approach. The important thing to note is that this gain in accuracy comes at almost no reduction in performance. The timings on the right side of the figure show the time taken to generate 1 sample of audio with the different techniques. It is clear that our technique is about 10x faster for this scene. Also, note that there are no visible discontinuities in the field at the partition interfaces, which shows that partitioning is being handled accurately. Another demonstration of reduced dispersion with our technique with corrected eigenvalues on a 2D scene is shown in Figure 6.

## 5.4 3D Environment

To demonstrate the efficiency and robustness of our approach, we have implemented our algorithm and tested it on a moderately complex architectural model. Figure 7 shows the results for a sound simulation on the building. The user may move freely through the model and all the sounds in the scene are passed in to a sound simulator which implements all the techniques described in this paper. The physical dimensions of the building are $12m \times 13m \times 7m$, which is about the size of a large hall. The grid spacing for spatial discretization on the coarse mesh is $h = \frac{1}{8}m$. The total air volume of the scene is approximately $600m^3$, which corresponds to about 300,000 cells. The map was first partitioned into 300 partitions, each having about 1000 cells. For each partition, the eigenvalues were corrected by using the eigenvalues from a 3x refined mesh, each with about 27,000 cells. The refinement factor was fixed so that the matrix diagonalization did not take more than 2GB of memory. The pre-processing was performed on a cluster, with each partition being processed on a separate node, in parallel. Each node has a 2.3GHz Intel EM64T processor with 4M L2 cache. The total time for pre-processing this scene was about 50 minutes.

Even though the pre-processing is expensive, it is still tractable. Following the analysis in Section 4.4, observe that dividing the scene into 300 partitions and then taking eigenvalues from a 3x refined mesh reduces the pre-processing time by $\frac{300^2}{3^6} \approx 100$ times. Therefore, the pre-processing requirements are reduced considerably, even though we are sacrificing some pre-processing efficiency for improved accuracy at runtime. The simulator takes about 300 ms per sample of audio, at an audio update rate of 5000 Hz, while consuming about 1GB of memory. Note that the compute requirements are high even when we are using a coarse mesh which is close to the Nyquist threshold. In comparison, a finite difference implementation would require a 3x refined mesh, which would have $27 \times 300,000 = 8.1M$ cells.

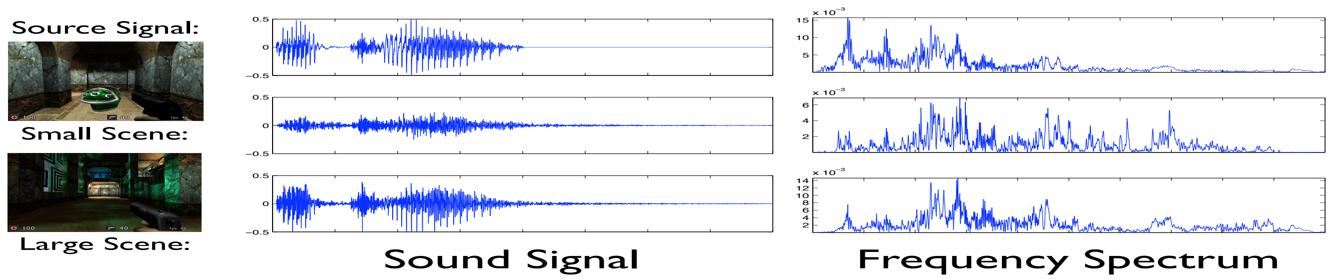

**Figure 7:** *Results for sound simulation on a complex architectural scene in 3D.* The physical dimensions of the map are $12m \times 13m \times 7m$ and it has an air volume of 300,000 cells, which we partition into 300 partitions. We are able to perform simulations for frequencies up to 1kHz on this map. The listener is at the player's location. Note that the input sound is modified considerably by the environment depending the size of the space (small or large). Also note that the frequency content of the input signal is effectively filtered depending on the resonant properties of the area of the map where the listener is located. Note the "reverberant tail" in both the simulated sounds.

The input signal was band-passed below 1kHz, since that is the maximum simulated frequency. Note the marked difference between the input sound and the signal received by the listener. This shows that the input sound is modified considerably by the environment depending the size of the space (small or large). The "tail" of the response shows that late reverberations (corresponding to hundreds of reflections) are being captured by the simulator. Also note that the frequency content of the input signal is markedly different depending on how it interferes with itself after multiple reflections, which is crucially influenced by the area of the map where the listener is located. To show the generality of our approach, we have also performed the simulations on a different environment, shown in Figure 1. This map has about 290,000 elements and was partitioned into 290 partitions. The pre-processing for this scene took about 65 minutes. The sound simulation was carried out at an update rate of 5000 Hz. The time taken to generate one sample of audio at runtime was about 250 ms.

## 6 Conclusion

We have proposed an efficient and accurate technique for modeling sound propagation in an arbitrary 3D environment by directly integrating the wave equation. Our results show that important acoustic effects like interference, diffraction and reverberation are accurately captured by our technique. Moreover, due to the combination of a simple and stable domain decomposition technique and a technique to reduce the dispersion on a coarse mesh by utilizing the eigenvalues from a refined mesh, we are able to handle complex 3D scenes with $\sim 300,000$ elements and frequencies up to 1 kHz, while many previous numerical techniques have been demonstrated simulating frequencies only up to a few hundred Hz. We believe that further optimization combined with the increased computing power of the upcoming many-core processors will enable our technique to perform interactive simulation on 3D scenes with millions of elements in the foreseeable future, enabling rapid design and simulation of building acoustics and offering a much more immersive auditory experience for virtual environments.

## 7 Acknowledgements

This work was supported by the Army Research Office, Defense Advanced Research Project Agency, National Science Foundation, RDECOM and Intel Corporation.

## References

ALFORD, R. M., KELLY, K. R., AND BOORE, D. M. 1974. Accuracy of finite-difference modeling of the acoustic wave equation. *Geophysics 39*, 6, 834–842.

ALLEN, J. B., AND BERKLEY, D. A. 1979. Image method for efficiently simulating small-room acoustics. *J. Acoust. Soc. Am 65*, 4, 943–950.

ANTONACCI, F., FOCO, M., SARTI, A., AND TUBARO, S. 2004. Real time modeling of acoustic propagation in complex environments. *Proceedings of 7th International Conference on Digital Audio Effects*, 274–279.

BERTRAM, M., DEINES, E., MOHRING, J., JEGOROVS, J., AND HAGEN, H. 2005. Phonon tracing for auralization and visualization of sound. In *IEEE Visualization 2005*.

BOTTELDOOREN, D. 1994. Acoustical finite-difference time-domain simulation in a quasi-cartesian grid. *The Journal of the Acoustical Society of America 95*, 5, 2313–2319.

BOTTELDOOREN, D. 1995. Finite-difference time-domain simulation of low-frequency room acoustic problems. *Acoustical Society of America Journal 98* (December), 3302–3308.

DDM. http://www.ddm.org.

DEINES, E., MICHEL, F., BERTRAM, M., HAGEN, H., AND NIELSON, G. 2006. Visualizing the phonon map. In *Eurovis*.

FUNKHOUSER, T., TSINGOS, N., AND JOT, J.-M. 2003. Survey of methods for modeling sound propagation in interactive virtual environment systems. *Presence and Teleoperation*.

FUNKHOUSER, T., TSINGOS, N., CARLBOM, I., ELKO, G., SONDHI, M., WEST, J. E., PINGALI, G., MIN, P., AND NGAN, A. 2004. A beam tracing method for interactive architectural acoustics. *The Journal of the Acoustical Society of America 115*, 2, 739–756.

KARJALAINEN, M., AND ERKUT, C. 2004. Digital waveguides versus finite difference structures: equivalence and mixed modeling. *EURASIP J. Appl. Signal Process. 2004*, 1 (January), 978–989.

KARYPIS, G., AND KUMAR, V. 1999. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing 20*, 1, 359–392.

KINSLER, L. E., FREY, A. R., COPPENS, A. B., AND SANDERS, J. V. 1999. *Fundamentals of Acoustics*. Wiley, December.

KLEINER, M., DALENBCK, B.-I., AND SVENSSON, P. 1993. Auralization - an overview. *JAES 41*, 861–875.

KROCKSTADT, U. 1968. Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound Vibration*.

KUTTRUFF, H. 2000. *Room Acoustics*. Taylor & Francis, October.

LEHOUCQ, R., SORENSEN, D., AND YANG, C. 1997. Arpack users' guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods. Tech. rep.

LOKKI, T. 2002. *Physically-based Auralization*. PhD thesis, Helsinki University of Technology.

MIN, P., AND FUNKHOUSER, T. 2000. Priority-driven acoustic modeling for virtual environments. In *EUROGRAPHICS 2000*.

MURPHY, D., KELLONIEMI, A., MULLEN, J., AND SHELLEY, S. 2007. Acoustic modeling using the digital waveguide mesh. *Signal Processing Magazine, IEEE 24*, 2, 55–66.

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*, ACM Press, 175–181.

RAGHUVANSHI, N., AND LIN, M. C. 2006. Interactive sound synthesis for large scale environments. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 101–108.

RINDEL, J. H. The use of computer modeling in room acoustics.

SABINE, H. 1953. Room acoustics. *Audio, Transactions of the IRE Professional Group on 1*, 4, 4–12.

SAKAMOTO, S., YOKOTA, T., AND TACHIBANA, H. 2004. Numerical sound field analysis in halls using the finite difference time domain method. In *RADS 2004*.

SAKAMOTO, S., USHIYAMA, A., AND NAGATOMO, H. 2006. Numerical analysis of sound propagation in rooms using the finite difference time domain method. *The Journal of the Acoustical Society of America 120*, 5, 3008–3008.

SAVIOJA, L., RINNE, T., AND TAKALA, T. 1994. Simulation of room acoustics with a 3-d finite difference mesh. *Proc. Int. Computer Music Conf*, 463–466.

SAVIOJA, L., BACKMAN, J., JRVINEN, A., AND TAKALA, T. 1995. Waveguide mesh method for low-frequency simulation of room acoustics. In *15th International Congress on Acoustics (ICA'95)*, vol. 2, 637–640.

SAVIOJA, L. 1999. *Modeling Techniques for Virtual Acoustics*. Doctoral thesis, Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, Report TML-A3.

TSINGOS, N., FUNKHOUSER, T., NGAN, A., , AND CARLBOM, I. 2001. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *Computer Graphics (SIGGRAPH 2001)*.

TSINGOS, N., DACHSBACHER, C., LEFEBVRE, S., AND DELLEPIANE, M. 2007. Instant sound scattering. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*.

VAN DEN DOEL, K., KRY, P. G., AND PAI, D. K. 2001. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 537–544.

VAN DUYNE, S., AND SMITH, J. O. 1993. The 2-d digital waveguide mesh. In *Applications of Signal Processing to Audio and Acoustics, 1993. Final Program and Paper Summaries., 1993 IEEE Workshop on*, 177–180.