University of Huddersfield Repository

Thomas, Matthew Reuben

Wayverb: A Graphical Tool for Hybrid Room Acoustics Simulation

Original Citation

Wayverb: A Graphical Tool for Hybrid Room Acoustics Simulation

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Master of Arts

Matthew Reuben Thomas

January 2017

Abstract

Acoustic simulation aims to predict the behaviour of sound in a particular space. These simulations can be carried out on commodity computing hardware, and are faster, cheaper, and more convenient than building and recording a physical space. This is useful to architects, who need to know that their designs will meet exacting specifications before building starts. It is also useful to musicians and sound designers, who can use the simulation results to create pleasing, precise, and immersive audio effects.

Currently, users are limited in their choice of simulation software, as existing solutions focus either on speed or on overall accuracy. Fast simulation techniques are often inaccurate at low frequencies, while more accurate techniques become prohibitively slow at higher frequencies. There is a clear need for a program which is fast, accurate at all frequencies, and easy to use without specialist training.

This project identifies a hybrid acoustic simulation technique which combines the efficiency of geometric simulation with the accuracy of wave-based modelling. This fusion of simulation techniques, not available in any existing piece of simulation software, gives the user the flexibility to balance accuracy against efficiency as they require. This hybrid method is implemented in the Wayverb program. The program is made free and open-source, with a simple graphical interface, differentiating it from hybrid simulators found in the literature which are all private and closed-source. In this way, Wayverb is uniquely accurate, efficient, and accessible.

Rather than presenting an entirely new simulation method, the Wayverb project surveys algorithms from the literature, employing those which are deemed most appropriate. It focuses on issues of practical implementation. In particular, for increased performance, Wayverb uses graphics hardware to accelerate calculations via parallelisation. Test data is presented to demonstrate accuracy.

The Wayverb program demonstrates that the hybrid method is efficient enough to be viable in consumer software, but testing reveals that simulation results are not directly ready for production usage. The differing properties of the wave-based and geometric methods can result in different onset and decay times in the upper and lower regions of the output spectrum, which reduces the perceived quality of the output despite increased low-frequency accuracy. Avenues for further research are suggested in order to improve the quality and usability of the software.

Contents

Acknowledgements

Introduction

The aim of room acoustics simulation is to simulate the reverberant properties of a space without having to physically build anything. This is useful for a variety of applications: architects need to be able to evaluate the acoustics of a building before construction begins; sound editors for film sometimes need to mix in recordings which were not made on location; electronic musicians like to conjure imaginary or impossible spaces in their music, and virtual-reality experiences must use audio cues to convince the user that they have been transported to a new environment.

Unfortunately, software allowing the synthesis of accurate impulse responses is not currently widely available. Often, software produced for research purposes is not made public. Such software that *is* available generally suffers from one or more of an array of issues:

- Most software relies only on fast geometric methods, which are inaccurate, especially at low frequencies. Conversely, programs opting to use more accurate wave-modelling methods require long time periods, in the order of days, or significant computing power to run.
- Licensing is also a problem. Most room acoustics packages are the product of years of combined research by multiple contributors, which is only made viable by releasing the software commercially. However, this inhibits further research, as the code is not freely available. This model also limits users to those able to pay, restricting widespread adoption.
- When software is made available freely, often the user experience suffers. Code requires manual compilation, or perhaps can only be run from a textual interface, or else the project is outdated and unmaintained.

The Wayverb project provides a solution to these problems, by making available a graphical tool for impulse response synthesis. It combines geometric and wave-modelling simulation techniques, providing an adjustable balance between speed and accuracy. It is also free to download, can be run immediately on commodity hardware, and the source code can be used and extended under the terms of the GNU General Public License (GPL).

The software has three main simulation engines, each with complementary strengths and weaknesses. Each engine had to be thoroughly researched, and implementation decisions evaluated, in order to balance the dual aims of accuracy and efficiency. This process is described in detail for each engine type, as is the process of automatically combining the engine outputs. Wayverb also implements two extensions to the basic room acoustics model, namely frequency-dependent reflections at boundaries, and microphone modelling, both of which are reviewed in detail.

1 Context

1.1 Overview

Room acoustics algorithms fall into two main categories: *geometric*, and *wave-based* [1]. Wave-based methods aim to solve the wave equation numerically, simulating the actual behaviour of sound waves within an enclosed space. Geometric methods instead make some simplifying assumptions about the behaviour of sound, which result in faster but less accurate simulations. These assumptions generally ignore all wave properties of sound, choosing to model sound as independent *rays*, *particles*, or *phonons*.

The modelling of waves as particles has found great success in the field of computer graphics, where *ray-tracing* is used to simulate the reflections of light in a scene. The technique works well for simulating light because of the relatively high frequencies of the modelled waves. The wavelengths of these waves - the wavelengths of the visible spectrum - will generally be many times smaller than any surface in the scene being rendered, so wave phenomena have little or no visible effect.

The assumption that rays and waves are interchangeable falls down somewhat when modelling sound. The wavelengths of sound in air range from 17m to 0.017m for the audible frequency range (20Hz to 20kHz), so while the simulation may be accurate at high frequencies, at low frequencies the wavelength is of the same order as the wall surfaces in the scene. Failure to take wave effects such as interference and diffraction into account at these frequencies therefore results in noticeable approximation error [2].

In many cases, some inaccuracy is an acceptable (or even necessary) trade-off. Wave-modelling is so computationally expensive that using it to simulate a large scene over a broad spectrum might take weeks on consumer hardware. This leaves geometric methods as the only viable alternative. Though wave-modelling been studied for some time [3], and even applied to small simulations of strings and membranes in consumer devices such as keyboards, it is only recently, as computers have become more powerful, that these techniques have been seriously considered for room acoustics simulation.

Given that wave-based methods are accurate, but become more expensive at higher frequencies, and that geometric methods are inexpensive, but become less accurate at lower frequencies, it is natural to combine the two models in a way that takes advantage of the desirable characteristics of each [4]. That is, by using wave-modelling for low-frequency content, and geometric methods for high-frequency content, simulations may be produced which are accurate across the entire spectrum, without incurring massive computational costs.

1.2 Characteristics of Room Acoustics Simulation Methods

A short review of acoustic simulation methods will be given here. For a more detailed survey of methods used in room acoustics, see [2].

Figure 1.1 shows the relationships between the most common simulation methods. The advantages and disadvantages of each method will be discussed throughout the remainder of this section.
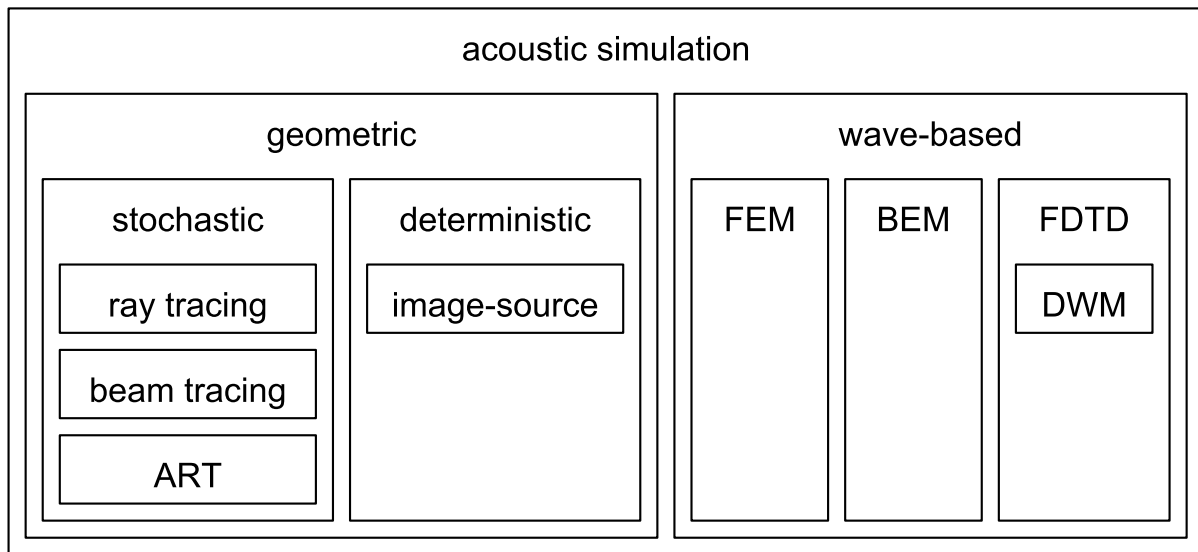
Figure 1.1: An overview of different acoustic simulation methods, grouped by category.

1.2.1 Geometric Methods

Geometric methods can be grouped into two categories: *stochastic* and *deterministic*.

Stochastic methods are generally based on statistical approximation via some kind of Monte Carlo method. Such methods are approximate by nature. They aim to randomly and repeatedly sample the problem space, recording samples which fulfil some correctness criteria, and discarding the rest. By combining the results from multiple samples, the probability of an incorrect result is reduced, and the accuracy is increased. The balance of quality and speed can be adjusted in a straightforward manner, simply by adjusting the number of samples taken.

In room acoustics, stochastic algorithms may be based directly on reflection paths, using *ray tracing* or *beam tracing*, in which rays or beams are considered to transport acoustic energy around the scene. Alternatively, they may use a surface-based technique, such as *acoustic radiance transfer* (ART), in which surfaces are used as intermediate stores of acoustic energy.

Surface-based methods, especially, are suited to real-time simulations (i.e. interactive, where the listener position can change), as the calculation occurs in several passes, only the last of which involves the receiver object. This means that early passes can be computed and cached, and only the final pass must be recomputed if the receiver position changes.

The main deterministic method is the *image source* method, which is designed to calculate the exact reflection paths between a source and a receiver. For shoebox-shaped rooms, and perfectly rigid surfaces, it is able to produce an exact solution to the wave equation. However, by its nature, it can only model specular (perfect) reflections, ignoring diffuse and diffracted components. For this reason, it is inexact for arbitrary enclosures, and unsuitable for calculating reverb tails, which are predominantly diffuse. The technique also becomes prohibitively expensive beyond low orders of reflection. The naive implementation reflects the sound source against all surfaces in the scene, resulting in a set of *image*

sources. Then, each of these image sources is itself reflected against all surfaces. For high orders of reflection, the required number of calculations quickly becomes impractical. For these reasons, the image-source method is only suitable for early reflections, and is generally combined with a stochastic method to find the late part of an impulse response (IR).

For a detailed reference on geometric acoustic methods, see [2].

1.2.2 Wave-based Methods

The main advantage of wave-based methods is that they inherently account for wave effects such as diffraction and interference [5], while geometric methods do not. This means that these wave-based methods are capable of accurately simulating the low-frequency component of a room IR, where constructive and destructive wave interference form *room modes*. Room modes have the effect of amplifying and attenuating specific frequencies in the room IR, and produce much of the subjective sonic "colour" or "character" of a room. Reproducing these room modes is therefore vital for evaluating the acoustics of rooms such as concert halls and recording studios, or when producing musically pleasing reverbs.

Wave-based methods may be derived from the *Finite Element Method* (FEM), *Boundary Element Method* (BEM) or *Finite-Difference Time-Domain* (FDTD) method. The FEM and BEM are known together as *Element Methods*.

The FEM is an iterative numerical method for finding natural resonances of a bounded enclosure. It models the air pressure inside the enclosure using a grid of interconnected nodes, each of which represents a mechanical system with a single degree of freedom. The interconnectedness of the nodes leads to a set of simultaneous equations, which can be solved for displacement at each node, and then the solved equations can be used to calculate pressure values at certain elements. The BEM is similar, but models nodes on the surface of the enclosure, instead of within it. This in turn allows it to model unbounded spaces, whereas the FEM is limited to bounded spaces [6, pp. 52–55].

The FDTD method works by dividing the space to be modelled into a regular grid, and computing changes in some quantity (such as pressure or particle velocity) at each grid point over time. The formula used to update each grid point, along with the topology of the grid, may be varied depending on the accuracy, efficiency, and complexity required by the application. FDTD methods are generally applied to problems in electromagnetics, but a subclass of the FDTD method known as the *Digital Waveguide Mesh* (DWM) is often used for solving acoustics problems.

The FDTD process shares some characteristics with the element methods. They all become rapidly more computationally expensive as the maximum output frequency increases [7]. They also share the problem of discretisation or quantisation, in which details of the modelled room can only be resolved to the same accuracy as the spatial sampling period. If a large inter-element spacing is used, details of the room shape will be lost, whereas a small spacing will greatly increase the computational load.

The major advantage of FDTD over element methods is that it is run directly in the time domain, rather than producing frequency-domain results, which in turn affords a much simpler implementation.

FDTD simulations can also be implemented with relative efficiency by taking advantage of their "embarrassingly parallel" nature. Each individual node in the simulation (of which there may be thousands or millions) can be updated without synchronisation. As a result, the nodes may be updated entirely in parallel, leading to massive reductions in simulation time.

The main disadvantage of the FDTD method is that it is susceptible to *numerical dispersion*, in which wave components travel at different speeds depending on their frequency and direction, especially at high frequencies. Several techniques exist to reduce this error, such as oversampling the mesh [8], using different mesh topologies [9], [10], and post-processing the simulation output [11]. Oversampling further increases the computational load of the simulation, while using different topologies and post-processing both introduce additional complexity.

Despite its drawbacks, the FDTD method is generally preferred for room acoustics simulation [7], due to its straightforward implementation, inherent parallelism, intuitive behaviour, and its ability to directly produce time-domain IRs.

1.3 Existing Software

A handful of programs exist for acoustic simulation. Table 1.1 shows a selection which, whilst not exhaustive, is representative.

Table 1.1: Some of the most prominent tools for acoustic simulation.

Name	Type	Availability
Odeon [12]	Geometric	Commercial
CATT-Acoustic [13]	Geometric	Commercial
Olive Tree Lab [14]	Geometric	Commercial
EASE [15]	Geometric	Commercial
Auratorium [16]	Geometric	Commercial
RAVEN [17]	Geometric	None
RoomWeaver [18]	Waveguide	None
EAR [19]	Geometric	Free
PachydermAcoustic [20]	Geometric	Free
Parallel FDTD [21]	Waveguide	Free
i-Simpa [22]	Geometric, extensible	Free

All commercial acoustics programs found use geometric techniques, probably because they are fast to run, and can often be implemented to run interactively, in real-time. However, low-frequency performance is a known issue with these programs. For example, the FAQ page for the Odeon software [23] notes that:

> For Odeon simulations as with real measurements, the source and receiver should be at least 1/4th wave length from the walls. But at the very lowest resonance of the room the level can change a lot from position to position without Odeon being able to predict it. For investigation of low frequency behavior (resonances), indeed Odeon is not the tool.

Clearly there is a need for wave-modelling acoustics software, which can accurately predict low frequency behaviour. However, such software seems to be somewhat rarer than geometric acoustics software. Of the two wave-modelling programs listed, only one is generally available, which must additionally be run from Python or Matlab scripts. This is a good approach for research software, but would probably not be straightforward for users with limited programming experience.

At time of writing (December 2016) it appears that no generally-available (commercially or otherwise) piece of software has taken the approach of combining wave-modelling and geometric methods, although this technique is well-known in the literature [1], [4], [24]–[27].

1.4 Acoustic Simulation and the Creative Arts

Musicians and sound designers can choose from an abundance of convolution reverb plugins, such as Waves' *IR1 Convolution Reverb* [28], Audio Ease's *Altiverb* [29], and Liquid Sonics' *Reverberate 2* [30]. These tools are extremely flexible: by convolving an IR with some other signal, that signal can be made to sound as though it was recorded in the same location as the IR. A music producer might use this technique to create a recording of an "orchestra" in which each instrument is recorded separately, composited, and convolved with the IR of a concert hall. Similarly, a foley artist could use convolution reverb to make studio-recorded effects sound more believable in the context of the environment on-screen.

The main drawback of convolution reverbs is their dependency upon high-quality IR recordings. Although most tools come with a library of IRs, this library will not be comprehensive. In some circumstances (for example, when attempting to seamlessly combine foley effects with location recordings) a suitable pre-recorded IR will not be directly available. In other circumstances, it may not even be possible to record a suitable IR using traditional methods, because the desired reverb is designed to evoke an environment that does not (or cannot) exist.

In these situations, the user has a few options. Firstly, a custom IR could be recorded. This will require specialist equipment, and access to the particular location. Secondly, the desired reverb could be approximated using an algorithmic (i.e. not convolution-based) reverb tool. Thirdly, the IR could be predicted using an acoustic simulator. The third approach seems like a sensible middle ground between the first two options. It should be less time-consuming than recording a custom IR, and will not require access to the modelled location (although a 3D virtual model would be necessary). Additionally, the simulated IR should match real-world behaviour more closely than an algorithmic reverb.

Despite the obvious application of virtual acoustics to music and sound production, all of the software in table 1.1 appears to be targeted at technical users with specialist knowledge in acoustics. For example, the i-Simpa homepage [22] says:

> It is a perfect tool for experts (i.e. acousticians), for teachers and students, as well as for researchers, in their projects (room acoustics, urban acoustics, industrial spaces, acoustic courses…)

The Olive Tree Lab "philosophy" page [31] describes a similar focus on technical users:

> …we hope to assist acousticians and engineers in predicting sound and noise propagation more accurately, especially in the field of Noise Control.

In the case of the "EASE" software, its name is an acronym standing for "Enhanced Acoustic Simulator for Engineers".

This targeting of technical users has an effect on the program design. The programs prioritise physical accuracy, and the ability to export visualisations and statistics about the modelled acoustics. These tools may also have steep learning curve, assuming that users are already familiar with acoustics theory. A simulation tool for creative users should have different goals:

- **Sound quality**: Generated IRs should be suitable for use without any additional cleanup or editing.
- **Intuitive controls**: The interface should make it obvious how each parameter will affect the output.
- **Simulation speed**: Part of the creative process is experimentation, and users need to hear the effects of their experiments quickly in order to iterate towards the desired sound.

Creative users only require a subset of the functionality provided by other simulators. That is, they only require the final IR result. Other features, such as the creation and export of statistics and visualisations, are not required. Therefore, such a tool could be reasonably streamlined, presenting a simple "import, configure, render" workflow and omitting additional analysis features.

1.5 Project Aims

Based on the evidence presented, it seems that there is a clear need for an acoustic simulation tool which uses wave-modelling to predict low-frequency behaviour, and which is targeted at creative users. The goal of the Wayverb project is build such a tool. The development of this program should prioritise the following goals:

- **Plausibility**: Provide a way of generating physically plausible impulse responses of arbitrary enclosed spaces.
- **Efficiency**: Ensure that the simulation is fast. Simulations should take less than ten minutes in general, and certainly never more than an hour.
- **Accessibility**: The program's controls should be intuitive, and it should be possible for someone with no programming or acoustics experience to generate IRs.

The ideal simulation program would be capable of replicating, with perfect accuracy, the real-world behaviour of any acoustic scenario. However, for the purposes of sound-design, this level of accuracy is not necessary. When creating a reverb, a sound designer's focus is generally on experimenting and developing the desired atmosphere, rather than on perfectly reconstructing a physical location. Therefore, simulation results should be believable first:

- The room size should directly affect the reverb duration, with smaller rooms exhibiting lower reverb times than larger spaces.
- The absorptions of the surfaces in the room should also affect reverb times. Absorptive surfaces should produce lower reverb times than more reflective surfaces. This effect should also be frequency-dependent, so that a room which is largely reflective at low frequencies and absorptive at high frequencies will exhibit appropriate relative reverb times in each frequency band.
- Rooms with parallel surfaces should show modal resonances at the correct frequencies.

- Adjusting the separating distance between a source and receiver should lead to corresponding changes in the balance between early and late reflections. For larger separations, the direct contribution should become less pronounced.
- If the source is not directly visible from the receiver, there should be no direct contribution.
- Reflective tunnel-like rooms should produce distinct echoes in the reverb tail.
- Modelled microphones should attenuate directional contributions appropriately, according to the polar pattern of the capsule.

Another aspect of plausibility is overall quality: if a generated IR contains obvious artefacts, it is by definition physically implausible, and of limited use to a sound designer.

Plausibility and efficiency are competing goals, which must be balanced. Extreme performance, allowing real-time usage, has already been implemented in several of the commercial programs listed above, and generally relies on simplified acoustic models, which in turn reduce accuracy. This runs counter to the aims of the project. Similarly, high-quality simulations generally require long compute times and specialised hardware, both of which are inaccessible to the target user. Therefore, the focus of the project cannot be solely on plausibility. Software which balances these two aims does not exist, at time of writing, and there is a clear need for a solution which runs on commodity hardware, and is both reasonably fast and produces believable results. Ideally, this software would allow the user control over the trade-off between accuracy and efficiency, enabling fast, lower-quality simulations to be used when auditioning, and a slower, higher-quality render to be produced once the user is happy with all the simulation settings.

Regarding accessibility, the program must be simple to install and run, and users should not require specialist training in acoustics or programming in order to become productive. Controls must be intuitive, and it should be obvious how each parameter will affect the final IR. The project as a whole should be accessible too: the code and supporting materials must be made freely available to researchers, to encourage further research and modification. Note that accessibility is a lesser goal than plausibility and efficiency: if the program is fast and produces high quality, usable results, users will be prepared to invest time to learn the program. Meanwhile, if the program is easy to use but is too slow or produces poor results, users will have no reason to learn the software in the first place.

1.5.1 Proposed Solution

It appears that an approach combining geometric and wave-based methods will be most flexible in achieving both plausibility and efficiency: wave-based methods are accurate but slow; and geometric methods are faster but more approximate. Efficiency can be balanced against output quality by adjusting the proportion of the output generated with each method. The Wayverb project puts forward an acoustic simulator based on this hybrid method.

To achieve the goal of accessibility, the Wayverb program runs on consumer hardware, and is accessed through a graphical interface which allows simulations to be configured, stored, and run. Code for the project is public and permissively licensed.

1.6 Original Contributions

Most importantly, at time of writing, Wayverb is the only public graphical acoustics tool incorporating geometric and wave-based methods. Although hybrid acoustics methods are well documented [4], [24], [25], they have only been used in specific research settings, for producing experimental results. It may be assumed that these tools have been built to model specific test-cases, rather than general simulation tasks, but this is uncertain as no tools incorporating these techniques have been made public. However, Wayverb is able to model arbitrary enclosures.

The project acts as a survey of room acoustics techniques, and related issues regarding practical implementation. Rather than designing completely new simulation methods, existing techniques were investigated, compared, and evaluated in terms of their plausibility and performance. Then, optimum techniques were chosen and further developed for use in the final program. An especially important consideration is the matching of parameters between models. For example, all models should produce the same sound energy at a given distance, and should exhibit the same reverb time for a given scene. Therefore, the acoustics techniques were chosen so that they produce consistent results.

Sometimes the models required development beyond the methods presented in the literature in order to become useful. An example of this is the waveguide set-up process. Most experimental set-ups in the literature only model cuboid-shaped enclosures, and no guidance is given for setting up simulations in arbitrarily-shaped enclosures. Of course, it must be possible to model real, complex room shapes, and so an original set-up procedure had to be developed. The same goes for memory layout and implementation details: in the literature, techniques for efficient implementation are rarely discussed. As a result, new techniques had to be invented, rather than reimplementing known methods. Where extensions to existing techniques have been developed for use in Wayverb, this is mentioned in the text.

Much of the literature on acoustic simulation focuses predominantly on accuracy. Performance appraisals are rarely given, presumably because they are somewhat subjective, and "reasonable" efficiency will vary between applications. Ideally, the simulation methods in Wayverb should be selected and implemented to allow tunable performance, so that results with acceptable accuracy can be generated within a few minutes, but it is possible to run longer simulations if higher-accuracy results are needed. This is similar to approaches taken in computer graphics, where "overview" renders may take seconds to generate, but physically-modelled simulations for film often take hours to render, even on purpose-built compute clusters.

The notable components of the Wayverb project are as follows, each of which has a dedicated chapter with detailed explanation:

- Image-source model, accelerated with parallel ray-casting, for early reflections. Uses a novel method for speeding up audibility tests by re-using reflection paths from the ray tracer.
- Parallel stochastic ray-tracer, for late reflections.
- Parallel digital waveguide mesh, for low frequency modelling. Uses a novel set-up procedure to create meshes with correctly-placed boundary nodes in arbitrary scenes.
- Calibration, automatically matching the output levels of the different models.
- A microphone model, capable of simulating capsules with direction-dependent frequency responses, within all three simulation-types.
- A boundary model with matched performance in all three simulation-types.

1.6.1 Chosen Simulation Techniques

The image-source and stochastic ray-tracing methods were chosen for modelling high-frequency content. These models are complementary: the image model can find early reflections with great accuracy but is slow at finding later reflections; while the ray-tracer is much faster but more approximate, making it better suited to finding naturally-diffuse late reflections. Specifically, a simple ray tracing method was chosen over a phonon- or surface-based method for the late-reflection simulation, for several reasons. Firstly, ray tracing is broadly discussed in the literature [32]–[36], so would not require a great deal of experimentation to implement. Secondly, ray tracing has the property of being an *embarrassingly parallel* algorithm, because each individual ray can be simulated entirely independently, without requiring communication or synchronisation. By running the algorithm on graphics hardware, which is designed to run great numbers of calculations in parallel, all rays could be simulated in one go, yielding much greater performance than processing each ray sequentially. Finally, though surface-based methods are capable of real-time operation, they do not pose any performance benefit for non-real-time or "one-off" simulations. Their performance comes from re-using pre-computed information when the receiver position changes, but in a one-off simulation the receiver position is fixed. Ray tracing is also less complex and better documented than surface-based methods, making it the superior choice for this application. A logistical reason for choosing the image-source and ray tracing solution for high-frequency modelling was that the author had previously implemented such a system for an undergraduate project. It was hoped that much of the code from that project could be re-used, but it transpired that the project suffered from accuracy and implementation issues, making it unsuitable for direct integration within Wayverb. Therefore, the majority of this code was completely re-written. The author was, however, able to re-use much of the knowledge and experience gained from the previous project, which would not have been possible if a completely new stochastic method had been introduced.

For low-frequency simulation, a FDTD-based DWM model was chosen. There is a great deal of writing on this method [6], [8], [37]–[39], it is relatively simple to implement, and shares with ray tracing the characteristic of being embarrassingly parallel. As wave-modelling is especially costly, a parallel implementation is necessary in order to achieve simulation times in the order of minutes rather than hours or days.

An in-depth description of the algorithms implemented is given in the Image-Source, Ray Tracer, and Waveguide sections. Figure 1.2 shows how the outputs from the different methods work together to produce a broadband IR. It shows that the lower portion of the spectrum is generated entirely with the waveguide, while the upper portion is simulated using the image-source method for early reflections, and the ray tracing method for the reverb tail.

Deciding on the simulation techniques led to three questions:

- To produce a final output, the three simulations must be automatically mixed in some way. How can this be done?
- Binaural simulation requires some method for direction- and frequency-dependent attenuation at the receiver. How can receivers with polar patterns other than omnidirectional be modelled consistently in all three simulation methods?
- The reverb time and character depends heavily on the nature of the reflective surfaces in the scene. How can frequency-dependent reflective boundaries be modelled consistently in all methods?
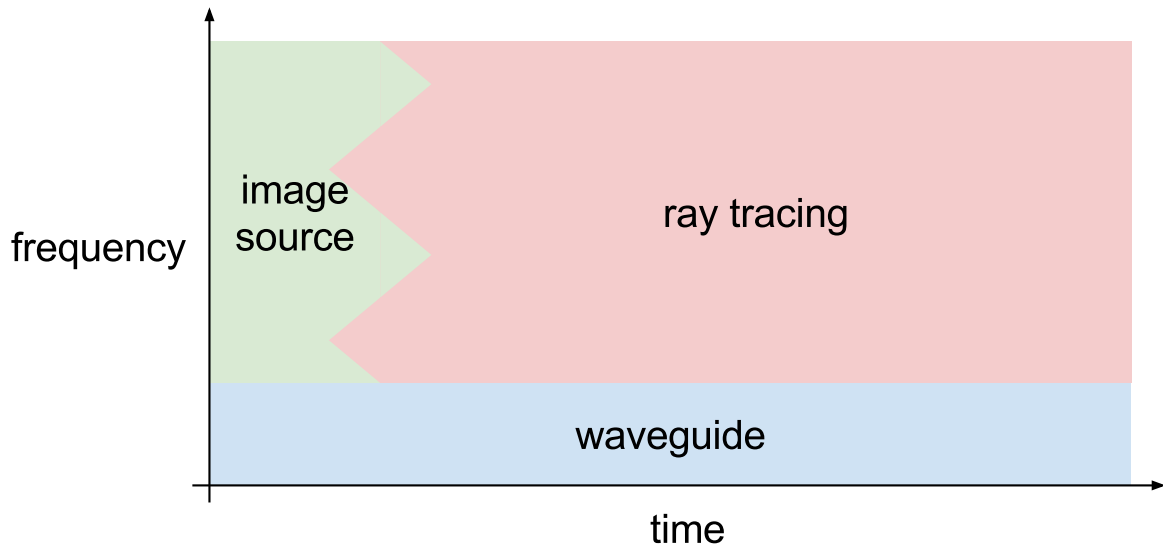
Figure 1.2: The structure of a simulated IR.

These questions will be discussed in the Hybrid, Microphone Modelling, and Boundary Modelling sections respectively.

1.6.2 Chosen Technology

The programming language chosen was C++. For acceptable performance in numerical computing, a low-level language is required, and for rapid prototyping, high-level abstractions are necessary. C++ delivers on both of these requirements, for the most part, although its fundamentally unsafe memory model does introduce a class of bugs which do not really exist in languages with garbage collection, borrow checking, or some other safety mechanism.

OpenCL was chosen for implementing the most parallel parts of the simulation. The OpenCL framework allows a single source file to be written, in a C-like language, which can target either standard *central processing units* (CPUs), or highly parallel *graphics processing units* (GPUs). The main alternative to OpenCL is CUDA, which additionally can compile C++ code, but which can only target Nvidia hardware. OpenCL was chosen as it would allow the final program to be run on a wider variety of systems, with fewer limitations on their graphics hardware.

The only deployment target was Mac OS. This was mainly to ease development, as maintaining software across multiple platforms is often time-consuming. Mac OS also tends to have support for newer C++ language features than Windows, which allows code to be more concise, flexible, and performant.[1] Targeting a single platform avoids the need to use only the lowest common denominator of language features. As far as possible, the languages and libraries have been selected to be portable if the decision to support other platforms is made in the future.

The following additional libraries were used to speed development. They are all open-source and freely available.

---

[1]Visual Studio 2015 for Windows still does not support all of the C++11 language features [40], while the Clang compiler used by Mac OS has supported newer C++14 features since version 3.4 [41], released in May 2014 [42].

GLM Provides vector and matrix primitives and operations, primarily designed for use in 3D graphics software, but useful for any program that will deal with 3D space.

Assimp Used for loading and saving 3D model files in a wide array of formats, with a consistent interface for querying loaded files. Using a 3D mesh importer means that users can load and simulate models created in practically any mesh editor, providing the mesh is manifold and represents a single watertight enclosure.

FFTW3 Provides Fast Fourier Transform routines. Used mainly for filtering and convolution.

Libsndfile Used for loading and saving audio files, specifically for saving simulation results.

Libsamplerate Provides high-quality sample-rate-conversion routines. Waveguide simulations are often run at a relatively low sample-rate, which must then be adjusted.

Gtest A unit-testing framework, used to validate small individual parts of the program, and ensure that changes to one module do not cause breakage elsewhere.

Cereal Serializes data to and from files. Used for saving program configuration options.

ITPP A scientific computing library. Used for its implementation of the Yule-Walker method for estimating filter coefficients for a given magnitude response.

JUCE Provides a framework for building graphical applications in C++. Used for the final application.

The project uses CMake to configure its build, and to automatically download project dependencies. Python and Octave were used for running and automating tests and generating graphs.

This documentation is written in Markdown, and compiled to HTML and to PDF using Pandoc. The project website is generated with Jekyll.

1.7 Summary

An account of techniques commonly used for room acoustics simulation has been provided. The strengths and weaknesses of these techniques have been discussed, leading to the observation that geometric and wave-based models have complementary characteristics. The weaknesses of the individual models could be minimised by creating a combined "hybrid" model. This hybrid approach has not previously been used in publicly-available software, and such acoustic simulation software that is available is consistently targeted at technical users. This evidence suggests the need for a program focused on the requirements of creative users, which uses a hybrid modelling approach, and which is made publicly available. Specific goals for such a program have been suggested and explained, and the original contributions of the program have been examined. Finally, a plan to build the program has been put forward.

2 Theory

This chapter explains some aspects of room acoustics theory, which will help to clarify the implementation decisions governing the simulation techniques discussed in later chapters. The account given here deals only with topics which are directly applied in Wayverb. For more detailed derivations of equations, along with information about more advanced acoustic phenomena such as diffraction, refraction, and the Doppler effect, the books by Vorländer [34] and Kuttruff [33] are recommended.

2.1 Waves and Media

Sound waves can be described completely by specifying the instantaneous velocity, $\vec{v}$, of each particle in the propagation medium. Not all particles will have the same velocity, which causes fluctuations in density, pressure, and temperature, which are dependent upon time, and position. *Sound pressure*, p, is the difference between the "at rest" pressure p_0, and the pressure measured in the medium at a particular position and time p_{tot} [34, p. 9]:

$$p = p_{\text{tot}} - p_0 \tag{2.1}$$

Sound pressure is measured in pascals (1 Pa = 1 N/m²) [34, p. 16]. The goal of room acoustic simulation is to predict the change in sound pressure over time at particular points in a sound field. A similar equation to eq. 2.1 can be written for change in density due to sound ρ, where ρ_0 is the static density of the medium, and ρ_{tot} is the instantaneous density:

$$\rho = \rho_{\text{tot}} - \rho_0 \tag{2.2}$$

For the purposes of room acoustics, it may be assumed that the propagation medium is air. Sound propagation through liquids and solid structures will be ignored. The speed of sound in air, c, is approximately

$$c = (331.4 + 0.6\theta)\text{m/s} \tag{2.3}$$

where θ is the temperature in degrees Celsius [33, p. 7]. In most real-world acoustics problems, variations in temperature will be very small, and can be ignored. That is, the propagation medium is assumed to be homogeneous. A propagation medium can be specified by its *characteristic impedance* or *wave impedance Z_0* [34, p. 14]:

$$Z_0 = \rho_0 c \tag{2.4}$$

This quantity denotes the medium's resistance to pressure excitation, or alternatively the pressure required to induce movement in the medium's particles. For air, the characteristic impedance is generally around 400 kg/m²s (although this depends on the speed of sound, and therefore the air temperature) [34, p. 15].

The difference in sound pressure between the quietest audible sound and the threshold of pain is around 13 orders of magnitude. To make working with the values involved more manageable, sound pressure is usually given in terms of the *sound pressure level* (SPL), which is measured on a logarithmic scale in decibels (dB) [33, p. 23]:

$$\text{SPL} = 20 \log_{10} \frac{p_{\text{rms}}}{p_0} \text{dB} \tag{2.5}$$

Here, p_{rms} is the *root mean square* sound pressure, and p_0 is a reference pressure of 2×10^{-5} Pa.

Wave propagation in a medium causes energy to be transported through that medium. This energy flow can be measured in terms of the energy transported per second through an area of 1m² (W/m²), and is called *sound intensity*, I [34, p. 19]:

$$\vec{I} = \overline{p\vec{v}} \tag{2.6}$$

Note that the overline notation signifies time-averaging. The sound intensity can also be given in terms of the *intensity level*, L_I [34, p. 20]:

$$L_I = 10 \log_{10} \frac{|\vec{I}|}{I_0} \tag{2.7}$$

where $I_0 = 10^{-12} W/m^2$ is a reference intensity chosen to match the levels of sound pressure and intensity in a plane wave.

For a harmonic wave, the temporal and spatial periods of the wave are related by the speed of sound, c:

$$c = \lambda f = \frac{\lambda}{T} \tag{2.8}$$

where f is the frequency of the wave in Hz, T is the temporal period, and λ is the spatial period or *wavelength*.

In the simulation presented in this project, sound waves are assumed to propagate equally in all directions from a point-like source, or *monopole*. The pressure wave, propagating with increasing radius, is known as a *spherical wave*.

The pressure p observed at distance r and time t from a monopole source with signal strength $Q(t)$ is given by [34, p. 24]:

$$p(r,t) = \frac{\rho_0}{4\pi r} \dot{Q} \left(t - \frac{r}{c} \right) \tag{2.9}$$

For harmonic waves, the total radiated sound power P of a monopole source is related to the sound intensity by [33, p. 15]:

$$P = 4\pi r^2 I \tag{2.10}$$

2.2 Boundary Characteristics

Room acoustics is not just concerned with the behaviour of pressure waves in air. Additionally, room acoustics problems normally bound the air volume with a set of surfaces (walls, floors, baffles etc.), from which an incident pressure wave may be reflected and/or absorbed. The reflected pressure waves generally lead to complex sound fields, which in turn contribute to the particular sonic "character" or *acoustic* of an enclosure.

Several assumptions are made to simplify the equations governing wave behaviour at a boundary. First, incident waves are assumed to be plane waves. This is never true in the Wayverb app, as simulations use point sources which produce spherical waves. However, the curvature of the wave front may be ignored so long as the source is not close to the reflecting surface [33, pp. 35–36] ("close" here will depend on the error constraints of the particular simulation). Secondly, boundary surfaces are assumed to be flat, and infinitely large [33, p. 35]. This is a valid approximation only if the size of each surface is large relative to the longest wavelength in the simulation.

2.2.1 Magnitude and Phase

The reflection factor R of a boundary is a complex value given by

$$R = |R| \exp(i\chi) \tag{2.11}$$

which describes a modification to the amplitude and phase of a wave reflected in a boundary ($|R|$ is the magnitude term, χ is phase).

This factor depends both on the frequency and direction of the incident wave. When $\chi = \pi$, R is negative, corresponding to a phase reversal. This is known as a "soft" wall, but is rarely seen in room acoustics. It is reasonable to assume that reflections are in-phase in the majority of architectural acoustics problems [33, p. 36].

The wall impedance Z is defined as the ratio of sound pressure to the normal component of particle velocity at the wall surface. It is related to the reflection factor by

$$R = \frac{Z \cos\theta - Z_0}{Z \cos\theta + Z_0} \tag{2.12}$$

where θ is the angle of incidence, and Z_0 is the characteristic impedance of the propagation medium, normally air. In the case that the wall impedance is independent of the wave angle-of-incidence, the surface is known as *locally reacting*. A locally reacting surface does not transmit waves tangentially along the wall surface. In Wayverb, all surfaces are modelled as locally reacting.

The absorption coefficient α of a wall describes the proportion of incident energy which is lost during reflection. It is defined as

$$\alpha = 1 - |R|^2 \tag{2.13}$$

The *specific acoustic impedance* ξ for a given surface is defined as the impedance of that surface Z divided by the acoustic impedance of the propagation medium (air) Z_0.

$$\xi = \frac{Z}{Z_0} \tag{2.14}$$

Inserting this equation into eq. 2.12 gives:

$$R = \frac{\xi \cos\theta - 1}{\xi \cos\theta + 1} \tag{2.15}$$

where θ is the angle of incidence [26].

For a general surface, ξ will be a function of the incident angle. However, in the case of a locally reacting surface, the impedance is independent of the angle of incidence. The ξ term in eq. 2.15 can then be replaced by ξ_0 which represents the normal-incidence specific impedance (which will be constant for all angles). Thus, the reflection factor of a locally reacting surface is

$$R = \frac{\xi_0 \cos\theta - 1}{\xi_0 \cos\theta + 1} \tag{2.16}$$

Surfaces in Wayverb are defined in terms of absorption coefficients. To express the reflectance in terms of absorption, an equation for ξ_0 in terms of absorption must be found, and substituted into eq. 2.16.

Assuming that the absorption coefficients denote normal-incidence absorption, then by rearranging eq. 2.13, the normal-incidence reflection coefficient R_0 is given by

$$R_0 = \sqrt{1 - \alpha} \tag{2.17}$$

R_0 can also be expressed by setting θ to 0 in eq. 2.16:

$$R_0 = \frac{\xi_0 - 1}{\xi_0 + 1} \tag{2.18}$$

To express ξ_0 in terms of α, eq. 2.18 is rearranged in terms of the normal-incidence reflection coefficient:

$$\xi_0 = \frac{1 + R_0}{1 - R_0} \tag{2.19}$$

Then, eq. 2.17 may be substituted into eq. 2.19 to give ξ_0 in terms of α. This new definition of ξ_0 can then be used in conjunction with eq. 2.16 to define the angle-dependent reflection factor of a locally reacting surface.

2.2.2 Scattering

The reflection factor, absorption coefficient, and wall impedance describe the behaviour of perfectly-reflected (specular) waves. If the reflecting surface has imperfections or details of the same order as the wavelength, as many surfaces in the real world do, then some components of the reflected wave will be *scattered* instead of specularly reflected.

Describing the nature of the scattered sound is more complicated than specular reflections. A common method is to use a *scattering coefficient*, which describes the proportion of outgoing energy which is scattered, and which may be dependent on frequency (see fig. 2.1). The total outgoing energy E_{total} is related to the incoming energy E_{incident} by

$$E_{\text{total}} = E_{\text{incident}}(1 - \alpha) \tag{2.20}$$

Then the scattering coefficient s defines the proportion of this outgoing energy which is reflected specularly E_{specular} or scattered $E_{\text{scattered}}$:

$$
\begin{aligned}
E_{\text{specular}} &= E_{\text{incident}}(1 - \alpha)(1 - s) \\
E_{\text{scattered}} &= E_{\text{incident}}(1 - \alpha)s
\end{aligned}
\tag{2.21}
$$
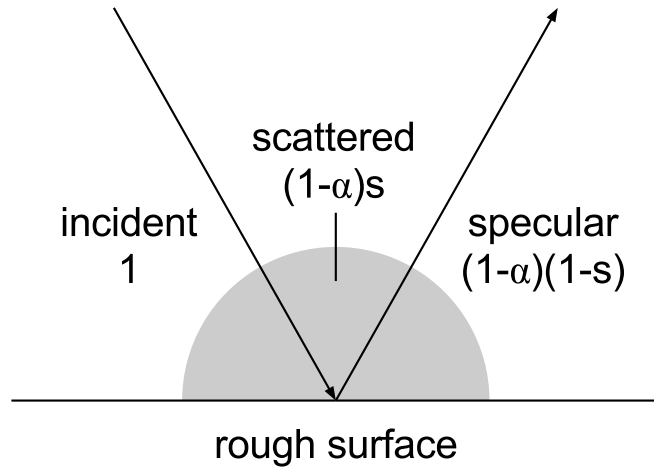
Figure 2.1: Reflected components from a rough surface.

Alone, the scattering coefficient fails to describe the directional distribution of scattered energy. In the case of an ideally-diffusing surface, the scattered energy is distributed according to Lambert's cosine law. That is, the intensity depends only on the cosine of the outgoing scattering angle, and is independent of the angle of incidence (see fig. 2.2). More complex scattering distributions, which also depend on the outgoing direction, are possible [43], [44], but there is no single definitive model to describe physically-accurate scattering.
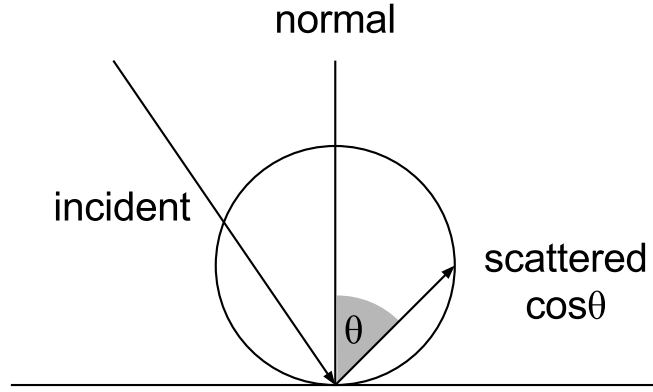
Figure 2.2: Lambert scattering is used to model ideally-diffusing surfaces, for which scattered intensity is independent of incident angle.

2.3 Impulse Response Metrics

2.3.1 Sabine's Equation

If an air-filled enclosure is excited impulsively, the sound field in the enclosure will not die away immediately, unless the boundaries are perfectly anechoic. Usually, the boundaries will reflect sound pressure waves back and forth inside the enclosure, and at each reflection some energy will be lost to the boundary. In this way the sound energy in the simulation will decrease gradually over time.

Assuming that the sound field is diffuse (that is, isotropic; having the same intensity in all directions [33, p. 52]), there will be a constant number of reflections per unit time. Under the additional assumption that the boundary absorption, α, has a homogeneous distribution, each reflection will cause that wave front to lose a proportion of its incoming energy equal to the absorption coefficient α.

The *energy density*, w of a sound field is defined as the amount of energy contained in one unit volume [33, p. 14]. In a diffuse sound field, the energy density at time t can be given by

$$w(t) \approx w_0(1 - \alpha)^{\overline{n}t} \tag{2.22}$$

where w_0 is the initial energy density when $t = 0$ (the time of the impulsive excitation), and $\overline{n}$ is the average number of reflections per unit time. Using the definition of sound pressure level, the energy density decay can be rewritten as a linear level decay [34, p. 61]:

$$L(t) = L_0 + 10\log_{10}\frac{w(t)}{w_0} = L_0 + 4.34\overline{n}t\ln(1 - \alpha) \tag{2.23}$$

A common measure of reverberation time is the *RT60*, the time taken for the level to decrease by 60dB. By rearranging eq. 2.23, the RT60 can be expressed [34, p. 61]:

$$\text{RT60} = -\frac{60}{4.34\overline{n}\ln(1 - \alpha)} \tag{2.24}$$

The average number of reflections per second, $\overline{n}$ can be approximated by

$$\overline{n} = \frac{cS}{4V} \tag{2.25}$$

where S is the total surface area of the enclosure boundaries, and V is the enclosure volume [33, p. 112].

By substituting eq. 2.25 into eq. 2.24 and simplifying (assuming α is very small), and assuming $c = 343m/s$, *Sabine's equation* is found [34, p. 61]:

$$\text{RT60} = 0.161\frac{V}{S\alpha} = 0.161\frac{V}{A} \tag{2.26}$$

The term $A = S\alpha$ is known as the *equivalent absorption area* of the room [33, p. 130]. Sabine's equation can be used to estimate the reverberation time of an enclosure, given the volume, surface area, and homogeneous absorption of that enclosure, and assuming that the sound field is diffuse.

The Sabine equation has some important limitations. Firstly, it fails at high absorptions. With the absorption coefficient set to 1, it estimates a finite reverb time, even though a completely absorptive enclosure cannot reverberate. Secondly, the assumption that the sound field in the enclosure is perfectly diffuse is untrue in practice. At low frequencies rooms behave modally, concentrating sound energy at specific points in the room. Under such circumstances, the sound field is clearly not diffuse, and so the Sabine equation is a poor predictor of reverb time at low frequencies.

2.3.2 Computing Reverb Times from Measurements

It is possible to compute the RT60 of a measured impulse response (IR). This is useful for comparing the reverb times of different IRs, or for evaluating an IR against a predicted reverb time.

Methods for evaluating the RT60 from measurements are given in the ISO 3382 standard [45]. Normally, the decay time is measured over a shorter range than 60dB, and then extrapolated. For example, the *T20* measurement is an estimate of the RT60 based on a 20dB level drop, and the *T30* is based on a 30dB drop.

The analysis itself proceeds as follows: the IR is reversed, each sample value is squared, then the squared sample values are integrated/accumulated. The backward-integrated IR can then be reversed once more to produce a decay curve. For a T20 measurement, the -5dB and -25dB points on the curve (relative to the maximum level) are found, and a regression line is fitted to the region between these points using the least-squares method. The gradient of this line, d, is a value measured in dB/s. The T20, in seconds, is given by $60/d$. The T30 is found in the same way, but the regression line is fitted to the region between -5dB and -35dB instead. To find the reverb time in a particular frequency band, the IR may be band-pass filtered before computing the decay curve.

A final measurement, known as the *early decay time* (EDT) is also based on fitting a regression line to the region between 0dB and -10dB, and then extrapolating to 60dB as usual. This measurement is useful as a descriptor of the ratio of direct to reverberant sound: an IR with a louder direct contribution and quieter reverberation will have a steep early gradient, and therefore a relatively short EDT. EDT is also useful as a measure of "perceived reverberance": the transients in music and speech normally have a much shorter period than the time taken for the level to drop 60dB, so the early part of the IR has greater perceptual significance than the late [34, p. 94].

ISO 3382 also defines the term *just noticeable difference* (JND), which is the smallest perceivable difference between two acoustic measurements [34, pp. 99–100]. That is, in order for simulation results to be indistinguishable from predictions or real-world measurements, the difference in each characteristic should be within the JND for that characteristic. For example, the JND for reverb time measurements (EDT, T20, T30) is 5%, and the JND for level is 1dB.

2.4 Summary

An overview of the properties governing the behaviour of a sound field in an enclosed space has been given. The relationships between different physical properties have been shown. Sound behaviour at boundaries has been explained, along with methods for deriving boundary characteristics from absorption and scattering coefficients. Finally, Sabine's equation for estimating reverb time has been derived, and methods for computing statistics from recorded impulse responses have been set out.

3 Image-source Model

3.1 Background

3.1.1 Basic Method

The image-source method aims to find the purely specular reflection paths between a source and a receiver. This process is simplified by assuming that sound propagates only along straight lines or rays. Sound energy travels at a fixed speed, corresponding to the speed of sound, along these rays. The energy in each ray decreases with $1/r^2$, where r is the total distance that the ray has travelled [34, p. 58].

Rays are perfectly reflected at boundaries. When a ray is reflected, it spawns a secondary source "behind" the boundary surface. This source is located on a line perpendicular to the wall, at the same distance from it as the original source, as if the original source has been "mirrored" in the surface (an example is shown in fig. 3.1). This new "image" source now represents a perfect reflection path, in that the distance along the straight line between the receiver and the image source has the same length as the path from the *real* source to the receiver, reflected in the boundary. If the source is reflected in a single boundary, this represents a first-order reflection. A ray which is reflected from several boundaries is represented by a "higher-order" image-source, which has been mirrored in each of those boundaries in turn [33, p. 104].
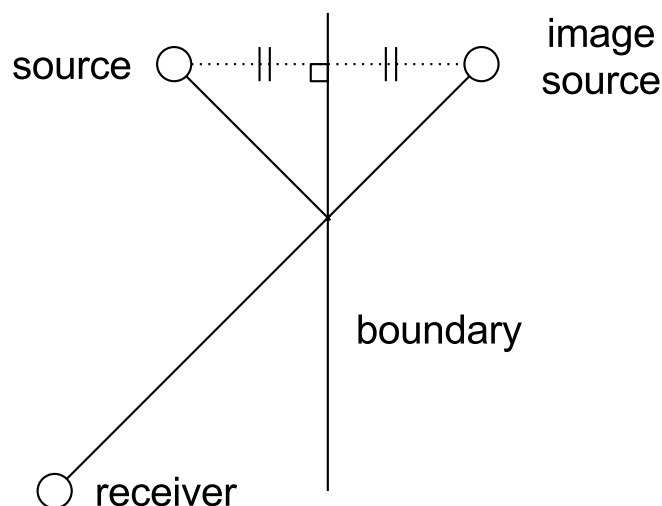
Figure 3.1: Image sources are found by reflecting the source position in a boundary.

All sources, original and image, emit the same impulsive source signal at the same time. The total impulse response (i.e. sound pressure against time) is found by summing the signals from each source, delayed and attenuated appropriately depending on the distance between that source and the receiver, which is equivalent to the length of the specular reflection path. The frequency response of the signal from each image source will additionally be modified depending on the characteristics of each boundary in which that source was reflected.

In the real world, not all energy is perfectly reflected at a boundary. Some energy will be randomly diffused in non-specular directions. The image-source model is not capable of modelling this phenomenon, though this is not particularly problematic. Consider that, once scattered, sound energy cannot become un-scattered. The conversion from incoming energy to scattered energy is unidirectional, so repeated reflections cause the ratio of scattered to specular energy to increase monotonically. Kuttruff shows that, though the earliest reflections may be largely specular, after a few reflections the large majority of sound energy becomes diffuse [33, p. 126]. This suggests that the image model should be used only for very early reflections, where most energy is not scattered, and a secondary model used to compute late, diffuse reflections. In Wayverb, the image model is used for early reflections, and stochastic ray-tracing is used for the diffuse tail. The combination of the two models is described in the Hybrid Model section.

3.1.2 Audibility Checking

The position of an image source is found by reflecting it in one or more surfaces. Next, it must be checked to ensure it represents a valid specular path to the receiver. This is known as an *audibility test* [34, p. 202].

Consider first a source S, a receiver R, and a single wall A. The source is reflected in A, creating an image-source S_A. A line is constructed from R to S_A. If this line intersects A, then S_A represents a valid image source. Otherwise, there is no possible specular reflection involving S, R and A.

Now consider two walls, A and B. The image source S_{AB} has been reflected in A then B. For the image-source to be valid:

- $R \rightarrow S_{AB}$ must intersect B at some point $B_{\text{intersection}}$,
- $B_{\text{intersection}} \rightarrow S_A$ must intersect A at $A_{\text{intersection}}$, *and*
- $A_{\text{intersection}} \rightarrow S$ must not intersect with any scene geometry.

The validation of a third-order image-source will require three intersection checks, a fourth-order image will require four checks, and so on. This method of tracing backwards from the receiver to each of the image sources is known as *backtracking*. This process is shown in fig. 3.2.

For a point p, its reflection p' in a plane with unit normal vector n and intersecting the point t can be found by

$$p' = p - 2n(n \cdot (p - t)) \tag{3.1}$$
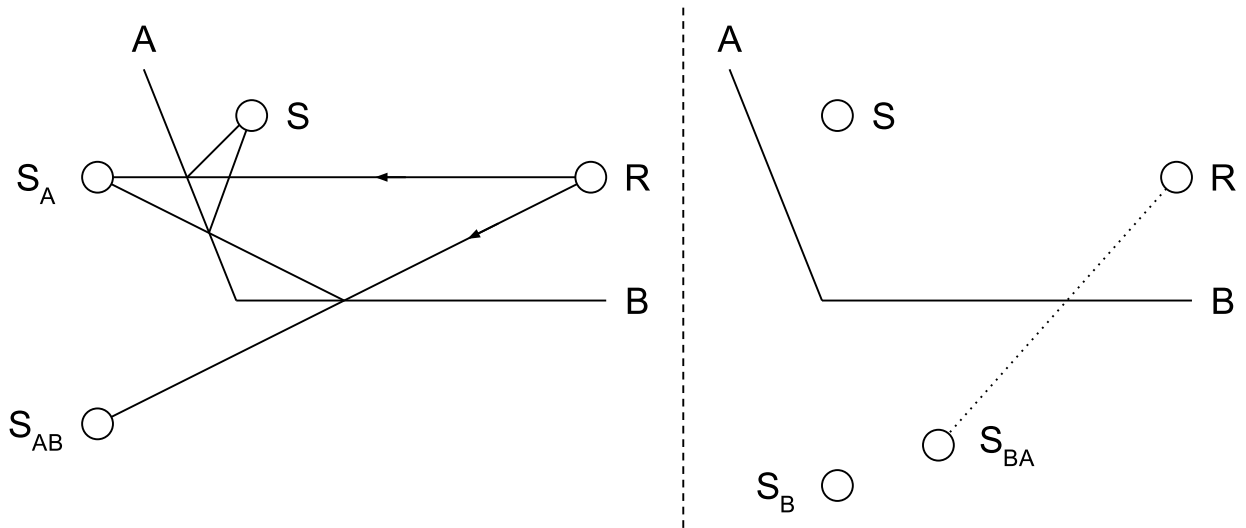
Figure 3.2: **Left:** The paths $S \rightarrow A \rightarrow R$ and $S \rightarrow A \rightarrow B \rightarrow R$ are both valid. **Right:** $S \rightarrow B \rightarrow A \rightarrow R$ is an invalid path because $R \rightarrow S_{BA}$ does not intersect A.

In eq. 3.1, the symbol · is the vector dot operator. Equation 3.1 can be used to find image source positions. However, in Wayverb surfaces are represented by triangles rather than by infinite planes. The triangle normal vector n is found by taking the cross-product of two of the triangle edge vectors, where an edge vector is the difference between two triangle vertices. The point on the plane t can be set to any of the triangle vertex positions.

To implement backtracking, an algorithm is required for computing the intersection point between a line and a triangle. A description of the ray-triangle intersection method used in Wayverb would be involved, and is omitted here in the interests of brevity. A detailed explanation can be found in [46].

3.1.3 Accelerating the Algorithm

The naive method to find all the image sources for a scene is very expensive: Consider that to find a single first-order image source, the original source must be mirrored in a surface, and then an intersection test must be conducted between that surface and the image-source-to-receiver ray. To find all first-order image sources, this process must be carried out for all surfaces in the scene. To find all second-order image sources, each of those first-order images must be tested against every surface. This continues for higher-order images, so that the number of checks for image sources of a given order is equal to the number of surfaces raised to the power of that order. The relationship between the image-source order and the computation time is therefore exponential, with average-case complexity of $O(N^o)$ where N denotes the number of boundaries, and o is the image-source order. As a result, it is practically impossible to validate all possible image source positions when the room geometry is complex or the image-source order is high. As an example, imagine that a particular (fictional) simulator might take a second to simulate a scene with 100 surfaces to an image-source depth of 2. If the image-source depth is increased to facilitate a longer reverb tail, third-order image sources will take 100 seconds to compute, and fourth-order sources will take 3 hours. Fifth-order sources will take 12 days. Clearly, it is not possible to achieve Wayverb's efficiency goal of "ten minutes or fewer" under all circumstances using this naive image source technique.

The majority of higher-order image sources found with the naive algorithm will be invalid. That is, they will fail the audibility test. For example, for tenth-order image-sources in a shoebox-shaped room, there are around 1.46e7 different image sources, only 1560 of which are valid [33, p. 323]. If the invalid image-sources can be discarded early, without requiring individual checking, then the amount of computation can be greatly reduced to a viable level. As explained above, image sources above order four or five are rarely required, but even these can be very time-consuming to find with the naive method. Optimisations are, therefore, a necessity for all but the simplest simulations.

To accelerate the image-source process, [34] suggests tracing a large number of rays in random directions from the source, and logging the unique paths of rays which eventually intersect with the receiver. The complexity of ray tracing grows linearly rather than exponentially with reflection depth, meaning it can find deeper reflections with far fewer operations than the image-source method. Each unique path found in this way is used to generate an image source sequence, which is then checked as normal. This technique has the advantage that the majority of surface sequences are *not* checked, so the image-source process is fast. However, if the preliminary ray-tracer is not run with enough rays, it is likely to miss some valid paths, especially in complex scenes. Additionally, if the receiver volume is too great, then some invalid paths may still be detected.

The technique used by Wayverb is similar to that presented in [34], but makes a small change to the acceleration process. Note that this does not affect the physical interpretation of the image-source model. It simply changes the way in which ray paths are initially selected for further audibility checking.

A large number of random rays are traced, as before, but at each reflection point, the receiver is checked to see whether it is visible. If it is, then the surface sequence is checked for a valid image-source. This adds constant work per-reflection to the ray-tracing process, which is insignificant in terms of overall time-complexity. This technique has two main advantages. Firstly, more paths are checked, so it is more likely to find all the valid image-sources. Instead of just checking ray paths which intersect the receiver, this method checks all paths which are *capable* of intersecting the receiver. Secondly, initial ray paths don't have to be specular, so techniques like *vector-based scattering* can be used. The disadvantage is that a greater number of validity checks are required, though this number is still many times smaller than would be required by a naive implementation.

3.2 Implementation

Here the concrete implementation of the image-source method is presented, as it is used in Wayverb. Figure 3.3 gives an overview of the entire process.

The main prerequisite for the simulation is a scene, made up of triangles, where each triangle has an associated material comprised of multi-band absorption coefficients. First, an axis-aligned bounding box is computed for this scene, and split into uniformly sized cuboid *voxels*. Each voxel holds a reference to any triangles in the scene which happen to intersect with that voxel. The voxel mesh acts as an "acceleration structure", speeding up intersection tests between rays and triangles. To check for an intersection between a ray and a number of triangles, the simplest method is to check the ray against each triangle individually, which is very time consuming. The voxel mesh allows the number of checks to be greatly reduced, by checking only triangles that are within voxels that the ray intersects. These voxels can be found very quickly, by "walking" the voxels along the ray, using an algorithm

1 ray tracing

2 possible paths

S → R
S → A → R
S → C → R
S → D → R
S → E → R
S → A → B → R
S → C → B → R
S → D → C → R
S → D → E → R
S → E → A → R

3 potential path tree

deeper reflections omitted

4 backtracking

S_{EA} S_A

S_{AB}

S_E

S_{DE} S_D

S_{CB}

S_C

S_{DC}

5 valid paths

path	length
S → R	11
S → A → R	25
S → C → R	19
S → D → R	16
S → E → R	17
S → D → C → R	22
S → D → E → R	21
S → E → A → R	28

Note: Per-path magnitude is proportional to the product of all surface reflectances on the path, multiplied by the reciprocal of the path length.

6 multiband impulse response

pressure

frequency band

time

7 broadband impulse response
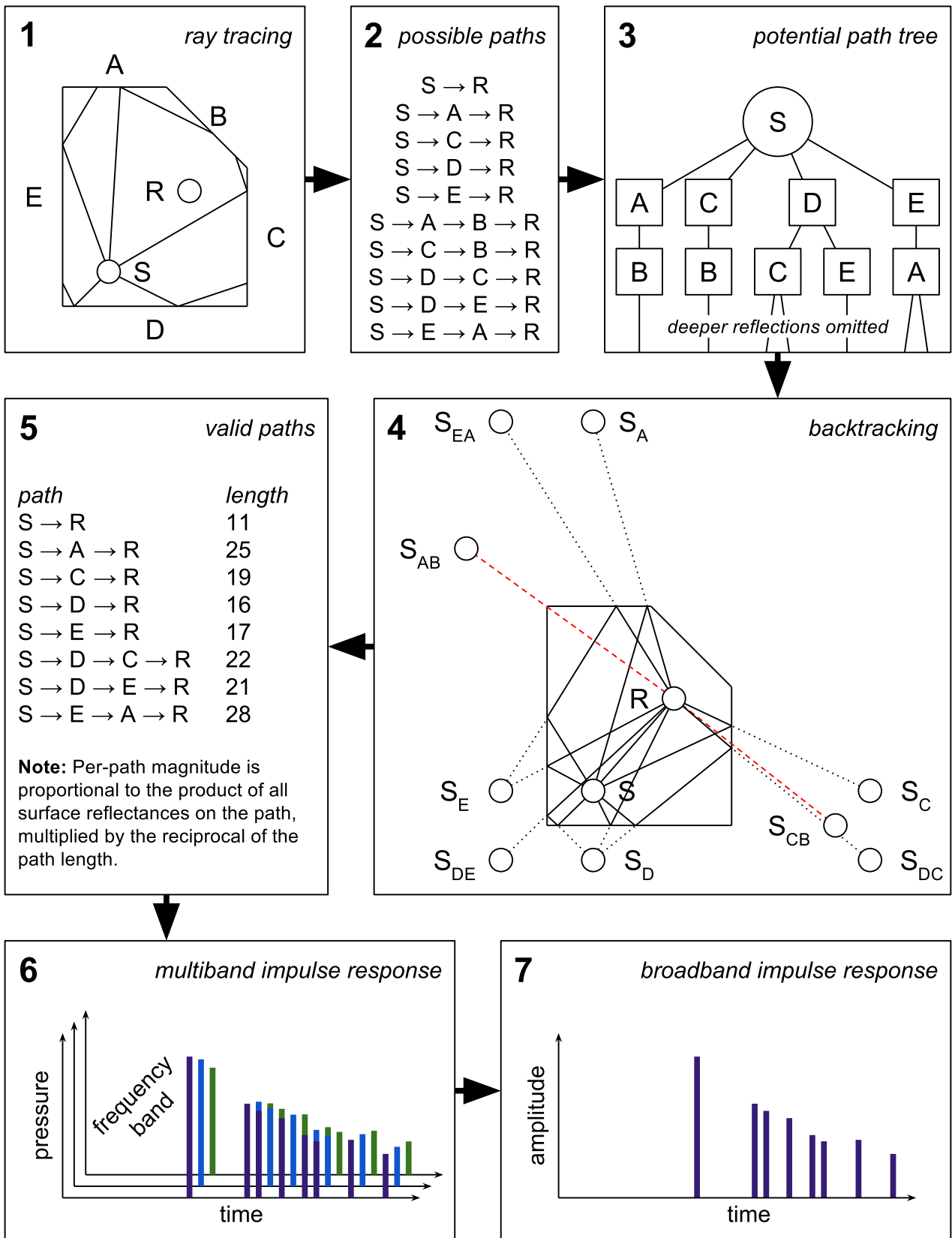
amplitude

time

Figure 3.3: Creation of an impulse response using image sources.

presented by Amanatides, Woo et al. [47]. For large scenes with many triangles, this method can lead to speed-ups of an order of magnitude or more. All ray-intersection tests mentioned throughout this thesis use the voxel-acceleration method, unless explicitly noted. In particular, both the initial ray-casting *and* the image-source audibility-checking are accelerated using the voxel method.

Rays are fired in uniform random directions from the source (item 1 in the figure). Each ray is checked for an intersection with the scene, and if an intersection is found, some data about the intersection is recorded. Specifically, the record includes the triangle which was intersected, and whether or not the receiver is visible from the intersection point. Then, the vector-based scattering method [43] (see the Ray Tracing section for details on this) is used to find the directions of new rays, which are fired from the intersection points. The ray-tracing process continues up to a certain depth, which is artificially limited to ten reflections in Wayverb. For most simulations, three or four reflections should be adequate, though this depends somewhat on the scattering coefficients of the surfaces. After a few reflections, most sound energy is diffuse rather than specular, and other methods are better suited to modelling this scattering, as explained in the Basic Method subsection above.

The ray tracer produces a list of reflection paths for each ray, where a single reflection path is defined by a sequence of visited surfaces (item 2 in the figure). Some rays may follow the same paths (reflect from the same surfaces), and so duplicate paths must be removed. This is achieved by condensing per-ray information into a tree of valid paths (item 3). Each node in the tree stores a reference to a triangle in the scene, and whether or not the receiver is visible from this triangle. Each unique path starting from a root node in the tree represents a possible image source contribution, which must be checked. This checking is carried out using the backtracking method explained above (item 4). The tree structure can be traversed using depth-first recursion, allowing the results of some intermediate calculations to be cached between ray paths, speeding up the calculation with only minimal memory overhead. This is similar to the approach mentioned in [2]. Also, because the tree is immutable, it can be shared between multiple worker threads, which independently check each branch for valid image sources. The nature of the recursive algorithm makes it a poor fit for an OpenCL implementation, so native (CPU) threads are used instead.

Some paths in the tree may not actually produce valid image sources, and these paths are discarded (item 5). For paths which *do* contribute valid image sources, the propagation delay and frequency-dependent pressure of the image source signal must be found. As described by Kuttruff, the propagation delay is equal to the distance from the receiver to the image source, divided by the speed of sound, [33, p. 325]. The pressure content is found by convolving together the reflectances of all intermediate surfaces. This is equivalent to a single multiplication per frequency band, as long as the reflectance value for each band per surface can be represented by a single real value.

The surface reflectances are found by converting per-band absorptions into per-band normal-incidence reflectance magnitudes using

$$|R| = \sqrt{1 - \alpha} \tag{3.2}$$

where R is the surface reflectance, and α is the absorption coefficient of that frequency band. This equation is simply a rearrangement of eq. 2.13. These per-band reflectances are converted to per-band normal-incidence impedances using eq. 2.19. Finally, the impedances are converted back to *angle-dependent* reflectances by eq. 2.16. This is the same approach taken in [26]. The angle of incidence must be found for each individual reflection, by taking the inverse cosine of the dot product between the incident ray direction and the surface normal, when both are unit vectors.

The contribution g of a single image source with intermediate surfaces $m_1 m_2 \ldots m_n$ is given by

$$g_{m_1 m_2 \ldots m_n} = \frac{\sqrt{Z_0/4\pi}}{d_{m_1 m_2 \ldots m_n}} \cdot R_{m_1} * R_{m_2} * \cdots * R_{m_n} * \delta(\frac{d_{m_1 m_2 \ldots m_n}}{c}) \tag{3.3}$$

where Z_0 is the acoustic impedance of air, c is the speed of sound, $d_{m_1 m_2 \ldots m_n}$ is the distance from the receiver to the image source, and R_{m_i} is the reflectance of surface i. This assumes that the original source emits a pressure impulse δ at the starting-time of the simulation. The contributions of all image sources must be summed together to find the final impulse response.

To create a digital audio file representing an impulse response, the output must be discretised at some sampling frequency f_s. The individual image source contributions must be added, at positions corresponding to their propagation delays, into an output buffer at that sampling frequency. The ideal buffer position for a given contribution is equal to τf_s where τ is the propagation delay of that contribution, equal to the total ray distance divided by the speed of sound. However, this value is unlikely to be an integer, and so may not coincide with a sample index. The simplest solution would be to round to the closest integer, and use this as the sample index. However, for applications such as multi-microphone simulation which are sensitive to arrival time, this can lead to phase errors. A better solution is suggested by Fu and Li [48]: The contribution can be positioned with sub-sample accuracy, by replacing the impulsive δ signal with the impulse-response of an ideal low-pass filter, with cut-off equal to the output Nyquist frequency. Such an impulse response is infinitely long, but tends to zero quickly, so a Hann window can be applied to limit its length. This form of the impulse is as follows:

$$\delta_{\mathrm{LPF}}(n - \epsilon) = \begin{cases} \frac{1}{2}(1 + \cos\frac{2\pi(n-\epsilon)}{N_w})\mathrm{sinc}(n - \epsilon), & -\frac{N_w}{2} < n < \frac{N_w}{2} \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

where n is an index in the output buffer, ϵ is the centre of the impulse in samples ($\epsilon = \tau f_s$), and N_w is the width of the window in samples.

Each image-source contribution has per-band pressure values. Rather than summing all contributions directly to the output buffer, several buffers are created, one per frequency band. The contributions for each band are summed into each buffer individually (item 6 in the figure). The final output of the simulation is created by band-passing and then mixing down the buffers (item 7). A single method for multi-band filtering is used throughout Wayverb, and more details are given in the Ray Tracer section.

3.3 Summary

The image source model can be used to find the path lengths and pressures of purely specular reflections. It cannot model diffuse reflections, and late reflections are generally diffuse. Therefore, the image source model is only suitable for predicting early reflections. The naive implementation of the image source model has exponential complexity, and a large proportion of the computations are redundant. For this reason, a ray-tracing-based implementation with greatly improved complexity has been developed. This implementation is more efficient than the naive implementation (i.e. it does less redundant work) although it may fail to find some valid image sources if the number of rays is too low. This has been deemed a reasonable trade-off. The implementation is also designed to re-use ray paths found by the ray-tracer model, minimising duplicated work between the two models. Finally, implementation details such as a method for frequency-dependent boundary modelling, and sub-sample-accurate impulse placement, have been described.

4 Ray Tracer

4.1 Background

Similarly to the image-source method, ray tracing assumes that sound energy travels around a scene in "rays". The rays start at the sound source, and are all emitted in uniformly random directions at the same time, travelling at the speed of sound. When a ray hits a boundary, it loses some of its energy, depending on the properties of the boundary's material. Then, the ray is reflected. When it intersects the receiver, the energy and time-delay of the ray is recorded. In these ways, the models are similar. However, there are some important differences between the two methods, explained below.

4.1.1 Stochastic Simulation

Image sources are deterministic, while ray tracing is stochastic. The image-source method finds exact specular reflections, which are fixed for given source, receiver, and boundary positions. Ray tracing is less accurate, aiming to compute a result which is correct, within a certain probability. A large number of rays are fired into the scene in random directions, and traced until they have undergone a certain number of reflections. Some of these rays may intersect with the receiver volume (see below), but some may not. Only rays that *do* intersect the receiver contribute to the final output. The proportion of rays which intersect the receiver, and therefore the measured energy at the receiver, can be found with greater accuracy simply by increasing the number of rays fired.

4.1.2 Receiver Volume

The random nature of ray tracing requires that the receiver must have a finite volume. The likelihood of any given random ray intersecting with a single point with no volume tends to zero (there is an infinite number of possible ray directions, only one of which passes through the point). If the probability of a ray-receiver intersection is to be non-zero, the receiver must have some volume. This is different to the image-source method, which traces reflections backwards from the receiver, allowing it to be represented as a point.

4.1.3 Energy and Distance

In ray tracing, each ray represents a finite portion of the initial source energy. The reduction of energy over a given distance is accounted for by the spreading-out of the rays. This can be illustrated very simply: Imagine a sphere placed very close to a point. Assuming rays are fired with a uniform random distribution from that point, a certain proportion of those rays will intersect with the sphere. If the sphere is moved further away, a smaller proportion of rays will hit it (see fig. 4.1).
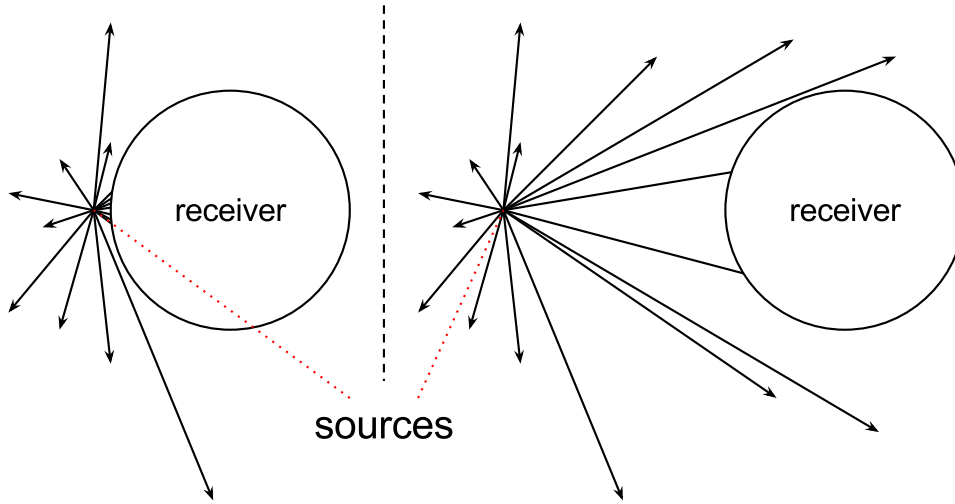
Figure 4.1: The proportion of randomly-distributed rays intersecting with a sphere depends on the distance between the ray source and the sphere.

The exact proportion of intersecting rays is equal to $s/4r^2$ [35, p. 75], where s is the constant area covered by the receiver, and r is the distance between the source and receiver. That is, the proportion of rays intersecting the receiver is inversely proportional to the square of the distance between the source and receiver. The energy registered is proportional to the number of ray intersections recorded, therefore the ray model intrinsically accounts for the inverse-square law for energy, and the per-ray energy does not need to be scaled proportionally to the distance travelled. This differs to the image-source model, in which only valid specular reflections are recorded, and the inverse-square law must be applied directly.

4.1.4 Rendering

The final major difference between ray tracing and the image-source method is to do with the way in which results are recorded. The image-source method finds exact specular reflections, each of which contributes an impulsive signal with specific frequency content at a precise time. This reflection data is precise and accurate, so it can be used to render an output signal at arbitrarily high sampling frequencies. Ray tracing, on the other hand, is inexact because it is based on stochastic methods. The accuracy of the output increases with the average number of rays detected per unit time. It is shown in [34, p. 191] that the mean number of intersections k per time period Δt is given by

$$k = \frac{N\pi r^2 c\Delta t}{V} \tag{4.1}$$

where N is the number of rays, r is the radius of the receiver, c is the speed of sound, and V is the room volume.

For an output which covers the human hearing range, the sampling rate must be at least 40kHz, which corresponds to a sampling period of 25μs. Therefore, for a receiver radius of 0.1m (around the size of a human head), and assuming that one detection-per-sampling-period is adequate, the minimum number of rays is

$$N = \frac{kV}{\pi r^2 c \Delta t} = \frac{V}{\pi \cdot 0.1^2 \cdot 340 \cdot 0.000025} \approx 3745V \tag{4.2}$$

In actual simulations, especially of large and complex rooms, this number of rays is likely to produce results with large, inaccurate energy fluctuations. For higher accuracy, higher output sample rates, and smaller receivers the number of rays required becomes even greater. This sheer quantity of rays requires a vast number of independent calculations which will be prohibitively time-consuming, even on modern hardware.

If, on the other hand, audio-rate results are not required, then the number of necessary rays is much lower. Vorländer suggests a sampling period of the order of magnitude of milliseconds, which requires at least 40-times fewer rays [34, p. 186].

Now, the ray tracer can be thought to produce an *energy envelope* describing the decay tail of the impulse response. To produce the impulse response itself, this energy envelope is simply overlaid onto a noise-like signal. The process will be described in greater detail in the following Implementation section.

4.2 Implementation

Here, Wayverb's ray tracer will be described. Details of the boundary- and microphone-modelling processes are discussed separately, in the Boundary Modelling and Microphone Modelling sections respectively.

4.2.1 Finding Reflections

The simulation begins identically to the image-source process. A voxel-based acceleration structure is created, to speed up ray intersection tests.

Rays are fired in uniformly-distributed random directions from the source point. Each ray carries an equal quantity of energy (the method for determining the starting energy is described in the Hybrid section). If a ray intersects with the scene geometry, data is stored about that intersection: its position, the unique ID of the triangle which was intersected, and whether or not the receiver point is visible from this position. This data will be used later on, when calculating energy loss, and the directional distribution of received energy.

Next, a new ray direction is calculated using the *vector-based scattering* method, described by Christensen and Rindel [43]. A uniformly random vector is generated, within the hemisphere oriented in the same direction as the triangle normal. The ideal specular direction is also calculated, and the two vectors are combined by

$$\vec{R}_{\text{outgoing}} = s\vec{R}_{\text{random}} + (1-s)\vec{R}_{\text{specular}} \tag{4.3}$$

where s is the scattering coefficient, as described in the Theory chapter. Normally, the scattering coefficient would be defined per-band, but this would require running the ray tracer once per band, so that each frequency component can be scattered differently. Instead, the mean scattering coefficient is used, so that all bands can be traced in one pass. For eight frequency bands, this provides an eight-times speed-up, at the cost of inaccurate interpretation of the scattering coefficients. This is a reasonable trade-off, as scattering will also be modelled using the *diffuse rain* technique described in the Boundary Modelling section, which *is* able to account for different per-band scattering coefficients. The final output will therefore retain per-band scattering characteristics, but with much improved performance.

Having calculated a new ray direction, the energy carried in the ray is decreased, depending on the absorption coefficients of the intersected triangle. If the surface has an absorption coefficient of α in a particular band, then the energy in that band is multiplied by $(1 - \alpha)$ to find the outgoing energy. This process is repeated, using the incoming energy and absorption coefficient for each band, to find outgoing energies in all bands. The new ray, with the computed outgoing energies and vector-scattered direction, is now traced.

The ray tracing process continues for a set number of reflections. Typically, each ray would be traced until the energy in all bands has fallen below a certain threshold, requiring an additional check per reflection per ray [34, p. 183]. Under such a scheme, some rays might reach this threshold faster than others, depending on the absorptions of intermediate materials. However, in Wayverb all rays are traced in parallel, so it is not feasible or necessary to allow rays to quit early. The time taken for a parallel computation will always be limited by the longest-running process. If some rays are "stopped" early, this does not improve the processing-speed of the continuing rays, so the simulation still takes the same time to complete. Instead, the maximum possible required depth is found before the simulation, and all rays are traced to this maximum depth.

To find the maximum required ray tracing depth, first the minimum absorption of all surfaces in the scene is found. The outgoing energy from a single reflection is equal to $E_{\text{incoming}}(1 - \alpha)$ where E_{incoming} is the incoming energy and α is the surface absorption. It follows that the outgoing energy from a series of reflections is given by $E_{\text{incoming}}(1 - \alpha)^{n_{\text{reflections}}}$. Then, the maximum ray tracing depth is equal to the number of reflections from the minimally absorptive surface required to reduce the energy of a ray by 60dB:

$$(1 - \alpha_{\text{min}})^{n_{\text{reflections}}} = 10^{-6} \therefore n_{\text{reflections}} = \left\lceil -\frac{6}{\log_{10}(1 - \alpha_{\text{min}})} \right\rceil \tag{4.4}$$

The 60dB level decrease is somewhat arbitrary, but was chosen to correspond to the *RT60*, which is a common descriptor of recorded impulse responses. In a future version of the software, the level decrease might be set depending on the dynamic range of the output format. This would allow 16-bit renders (with around 48dB of dynamic range) to use fewer reflections, while 32-bit outputs with lower noise floors would require more reflections.

4.2.2 Logging Energy

The output of the ray tracing process is a set of histograms, one per frequency band, plotting recorded energy per time step. This recorded energy may come from two different sources.

Firstly, if a ray intersects with the receiver volume, then the current per-band energy of that ray, which may have been attenuated by previous reflections, is added to the result histograms at the appropriate time step. The time of the energy contribution is given by the total distance travelled by the ray, divided by the speed of sound. This is the approach taken in typical acoustic ray tracers.

Secondly, each reflection point is considered to spawn a "secondary source" which emits scattered sound energy, depending on the scattering coefficients of the surface. If the receiver is visible from the reflection point, then an energy contribution is logged, at a time proportional to the distance travelled by the ray. Similarly to absorption coefficients, scattering coefficients are defined per frequency band, and these coefficients are used to weight the contributions logged in each band of the output histograms. This mimics the real world behaviour of rough surfaces, which cause some energy to be randomly diffused in non-specular directions during reflection of the wave-front. The exact level of this contribution is explained in the Geometric Implementation subsection of the Boundary Modelling page.

4.2.3 Producing Audio-rate Results

When ray tracing has completed, the result is a set of histograms which describe the energy decay envelope of each frequency band. These histograms will have the relatively low sampling rate, as explained above (Wayverb uses a sampling rate of 1kHz). As a result, these histograms are not directly suitable for auralisation. To produce audio-rate impulse responses, the "fine structure" of the decay tail must be synthesised and then the gain adjusted using the histogram envelopes. The process used in Wayverb to convert the histogram into an audio-rate impulse response is described in [49], and in greater depth in [35, p. 70], though an overview will be given here. Figure 4.2 outlines the process of estimating an audio-rate representation of low-sample-rate multi-band histograms.

4.2.3.1 Generating a Noise Signal

First, a noise-like sequence of Dirac impulses is generated at audio-rate. This sequence is designed to mimic the density of reflections in an impulse response of a certain volume. Therefore it is modelled as a temporal Poisson process which starts sparse, and with increasing density of impulses over time. Specifically, the time between one impulse event and the next is given by

$$\Delta t_{\text{event}}(z) = \frac{\ln \frac{1}{z}}{\mu} \tag{4.5}$$

where z is a uniformly distributed random number $0 < z \leq 1$. μ here is the mean event occurrence, and is dependent upon the current simulation time t, the enclosure volume V and the speed of sound c:

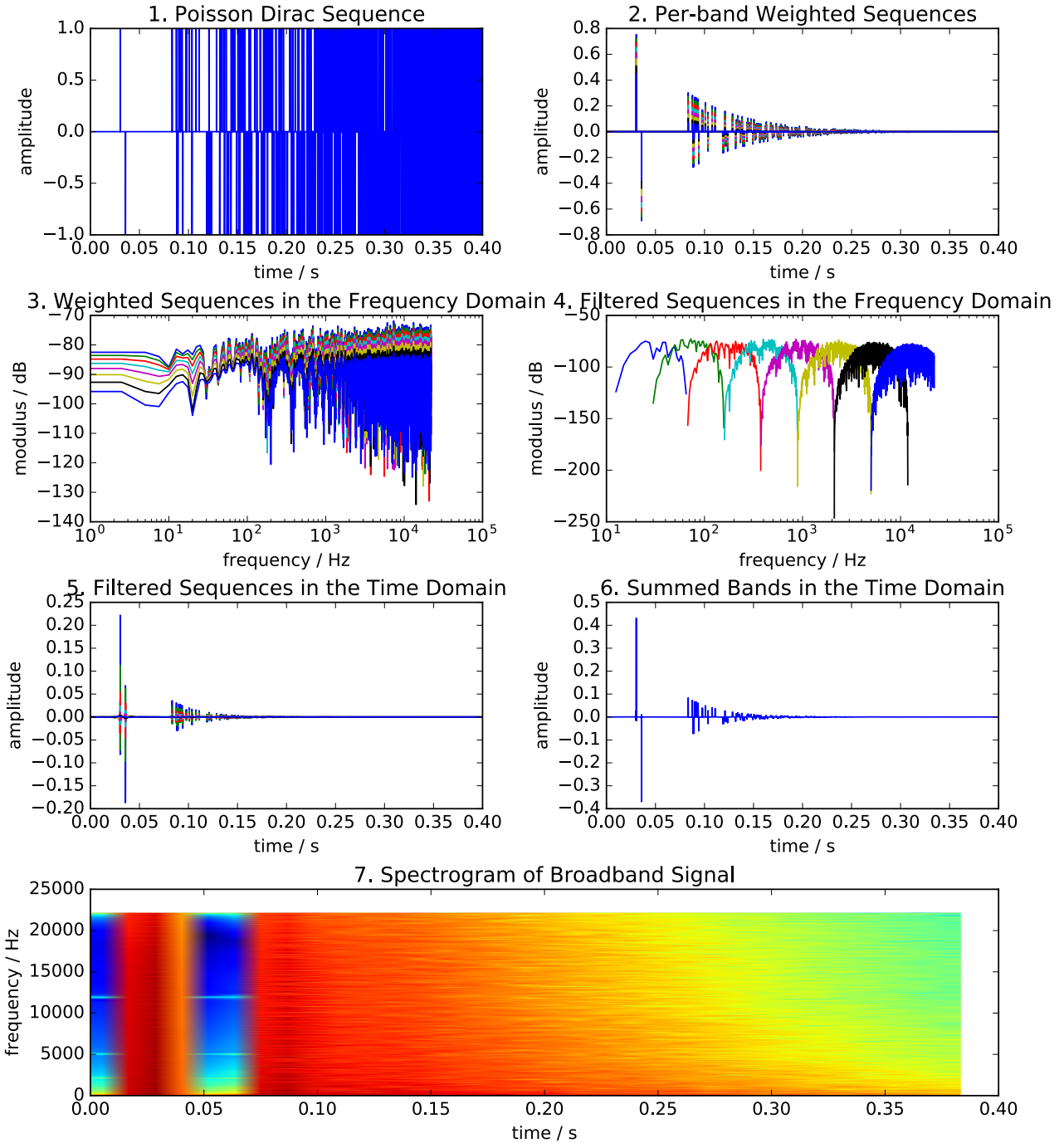$$\mu = \frac{4\pi c^3 t^2}{V} \tag{4.6}$$

Figure 4.2: Generating an audio-rate signal from multi-band ray tracing energy histograms at a low sampling rate.

It can be seen that the mean occurrence is proportional to the square of the current time, producing an increase in event density over time. The first event occurs at time t_0:

$$t_0 = \sqrt[3]{\frac{2V \ln 2}{4\pi c^3}} \tag{4.7}$$

The full-length noise signal is produced by repeatedly generating inter-event times Δt_{event}, and adding Dirac impulses to a buffer, until the final event time is greater or equal to the time of the final histogram interval. Dirac deltas falling on the latter half of a sampling interval are taken to be negative-valued. The number of Dirac deltas per sample is limited to one, and the value of μ is limited to a maximum of 10kHz, which has been shown to produce results absent of obvious artefacts [49].

4.2.3.2 Weighting Noise Sequences

The noise sequence is duplicated, once for each frequency band. Then, the noise sequence for each band is weighted according to that band's histogram. This enveloping is not quite as simple as multiplying each noise sample with the histogram entry at the corresponding time. Instead, the enveloping process must conserve the energy level recorded over each time step.

For each time interval in the histogram, the corresponding range of samples in the noise sequence is found. If the output sample rate is f_s and the histogram time step is Δt, then the noise sequence sample corresponding to histogram step h is $\lfloor h \cdot f_s \cdot \Delta t \rfloor$. The corrected energy level for each histogram step is found by dividing the histogram energy value by the sum of squared noise samples for that step. This is converted to a corrected pressure level by $P = \sqrt{Z_0 I}$, where I is the corrected energy level, and Z_0 is the acoustic impedance of air. The weighting is now accomplished by multiplying each noise sequence sample corresponding to this histogram step by the corrected pressure level.

4.2.3.3 Multi-band Filtering

Now, we are left with a set of broadband signals, each with different envelopes. The output signal is found by bandpass filtering each of these signals, and then mixing them down.

The filter bank should have perfect reconstruction characteristics: a signal passed through all filters in parallel and then summed should have the same frequency response as the original signal. In the case where all the materials in a scene have the same coefficients in all bands, the input to each filter would be identical. Then, the expected output would be the same as the input to any band (though band-passed between the lower cutoff of the lowest band and the upper cutoff of the highest band). Perfect-reconstruction filters maintain the correct behaviour in this (unusual) case. It is especially important that the bandpass filters are zero-phase, so that Dirac events in all bands are in-phase, without group delay, after filtering. Finally, the filters should have slow roll-off and no resonance, so that if adjacent bands have very mismatched levels, there are no obvious filtering artefacts.

An efficient filtering method is to use a bank of infinite-impulse-response filters. These filters are fast, and have low memory requirements. They can also be made zero-phase when filtering is offline, by running the filter forwards then backwards over the input signal (though this causes filter roll-off to become twice as steep). This was the initial method used in Wayverb: the filter bank was constructed

from second-order Linkwitz-Riley bandpass filters. This method had two main drawbacks: the roll-off is limited to a minimum of 24 dB/octave [50], which may cause an obvious discontinuity in level in the final output; and the forward-backward filtering method requires computing the initial filter conditions in order to maintain perfect-reconstruction, which is non-trivial to implement [51].

A better method, which allows for shallower filter roll-offs while retaining perfect-reconstruction capabilities is to filter each band directly in the frequency domain. The filtering of each signal is accomplished by computing the signal's frequency-domain representation, attenuating bins outside the passband, and then transforming the altered spectrum back to the time domain. To ensure perfect reconstruction, and to avoid artificial-sounding discontinuities in the spectrum, the filter shape suggested in [52] is used. This paper suggests equations which describe the band-edge magnitudes:

$$G_{\text{lower}}(\omega_{\text{edge}} + p) = \sin^2\left(\frac{\pi}{2}\phi_l(p)\right), G_{\text{upper}}(\omega_{\text{edge}} + p) = \cos^2\left(\frac{\pi}{2}\phi_l(p)\right) \tag{4.8}$$

Here, G is a function of frequency, ω_{edge} is the band-edge frequency, and p is the relative frequency of a nearby frequency bin. The equations are computed for a range of values $p = P, \ldots, P$ where P is the width of the crossover. The definition of $\phi_l(p), l \geq 0$ is recursive:

$$\phi_l(p) = \begin{cases} \frac{1}{2}(p/P + 1), & l = 0 \\ \sin(\frac{\pi}{2}\phi_{l-1}(p)), & \text{otherwise} \end{cases} \tag{4.9}$$

The variable l defines the steepness of the crossover, and is set to 0 in Wayverb, so that the transition between bands is as slow and smooth as possible. The absolute width of the crossover is denoted by P, but it is more useful to specify the crossover width in terms of overlap $0 \leq o \leq 1$. Assuming logarithmically-spaced frequency bands, spread over the range $\omega_{\text{lowest}}, \ldots, \omega_{\text{highest}}$ where the edge frequency of band i is defined as

$$\omega_{\text{edge}_i} = \omega_{\text{lowest}}\left(\frac{\omega_{\text{highest}}}{\omega_{\text{lowest}}}^{\frac{i}{N_{\text{bands}}}}\right) \tag{4.10}$$

the maximum width factor w is given by

$$w = \frac{x-1}{x+1}, x = \frac{\omega_{\text{highest}}}{\omega_{\text{lowest}}}^{\frac{1}{N_{\text{bands}}}} \tag{4.11}$$

For $\omega_{\text{lowest}} = 20$Hz, $\omega_{\text{highest}} = 20$kHz, and $N_{\text{bands}} = 8$, $w \approx 0.4068$. Then, the band edge width P can be defined in terms of the overlap-amount o, the frequency of this band edge ω_{edge}, and the maximum allowable width factor w: $P = \omega_{\text{edge}}ow$. Wayverb sets the overlap factor $o = 1$ to ensure wide, natural-sounding crossovers.

The final broadband signal is found by summing together the weighted, filtered noise sequences.

4.3 Summary

Like the image-source model, ray tracing assumes that energy is transported through the sound field in rays. However, ray tracing is stochastic, meaning that the overall accuracy of the simulation (including the maximum viable sampling rate) is governed by the number of rays used in the simulation. Tracing enough rays to produce audio-rate results is unnecessarily computationally expensive. It is much more efficient to trace at a reduced sampling rate, and to approximate the fine structure of the reverb tail during post-processing. This matches the goals of the Wayverb project, by increasing efficiency without negatively affecting the plausibility of the results.

In Wayverb, the ray tracer produces multi-band energy histograms, which are used to weight Poisson noise sequences. Perfect-reconstruction frequency-domain filtering is used to combine individual frequency bands into a single broadband IR. The final model is capable of modelling loss of sound energy due to distance, along with frequency-dependent boundary absorption and scattering.

5 Digital Waveguide Mesh

5.1 Background

The *digital waveguide mesh* (DWM) is one of several wave-based simulation techniques. Each technique in this family is derived directly from the wave equation, allowing them to inherently support wave phenomena such as diffraction and interference. Wave effects such as these have a great effect upon the low-frequency response of a room. This means that at low frequencies wave-based methods are far more accurate than geometric methods, which are not able to model wave effects [1].

The drawback of wave-based methods is that their computational complexity increases rapidly with the maximum output frequency, and with the volume of the modelled space. A waveguide simulation of a space with volume V, at sampling frequency f_s, will have a complexity of $O(V f_s^3)$. This means that, on current consumer hardware, it is not feasible to compute a full-spectrum simulation using wave-based techniques. The hybrid simulation method implemented in Wayverb aims to optimise computation time, while retaining reasonable accuracy across the spectrum, by combining wave-based methods with geometric methods. Wave-based methods are used to calculate accurate low-frequency content, while geometric methods estimate the higher frequency content, which is less dependent upon wave effects.

There are, largely speaking, two main types of wave-based simulation used for room acoustics: element methods, and finite difference methods. The waveguide mesh is the latter, a simplified sub-class of the *finite-difference time-domain* (FDTD) technique. Although the DWM and FDTD have converged over time, and are equivalent [53], [54], their histories are quite different. The DWM was designed to be efficient for small-scale acoustic simulations in which the only quantity of interest is pressure [3], while FDTD is a more general technique designed for electromagnetic simulation, in which the electric and magnetic fields are both of interest [55].

5.1.1 Method

The derivation of the waveguide mesh begins with the one-dimensional *digital waveguide*. A one-dimensional waveguide can exactly describe the behaviour of a band-limited wave in one dimension. Such a model is well-suited for predicting the behaviour of certain musical instruments which use columns of air or vibrating strings to produce sound. The model itself is based on d'Alembert's solution of the wave equation in one dimension [5, p. 86]. The displacement of a string y at time t and position x can be written as

$$y(t,x) = y_r \left(t - \frac{x}{c} \right) + y_l \left(t + \frac{x}{c} \right) \tag{5.1}$$

where $y_r \left(t - \frac{x}{c}\right)$ and $y_l \left(t + \frac{x}{c}\right)$ are the right- and left-going travelling waves respectively, with speed c [3]. The discrete form of this equation is given in terms of constant time and space divisions, T and X respectively:

$$y(nT, mX) \overset{\Delta}{=} y^+(n - m) + y^-(n + m) \tag{5.2}$$

where superscript $+$ and $-$ denote propagation to the right and left respectively, and n and m are integers, used to index the spatial and temporal sampling intervals [53].

An implementation of these equations will take the form of two parallel delay lines, which propagate wave components in opposite directions. This is shown in fig. 5.1. The "output" of the simulation, that is, the physical displacement of the modelled string over time, is found by adding the wave components in both delay lines at a single point.
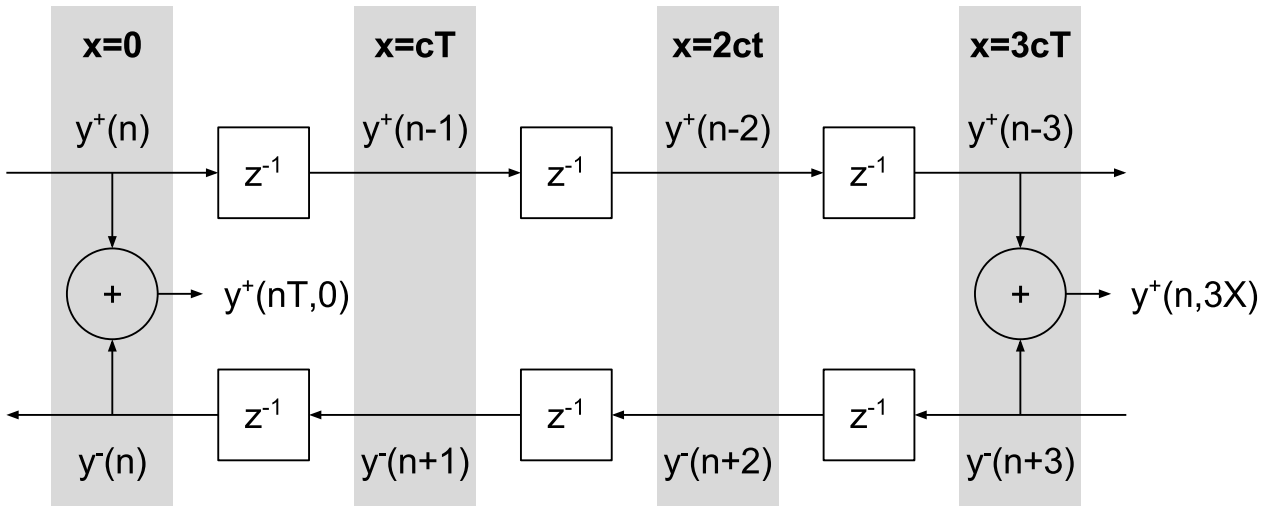
Figure 5.1: Delay lines cause wave components to be propagated along the "string" over time. The total displacement of the string is found by adding together values from the same point on each delay line.

Waveguides in higher dimensions can be created in a straightforward manner, by connecting digital waveguide elements at a *scattering junction*. Wave components entering the junction are distributed amongst the waveguide elements, preserving energy and power according to Kirchoff's power conservation laws [5, p. 87]. The sound pressure p_J at a lossless scattering junction J with N connected elements or "ports" is the summed incoming components of all connected elements:

$$p_J = \frac{2 \sum_{i=1}^{N} \frac{p_i^+}{Z_i}}{\sum_{i=1}^{N} \frac{1}{Z_i}} \tag{5.3}$$

where p_i is the pressure in waveguide element i and Z_i is its associated impedance. This simplifies, if all impedances are equal, which is true for homogeneous media:

$$p_J = \frac{2}{N} \sum_{i=1}^{N} p_i^+ \tag{5.4}$$

A *digital waveguide mesh* is any configuration of regularly-arranged N-port scattering junctions which are separated by unit delay lines. In some literature, this specific type of mesh is known as a *W-DWM* because it operates in terms of "W" (wave) variables.

The input into a scattering junction is equal to the output of a neighbour junction at the previous time step. This fact allows the waveguide mesh to alternatively be formulated directly in terms of the pressure at each junction (assuming all junction impedances are equal) [18]:

$$p_J(n) = \frac{2}{N} \sum_{i=1}^{N} p_i(n-1) - p_J(n-2) \tag{5.5}$$

That is, the next pressure at a given node depends on the previous pressure at that node, and the current pressure at surrounding nodes. This alternative formulation operates on Kirchhoff variables, and is therefore known as a *K-DWM*. In the 1D case, the K-DWM and W-DWM are computationally identical [53]. In higher dimensions, they are equivalent only under certain circumstances [56, p. 5].

The K-DWM is advantageous compared to the W-DWM for reasons of efficiency. It requires less memory, and fewer calculations per node per step than the W-DWM: experiments in [18] show that the K-DWM is 200% faster and uses 50% of the memory of the equivalent W-DWM. This is mainly to do with the number of values stored per node. Each node in a K-DWM must store a "current" and a "previous" pressure value, whereas in a W-DWM each *connection* must store a value [54]. For mesh layouts in which each node has many neighbours (see Mesh Topology below), the W-DWM can require many times more memory. The K-DWM also requires one fewer addition per node per step [5, p. 91], so will be slightly faster, all else being equal.

In the general case of an N-dimensional waveguide mesh, the spatial and temporal sampling periods are related by the Courant number λ. The Courant criterion specifies the conditions required for numerical stability of the simulation:

$$\lambda = \frac{cT}{X} \leq \frac{1}{\sqrt{N}} \tag{5.6}$$

The highest sampling rate and lowest error is achieved by setting the Courant number to its maximum value [57]. This is normally desirable, and so the inequality above can be simplified:

$$T = \frac{X}{c\sqrt{N}} \tag{5.7}$$

Here, T is the temporal sampling period, X is the spatial sampling period, c is the speed of sound and N is the number of spatial dimensions.

A higher output sampling rate requires a smaller inter-nodal spacing and therefore more modelled points per-unit-volume, which in turn requires more memory and more calculations per time step.

An output signal created using a mesh with a sampling frequency $f_s = 1/T$ has a maximum available bandwidth which spans from DC to the Nyquist frequency, $0.5 \cdot f_s$. However, the *valid* bandwidth of the waveguide output is often considerably lower. For example, the highest valid frequency in the rectilinear mesh is $0.196 \cdot f_s$. Detailed bandwidth information for other mesh topologies is given

in [39]. The output signal will contain high-frequency content above the maximum valid frequency, however numerical dispersion in this region is so high that the results are completely non-physical. To ensure that the output only contains frequencies within the valid bandwidth, the invalid high frequency content must be removed using a low-pass filter at the output. Alternatively, the mesh may be excited using an input signal with no frequency content above the maximum valid frequency.

5.1.2 Strengths and Weaknesses of the DWM

The main advantage of the DWM is its relative simplicity. The air in the simulation is evenly divided into nodes. Each node has an associated pressure, and also stores its previous pressure. The next pressure at a node depends on its previous pressure, and the current pressures at its neighbours. The simulation progresses by repeatedly updating all nodes, and storing the change in pressure over time at some output nodes.

This simplicity presents an optimisation opportunity. Each node can be updated completely independently, as long as all updates in one time-step are completed before the next time-step begins. This means that the updates can happen in parallel. In fact, with enough memory and processing cores, the entire mesh could be updated in the time that it takes to update a single node. The update method is also very simple, only requiring some additions and a multiplication per node per step. This kind of simple, parallel code lends itself to implementations on graphics hardware, which is designed for running similar calculations simultaneously over large inputs. *Graphics processing units* (GPUs) have best throughput when all threads execute the same instruction, and when memory is accessed at consecutive addresses from running threads. The K-DWM update equation has no branching, and has consistent memory access patterns, which should allow for a very efficient implementation for GPUs.

The greatest limitation of the DWM is *dispersion error*. Unlike waves in homogeneous physical media, the velocity of wave propagation in the DWM depends on the direction of propagation, and also on the frequency of the wave component. This leads to errors in the frequency response of recorded signals, especially towards the upper limit of the output bandwidth. The exact pattern of dispersion error is dependent upon the topology of the mesh (topology is explained in the Mesh Topology subsection), and can be examined using *Von Neumann* analysis [37]. One solution to the dispersion problem is to increase the sampling rate of the mesh, moving the high-error area out of the region of interest. Of course, this can quickly become very expensive, as the number of nodes, and therefore memory usage and computation time is proportional to the inverse cube of the sampling period. Another option is to use a mesh topology designed to reduce both direction- and frequency-dependent error, such as those presented in [39]. One interesting variation on this option is to reduce only direction-dependent error, and then to compensate for frequency-dependent error with a post-processing step, which is the approach taken in [11]. However, these mesh topologies with higher accuracy and isotropy require relatively high numbers of calculations per node. The interpolated schemes in [39] require 27 additions and 4 multiplications per node, whereas a tetrahedral mesh would require 5 additions and 1 multiplication. It is clear that high-accuracy results will be very costly to compute, whichever method is used.

5.2 Design Choices

5.2.1 Mesh Topology

There is no single optimal implementation of the digital waveguide mesh. Perhaps the most important decision is the mesh topology or *stencil* that will be used, and by extension the mesh update equation. Here, the mesh topology refers to the pattern which is used to distribute nodes throughout the modelled space. The choice of topology will affect the accuracy, memory usage, calculation speed, and implementation complexity of the final design. It must therefore be chosen with care, in order to satisfy the constraints of the particular application.

The simplest topology is rectilinear, in which nodes are laid out on the vertices of a cubic grid, and each node has 6 direct neighbours. During the mesh update, the pressure at each neighbour node must be checked in order to calculate the next pressure at the current node. This is straightforward to implement, as the nodes can be stored in memory in a three-dimensional array, in which the array extents define the mesh dimensions, and the array indices refer to the positions of individual nodes. Other options for the topology include tetrahedral, octahedral, and dodecahedral, in which nodes have 4, 8, and 12 neighbours respectively, as shown in fig. 5.2. In most of these meshes, the nodes align to a cubic grid in a straightforward manner. However, in the case of the tetrahedral mesh, the tessellation scheme is less obvious. The method used to construct the tetrahedral mesh is shown in detail in fig. 5.3.
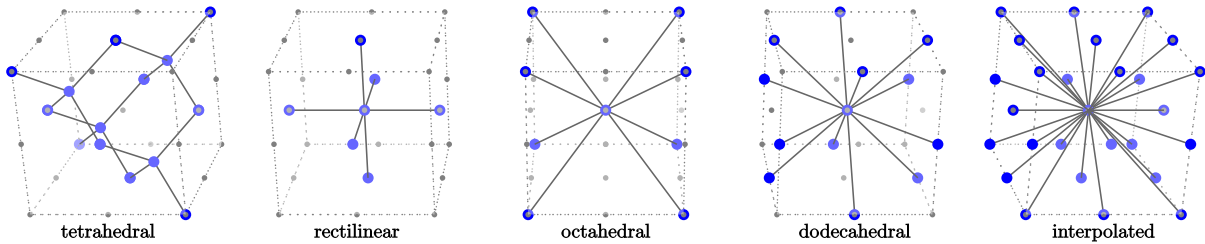
Figure 5.2: Some of the most common mesh topologies. Black lines show connections to nodes that will be checked during update. Note that the tetrahedral topology is unique, in that nodes can have two different orientations.

In their evaluation of different mesh topologies, Kowalczyk and Walstijn [39] define an accuracy criterion which can be used to compare topologies. This criterion is defined as the frequency band within which the maximum relative numerical error does not exceed 2%. They also describe how this "acceptably-accurate bandwidth" may be increased by overlaying or "superposing" rectilinear, octahedral, and dodecahedral schemes together. This is possible because all nodes are oriented uniformly, and have cubic tessellation. Such schemes are known as *interpolated*, and in these schemes each node has 26 neighbours. The rectilinear, octahedral, dodecahedral, and interpolated schemes may additionally all be represented by a single "unified" update equation, which is presented in [39]. In this respect the tetrahedral scheme is unique, requiring a dedicated update method. This is because the node connections in a tetrahedral mesh may be oriented in either of two directions, effectively requiring two update equations instead of one.

Figure 5.3: **Left:** A single "block" of a tetrahedral mesh. **Right:** Individual blocks are combined in an alternating chessboard-like pattern to create a mesh with tetrahedral topology.

If the primary concern is speed rather than accuracy, a scheme with fewer neighbour nodes should be used, as the number of calculations per node is proportional to the number of neig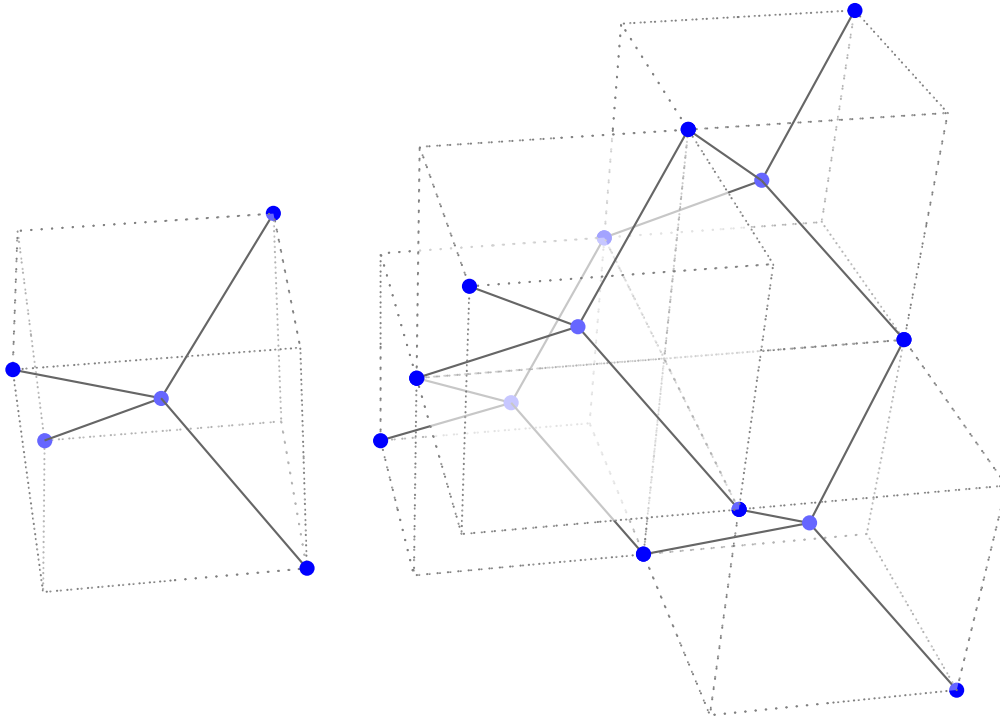hbours [8]. The tetrahedral mesh has fewest neighbours per node, and also has the lowest density. That is, it requires the fewest nodes to fill a volume at any given sampling rate. Fewer nodes to update means fewer calculations, and a faster overall simulation. Lower density meshes also require less storage, so the tetrahedral scheme is the most time and memory efficient.

To optimise for accuracy, it appears that there are two possible approaches: The first option is to use an interpolated scheme, which is relatively inefficient in terms of time and space requirements, but which is most accurate for any given sampling rate. The second option is to use a tetrahedral mesh, which is inaccurate but the most time- and space-efficient, and to oversample until the required accuracy is achieved. Unfortunately, there is no prior research comparing the accuracy and efficiency of the tetrahedral topology against interpolated schemes. Due to time constraints, this could not be investigated as part of the Wayverb project (implementation and numerical analysis of mesh topologies is not trivial). It was, however, noted that the tetrahedral mesh is the more flexible of the two approaches. That is, when accuracy is not important, the tetrahedral mesh will always be most efficient, but it can be made more accurate by oversampling. On the other hand, the interpolated schemes cannot be tuned to produce less accurate results quickly - they will always be accurate but inefficient. For these reasons, the tetrahedral mesh was initially chosen for use in Wayverb.

The tetrahedral mesh was implemented during the first two months of the project, with support for microphone modelling. When it came to implementing frequency dependent boundary conditions, no prior research could be found discussing boundary implementations in a tetrahedral topology. As noted in the conclusion of [39]:

> One aspect that has not been dealt with in this paper, and could be of interest for future research, is the comparison of the identified schemes with the tetrahedral topology... However, the applicability of the tetrahedral stencil to room acoustic simulations is debatable due to the nontrivial formulation of boundary conditions for complex room shapes.

The design of a new boundary formulation is outside the scope of this research, which was primarily concerned with the implementation of existing techniques rather than the derivation of new ones. Instead, the standard leapfrog (rectilinear) scheme was adopted at this point. Most of the tetrahedral code had to be rewritten, as the update schemes are very different (tetrahedral nodes have two possible update equations depending on node orientation), and the memory layout and indexing methods are more involved. The new scheme is much simpler than the tetrahedral mesh, and was quick to implement.

The rectilinear mesh uses the same cubic tessellation as the more complex topologies mentioned earlier, so in the future the update equation could conceivably be replaced with a more accurate "interpolated" alternative. Such a scheme would be more suitable than the rectilinear mesh for simulations where high accuracy is required. The conversion would only require changes to the update equations (and would complicate the boundary modelling code), but would be more straightforward than the move from the tetrahedral to the rectilinear topology. Due to time constraints, this was not possible during the project, meaning that the waveguide in Wayverb is not the most optimal in terms of accuracy *or* speed. Interpolated meshes are more accurate, and tetrahedral meshes are faster, although the rectilinear mesh is the fastest mesh with a cubic stencil. Instead, Wayverb's waveguide was optimised for ease and speed of implementation. Use of a more suitable mesh topology would be a sensible starting point for future development work on the project.

5.2.2 Source Excitation Method

Input and output methods for the digital waveguide mesh are superficially simple. The waveguide mesh is a physical model, and so it is easy to draw an analogy between the waveguide process, and the process of recording a physical impulse response. Typically, for physical spaces, a speaker plays a signal at some location within the space, and a microphone reads the change in air pressure over time at another point in the space. The impulse response is found by deconvolving the input signal from the recorded signal. The analogue of this process within the waveguide mesh is to excite the mesh at one node, and to record the pressure at some output node at each time step, which is then deconvolved as before. Recording node pressures is simple, and deconvolution is a well-known technique. Injecting a source signal, however, requires careful engineering in order to maintain the physical plausibility and numerical robustness of the model. Source design has two main concerns: the method that is used to add the signal into the mesh, and the signal that is injected.

5.2.2.1 Input Node Update Method

Firstly, the input signal may be injected at a single node, or distributed across several [58]. It is complicated to calculate an appropriate distributed signal [59], and so single-node sources are more common in the literature [58], [60]–[63].

There are two main options for updating the source node with the input signal. The first method, known as a *hard source*, simply overwrites the pressure value at the source node with that of the input signal. Hard sources are simple to implement, and ideally couple the input signal to the mesh, but also scatter any incident wave-fronts [64]. Although the initially radiated field is optimal, this spurious scattering is an undesirable artefact. Furthermore, the abrupt pressure discontinuity introduced by the hard source can cause more artefacts through the accumulation of numerical errors. For these reasons, hard sources are generally unsuitable for high-accuracy simulations.

The second method, the *soft source*, instead adds or superimposes the input signal on the standard mesh update equation. Soft sources obey the mesh update equations, and so do not suffer from the scattering problem. However, they do not couple the input signal to the mesh, so that the radiated wave front does not resemble the input signal. For input signals with DC components, soft sources can lead to *solution-growth*, in which the DC level of the entire simulation increases exponentially over time. This is not to do with numerical error or stability. Rather, the DC component suggests that the input signal is of infinite length, and that the source continues to create volume velocity for the duration of the simulation [65]. That is, the model is valid, but it is being used to model a physically-implausible situation. Solution-growth is unacceptable for modelling purposes. In all cases it creates an increasing DC component which is difficult to remove from the output. It also leads to a loss of precision in floating-point simulations, as intervals between floating-point numbers are larger for higher-magnitude numbers. The low-magnitude content of interest cannot be specified with high precision when it is superimposed on a high-magnitude DC component. In the worst case, the DC component will build up so much that the numerical representation "overflows", placing an upper bound on the duration of the simulation. Some of the drawbacks of the soft source can be alleviated by carefully constructing the input signal, which will be discussed later.

Solution growth is not seen in hard source models, because the source node pressure is replaced by that of the input function. Therefore, at the source node, there is no addition of DC level. Unfortunately, rarefaction cannot occur at the source node, which can cause low-frequency oscillations instead. These oscillations are easier to remove than an increasing DC offset, and have less impact on precision.

A special case of the soft source is the *transparent source*, described in [64]. This method is the same as the soft source, but additionally subtracts the convolution of the mesh impulse response with the input signal from the input node. The resulting wave-front has the same characteristics as that produced by a hard source, with the additional benefit that incident wave-fronts are not scattered. The drawback of this technique is that the mesh impulse-response must be precomputed before the simulation can begin, which is achieved by running a separate (costly) waveguide mesh simulation. Once found, the mesh response can be used for any other simulation sharing the same mesh topology. Although the transparent source is complex to set up, its characteristics seem perfect. It behaves as if the input signal is perfectly coupled to the mesh, and there is no scattering problem. However, it retains the solution-growth issue of the general soft source.

The hard and soft source methods can be combined, in order to benefit from the characteristics of both methods. A *time-limited pulse* method is introduced in [58], in which the source node starts as a hard source, and reverts to a soft source after a certain period of time has elapsed. The input signal is ideally coupled to the mesh for its duration, but thereafter the mesh can update as usual. As long as the source is positioned away from reflective boundaries, this solves the scattering issue. However, if the source is near to a reflective boundary, reflected wave-fronts might reach the source node before it has reverted to the standard update equation, scattering them. This input scheme has similar performance to the transparent source, with much reduced complexity, but it also shares a major drawback. If the input signal has a DC component, the time-limited hard source can still cause solution-growth [65]. In addition, if the nodes next to the source are not zero when the update equation switches from hard to soft, the pressure discontinuity introduced may introduce error which will propagate for the remainder of the simulation [61].

On balance, it seems as though the transparent source is the optimum input method. Once the mesh impulse response has been computed, the update method itself is simple to implement, and does not carry any performance penalties. The only disadvantage is that the input function must be carefully designed to have no DC component, to avoid solution-growth. Unfortunately, overcoming this weakness is somewhat difficult, as will be shown in the following subsection.

5.2.2.2 Input Signal

Of course, the input signal must be bounded in time, so that the simulation can end. The shorter the input signal, the fewer simulation steps must be computed, so shorter inputs are more efficient. If a single-sample Dirac-delta is used, all frequencies in the output bandwidth will be excited equally, and the output will not require deconvolution, which is another computational saving. It is plain that a short impulsive input is desirable in terms of efficiency. Unfortunately, impulsive signals have a DC component, which can cause solution-growth with soft, particularly transparent, sources.

The constraints of the input signal are, therefore, as follows: The signal shall have no DC component. It shall be as short as possible, to minimise the simulation time. It shall have a wide and flat passband, to increase the accuracy of the deconvolution. The time-compactness and bandwidth constraints are mutually exclusive, so in practice the input length must be chosen to balance bandwidth and calculation time.

Particular constraints of source signals are presented in greater detail in [61], which puts forward an additional constraint, known as the *differentiation constraint*: The input signal shall be equal to the first time derivative of fluid emergence. Fluid emergence should start and end at zero, which in turn enforces a null DC component.

Some particular possibilities for the input signal are the *sine-modulated Gaussian pulse* [58], and the *differentiated Gaussian pulse* and *Ricker wavelet* [61]. All of these signals satisfy the differentiation constraint and the length constraint. However, they all have non-flat pass-bands, as shown in fig. 5.4. A final option is the *physically constrained source* (PCS) model presented in [61]. This method can be used to create input signals with pass-bands much flatter than those of the more conventional pulse

and wavelet signals. PCS signals obey the differentiation constraint, have wide and flat passbands, and are short in time. They use soft-source injection, so will not cause scattering artefacts, and as they have no DC component, they should not introduce solution-growth. A PCS input signal seems like an obvious choice for this application.

A test was devised to ensure that the source injection method did not cause solution-growth. A standard rectilinear waveguide mesh with a sampling frequency of 10kHz was set up within a cuboid room, measuring $5.56 \times 3.97 \times 2.81$ metres. A source was placed at (4.8, 2.18, 2.12), and a receiver at (4.7, 2.08, 2.02). The walls of the room were set to have a uniform broadband absorption of 0.006 (see the Boundary Modelling section). "Transparent" input signals were created from a differentiated Gaussian pulse, a sine-modulated Gaussian pulse, and a Ricker wavelet, all of which were set to a centre frequency of 0.05 f_s. A physically-constrained source signal was also generated, using parameters suggested in section V of [61]: a max-flat finite-impulse-response pulse-shaping filter kernel with 16 taps and centre frequency of 0.075 f_s and magnitude 250N was generated; it was passed through a mechanical shaping filter with radius 5cm, mass 25g, lower cutoff 100Hz, and resonance 0.7; then this signal was passed through an infinite-impulse-response *injection filter*, and used as a soft source. (These parameters are reproduced here to ensure that the test is repeatable, but a full discussion of their meaning is beyond the scope of this paper. The interested reader is directed to [61] or to the implementation of the physically-constrained source in the Wayverb repository.) Finally, the simulation was run for around 85,000 steps (less than the expected Sabine RT60 of the room) with each of the four sources, and the response at the receiver was recorded.

The results of the experiment are shown in fig. 5.5. The response of a transparent Dirac source (which has a strong DC component) is also shown. Solution growth can be seen in all the outputs. However, the magnitude of growth is different depending on the input signal. As expected, the Dirac signal exhibits the largest rate of growth, followed by the differentiated Gaussian, sine-modulated Gaussian, Ricker wavelet, and finally the PCS signal. All the sources with no DC component show significantly less solution-growth than the transparent Dirac source. The PCS has a much lower rate of growth than the alternatives. However, all inputs *do* show the effects of solution-growth.

The solution-growth seen here only becomes prominent towards the end of the simulation, after around 60,000 steps. However, papers which propose countermeasures to the solution-growth problem generally only test their solutions up to 15,000 steps or so [58], [61], [65]. The results of testing the solution in [63] are not even presented. However, the experiments in [66] are run for 100,000 steps, exhibiting similar DC instability to the results presented here.

The reason for the solution-growth is not clear. In general, the problem is caused by repeated superposition of the DC level, which is reflected from boundaries. The waveguide has no inherent way of removing DC, so *any* small DC component will accumulate over time. If the original signal does not have a DC component, then the DC is being added from elsewhere. The most likely origin is numerical error in the waveguide mesh. The experiment above uses 32-bit single-precision floating-point to represent the pressure of mesh nodes, which is necessary because using double-precision would double the memory usage and halve the computational throughput. It is possible that error in these single-precision calculations manifests as a tiny DC component, which then multiplies as the simulation progresses.
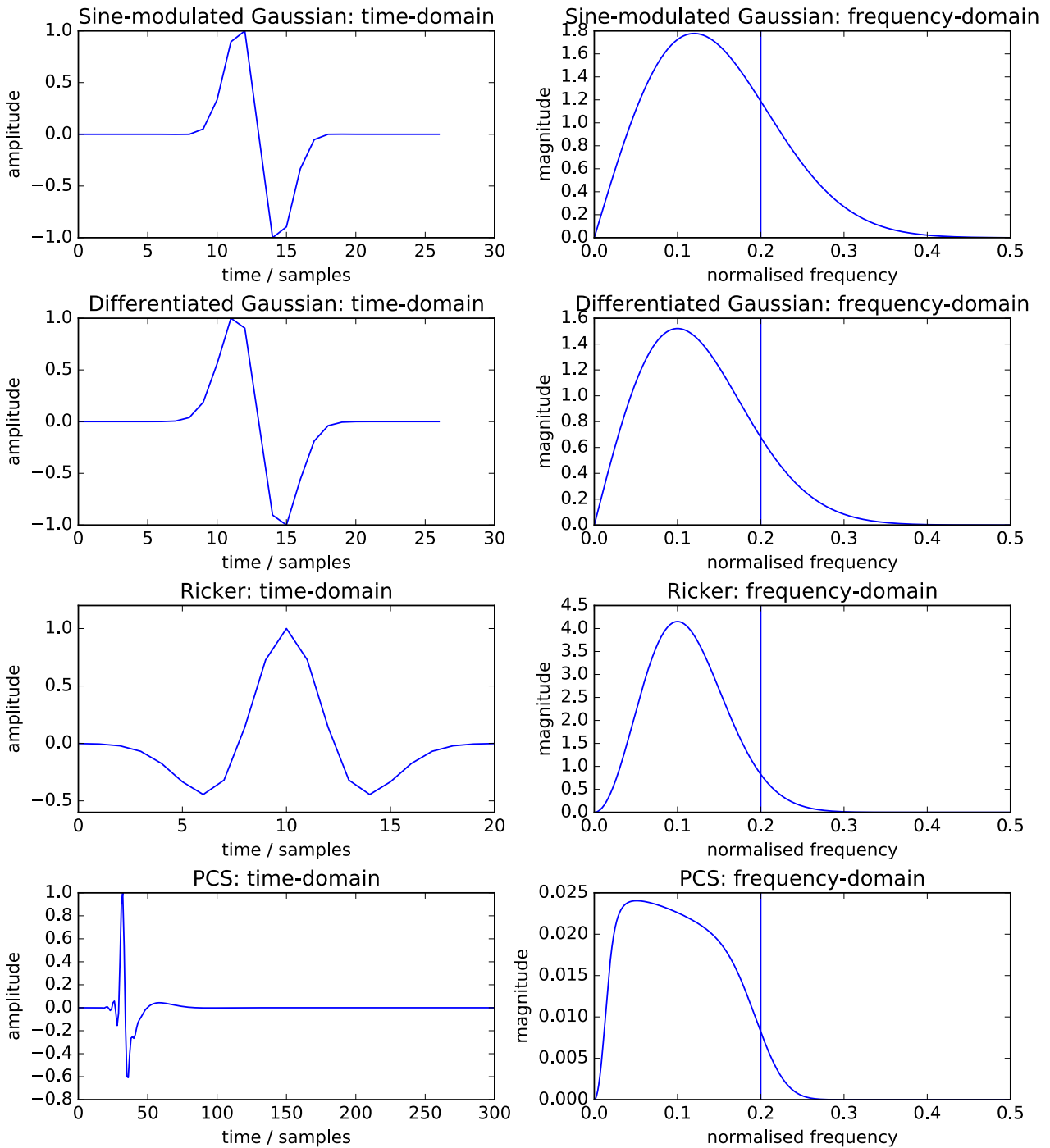
Figure 5.4: The time-domain and frequency-domain responses of some signals commonly used as FDTD excitations. All signals are shown with an upper cutoff of $0.2f_s$. The pulse signals have their centre frequencies set to $0.1f_s$. The PCS signal shown has a sampling rate of 10kHz, a mass of 25g, a low cutoff of 100Hz, and a Q of 0.7. It *includes* the injection filter, which means the signal shown could be injected like a soft source.
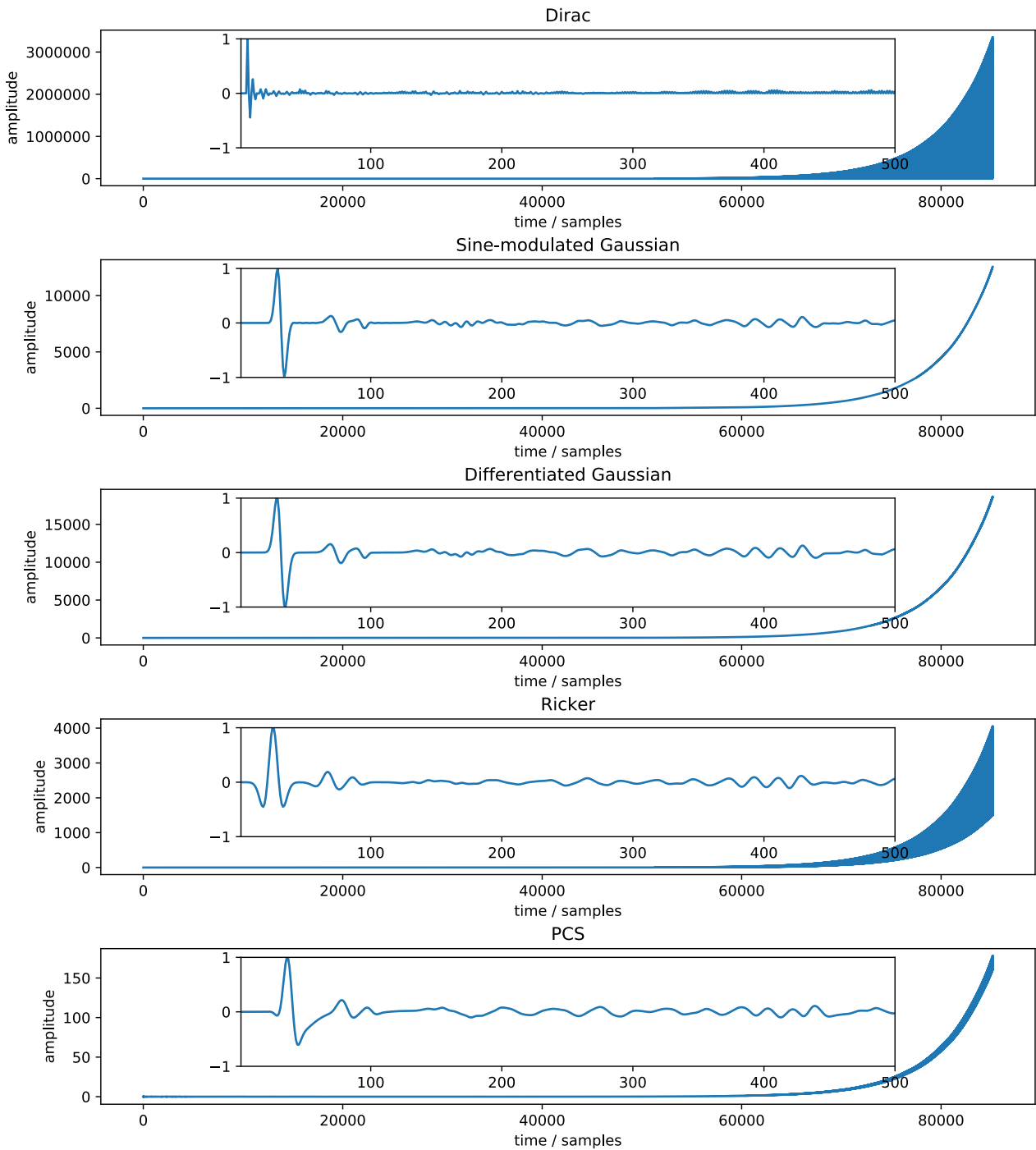
Figure 5.5: Solution growth in the waveguide mesh with a selection of different inputs. Results are normalized so that the initial wave-fronts have the same magnitude. The overlays show the initial 500 samples of the response on the same scale, highlighting the different shapes of the excitation signals. The full signals are shown behind, with *different* amplitude scales. In the Dirac, Ricker, and PCS examples, the graphs appear "filled-in" due to high-frequency oscillations which increase in magnitude over time.

The theory that low numerical-precision leads to solution-growth is reinforced by evidence given in [66]. This study shows that numerical error can perturb otherwise-stable simulations, causing gradual amplification. The solution proposed there is to conduct the entire simulation using double-precision floating-point, which would effectively delay rather than remove the effects of DC instability. As explained above, this approach is not practical in Wayverb.

Whatever the reason, it is clear that using a soft source generally causes a DC offset to accumulate, even when the input signal has no DC component. Soft sources may be suitable for shorter simulations. However, without running the simulation, it is impossible to know how much DC will accumulate, and whether or not it will remain within acceptable bounds. Therefore, in general, current forms of the soft source are not appropriate for arbitrary room simulation.

As an alternative to a soft source, Wayverb currently uses a hard source with a Dirac impulse as its input method. Though this still causes low-frequency error [61], this error manifests as an oscillation rather than an exponential growth. The oscillation tends to be below the audible range, and therefore can be removed without affecting the perceived quality of the simulation. This removal is achieved with no phase modifications by transforming the signal into the frequency domain, smoothly attenuating the lowest frequencies, and then converting back to the time domain. This is the same filtering process used throughout Wayverb, described in detail in the Ray Tracer section. The main drawback of the hard source is its scattering characteristic. Though undesirable, this behaviour has a physical analogue: in a physical recording, reflected wave-fronts would be scattered from the speaker cabinet or starter pistol used to excite the space.

The creation of soft sources which do not cause solution growth is an important area for future research. Code for modelling soft sources remains in the Wayverb repository, to ease further development work in this area. If a better excitation method is discovered, it will be easy to replace the current source model with the improved one.

5.3 Implementation

Here, the final waveguide, as implemented in Wayverb, is described. Room acoustics papers tend not to discuss the set-up process for 3D waveguide meshes, so this process will be described in detail here.

5.3.1 Inner and Outer Nodes

Prerequisites for the simulation are: a 3D scene, made up of triangles, each of which has multi-band absorption coefficients; a source and receiver position within the scene; the speed of sound c and acoustic impedance of air Z_0, and the sampling frequency of the mesh f_s. This sampling frequency may be derived from a maximum cutoff frequency and an oversampling coefficient.

The first step is to calculate the position of each node in the mesh. The inter-nodal spacing X (that is, the spatial sampling period) is given by

$$X = \frac{c}{f_s \lambda} = \frac{c\sqrt{3}}{f_s} \tag{5.8}$$

where λ is the Courant number, set to its maximum stable value. Now, the axis-aligned bounding box of the scene is found, and padded to exact multiples of the grid spacing along all axes. The exact padding is chosen so that one node will fall exactly at the receiver position, and so that there is room for an "outer layer" at least two nodes deep around the scene. This outer padding is to accommodate for boundary nodes, which will always be quantised to positions just outside the modelled enclosure. If the new padded bounding box has minimum and maximum corners at 3D points c_0 and c_1, then the position of the node with integer indices (i, j, k) is given by $c_0 + X(i, j, k)$. The number of nodes in each direction is given by $\frac{c_1 - c_0}{X}$. The actual node positions are never computed and stored, because this would take a lot of memory. Instead, because the calculation is so cheap, they are recomputed from the node index, bounding box, and mesh spacing whenever they are needed.

Each node position must be checked, to determine whether it falls inside or outside the scene. The algorithm for checking whether a node is inside or outside is conceptually very simple: Follow a ray from the node position in a random direction chosen from a uniform distribution, until it leaves the scene bounding box. If the ray intersects with an odd number of surfaces, the point is inside; otherwise it is outside. There is an important special case to consider. Floating-point math is imprecise, so rays which "graze" the edge of a triangle may be falsely reported as intersecting or not-intersecting. This is especially problematic if the ray intersects an edge between two triangles, in which case zero, one, or two intersections may be registered. To solve this problem, the intersection test can return three states instead of two ("uncertain" as well as "definite intersection" and "definitely no intersection"). If the ray grazes any triangle, then "uncertain" is returned, and a new random ray is fired. The process then repeats until a ray with no grazing intersections is found. Note that this algorithm relies on ray-casting, which means that it can be accelerated using the voxel-based method discussed in the Image Source section. All tests are carried out in parallel on the GPU, and the results are stored.

5.3.2 Boundary Node Classification

The previous section described how to classify the "inner" nodes of an enclosure. However, the remaining nodes are not all "outside" the simulation: some are "boundary" nodes (see Boundary Modelling). These boundary nodes must be found and classified.

Boundary nodes fall into three main categories, shown in fig. 5.6:

- **1D** nodes are situated directly adjacent to a single inner node in one of the six axial directions.
- **2D** nodes are next to a single inner node in one of the twelve on-axis diagonal directions.
- **3D** nodes are next to a single inner node in one of the eight off-axis diagonal directions.

There is also a fourth category, known as *re-entrant* nodes, which are adjacent to two or more inner nodes. These nodes are special, in that they fall outside the enclosure (like boundary nodes) but are updated using the standard inner node equation. Re-entrant nodes are generally found on corners which face "into" the enclosed space.

The classification proceeds as follows: For a given node, if it is inside, return. Otherwise, check the node's six axial neighbours. If one neighbour is inside, the node is a 1D boundary; if two neighbours are inside, the node is re-entrant; if no neighbours are inside the node remains unclassified. If the node is unclassified, check the twelve on-axis diagonal neighbours. If one neighbour is inside, the node is a 2D boundary; if two neighbours are inside, the node is re-entrant; if no neighbours are inside the node
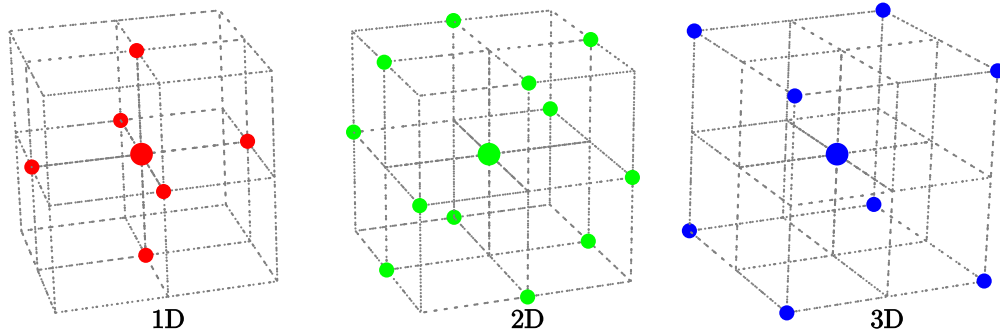
Figure 5.6: A given node (represented by a large dot) is a boundary node if it is *not* an inner node, but there is an adjacent inner node at one of the locations shown by smaller dots.

is still unclassified. Finally, check the eight off-axis diagonal neighbours. If one neighbour is inside, the node is a 3D boundary; if two neighbours are inside, the node is re-entrant; if no neighbours are inside, the node is an "outer" node and can be ignored for the remainder of the simulation. This classification process can of course be run in parallel on the GPU.

The classification is a little too involved to recompute regularly, so the results of the classification are cached. Each node stores its characteristics into a integer which behaves as a bitfield, allowing the characteristics to be stored in a compact form. Bits have the following significance:

```
typedef enum : cl_int {
    id_none = 0,
    id_inside = 1 << 0,
    id_nx = 1 << 1,
    id_px = 1 << 2,
    id_ny = 1 << 3,
    id_py = 1 << 4,
    id_nz = 1 << 5,
    id_pz = 1 << 6,
    id_reentrant = 1 << 7,
} node_type;
```

The descriptor field for an inner node will be set to `id_inside`, and for a re-entrant node will be set to `id_reentrant`. Boundary nodes are described by setting bits equal to the direction of the adjacent inner node. A 1D node with an inner neighbour in the negative-x direction will have the descriptor `id_nx`, a 2D node with an inner neighbour on the positive-y-z diagonal will have the descriptor `id_py | id_pz` (where | is a bitwise-or operator), and a 3D node with an inner neighbour on the positive-x, negative-y, positive-z diagonal will have the descriptor `id_px | id_ny | id_pz`.

5.3.3 Boundary Behaviour and Materials

Each boundary node behaves as if it has an internal IIR filter. More precisely, 1D nodes have a single internal IIR filter, while 2D and 3D nodes have two and three internal filters respectively. To operate, these filters must reference filter coefficients which approximate the wall behaviour, and must also have dedicated storage for their filter delay lines.

First, the numbers of 1D, 2D, and 3D boundary nodes are counted. An array of single-filter delay lines is created, containing one delay line for each 1D node. The same is done for the 2D and 3D nodes, but each element in these arrays has storage for two and three filter delay lines respectively. Now, each boundary node is given a unique index which is used to reference an element in its corresponding filter memory array. These unique indices are simple to compute: for each node in the simulation, if it is 1D (or 2D, or 3D), increment a counter, and use the counter value as the unique index.
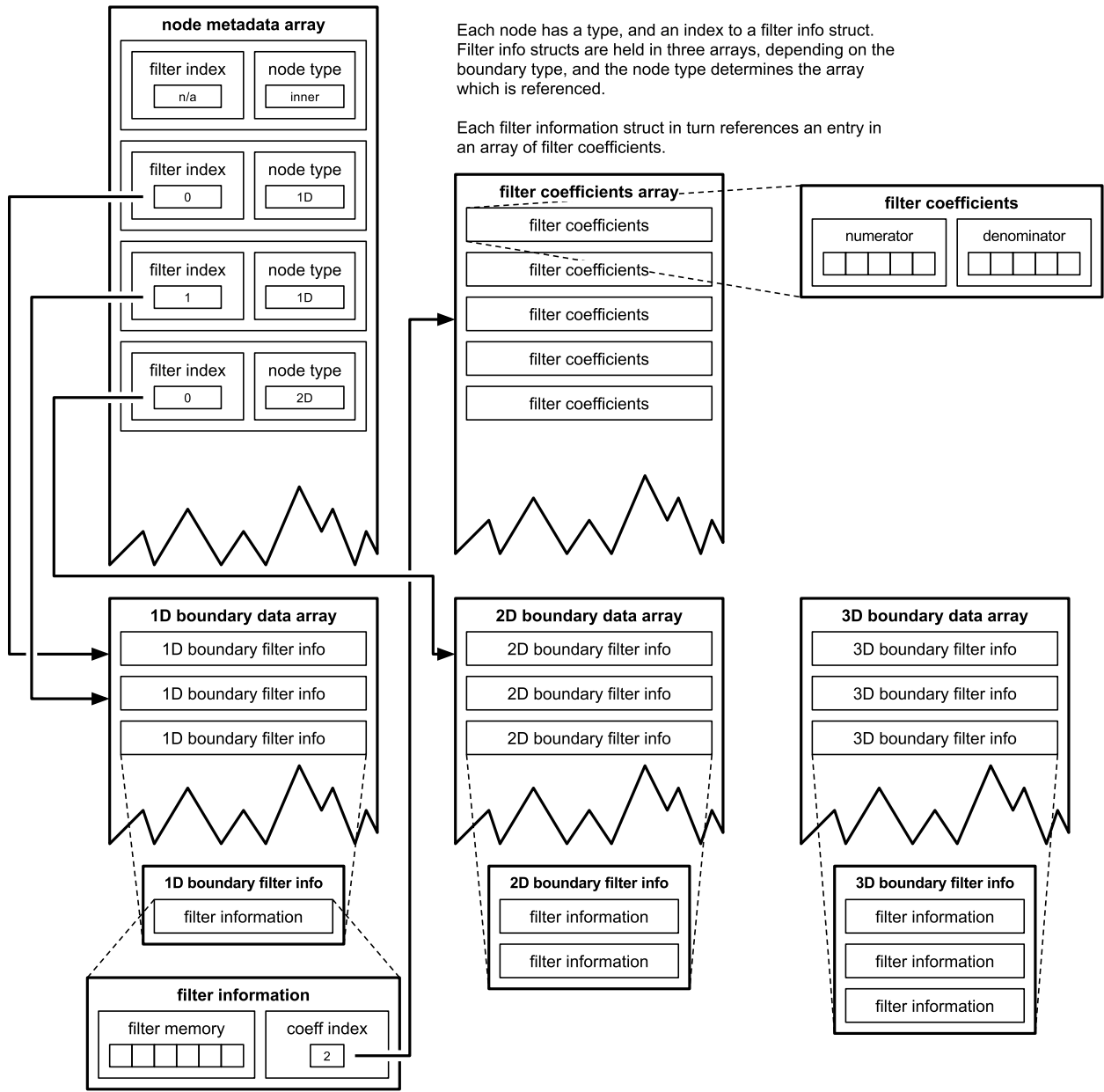
For each material in the scene, the Yule-Walker method is used to generate an IIR filter representing that material, resulting in an array of IIR filter coefficients. Each filter delay line is paired with an index field, which allows it to reference the filter coefficients which should be used when updating the filter.

The final step is to find which filter coefficients should be linked to which filter delay line. For 1D boundaries, the process is as follows: find the closest triangle to the node; find the material index of that triangle; get the node's filter data entry; set the coefficient index field to be equal to the closest triangle's material index. For 2D boundaries, adjacent 1D boundary nodes are checked, and their filter coefficient indices are used, which saves running further closest-triangle tests. For 3D boundaries, adjacent 1D *and* 2D nodes are checked.

At this point, all information required to run the simulation has been computed. Simulations may contain many millions of nodes, each with associated metadata, so this information must be stored in a way which minimises redundancy whilst still allowing for efficient queries. Wayverb's memory layout scheme is shown in fig. 5.7.

5.3.4 Running the Simulation

With node properties and boundary information set up, all that remains is to run the simulation itself. Two arrays of floating-point numbers are allocated, with length equal to the number of nodes in the simulation. These arrays represent the current and previous pressures at each node. The simulation is then run for a certain number of steps. In Wayverb the simulation length is found using the time of the final ray-traced histogram bin, divided by the mesh sampling frequency. In most cases this will lead to a sufficiently accurate duration estimate. However, in very large or irregularly-shaped rooms where few ray-receiver intersections are recorded, this may lead the waveguide simulation time to be underestimated. In this scenario, the simulation time can be found with greater accuracy by increasing the number of rays.

Figure 5.7: Efficient memory usage is important in large-scale simulations such as those conducted by Wayverb. The waveguide storage scheme aims to minimise redundant duplication of data.

During a single step of the simulation, each node is updated. These updates occur in parallel, using the GPU. The "descriptor" field of each node is checked, and if it is `id_none` then the node is ignored. If the descriptor is `id_inside` or `id_reentrant` then the node is updated like a normal air node; that is, the "next" pressure of the node is equal to the sum of current axially-adjacent pressures divided by three, minus the previous pressure of the node. This is shown in the following equation, where i, j, and k are spatial indices on the x, y and z axes respectively, and n is a time index.

$$p_{i,j,k}^{n+1} = \frac{1}{3}(p_{i-1,j,k}^n + p_{i+1,j,k}^n + p_{i,j-1,k}^n + p_{i,j+1,k}^n + p_{i,j,k-1}^n + p_{i,j,k+1}^n) - p_{i,j,k}^{n-1} \tag{5.9}$$

If the node is a boundary node, then it is instead updated according to the boundary update equations found in [67].

The update equation references three points in time (n, and $n \pm 1$), which suggests that three arrays of node pressures required. That is, during update, the "previous" and "current" pressures are read from two arrays, and used to compute a value for the "next" pressure, which is then stored to a third array. A useful property of the update equations is that the "previous" pressure of each node is *only* used when updating that node. This means that the result of the update can be written back to the "previous" pressure array, instead of being written to an extra "next" array, which is a significant memory saving. For the following step of the simulation, the "current" and "previous" arrays are swapped. If the arrays are referenced through pointers, then this can be achieved by just swapping the pointers, which is much faster than swapping the actual array contents.

The simulation inputs and outputs are handled using generic callbacks. Before each step, a reference to the "current" pressure array is passed to a preprocessor callback, which may modify the pressure at any node. This architecture allows different source types to be implemented and swapped very easily. For example, hard and soft sources are just two different types of stateful callback. This also gives the option of excitations which span several nodes on the mesh. At the end of the step, the "current" pressure array is passed to the post-processor callback, which in general will append the value of a single node to an array, which can be retrieved at the end of the simulation. Again, the architecture is flexible, in that it allows for different receiver types, such as those discussed in Microphone Modelling.

5.4 Summary

A derivation for the waveguide mesh update formulae has been presented, along with the equations which relate the spatial sampling frequency, temporal sampling frequency, speed of sound, and number of dimensions. The K-DWM has been shown to be faster and more memory-efficient than the W-DWM, although both formulations exhibit dispersion error which limits the usable bandwidth of simulation results.

Different mesh topologies have been compared. The tetrahedral topology has the lowest density and fewest inter-node connections, so it is the optimal choice in terms of speed and memory consumption. Interpolated meshes are the most accurate and isotropic, but are more complicated to implement. For Wayverb, the rectilinear mesh was chosen, due to its relative simplicity, and because it was the only topology for which boundary formulations already existed.

Mesh input methods have been explored, and it has been shown that solution growth places a major design constraint upon mesh inputs. The literature suggests that the PCS is the optimal input signal. However, tests of five different non-hard inputs (including a PCS) showed that they all lead to solution growth, likely because of accumulated numerical error. The presence of solution growth means that pressure values may eventually overflow, invalidating results. To avoid this eventuality, Wayverb must use a hard source.

A simple procedure for building space-filling waveguide meshes with classified boundaries in arbitrary watertight manifold meshes has been presented. This is a novel contribution. Details relating to the efficient implementation of a DWM on graphics hardware have also been provided.

As shown, the DWM in Wayverb is obviously sub-optimal. Its topology is neither the fastest nor the most accurate, and its hard source causes scattering artefacts. Using a more efficient mesh topology would have required formulating an entirely new boundary modelling scheme, and using a soft source would have required development of a novel injection method. Such research, though useful, would have run counter to the specific research aims of this project. As noted in the Context chapter, the purpose of Wayverb is primarily to create a useful tool for musicians, by surveying and combining existing techniques for room simulation. Further research into these issues, though important, would have precluded the completion of a usable simulation program. Of course, future research may reveal solutions to the problems identified over the course of this project. Improvements to Wayverb could then be made, based on these solutions.

6 Hybrid Model

6.1 Background

The previous sections have looked at the theory and implementation of three different acoustic simulation techniques. The image-source method is accurate for early reflections, but slow for longer responses. The ray tracing method is inaccurate, but produces acceptable responses for "diffuse" reverb tails. The waveguide method models physical phenomena better than the geometric methods, but is expensive at high frequencies. By combining all three models, accurate broadband impulse responses can be created, but for a much lower computational cost than would be possible with any individual method.

This section will focus on the two most important factors governing the combination of modelling techniques. The first is positioning transitions: in the time domain, from early to late reflections; and in the frequency domain, between geometric and waveguide modelling. The second is matching the output levels of the different methods, so that there are no sudden discontinuities in level, and the transitions are seamless.

6.2 Transitions

6.2.1 Early and Late Reflections

The beginning of the image-source process relies on randomly ray tracing a certain number of reflections. This ray tracing process is similar to that used for estimating late, diffuse reflections. When the simulation is run in Wayverb, rays are actually traced to a depth of 100 reflections or more. The first few reflections are routed to image-source processing, while the entire set of reflections is used for finding the reverb tail.

It is important to note that the stochastic ray tracing process will record both specular and diffuse reflections. At the beginning of the impulse response, this will lead to a duplication of energy, as the energy from specular reflections will be recorded by both the image-source and ray-tracing processes. To solve this problem, the stochastic ray tracer records specular and diffuse contributions separately. Specular contributions from the ray tracer are only added to the output for reflections of higher order than the highest image-source order.

Surface scattering also poses a problem. When simulating scenes with high surface scattering coefficients, specular reflections should be quiet, with a greater degree of scattered energy. Unfortunately, the image-source process cannot account for scattered sound energy by design. The solution is to use diffuse contributions from the stochastic ray tracer, so that the image-source and ray-traced outputs "overlap". To ensure that the amount of energy in the simulation remains constant, the image-source finder must

account for energy lost to scattering during reflections. Therefore, after finding the reflectance of each surface using the method outlined above, the reflectance is further multiplied by $(1 - s)$ where s is the frequency-dependent scattering coefficient of the surface. This causes the image-source contributions to die away faster, and the "missing" energy will be made up by the diffuse output of the ray tracer.

The transition between the image-source and ray tracing models will generally overlap. The image-source response will fade away as the ray-traced diffuse reflections become louder. The exact number of early reflections to be found with the image-source method is largely a subjective decision. For diffuse rooms, the early specular reflections will be very quiet, regardless of which method is used, so it is appropriate to set the number of specular reflections very low, or even to disable image-source contributions altogether. For rooms with surfaces which are large, smooth, and flat, specular reflections will form a more significant part of the room response, and so it is reasonable to use deeper image-source reflections in this case. Even under these conditions, an image-source depth of more than 5 or 6 is unnecessary: in virtually all scenes, some incident sound energy will be scattered diffusely, and the conversion of "specular energy" into "scattered energy" is unidirectional, meaning that late reflections in all scenes will be diffuse, and therefore suitable for simulation with stochastic ray-tracing methods [33, p. 126].

6.2.2 Crossover Position

In the interests of efficiency, the most accurate waveguide method is only used at low frequencies, where it is relatively cheap. The rest of the audible spectrum is modelled with geometric methods, which are most accurate at higher frequencies. However, there is no concrete rule governing where to place the crossover between "low" and "high" frequencies in this context. It should be clear that, when the time and computing power is available, the cutoff should be placed as high as possible, so as to use accurate wave-based modelling for the majority of the output. However, in practice, it might be useful to have an estimate for the frequencies where wave-modelling is strictly required, to guide the placement of the cutoff frequency. The *Schroeder frequency* is such an estimate, and is based on the density of room resonances or "modes". Below the Schroeder frequency, room modes are well separated and can be individually excited. Above, the room modes overlap much more, resulting in a more "even" and less distinctly resonant sound. The Schroeder frequency is defined as follows (see [33, p. 84] for a detailed derivation):

$$2000\sqrt{\frac{RT60}{V}} \qquad (6.1)$$

Here, $RT60$ is the time taken for the reverb tail to decay by 60dB, and V is the room volume in cubic metres. In the Waveguide chapter, the complexity of a waveguide simulation was given as $O(Vf_s^3)$ for a space with volume V, at sampling frequency f_s. Given that f_s is proportional to the waveguide cutoff frequency, which in turn may be set proportional to $V^{-\frac{1}{2}}$ by eq. 6.1, the waveguide simulation complexity becomes $O(V \cdot V^{-\frac{3}{2}})$ or $O(V^{-\frac{1}{2}})$. This implies that, for a waveguide simulation capped at the Schroeder frequency, a larger room will be cheaper to simulate than a smaller one when the reverb times of both rooms are equal.

The Schroeder frequency is only an estimate. The actual frequency dividing "resonant" and "even" behaviours will vary depending on the surface area, absorption coefficients, and shape of the modelled space. The optimum crossover frequency should also be guided by the accuracy and time constraints imposed by the user. For this reason, the Schroeder frequency is not used to guide the placement of the crossover frequency in Wayverb. Instead, the user may select the maximum frequency modelled by the waveguide, along with an oversampling ratio. In this way, the user can use the waveguide to model as wide a bandwidth as their time constraints allow. The literature indicates that this is a valid approach: [26] suggests that "under ideal conditions the FDTD method would compute the entire RIR in all bands to ensure physical accuracy", and [1] notes that further research is required in order to find an objective method for placing the crossover. Finally, the systems described in [24] and [25] allow arbitrary placement of the crossover frequency.

6.2.3 Combining Outputs

Once the geometric and waveguide outputs have been produced, they must be combined into a single signal. This combination process requires that the geometric and waveguide outputs have the same sampling frequency. However, the waveguide sampling frequency will almost certainly be lower than the final output sampling frequency, so the waveguide results must be up-sampled. Wayverb uses the Libsamplerate library for this purpose. The sampling rate conversion preserves the signal magnitude, but not its energy level. The re-sampled waveguide output is therefore scaled by a factor of $f_{s\text{in}}/f_{s\text{out}}$ (where $f_{s\text{in}}$ is the waveguide sampling rate, and $f_{s\text{out}}$ is the output sampling rate), so that the correct energy level is maintained.

Once the waveguide sampling rate has been corrected, the waveguide and geometric outputs are filtered and mixed. The filtering is carried out using the frequency-domain filtering method described in the Ray Tracing section. The waveguide is low-passed, and the geometric outputs are high-passed, using the same centre frequency and crossover width in both cases. The final output is produced by summing the filtered responses.

6.3 Level Matching

6.3.1 Image-Source and Ray Tracer

The Image Source model operates in terms of pressure. This means that the pressure contribution of each individual image is inversely proportional to the distance between that image source and the receiver. In contrast, the Ray Tracing method operates in terms of acoustic intensity, and the total intensity measured at the receiver depends only on the number of rays which intersect with it. The distance travelled by each ray is not taken into account.

A method for equalising the output of the two models is given in [35, p. 75]. The goal of the method is to ensure that the energy of the direct contribution (the wave-front travelling directly from source to receiver, with no intermediate reflections) is equal between the two models.

First, equations for the intensity at the receiver must be created. Given a source and receiver separated by distance r, the intensity of the direct image-source contribution is given by:

$$E_{\text{image source}} = \frac{E_{\text{source}}}{4\pi r^2} \tag{6.2}$$

This is the standard equation for describing the power radiated from a point source.

For the ray tracing method, the intensity of the direct contribution is a function of the number of rays N, and the intensity of each ray E_r. Only rays intersecting the receiver will be registered, so ray intensity must be normalised taking into account the proportion of rays which will intersect the receiver. For a spherical receiver, and uniformly distributed rays, the proportion of rays which intersect the receiver can be estimated as the ratio between the area covered by the receiver, and the total area over which the rays are distributed. If the receiver is at a distance r from the source, with an opening angle γ, then its area is that of a spherical cap (see fig. 6.1):

$$A_{\text{intersection}} = 2\pi r^2 (1 - \cos\gamma) \tag{6.3}$$

Then, the total direct energy registered by the ray tracer can be expressed:

$$\begin{aligned} E_{\text{ray tracer}} &= N E_r \left(\frac{A_{\text{intersection}}}{4\pi r^2} \right) \\ &= N E_r \left(\frac{2\pi r^2 (1 - \cos\gamma)}{4\pi r^2} \right) \\ &= N E_r \left(\frac{1 - \cos\gamma}{2} \right) \end{aligned} \tag{6.4}$$

The direct energy in both models should be equal, so the two equations can be set equal to one another, giving an equation for the initial intensity of each ray, in terms of the source intensity E_{source}, the opening angle of the receiver γ, and the number of rays N.

$$\begin{aligned} E_{\text{ray tracer}} &= E_{\text{image source}} \\ N E_r \left(\frac{1 - \cos\gamma}{2} \right) &= \frac{E_{\text{source}}}{4\pi r^2} \\ E_r &= \frac{E_{\text{source}}}{2\pi r^2 N (1 - \cos\gamma)} \end{aligned} \tag{6.5}$$

As long as the initial ray intensities are set according to this equation, both methods will produce outputs with the correct relative levels. The outputs from the two methods are combined by simple signal addition, with no need for additional level adjustment.

6.3.2 Geometric and Waveguide

Setting the waveguide mesh level should follow a similar procedure. The waveguide output level should be normalised so that the direct intensity observed at the receiver is equal to that observed in the image-source model. The normalisation is achieved through the use of a *calibration coefficient* which can be used to scale the waveguide output, or alternatively to adjust the magnitude of the input signal.
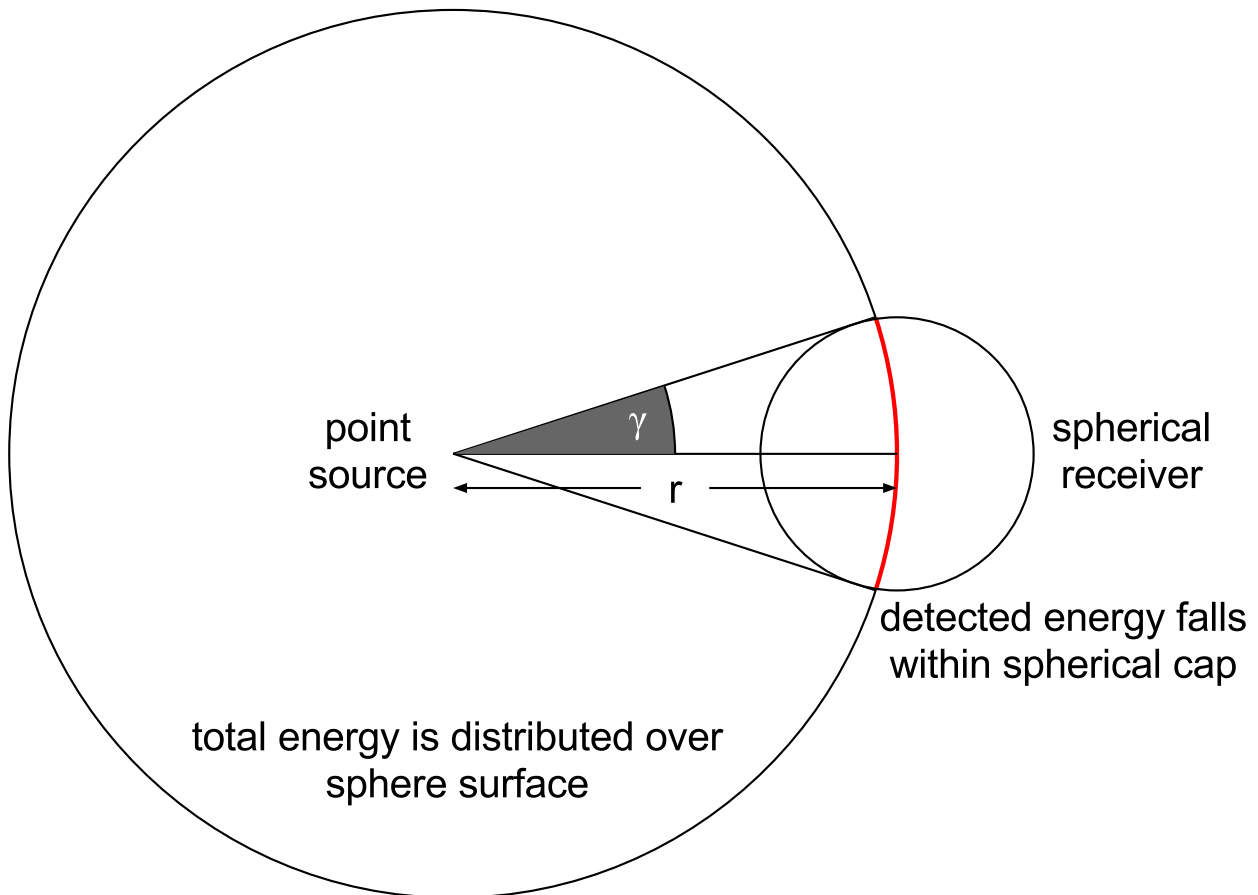
Figure 6.1: The proportion of uniformly-distributed rays intersecting the receiver depends on the distance to the source, and opening angle formed by the receiver. The acoustic intensity registered by the ray tracer is given by the number of rays which intersect the receiver, and the energy carried by each ray.

Several suggestions for finding the calibration coefficient are suggested in the literature. One option, which is perhaps the simplest (short of manual calibration), is found in [1]. The average magnitude of a certain frequency band is calculated for both the waveguide and geometric output. The calibration coefficient is then equal to the ratio of these two magnitudes. This approach is flawed, in that the same frequency band must be used for both signals. This frequency band will therefore be towards the lower end of the geometric output, which is known to be inaccurate (this is the entire reason for hybrid modelling), and at the top end of the waveguide output (which may be inaccurate due to numerical dispersion). It is impossible to compute an accurate calibration coefficient from inaccurate data.

Another method, suggested in the same paper [1], is to find the intensity produced by the waveguide at a distance of 1m, and then to normalise both models so that they produce the same intensity at this distance. The image-source direct response is low-pass filtered, so that it only contains energy in the same frequency band as the waveguide output. Then, the maximum magnitude in the initial portion (up to the first reflection) of the output signal is found, for the image-source and waveguide output. The calibration parameter is found by taking the ratio of these maximum magnitudes. This differs from the first technique, in that the calibration coefficient is derived from a single contribution in the time-domain, instead of from frequency-domain magnitudes accumulated over the entire duration of the signal.

The second method is superior, in that it will produce an accurate calibration coefficient. Unfortunately, it requires analysis of the waveguide output signal which, while not time-consuming, seems unnecessary. A given mesh topology and excitation signal should always require the same calibration coefficient, assuming that the geometric source level remains constant. It should be possible to calculate the calibration coefficient for a certain mesh topology, and then to re-use this coefficient across simulations. This is the approach taken in [68] which provides Wayverb's calibration method.

This general calibration coefficient is found by exciting a waveguide mesh with an impulsive signal, and recording the pressure at a receiver node immediately adjacent to the source node. The simulation continues until the magnitude of vibrations at the receiver have reduced below some threshold (perhaps falling below the noise floor). Now, the change in pressure at a distance X is known, where X is the inter-nodal spacing of the waveguide mesh. The geometric pressure level at the same distance is given by

$$p_g = \sqrt{\frac{PZ_0}{4\pi}}/X \tag{6.6}$$

where P is the source strength and Z_0 is the acoustic impedance of air. The waveguide pressure level cannot be directly compared to the geometric pressure level, because the upper portion of the waveguide output frequency range is invalid. Instead, the DC levels are compared. The DC component of the waveguide output can be found simply by accumulating the signal at the receiver. Now, the calibration coefficient η can be expressed like so:

$$\eta = \frac{p_{\text{init}}R}{p_{\text{DC}}X} \tag{6.7}$$

where p_{init} and p_{DC} are the initial and DC pressure levels respectively, X is the inter-nodal spacing, and R is the distance at which the *geometric* source has intensity 1W/m².

Experimentally-obtained values of $\frac{p_{\mathrm{DC}}}{p_{\mathrm{init}}}$ are given in [68] for several different mesh topologies. To produce normalised waveguide outputs, a calibration coefficient is calculated, using the experimental result corresponding to a rectilinear mesh. The waveguide excitation is then scaled by this calibration coefficient.

6.3.2.1 Testing

To validate the waveguide calibration procedure, a simple cuboid-shaped room is simulated using the image-source and waveguide methods. The outputs are compared in the frequency-domain, to ensure that the modal responses of the two models match, in shape and in magnitude.

Although geometric methods are generally not capable of modelling low-frequency modal behaviour, the image-source model in a geometric room is a special case. For cuboid rooms with perfectly reflective surfaces, the image-source method is exact [33], and it remains reasonably accurate for slightly-absorbing surfaces. In cuboid rooms the image-source model can, therefore, predict modal behaviour. Additionally, for this room shape, the image source method can be dramatically accelerated, making it possible to calculate extended impulse responses [69]. This accelerated method differs from Wayverb's image-source finder, in that it can calculate long impulse responses for one specific room shape, whereas Wayverb's can calculate short responses for arbitrary geometry.

If the accelerated method is implemented, it can be used to generate impulse responses which are close to ideal (depending on the surface absorptions used). These impulse responses can be compared to those produced by the waveguide, and if the calibration coefficient has been chosen correctly, then their frequency responses should match.

A room, measuring $5.56 \times 3.97 \times 2.81$ metres is simulated, using the accelerated image-source and waveguide methods. The source is placed at $(1, 1, 1)$, with a receiver at $(2, 3, 1.5)$. Both methods are run at a sampling rate of 16kHz. The simulation is carried out three times, with surface absorptions of 0.2, 0.1, and 0.05, and in all cases the simulation is run up until the Sabine-estimated RT60, which is 0.52, 1.03 and 2.06 seconds respectively. The resulting frequency responses are shown in fig. 6.2.

In the graphs, room modes are shown. One of the properties of the waveguide is that it models wave effects which directly contribute to this low-frequency modal behaviour. In the case of arbitrarily-shaped rooms, the image model is not exact, and cannot be used to model low-frequency behaviour in this way, while the waveguide will accurately model low-frequency behaviour in any enclosed space. This is the main reason for using a wave-modelling technique at all, instead of using geometric methods for the entire spectrum.

Below 30Hz, the responses show significant differences. Between 30 and 70Hz, the responses match closely, to within a decibel. There is some divergence at 80Hz, after which the results match closely again until the upper limit of 200Hz. At the upper end of the spectrum, the levels match closely between outputs, but the peaks and troughs are slightly "shifted".

The low-frequency differences can be explained as error introduced by the hard-source/Dirac-delta waveguide excitation method (see Digital Waveguide Mesh). This source type has previously been demonstrated to cause significant error at very low frequencies [61]. However, it is not known how this error will be affected by changes in mesh sampling rate.
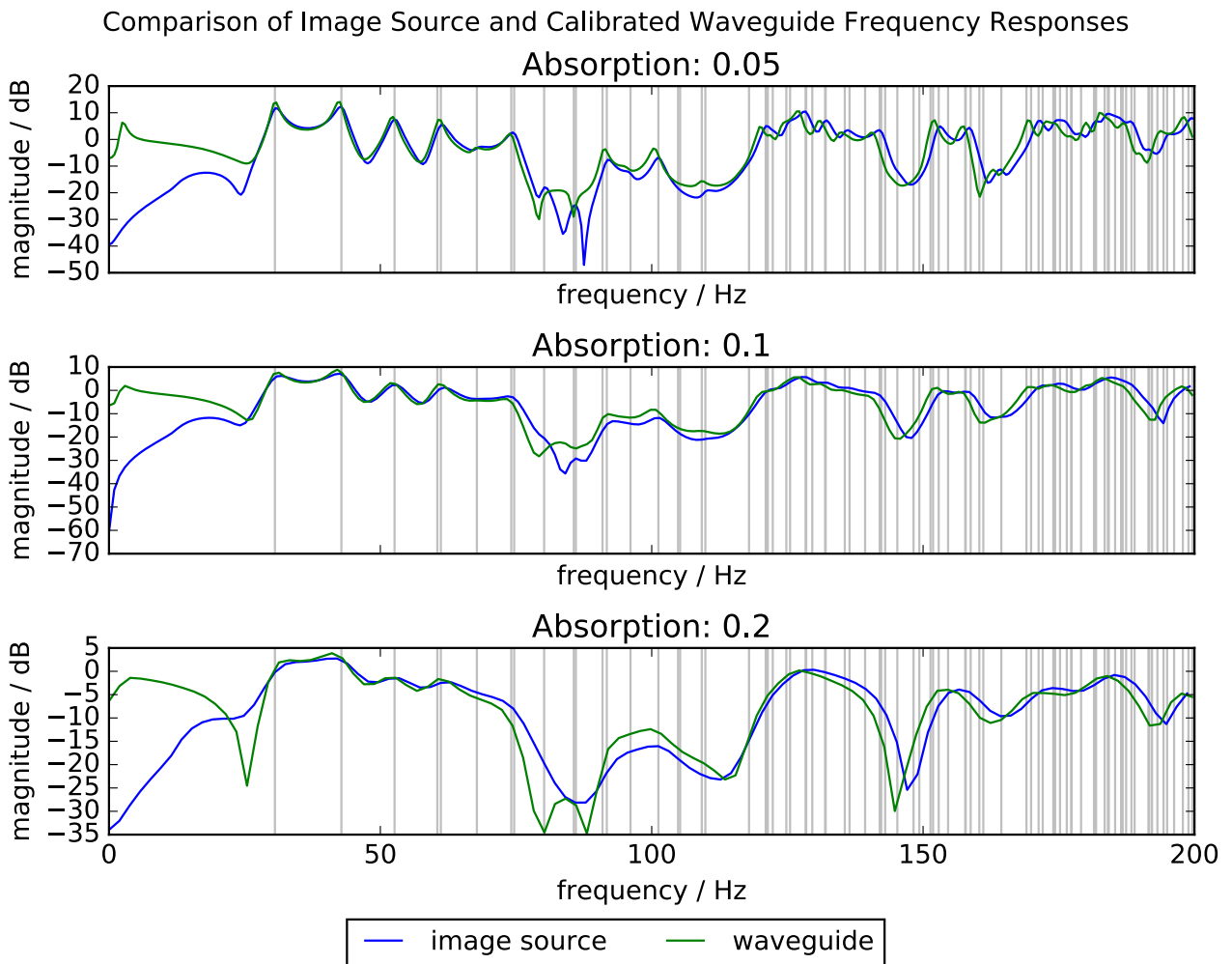
Figure 6.2: Frequency responses analysis of image-source and waveguide outputs. The initial waveguide level has been calibrated using the technique described above. Room mode frequencies are shown in grey.

There are several possible causes for the remaining differences seen between the outputs of the different models. Firstly, the image-source technique is only exact for perfectly reflecting boundaries. The boundary model used in this image-source implementation is the same locally-reacting surface model that Wayverb uses: the reflection factor is real-valued and angle-dependent. A more physically correct method would be to use complex reflection factors, which would allow phase changes at boundaries to be represented. The boundary model is almost certainly the cause of the largest discrepancy, at around 80Hz: results given in [4, p. 78] show similar artefacts of the real-value angle-dependent reflection factor, compared against other more accurate image-source boundary types. Due to time constraints, these more complicated boundary models could not be tested here.

The small frequency shift at the top of the spectrum is most likely to be caused by numerical dispersion in the waveguide mesh. Numerical dispersion becomes more pronounced as frequency increases, which is consistent with the shift seen in the results, which is greater at higher frequencies. The shift may also be caused by slight differences in the dimensions of the modelled room, or the source and receiver positions. In the image-source model, all measurements and positions are exact, but the waveguide must "quantise" the boundary, source, and receiver positions so that they coincide with mesh nodes. If the dimensions of the room were adjusted slightly, this would also cause the room modes to change (which again would be more pronounced at higher frequencies), which might lead to a perceived spectral shift relative to a room with exact dimensions.

Despite the small differences between the frequency responses, the close level match between models suggests that the calibration coefficient is correct.

The reverb times of the outputs are also compared and shown in table 6.1.

Table 6.1: Reverb times of outputs generated by image source and waveguide models

absorption	method	T20 / s	T30 / s
0.05	exact image source	1.044	1.065
0.05	waveguide	1.165	1.180
0.10	exact image source	0.5401	0.5633
0.10	waveguide	0.5689	0.5905
0.20	exact image source	0.2768	0.2990
0.20	waveguide	0.2674	0.2880

There is a difference of 11% for the lowest absorption, which falls to 6% for an absorption of 0.10, and to 4% for an absorption of 0.20. The *just noticeable difference* (JND) for reverb time is 5%. For the lowest two absorptions the difference is larger than the JND, so the match cannot be considered accurate. The waveguide's longer reverb times may be a by-product of the relatively increased level below 30Hz. Octave-band reverb-time measurements may help to clarify whether the error is indeed frequency-dependent. In any case, the match between the modal responses above 30Hz suggests that the boundary models have equivalent behaviour, at least within the audible range.

6.4 Summary

By combining the results from different models, IRs can be generated which have better broadband accuracy than the geometric output alone, and which are faster to produce than broadband waveguide results. In Wayverb, the crossover frequency between models can be adjusted by the user, depending on the desired accuracy and simulation speed. Similarly, the image-source depth is somewhat subjective and may be set manually. Investigation of the Schroeder frequency suggests that the waveguide bandwidth can be reduced in larger rooms, which means that the overall simulation complexity is effectively below linear in the volume of the modelled enclosure.

A technique for matching levels in the geometric models has been derived from the equation for power radiated from a monopole source. This results in a formula for the initial energy carried by each ray, expressed in terms of the energy at the source, the distance between the source and receiver, the receiver radius, and the number of rays. When this formula is used to set initial ray energies, the ray tracer and image-source models will register the same energy level at the receiver by definition, and therefore this particular calibration has not been tested.

The process for calibrating the waveguide is also reasonably straightforward, but depends on a constant derived from experimental data put forward in [68]. This method is not provably correct, and therefore has been verified through independent experimentation. The calibrated waveguide output has been compared to the output of an ideal image-source model, showing close agreement in level between 30Hz and 200Hz. Below 30Hz, the waveguide level is higher than the image-source level, suggesting that some component of the waveguide (likely the injection method or boundary model) behaves incorrectly at low frequencies. The calibration method adjusts the broadband gain of the waveguide, so this error is definitely not due to an error in calibration. Further testing and analysis is necessary to show how the low-frequency error is related to the simulation parameters, and whether it is likely to be problematic in practice. The waveguide calibration method itself appears to function correctly.

7 Microphone Modelling

7.1 Introduction

In the preceding sections, simulation results have been recorded using a virtual omnidirectional microphone model. This model records the sound pressure at a single point in space. The gain and frequency response of the receiver are not affected by the direction of the pressure gradient. Signals recorded with omnidirectional microphones are primarily useful for assessing the overall "character" of an acoustic space, including reverb time, intelligibility, and overall gain.

However, one of the goals of room acoustics simulation is *virtual acoustics*: recreating the modelled auditory environment in such a way that the listener believes they are hearing sounds within a physical space. Real reverbs are highly directional, containing direct sound from the source, and reflected sounds from the room boundaries. To produce a convincing virtual environment, these directional cues must be encoded, by adjusting the output gain and frequency response depending on the direction of the cue. This is achieved using a direction-sensitive receiver model.

7.2 Background

Humans are able to detect the originating direction of sounds using two main techniques: Interaural Time Difference (ITD) defines the time delay between ears, when the sound source is located nearer to one ear than the other; and Interaural Level Difference (ILD) denotes the difference in sound level (which may vary by frequency) caused by attenuation by the outer ear, head, and torso. The suitability of impulse responses for stereo or multichannel playback depends on the ability to produce signals with appropriate ITD and ILD, in such a way that each reflection is perceived as coming from a certain direction.

When recording impulse responses of physical spaces, several techniques might be used to retain ITD and ILD information.

One option is to record the impulse response using a matched pair of microphones. An AB pair of spaced omnidirectional capsules will capture interchannel time difference, but will only capture interchannel level difference if the source is positioned near to the microphones. Alternatively, an XY or Blumlein pair (which consist of coincident cardioid or bidirectional capsules respectively) will capture level difference, but will be incapable of recording time difference because wave-fronts will always arrive at both capsules simultaneously.

Microphone pair methods are only suitable for recording stereo signals, as they only capture a two-dimensional "slice" through the modelled scene, where all directional information is restricted to the same plane as the microphone capsules. The technique can be extended to higher dimensions by using more microphone capsules. This is the basis of the ambisonic approach, which for B-format recordings uses four coincident microphone capsules, three bidirectional and one omnidirectional, to capture the three-dimensional directional pressure gradient and overall pressure level. Instead of being used directly, the recorded signals are post-processed depending on the configuration of the output speakers. For playback on headphones, the signals can be filtered with *head related transfer functions* (HRTFs), which modify the frequency content of the sound depending on its originating direction, mimicking the absorptive characteristics of the human head and torso [70].

If it is known that the recording will only be reproduced on headphones, the preferred method for capturing impulse responses which retain appropriate ILD and ITD is to use in-ear microphones, or a "dummy head" microphone pair. Recordings made in this way capture the ITD and ILD that are caused by sonic interactions with the outer ear, head, and torso, and produce a convincing reconstruction of the sonic space when played back over headphones. However, responses recorded using this technique are unsuitable for loudspeaker playback.

With the exception of the spaced-pair methods, all of these techniques require that the receiver gain is dependent upon the incident direction. The dummy head technique additionally requires that the frequency response of the receiver is dependent upon the incident direction. To allow reproduction on both headphones and arbitrary loudspeaker arrays, the receiver model in Wayverb should encompass all of the techniques described above. Given that spaced microphone techniques can be modelled simply by including multiple receivers in the simulation, each virtual capsule should have configurable direction-dependent gain and frequency response, allowing it to behave like a standard microphone, or like the ear of a dummy head.

7.3 Calculating Directional Gain

Wayverb contains two different models for calculating an appropriate gain from an incident direction, described here.

7.3.1 Microphone

The first method, the "perfect microphone", is very simple. Given that the virtual microphone is pointing in direction $\hat{p}$, and has a "shape" parameter $0 \leq s \leq 1$, the direction-dependent attenuation $a(\hat{d})$ from direction $\hat{d}$ is given by

$$a(\hat{d}) = (1 - s) + s(\hat{d} \cdot \hat{p}) \tag{7.1}$$

where $\hat{p}$ and $\hat{d}$ are unit vectors, and $\cdot$ is the dot-product operator. When s is set to 0, the $a(\hat{d})$ is equal to one for all values of $\hat{d}$, modelling an omnidirectional polar pattern. When s is 1, the modelled polar pattern is bidirectional, with a gain of 1 in front of the microphone, 0 at either side, and -1 behind. When s is 0.5, the pattern is cardioid. Sub- and supercardioid patterns are created by setting s lower or higher than 0.5 respectively. This microphone model is flexible, allowing almost all standard microphone techniques (including the XY pair, Blumlein pair, and B-format ambisonic) to be simulated.

Note that this model has an ideally flat frequency response in every direction, so it can only be used to model perfect microphones. It is not suitable for modelling physical microphones with direction-dependent frequency responses. In particular, with this technique it is not possible to produce binaural dummy-head-style recordings, for immersive headphone playback.

7.3.2 HRTF

A more general microphone model would allow specific per-frequency-band gains to be set for each incoming direction. Such a model is implemented in Wayverb, but is used only for the modelling of HRTF characteristics. In the future, the approach could easily be extended to simulate general non-flat microphones.

The method itself is based on a two-dimensional lookup table, indexed by the azimuth and elevation angles formed by the incident direction. The table is produced using experimentally-obtained HRTF data from the IRCAM Listen database [71]. The data takes the form of a collection of stereo audio files, containing the impulse responses measured using microphones placed in the left and right ears of a human volunteer. The recordings are made in an anechoic environment. Each file contains the response measured for an impulse originating at a specific azimuth and elevation.

To prepare the HRTF database for use in Wayverb, the audio files are multi-band-filtered, using the same 8-band frequencies used throughout Wayverb. Then, the per-band per-channel energy is recorded to a lookup table.

To find the receiver frequency response for a given incident direction, the direction vector is converted to an azimuth and an elevation. Then, this azimuth and elevation are rounded to the closest angles which exist in the lookup table. They are used to index the lookup table, and the energies at that index are returned.

This model contains two approximations: the first is the directional discretisation process, which "quantises" incoming angles to a set of discrete, equally-spaced bins. In the final implementation, fifteen-degree increments are used, matching the directional resolution of the experimentally-obtained data. However, humans can localise sounds from frontal sources to an accuracy of one degree [35, p. 22], so higher directional resolution may be required to produce a convincing binaural effect. This resolution is of course limited by the directional resolution of the input data, and HRTF data with one-degree resolution would be time-consuming to measure, and expensive to store.

The second approximation is in the frequency resolution of the lookup table. HRTF data is supplied as a set of finite-impulse-response filter kernels, which represent continuous frequency spectra, up to the sampling frequency of the kernels. Generally, these filter kernels would be used via convolution, resulting in a precise adjustment of the frequency content of the filtered signal. If this approach were to be taken in Wayverb, it would require creating a separate output signal for each of the incident

directions, and then convolving each direction separately, before a final mixdown. This would be extremely costly: for separation angles of fifteen degrees, this means recording signals for each of 288 incident directions (24 azimuth angles, each with 12 elevation angles), which might easily occupy more than a gigabyte of memory (288 3-second impulse responses at 32-bits and 44.1kHz require 1160MB of memory), and then separately convolving and mixing them. While this is definitely possible, it would be extremely time consuming to run. By restricting the HRTF data to 8 frequency bands, the application architecture can be simplified, while also improving memory usage and runtime performance.

7.4 Image Source Implementation

As described in the Image Source section, the magnitude of the contribution of each image source depends on the distance between the source and receiver, and the direction-dependent reflectance characteristics of any intermediate reflective surfaces. To model a directional receiver, the contribution magnitude is additionally scaled by the receiver's gain in the direction of that contribution.

The direction of each image source is found by subtracting the receiver position from the image source position, and normalising the result. This direction can either be used to directly compute the appropriate attenuation factor, or it can be used to look up the attenuation in a pre-computed table (see Calculating Directional Gain below).

In the case that the frequency response (rather than overall gain) of the receiver depends on incident direction, a different attenuation function or look-up table is used for each frequency band. At the end of the simulation, the outputs in each band are band-pass filtered and mixed down.

Wayverb includes a fast estimator for ITD in the image source model, which has not been encountered in the literature. During the mix-down process for the image-source contributions, the time of each contribution is calculated based on the distance from the image source to the receiver. When calculating these distances for the left ear, the receiver position can be temporarily moved 10cm to the left, relative to the receiver orientation, and the same can be done for the right ear. The output signal will then incorporate small time delays between channels, depending on the distance and direction of each image source. This method relies on the assumption that all image sources are visible from both ear positions, and that the new ear positions do not affect the angle-dependent reflection characteristics of intermediate surfaces. This is a reasonable assumption, given that the distance between ears is generally small compared to the size of the modelled room. The benefit of this method is that the costly image-source-finding process only has to be run once in total, instead of once for each ear.

This image-source "offsetting" technique cannot be applied to Wayverb's other simulation methods. In the waveguide, the new "ear" positions may not fall directly on grid nodes, and so the waveguide simulation would have to be run twice, with different grids. The raytracer would not benefit from interaural time difference at all, as the maximum possible interaural time delay is the interaural distance divided by the speed of sound, $0.2/340=0.0005$s, which is less than the 1ms energy histogram sampling interval. Therefore, any measured ITD would be lost when quantising to the histogram interval.

The drawback of the offsetting method is that, if the user manually positions two receivers with appropriate interaural spacing, an extra offset will still be automatically applied. This will lead to an interaural spacing that is twice as wide as desired. This could be addressed by including an option in the software to enable or disable the additional offset. Due to time constraints this feature has not been implemented in the current version of Wayverb.

7.5 Ray Tracer Implementation

In the ray tracer, a very similar approach could be taken. The incident direction of each ray is known, so this direction might be used with a function or lookup-table to adjust the ray's multiband intensity before it is added to the energy histogram (see Ray Tracer).

In terms of implementation, this approach would be needlessly expensive. In Wayverb, the initial simulation and directional processing stages are kept separate, allowing a single simulation to be processed by several different virtual microphone capsules. While flexible, it is also infeasible to store the direction of each incident ray, especially if there are hundreds of thousands of rays in the simulation. The ray tracer is used to estimate the late reverberation of the space, in which individual sonic events are "blurred" together, and in which there are few directional cues [35, p. 21], so this level of per-ray directional accuracy is unnecessary.

Instead, the surface of the spherical receiver is divided into discrete *directivity groups*, mirroring the approach of Schroder in [35, p. 72]. A separate multiband energy histogram is maintained for each directivity group, where the direction of ray incidence determines the histogram used to store that ray's energy. These histograms are at a substantially lower sampling-frequency than the output sampling rate, so it is feasible to store one for each discrete incident direction, as long as the direction quantisation intervals are on the order of ten to twenty degrees. To generate audio-rate results, the directional histograms must be combined into a single energy histogram, which can then be used to weight a noise sequence as discussed in the Ray Tracer section. This is achieved by taking the centre direction of each directivity group, calculating or looking-up the correct attenuation for that direction, and then multiplying the entire histogram by that value. The final direction-weighted histogram is given by the elementwise sum of all individual weighted histograms.

7.6 DWM Implementation

7.6.1 Implementation Decisions

Implementing directional receivers in the waveguide mesh is more complicated than for the geometric methods. Although absolute pressure values are available in the mesh, directional pressure values are not inherently supported. In order to simulate a directional receiver, the instantaneous direction of the sound field must be found at the receiver point. Several methods were considered, as outlined below.

7.6.1.1 Blumlein Difference Technique

The first method, described in [72] and [73], is designed with the goal of encoding the 3D sound-field by means of spherical harmonic decomposition to the second order, for use in ambisonic reproduction.

This method is based on sound intensity probe theory, which allows directional velocity components of a sound field to be captured using two closely-spaced omnidirectional receivers. In the DWM, each node has an associated pressure, so two closely-spaced nodes can be used to emulate a pair of omnidirectional receivers, and in turn used to generate directional velocity components. A full first-order B-format signal can be made using seven such "pressure sensors", arranged in a cross shape, with a single node at the centre, and six nodes placed around it. The pressure field velocity components are found by subtracting nodes on "opposite" sides of the cross, while the centre node contributes the overall pressure at the receiver. A second-order output is achieved by adding further pressure-sensing nodes in between the first-order nodes.

Note that this technique does not allow the immediate production of directional responses. Instead, it is designed to produce outputs in ambisonic format, which are then post-processed to create a signal with the desired modelled directional response.

Additionally, this technique has some difficulty adapting to meshes with large grid spacings. In a rectilinear mesh, the seven first-order nodes can be placed exactly, as their positions will coincide with nodes in the mesh. However, the second-order nodes will not fall exactly on existing nodes, and instead their positions will need to be quantised to the closest mesh nodes. This problem becomes worse in other mesh topologies, such as the tetrahedral mesh, where it may be that very few of the sensor nodes can be exactly placed in the desired positions.

To facilitate more exact placement of sensor nodes, the spatial sampling frequency of the mesh must be increased (by reducing the grid spacing, the quantisation error in the sensor placement is also reduced). The example given in [73] uses a sensor spacing of 0.04 metres, giving a maximum valid sampling frequency of 4.25kHz. However, to allow accurate placement of these sensors, a grid with a spacing of 0.0027m is required. This increase in mesh density necessitates more than 3000 times the original number of nodes, which is only practical for very small modelled spaces. An alternative to oversampling is interpolation, which may be used to obtain signal values for positions that do not coincide with grid nodes, as in [74]. However, the interpolated approach given in this paper is only applicable in the 2D case, so it cannot be applied in Wayverb, which requires a general solution in three dimensions.

Finally, this method places an additional constraint on the portion of the spectrum of the output signal which may be considered valid. The maximum valid frequency of the output spectrum is determined by the ratio between the receiver spacing and the wavelength at that frequency. To raise the upper frequency limit, the receiver spacings must be made as small as possible, which requires costly mesh oversampling. Small receiver spacings reduce the sensitivity at the lower end of the spectrum (even when interpolation is used), so ideally the simulation must use several sets of receiver nodes with different spacings, in order to maintain a flat frequency response.

This technique seems suitable for calculating lower-order ambisonic signals in small modelled spaces. For higher order outputs, with consistent frequency responses across the spectrum, the technique quickly becomes too expensive (due to mesh oversampling) and complex to implement (as several receiver spacings and configurations are required for a single simulation output).

7.6.1.2 Intensity Calculation Technique

The second method considered for implementation in Wayverb is described by Hacıhabiboğlu in [75]. This method is based around estimating the acoustic intensity at the output node of the DWM.

The technique is much more straight-forward than the technique discussed above. The pressure differences between the "output" node and its surrounding nodes are calculated. These pressures are transformed depending on the relative positions of each surrounding node, and combined to produce a three-element vector containing an approximation for the three-dimensional pressure gradient. The equations describing these operations in the general case are not replicated here, however they may be greatly simplified in the rectilinear case, as will be shown in the Implementation section below. The velocity for each time step is found by adding the pressure gradient vector to the velocity at the previous time step, where the initial velocity is assumed to be zero. Intensity is found by multiplying this velocity by the pressure at the output node.

In some respects, this technique is very similar to the Blumlein difference technique. Both techniques calculate the instantaneous pressure gradient at the output node by subtracting the pressures of the surrounding nodes. The main difference is that the Blumlein method requires that the mesh pressure is found at exact locations relative to the output node, which often requires mesh oversampling or interpolation. The intensity calculation technique instead uses a more general matrix-based method, which incorporates the relative node positions into the calculation. In short, the Blumlein method requires that the mesh is designed to suit the receiver configuration, whereas the intensity calculation method can adapt to arbitrary mesh topologies and spacings.

The frequency range issues mentioned in [72] (where large mesh spacings reduce the output bandwidth, and low mesh spacings reduce low frequency sensitivity) are mentioned by Hacıhabiboğlu in [75]. It is unclear whether the intensity calculation technique solves this problem, although the results presented in [75] appear consistent across the spectrum. Some inaccuracy is evident at the top of the spectrum, but this is attributed to direction-dependent dispersion error, rather than to a problem with the frequency response of the model itself.

Under this scheme, the output of the waveguide mesh is actually two streams of information: a stream of scalar values describing the pressure at the output node; and a stream of 3D vectors denoting the directional intensity at that point.

To find the directional response for a single time-step, the intensity vector is used to look up the microphone response in the direction of the vector, which is a scalar value. The microphone response is squared, and multiplied by the magnitude of the intensity vector, yielding a positive scalar output. Finally, the sign of the actual pressure at the output node is copied to the output value. To find the full directional response, this process is repeated for every time-step of the simulation.

The intensity calculation technique can also model receivers with direction-dependent frequency responses. The intensity and pressure outputs are processed with a set of polar patterns corresponding to responses at different frequency bands. Then, these processed signals are band-passed with linear-phase filters, and summed together.

7.6.1.3 Choice of Method

The intensity calculation technique was chosen for implementation in Wayverb, as it is simpler to implement, and less expensive to run. Also, the Blumlein difference technique is only practical in rectilinear meshes, whereas the intensity calculation technique can be adapted to any mesh topology. This was an important consideration at the beginning of the Wayverb project, which originally used the tetrahedral mesh due to its relative efficiency.

7.6.2 Implementation

The implementation is very simple in the rectilinear mesh. The mesh is inherently aligned to the axial directions, which means that the matrix maths is greatly simplified. The process for finding the instantaneous intensity vector for each step of the waveguide simulation is shown in pseudocode below.

```
def compute_intensities_at_node(node_position,
                                mesh_spacing,
                                mesh_sample_rate,
                                ambient_density):
    velocity_vector = (0, 0, 0)
    pressures = []
    intensities = []

    (nx_position, px_position,
     ny_position, py_position,
     nz_position, pz_position) = compute_neighbouring_positions(node_position)

    for step in waveguide:
        //  Find current pressure at output node.
        p = get_node_pressure(node_position)

        //  Calculate the pressure differences between surrounding nodes.
        nx_difference = (get_node_pressure(nx_position) - p) / mesh_spacing
        px_difference = (get_node_pressure(px_position) - p) / mesh_spacing
        ny_difference = (get_node_pressure(ny_position) - p) / mesh_spacing
        py_difference = (get_node_pressure(py_position) - p) / mesh_spacing
        nz_difference = (get_node_pressure(nz_position) - p) / mesh_spacing
        pz_difference = (get_node_pressure(pz_position) - p) / mesh_spacing

        //  Find the instantaneous pressure gradient.
```

```
        pressure_gradient_vector = ((px_difference - nx_difference) / 2,
                                     (py_difference - ny_difference) / 2,
                                     (pz_difference - nz_difference) / 2)


        //  Update the velocity accumulator.
        velocity_vector -= (pressure_gradient_vector /
                            (ambient_density * mesh_sample_rate))


        //  Find the instantaneous intensity.
        intensity_vector = velocity_vector * p


        pressures.append(p)
        intensities.append(intensity_vector)


    return pressures, intensities
```

For each waveguide step, the pressure and intensity at the output node is found and stored. If there are multiple coincident capsules, then the same pressure-intensity data can be post-processed once for each virtual capsule, without needing to re-run the simulation. However, this approach may not be well-justified: according to Hacihabiboglu et al. this microphone model is only suitable for single-diaphragm directional microphones, and is unsuitable for coincident pressure gradient microphones with multiple elements [75]. Despite its unsuitability, there is no other possible way of modelling coincident capsules using this microphone model, so coincident capsules *are* allowed in Wayverb. Development of a microphone model with better support of coincident capsules is a topic for future research.

Having generated the pressures and the intensity vectors at the output node, the post-processing progresses by finding the magnitude of the intensity vector. This value is multiplied by the square of the polar-pattern attenuation in the direction of the intensity vector, and then the square root is taken. Finally, the sign of the pressure is copied to this output value. This process is repeated for each step of the simulation, resulting in a sequence of directionally-processed pressure values.

For binaural results, this process is repeated once for each frequency band, resulting in several differently-weighted broadband signals. The directional gains are found using the same lookup-table technique as the geometric methods, using the azimuth and elevation of the intensity vector as indices. The final output is created by bandpass filtering each of these broadband signals using a zero-phase frequency-domain filter, and then mixing them down. The filtering process is the same as that used throughout Wayverb (see Ray Tracer).

7.6.3 Testing of the Microphone Model in the DWM

The testing procedure is similar to that described in [75]. A receiver is placed at the centre of a room measuring $1.5 \times 1.5 \times 1.5$ metres. Sixteen equally-spaced source positions are generated in a circle with radius 1m around the receiver. For each source, the room is simulated using a rectilinear waveguide mesh with a sampling rate of 50kHz, for 294 steps. The step number is chosen so that the initial wave-front will reach the receiver, but reflections from the room walls will not. After each simulation, the directionally-weighted signal recorded at the receiver is split into frequency bands, and

the normalised energy per band is calculated. The bands are split in the frequency domain, using the frequency-domain envelope method mentioned in the Ray Tracer section. The normalised energy of the band is calculated by dividing the sum of squared bin magnitudes in the output by the integrated frequency-domain envelope.

The per-band normalised energy can now be plotted against the angle of incidence, and compared to the ideal polar pattern of the modelled microphone. This is achieved by normalising the measured directional response relative to the response in the front direction. If there is a close match, then the model is successful.

Three different microphone polar patterns are simulated: omnidirectional, bidirectional, and cardioid. The results are shown in fig. 7.1 to 7.3. In these figures, the red lines show the measured energy in each direction, while the blue lines show the expected polar pattern, normalized so that the 0-degree level matches the experimentally-obtained level in that direction.
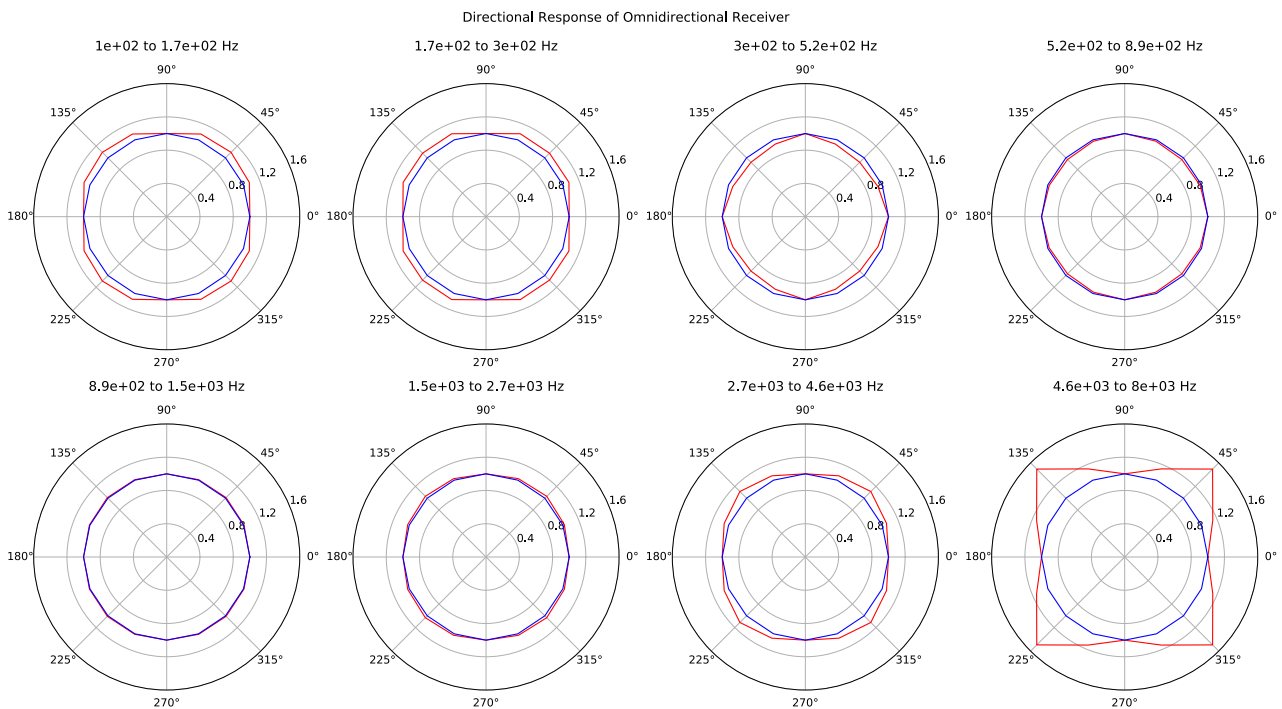

Figure 7.1: The directional response of an omnidirectional receiver in the waveguide mesh.

Results are consistent across all polar patterns tested. At the lower frequencies shown (100Hz to 1.5kHz), the match is particularly good, with virtually no error in all directions. As frequency increases, the error increases greatly. This happens as a result of frequency-dependent dispersion. At higher frequencies (above 2.7kHz), the speed of wave-fronts changes depending on their direction. In the rectilinear mesh, there is no dispersion along diagonal directions [39]. Along axial directions, however, high-frequency wave-fronts move more slowly. This is likely the cause of the predominant "spikes" seen in the diagonal directions: in these directions, all the acoustic energy reaches the receiver, while along axial directions, the wave-fronts are slower, and the full energy does not reach the receiver within the simulation window.
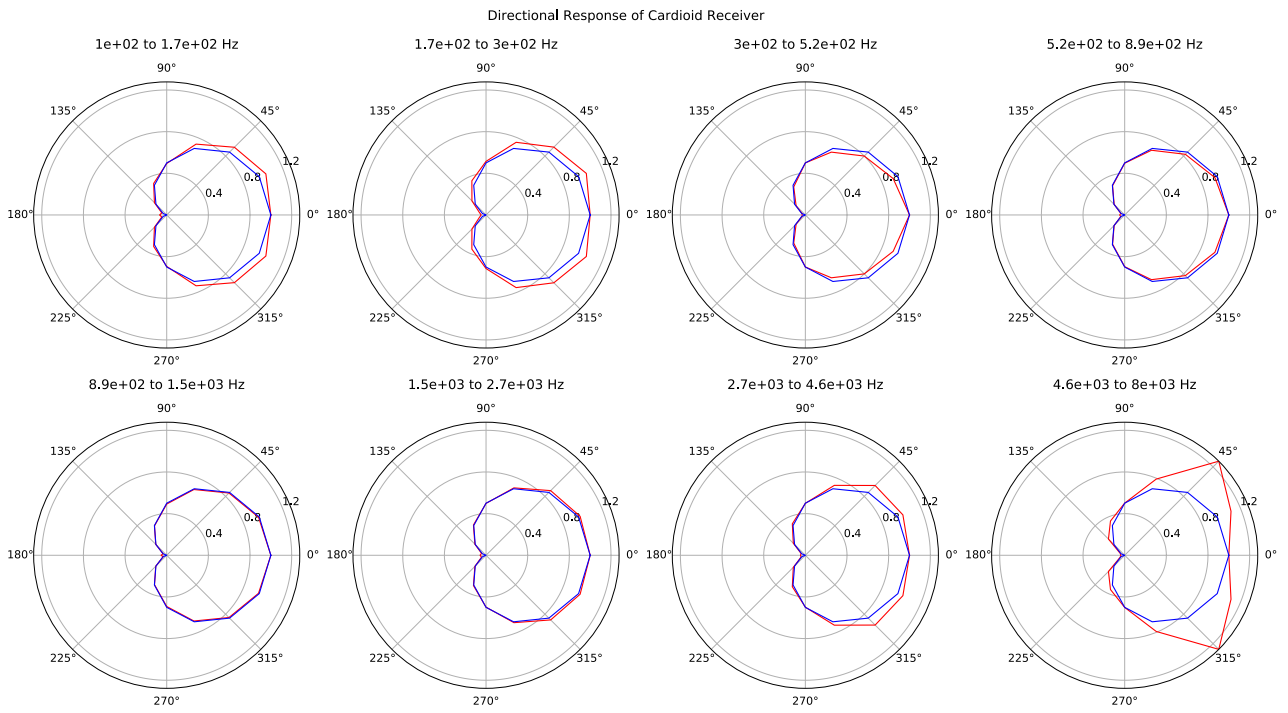
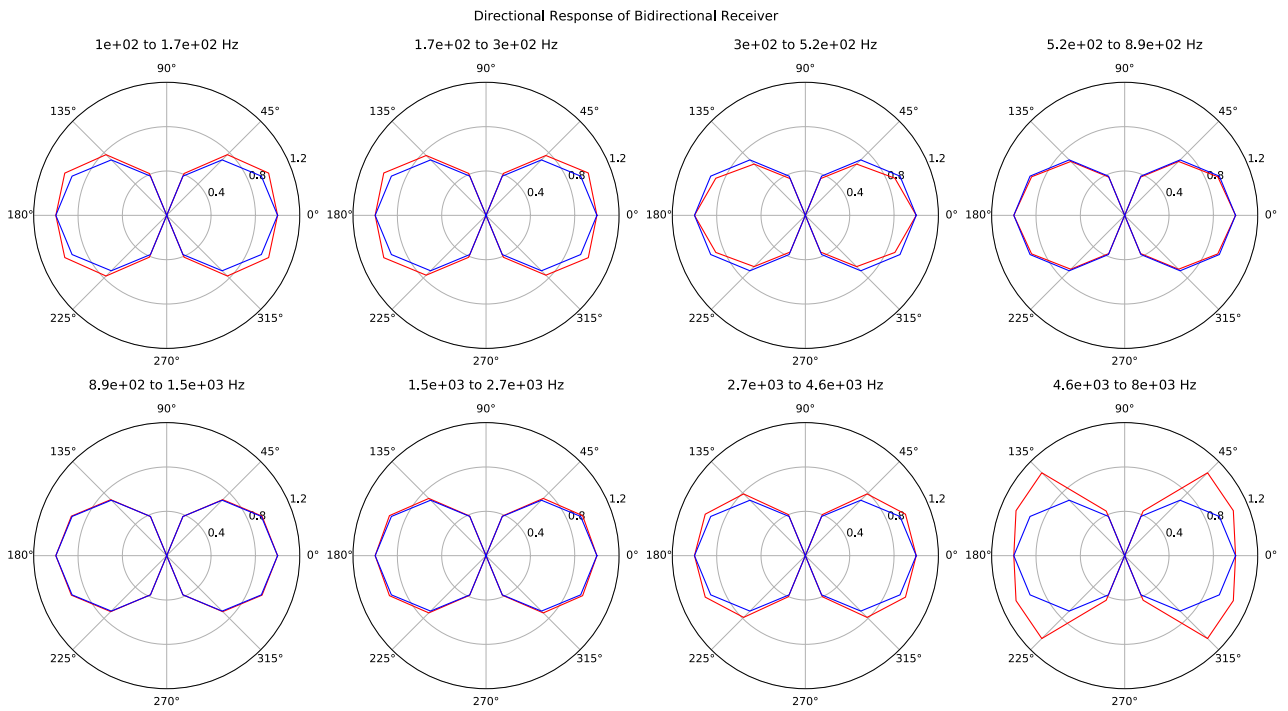Figure 7.2: The directional response of a cardioid receiver in the waveguide mesh.

Figure 7.3: The directional response of a bidirectional receiver in the waveguide mesh.

These results mirror those seen by Hacıhabiboğlu, who also records similar large errors in diagonal directions in the highest (8kHz) band. These errors are also attributed to directional dispersion. Therefore, error in the results is likely due to the properties of the waveguide and the particular receiver technique used, rather than an error in the implementation.

Although there is very large error in some directions at high frequencies, at low frequencies this method produces accurate directional responses. Even at higher frequencies (relative to the mesh sampling rate), the response follows the same overall shape as the desired response, so even here the results may provide a reasonable approximation of the directional room response. Finally, if accurate directional responses are required at higher frequencies, the mesh can be oversampled to reduce directional dispersion. Although costly, it is at least potentially possible to produce very accurate responses using this method.

7.7 Summary

The need for a virtual microphone model has been explained. Real-world microphone techniques rely on capsules with direction- and frequency-dependent responses, therefore a virtual model is put forward which is capable of simulating these same characteristics.

Directional information is inherent in the geometric models: each ray has a direction, which can be logged and used to attenuate each contribution according to the receiver polar pattern. These models already operate in multiple frequency bands, so to model frequency-dependent responses, the receiver can simply use a different directional response per band. A novel "offsetting" method has been put forward as an extension to the image-source model, which can reconstruct ITD information.

The waveguide does not model directional information natively. Instead, the pressure gradient at the output must be approximated using the pressures at surrounding nodes. The intensity calculation technique is used for this purpose in Wayverb, as it is simpler, faster, and more flexible than the main alternative (the Blumlein difference technique). It has been shown that Wayverb's implementation is capable of reproducing three different polar patterns with reasonable accuracy across the spectrum, although numerical dispersion causes accuracy to worsen as frequency increases. If receiver accuracy is a major concern, the mesh can be oversampled, reducing dispersion in the output bandwidth.

The test results presented are encouraging but superficial. The test procedure is modelled on that presented in [75], which has the shortcoming that it doesn't consider interactions with other aspects of the waveguide simulation. In particular, it has not been shown how the microphone model responds to source gain and distance, or whether it affects the measured frequency response at the receiver. These are important considerations which have implications for the validity of the waveguide model as a whole. Due to the number of components in Wayverb, and the time spent on implementation alone, there was not time to test all aspects of the program. However, the Evaluation contains a number of tests, all of which use this microphone model. In the future, more detailed tests should be conducted which focus solely on the microphone model.

8 Boundary Modelling

8.1 Introduction

The ideal boundary model would allow complete control over the frequency- and direction-dependent absorption and scattering of a surface. Though this is reasonably straightforward to implement in geometric models, it is far from a solved problem for the digital waveguide mesh (DWM). Several possible implementations are discussed in the literature, each with unique drawbacks.

This chapter relies on terms defined in the Theory section. It begins by discussing the ideal behaviour of modelled acoustic boundaries. Then, the implementation for geometric models is discussed. Possibilities for DWM boundary models are investigated, and the final choice of method explained. The geometric and DWM implementations are evaluated and compared, to ensure equivalence.

8.2 Geometric Implementation

In Wayverb, surfaces may have different absorptions in each frequency band. Each ray starts with the same pressure in each band. During a specular reflection, the per-band absorptions are converted into per-band reflection factors. Then, the pressure in each band is adjusted using that band's reflection coefficient. This is similar to the approach taken in graphical ray tracing, in which each ray carries separate red, green, and blue components. These components are modified independently, depending on the colour of the reflective surface.

By definition, image-source models find only specular reflections (i.e. image sources), so scattering is not implemented in these models. Scattering can be implemented in ray tracers, but there is no consensus on the optimum method. One option is to spawn two rays at every reflection: a specular ray, and a diffuse ray with random direction. Though this properly replicates the theory, it leads to an explosion in the number of rays which must be traced, so is impractical in most cases. A second option is to decide, using the scattering coefficient as a probability, whether the reflection should be specular or diffuse [2]. This solves the ray-explosion problem, but requires an additional random number to be generated per-reflection, which can be costly for large numbers of rays. An elegant solution is to simply mix the specular and diffuse rays together, using the scattering coefficient as a weighting [76], a technique known as *vector based scattering* [43]. This is the approach taken by Wayverb. A major drawback of all these scattering methods is that the scattering coefficient can only be frequency-dependent if a separate ray is traced for each band. If a single ray is used to carry all frequency components, then each component must be scattered in exactly the same way.

The plain scattering model affects only the ongoing ray direction and amplitude. However, it is worth considering that, at each reflection, the scattered energy may be directly visible to the receiver. This fact is exploited by the *diffuse rain* technique [35, pp. 61–66], in which each reflection is considered to spawn a "secondary source" which emits scattered energy towards the receiver. This scattered energy is recorded only if the secondary source is visible from the receiver.

The magnitude of scattered energy is proportional to the scattering coefficient, and the Lambert diffusion coefficient. It is also proportional to the fraction of the available hemispherical output area which is covered by the receiver volume. The absolute area covered by a spherical receiver $A_{\text{intersection}}$ is found using the equation for the surface area of a spherical cap.

$$A_{\text{intersection}} = 2\pi r^2 (1 - \cos\gamma) \tag{8.1}$$

Then, the detected scattered energy can be derived [35, p. 64]:

$$
\begin{aligned}
E_{\text{scattered}} &= E_{\text{incident}} \cdot s(1-\alpha) \cdot 2\cos\theta \cdot \left(\frac{A_{\text{intersection}}}{2\pi r^2}\right) \\
&= E_{\text{incident}} \cdot s(1-\alpha) \cdot 2\cos\theta \cdot \left(\frac{2\pi r^2 (1-\cos\gamma)}{2\pi r^2}\right) \\
&= E_{\text{incident}} \cdot s(1-\alpha) \cdot 2\cos\theta \cdot (1-\cos\gamma)
\end{aligned}
\tag{8.2}
$$

Here, θ is the angle from secondary source to receiver relative against the surface normal, and γ is the opening angle (shown in fig. 8.1). The magnitude of the scattered energy depends on the direction from the secondary source to the receiver (by Lambert's cosine law), and also on the solid angle covered by the receiver.
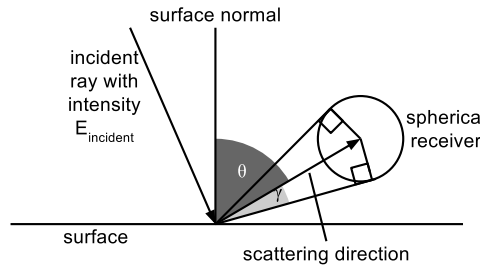

Figure 8.1: Angles used in the diffuse rain equation for a spherical receiver.

8.3 DWM Implementation

8.3.1 Possible Methods

Two methods from the literature were considered for use in Wayverb. A brief overview of each will be given here.

8.3.1.1 KW-Pipe Technique

This method is described in [77] and [78].

As described in the Digital Waveguide Mesh section, there are two technically-equivalent formulations of digital waveguides meshes, known as *W-models* and *K-models*. W-models allow for straightforward interaction with a variety of termination types, such as wave digital filters, which can be used to model frequency-dependent boundaries and air absorption. However, W-models use more than twice the memory of the equivalent K-model [18]. For large-scale simulations, K-models are preferable for their reduced memory usage. However, K-models cannot interact directly with wave digital filters.

The KW-pipe is a "converter" between wave- and Kirchhoff- variables, which is designed to allow the majority of a model (that is, the air-filled space inside it) to be constructed as a K-model waveguide mesh. At the boundaries of the model, the KW-pipe is used to connect K-model nodes to W-model nodes. These W-model nodes can then be connected to wave digital filters to simulate frequency-dependent absorption of wave energy. The complete model retains both the memory-efficiency of the K-model and the termination flexibility of the W-model, with the drawback of additional implementation complexity at the interface between the two model types.

This sounds extremely promising, but has a major drawback, as described by Kowalczyk and Van Walstijn [67]: while the inside of the mesh will be 2- or 3-dimensional, the boundary termination afforded by the wave-variable boundary is 1-dimensional. Each boundary node connects to just the closest interior node. As a result, the edges and corners are not considered to be part of the model, as these nodes do not have a directly adjacent interior node. Additionally, the 1D boundary termination equation implies a smaller inter-nodal distance than that of the 2D or 3D mesh interior. This means that when updating an interior node next to a boundary, the inter-nodal distance is greater than when updating the boundary node itself. For these reasons, the 1D termination is unphysical and can lead to large errors in the phase and amplitude of reflections [67].

8.3.1.2 Locally Reactive Surfaces Technique

This method, described in [67], aims to create physically correct higher-dimensional boundaries by combining a boundary condition, defined by a boundary impedance, with the multidimensional wave equation. This leads to a model for a *locally reacting surface* (LRS), in which boundary impedance is represented by an infinite-impulse-response (IIR) filter.

As noted above, a surface is locally reacting if the normal component of the particle velocity on the boundary surface is dependent solely upon the sound pressure in front of the boundary. In most physical surfaces, the velocity at the surface boundary will also be influenced by the velocity at adjacent points on the boundary, so LRS is not a realistic physical model in the vast majority of cases.

However, despite that it is not a realistic physical model, the implementation of the LRS modelling technique is both stable and accurate, as opposed to the 1D KW-pipe termination, which does not accurately model even locally-reacting surfaces.

The LRS model leads to an implementation that is efficient (as it is based completely on the K-model/FDTD formulation) and tunable (boundaries are defined by arbitrary IIR filters).

8.3.2 Choice of Boundary Technique for the DWM

The LRS technique was chosen, as it represented the best compromise between memory efficiency, customization and tuning, and physically-based behaviour (i.e. edges and corners are considered as well as flat surfaces). The particular strengths of this model are its performance and tunability, though as mentioned previously it is not physically accurate in many cases. That being said, neither of the boundary models considered are particularly realistic, so even for applications where physical modelling is the most important consideration, the LRS model seems to be the most appropriate.

8.3.3 LRS Implementation

See [67] and [79] for a more detailed explanation.

The reflectance of a LRS has been defined earlier, in terms of the normal-incidence specific impedance ξ_0 (eq. 2.16). For the geometric implementation, ξ_0 was defined in terms of a single normal-incidence reflection coefficient R_0 (eq. 2.19). If R_0 is replaced by a digital filter $R_0(z)$, then the specific impedance may also be expressed as a filter $\xi_0(z)$:

$$\xi_0(z) = \frac{1 + R_0(z)}{1 - R_0(z)} \tag{8.3}$$

To create the filter R_0, per-band normal reflection magnitudes are found using the relationship $|R| = \sqrt{1 - \alpha}$. Then, the Yule-Walker method is used to find *infinite impulse response* (IIR) coefficients for a filter with an approximately-matched frequency response. Then, this filter is substituted into eq. 8.3 to find IIR coefficients for the specific impedance filter. This impedance filter will eventually be "embedded" into the boundary nodes of the waveguide.

Surfaces with detailed frequency responses will require high-order filters. This generally leads to numerical instability in IIR filters. The usual solution to this problem would be to split the high-order filter into a series-combination of lower-order filters. However, the LRS requires access to intermediate values from the filter delay-line which makes this approach impossible. An alternative solution is suggested in [80], which suggests running the entire simulation multiple times, once for each octave band. This means that the boundary filters can be single-order, and resistant to accumulated numerical error. Compared to high-order boundary filters, this method gives much improved accuracy, but at the cost of running the entire simulation multiple times. In Wayverb, both approaches are possible, allowing the user to choose between a fast, inaccurate single-run simulation with high-order filters; or a slow, accurate multi-run simulation with low-order filters.

To implement the waveguide boundaries, the computed impedance filter coefficients are inserted into three special update equations, which are found by combining the discrete 3D wave equation with the discrete LRS boundary condition. These equations are used instead of the standard update equation when updating the boundary nodes. The exact update equations have not been reproduced here due to space constraints, but they can be found in [67], alongside a thorough derivation and explanation. The implementation in Wayverb does not make any modifications to these equations.

The three different update equations are chosen depending on the placement of the boundary nodes. In the case of a flat wall, the boundary node is adjacent to a single inner-node, and a "1D" update equation is used. Where two perpendicular walls meet, the nodes along the edge will each be adjacent to two "1D" nodes, and a "2D" update equation is used for these nodes. Where three walls meet, the corner node will be directly adjacent to three "2D" nodes, and a "3D" update equation is used for this node. The three types of boundary nodes are shown in fig. 8.2. Note that this method is only capable of modelling mesh-aligned surfaces. Other sloping or curved surfaces must be approximated as a group of narrow mesh-aligned surfaces separated by "steps". For example, a wall tilted at 45 degrees to the mesh axes will be approximated as a staircase-like series of "2D" edge nodes.
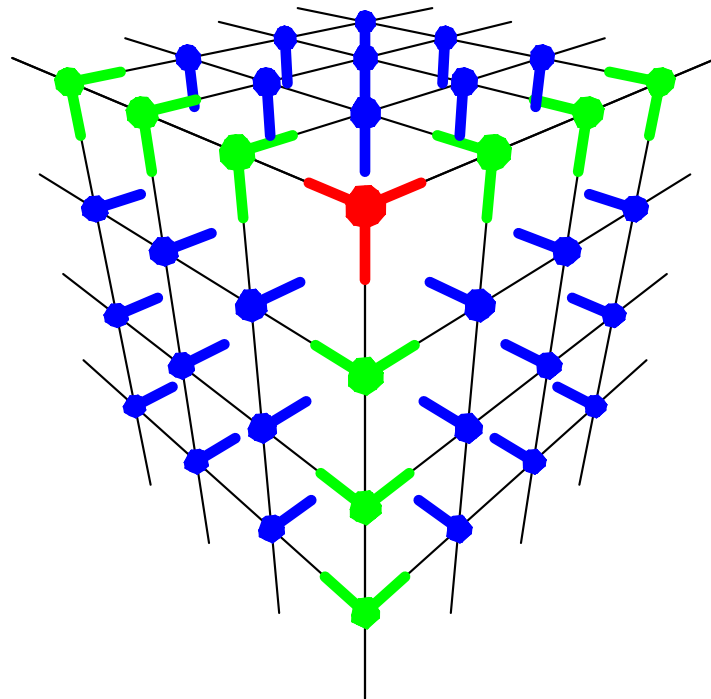

Figure 8.2: The three types of boundary nodes, used to model reflective planes, edges, and corners. 1D nodes are adjacent to inner nodes, 2D nodes are adjacent to two 1D nodes, and 3D nodes are adjacent to three 2D nodes.

8.4 Testing of the LRS Boundary for the DWM

The LRS waveguide boundary is complex to implement, as it embeds IIR filters into the waveguide boundaries, so it is worth ensuring that the boundary nodes behave as expected.

Although the 3D boundary equations are presented in [67], only 2D boundaries are tested. Therefore the test shown in this thesis is a novel contribution, as no previous empirical evidence exists for the 3D LRS boundary implementation in the waveguide mesh. The test used here is an extension of the test procedure presented in [67], but extended to three dimensions.

8.4.1 Method

A mesh with dimensions $300 \times 300 \times 300$ nodes, and a sampling frequency of 8kHz, was set up. A source and receiver were placed at a distance of 37 node-spacings from the centre of one wall. The source position was dictated by an azimuth and elevation relative to the centre of the wall, with the receiver placed directly in the specular reflection path. The simulation was run for 420 steps. The first output, r_f, contained a direct and a reflected response. Then, the room was doubled in size along the plane of the wall being tested. The simulation was run again, recording just the direct response at the receiver (r_d). Finally, the receiver position was reflected in the boundary under test, creating a *receiver image*, and the simulation was run once more, producing a free-field response at the image position (r_i). Figure 8.3 shows the testing setup.

When testing on-axis reflections (where azimuth and elevation are both 0), the position of the receiver will exactly coincide with the position of the source. If a hard source is used, the recorded pressures at the receiver (r_f and r_d) will always exactly match the input signal, and will be unaffected by reflections from the boundary under test. It is imperative that the signal at the receiver contains the reflected response, so a hard source is not viable for this test. Instead, a transparent dirac source is used, which allows the receiver node to respond to the reflected pressure wave, even when the source and receiver positions match. The main drawback of the transparent source, solution growth, is not a concern here as the simulations are only run for 420 steps. The tests in the Digital Waveguide Mesh section showed that solution growth generally only becomes evident after several thousand steps.
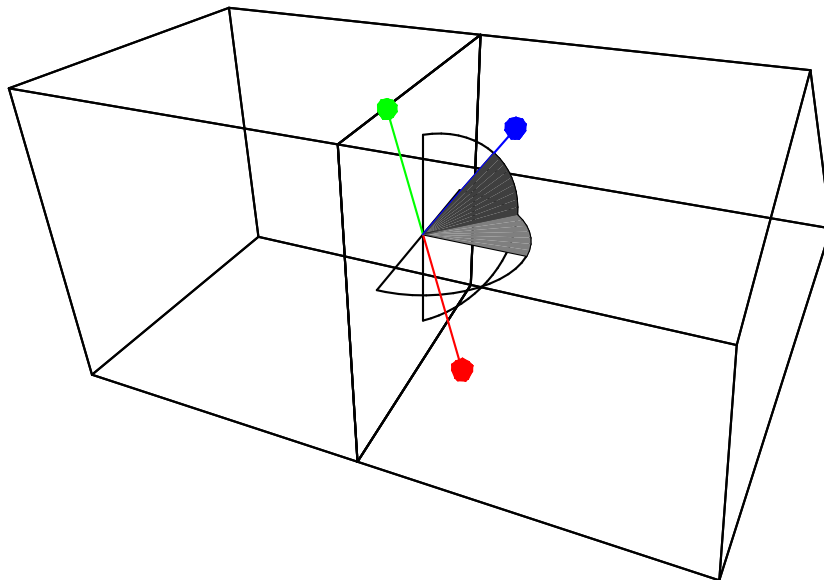

Figure 8.3: The setup of the two room-sizes, and the positions of sources and receivers inside.

The reflected response was isolated by subtracting r_d from r_f, cancelling out the direct response. This isolated reflection is r_r. To find the effect of the frequency-dependent boundary, the frequency content of the reflected response was compared to the free-field response r_i. This was achieved by windowing r_r and r_i with the right half of a Hann window, then taking FFTs of each. The experimentally determined numerical reflectance was determined by dividing the magnitude values of the two FFTs.

To find the accuracy of the boundary model, the numerical reflectance was compared to the theoretical reflection of the digital impedance filter being tested, which is defined as:

$$R_{\theta,\phi}(z) = \frac{\xi(z)\cos\theta\cos\phi - 1}{\xi(z)\cos\theta\cos\phi + 1} \tag{8.4}$$

where θ and ϕ are the reflection azimuth and elevation respectively.

The test was run for three different angles of incidence, with matched azimuth and elevation angles of 0, 30, and 60 degrees respectively. Three different sets of surface absorption coefficients were used, giving a total of nine combinations of source position and absorption coefficients. The specific absorption coefficients are those suggested in [80], shown in table 8.1.

Table 8.1: Absorption coefficients of different materials at different frequencies, taken from [80].

band centre frequency / Hz	31	73	173	411	974
plaster	0.08	0.08	0.2	0.5	0.4
wood	0.15	0.15	0.11	0.1	0.07
concrete	0.02	0.02	0.03	0.03	0.03

The boundary filter for each material was generated by converting the absorption coefficients to per-band reflectance coefficients using the relationship $R = \sqrt{1-\alpha}$. Then, the Yule-Walker method from the ITPP library [81] was used to calculate coefficients for a sixth-order IIR filter which approximated the per-band reflectance. This filter was converted to an impedance filter by $\xi(z) = \frac{1+R_0(z)}{1-R_0(z)}$, which was then used in the boundary update equations for the DWM.

8.4.2 Results

The results are shown in fig. 8.4. The lines labelled "measured" show the measured boundary reflectance, and the lines labelled "predicted" show the theoretical reflectance, obtained by substituting the impedance filter coefficients and angles of incidence into eq. 8.4. Although the waveguide mesh has a theoretical upper frequency limit of 0.25 of the mesh sampling rate, the 3D FDTD scheme has a cutoff frequency of 0.196 of the mesh sampling rate for axial directions. This point has been marked as a vertical line on the result graphs.

8.4.3 Evaluation

The most concerning aspect of the results is the erratic high-frequency behaviour. Even though the cutoff of the 3D FDTD scheme is at 0.196 of the mesh sampling rate, deviations from the predictions are seen below this cutoff in all the presented results. The cause of this error is unclear. One possibility is numerical dispersion, which is known to become more pronounced as frequency increases. However,
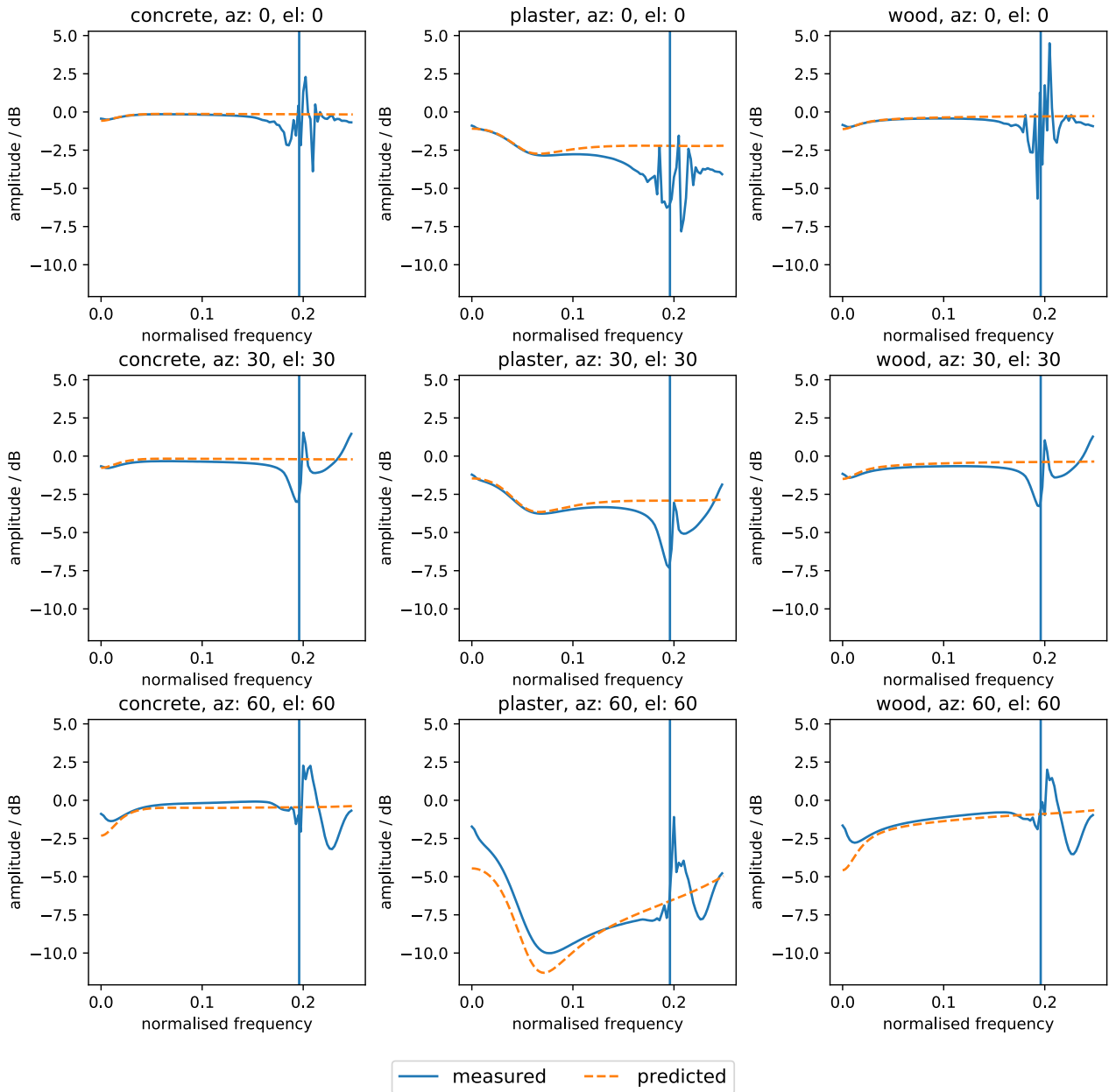
Figure 8.4: Measured boundary reflectance is compared against the predicted reflectance, for three different materials and three different angles of incidence.

the rapid onset of error around the cutoff suggests that the cause is not dispersion alone. Another possible explanation might be extra unwanted reflections in the outputs, although this seems unlikely. The use of a transparent source means that the source node does not act as a reflector, as would be the case with a hard source. In addition, the dimensions of the test domain were chosen to ensure that only reflections from the surface under test are present in the output. The recordings are truncated before reflections from other surfaces or edges are able to reach the receivers. Finally, it seems likely that such interference would affect the entire spectrum, rather than just the area around the cutoff. A final, more likely, explanation for the volatile high-frequency behaviour is that the filters used in this test are of much higher order than those tested in [67], leading to accumulated numerical error.

Whatever the cause of the poor performance at the top of the spectrum, the implications for Wayverb are minor, as the waveguide mesh is designed to generate low-frequency content. If wideband results are required, then the mesh can simply be oversampled. To prevent boundary modelling error affecting the results of impulse response synthesis in the Wayverb app, the mesh cutoff frequency is locked to a maximum of 0.15 of the mesh sampling rate.

Some of the results (especially concrete and wood at 60 degrees) show minor artefacts below 0.01 of the mesh sampling rate, where the measured responses diverge from the predictions. Kowalczyk and Van Walstijn note that some of their results display similar low-frequency artefacts, and suggest that the cause is that the simulated wave-front is not perfectly flat. However, flat wave-fronts are not easily accomplished. The experiments in [67] use large meshes (around 3000 by 3000 nodes, nine million in total) and place the sources a great distance away from the boundary being studied in order to maintain a mostly-flat wave-front. This is only practical because the experiments are run in two dimensions. For the 3D case, no experimental results are given. This is probably because running a 3D simulation on a similar scale would require a mesh of twenty-seven billion nodes, which in turn would require gigabytes of memory and hours of simulation time.

In conclusion, for the most part, the results presented adhere closely to the expected results, with the caveat that the surface reflectance is only accurate at low frequencies, below around 0.15 of the mesh sampling rate. Different absorption coefficients lead to clearly-different reflectance coefficients, which are additionally accurate at multiple angles of incidence. Whilst a more accurate method would be preferable, this model is both fast and tunable, making it a good candidate for boundary modelling in room acoustics simulations.

8.5 Summary

In the image source model, outgoing pressure from a boundary can be found as a function of incident angle and surface attenuation; scattering cannot be modelled. The ray tracer, however, is able to model scattering in two ways. Firstly, the average scattering coefficient across all frequency bands is used to govern the magnitude of random offset of the outgoing ray direction. Secondly, the diffuse rain technique is used to model frequency-dependent scattering.

Boundary models in the waveguide are more involved. Wayverb uses the LRS technique, as it is a better model of physical behaviour than the alternative KW-pipe technique. The LRS model uses special update equations, which contain embedded IIR filters, to calculate pressure values at designated boundary nodes. This is effective for boundaries which are aligned to the mesh, but for curved or angled surfaces the boundary will be quantised into steps. The effect of this quantisation has not been investigated.

The waveguide boundary implementation has been tested to see whether the measured reflectance matches the theoretical surface reflectance. In all tests, the match is reasonably close around the middle of the valid spectrum, between 0.01 and 0.1 of the mesh sampling rate. Outside this range, the results tend to deviate somewhat. As a result, the model crossover frequency in Wayverb has a maximum of 0.15 of the waveguide sampling frequency, and it is recommended to oversample the mesh if highly accurate boundary results are required. It is hoped that the low frequency artefacts will always occur below the audible range, although it is unclear whether this is really the case as tests have only been conducted at a single sampling frequency. Only 1D boundaries have been tested, as it is unclear how to test 2D and 3D boundaries in isolation. Given that all boundary types (1D, 2D and 3D) have the same derivation, it may be enough to assume that if the 1D boundaries function correctly, then the 2D and 3D boundaries will work too. However, this does not rule out the possibility of implementation mistakes in the higher-order boundaries.

Finally, the boundary model has not been tested in the context of an extended waveguide simulation. It would be expected that the absorption of surfaces in a room would have an effect on the overall reverb time in that room. This relationship has not been shown to be true of Wayverb's waveguide. A particularly interesting test would be to set different absorption coefficients in each of the waveguide frequency bands, to estimate Sabine reverb times in each of those bands, and to see whether the measured reverb time in each band matches the predictions. Such a test was not possible due to time constraints, but would help to clarify some of the results shown in the Evaluation.

9 Evaluation

This section describes the Wayverb program, and demonstrates some example simulation results. The simulations are chosen to highlight the behaviour of the simulator with respect to parameters such as reverb time, frequency content, and early reflection times. The project files for each of these tests are included in the Wayverb distribution.

9.1 Features

The Wayverb program has the following features:

- **Hybrid Geometric and Waveguide Simulation**: This is the most important feature of Wayverb, providing the ability to simulate the acoustics of arbitrary enclosed spaces.
- **Load Arbitrary Models**: The model-importing functionality is built on top of the Assimp library, which has support for a wide variety of 3D formats [82]. Care must be taken to export models with the correct scale, as Wayverb interprets model units as metres. The detected model dimensions are shown in the interface, so that model dimensions can be checked, and the model can be re-exported if necessary.
- **Visualiser**: Allows the state of the simulation to be observed, as it changes.
- **Unlimited Sources and Receivers**: Set up any number of sources and receivers. This has the trade-off that the simulation will automatically run once for each source-receiver pair, which will be time consuming when there are many combinations.
- **Multiple Capsules per Receiver**: Each receiver behaves like a set of coincident capsules. Each capsule may model an ideal microphone, or an HRTF ear. Multiple capsules at the same receiver require only a single simulation run, so multi-capsule receivers should be preferred over multiple receivers, wherever possible. For HRTF simulations, the receiver position will be automatically adjusted during the image-source simulation, replicating the interaural spacing of a real pair of ears (see the Image Source Implementation subsection of the Microphone Modelling section). This produces a realistic stereo time-delay effect in the early-reflection portion of the output, aiding localisation.
- **Custom Materials**: Wayverb reads unique material names from the loaded 3D model. Each unique material in the model may be assigned custom acoustic properties, consisting of multi-band absorption and scattering coefficients.
- **Tunable Ray Tracer**: The number of rays is controlled by a quality parameter, which defines the number of rays which are expected to intersect the receiver per histogram interval. Higher quality values will lead to more accurate reverb tails, at the cost of longer processing times. The desired image-source depth can also be varied from 0 to 10, although lower values are

recommended. In the real world, the ratio of scattered to non-scattered sound energy will increase as the impulse response progresses. The image-source model does not account for scattering. Therefore, lower image-source reflection depths are more physically plausible, as the simulation will switch to stochastic ray-tracing (which *does* account for scattering) sooner.

- **Tunable Waveguide**: The waveguide has two modes: a single-band mode which uses the Yule-Walker method to estimate boundary filter parameters, and a multi-band mode which uses "flat" filters. These filters are able to model a given wall absorption with greater accuracy, but only when the wall absorption is constant across the spectrum. The multi-band mode is therefore significantly slower, as it must run the waveguide process several times. It uses the wall absorption from each frequency band in turn, and then band-pass filters and mixes the results of each simulation to find the final output. Both waveguide modes allow the maximum waveguide frequency, and the oversampling factor, to be modified.

The interface of the Wayverb program is explained in fig. 9.1.

9.2 Tests

Some aspects of the Wayverb algorithm have already been tested in previous sections, and so do not require further testing here.

Specifically, the Hybrid Model section compares the waveguide to an ideal image-source model, showing that the output level is correctly matched between models. This test also shows that the modal response of the waveguide matches the "ideal" response for several different values of absorption coefficients, implying that the waveguide and image-source boundary models are consistent.

The Microphone Modelling section shows that the waveguide model is capable of simulating directionally-dependent receivers, with gain dependent on the angle of the incident wave-front.

Finally, the Boundary Modelling section shows that the waveguide boundaries exhibit the expected wall impedance (though with some error, which increases at higher frequencies).

In the tests below, all impulse responses are produced using the Wayverb software. Reverb times are calculated using the Room EQ Wizard [83]. The test projects can be found in the Wayverb repository.

9.2.1 Reverb Times for Varying Room Volumes

This test aims to check that rooms with different volumes produce the expected reverb times. Rooms with different volumes, but the same absorption coefficients and source/receiver positions, are simulated. Then, the RT60 is calculated from the simulated impulse responses, and compared against the Sabine estimate. A close match shows that the change in room volume has the correct, physically plausible effect on the generated outputs.

Three different cuboid rooms with the following dimensions are modelled:

- **small**: $2 \times 2.5 \times 3$ metres
- **medium**: $4.5 \times 2.5 \times 3.5$ metres
- **large**: $12 \times 4 \times 8$ metres

97

Wayverb User Interface

Figure 9.1: The interface of the Wayverb program.

Each room is set to have absorption and scattering coefficients of 0.1 in all bands. The source and receiver are placed 1 metre apart at the centre of each room. The waveguide is used to model frequencies up to 500Hz, using a mesh with a sampling rate of 3330Hz. The image-source model generates reflections up to the fourth order.

The results for the entire (broadband) output are shown in table 9.1. As mentioned above, all reverb times have been found by importing the generated impulse response into the Room EQ Wizard [83], and using the reverb time export function. This feature derives reverb times (EDT, T20, and T30) in accordance with the ISO 3382 specification [45].

Table 9.1: Comparison of Sabine reverb time against measured reverb time for enclosures of three different sizes.

room	Sabine RT / s	measured T20 / s	measured T30 / s
small	0.653	0.663 (1.53% error)	0.658 (0.766% error)
medium	0.887	0.897 (1.13% error)	0.903 (1.80% error)
large	1.76	1.89 (7.39% error)	1.96 (11.4% error)

The results for small and medium rooms are within 5% of the expected reverb time, although the measured T30 of the larger room has an error of 11%. To be considered accurate, the error in the measurement should be below the *just noticeable difference* (JND) for that characteristic. The JND for reverb time is 5%, therefore the simulated reverb time is accurate for the small and medium rooms, although it is inaccurate for the largest room. Increasing the room volume has the effect of increasing the reverb time, as expected.

Now, the results are plotted in octave bands (see fig. 9.2). The results in lower bands, which are modelled by the waveguide, have a significantly shorter reverb time than the upper bands, which are generated geometrically. The higher bands have reverb times slightly higher than the Sabine prediction, while the waveguide-generated bands show much shorter reverb tails than expected. The minimum and maximum reverb times across all octave bands are shown, along with percentage differences, in table 9.2. For all room sizes, the maximum difference in reverb time between the waveguide and geometric models is over 8 times the 5% JND. The difference in reverb times between the waveguide and geometric methods also becomes evident when spectrograms are taken of the impulse responses (see fig. 9.3). In all tests, the initial level is constant across the spectrum, but dies away faster at lower frequencies. In the medium and large rooms, some resonance at 400Hz is seen towards the end of the reverb tail.

Table 9.2: Minimum and maximum reverb times taken from the octave-band results in fig. 9.2.

room	min T30 / s	max T30 / s	percentage difference
small	0.4460	0.6750	40.84
medium	0.6200	0.9337	40.41
large	0.9145	1.978	73.52

In the medium and large tests, the spectrograms appear as though the low-frequency portion has a longer, rather than a shorter, reverb time. However, in the large test, the late low-frequency energy has a maximum of around -100dB, which is 40dB below the level of the initial contribution. The measured T20 and T30 values do not take this into account, and instead reflect the fact that the *initial* reverb decay is faster at low frequencies. The spectrograms show that the waveguide sometimes resonates for an extended period at low amplitudes.
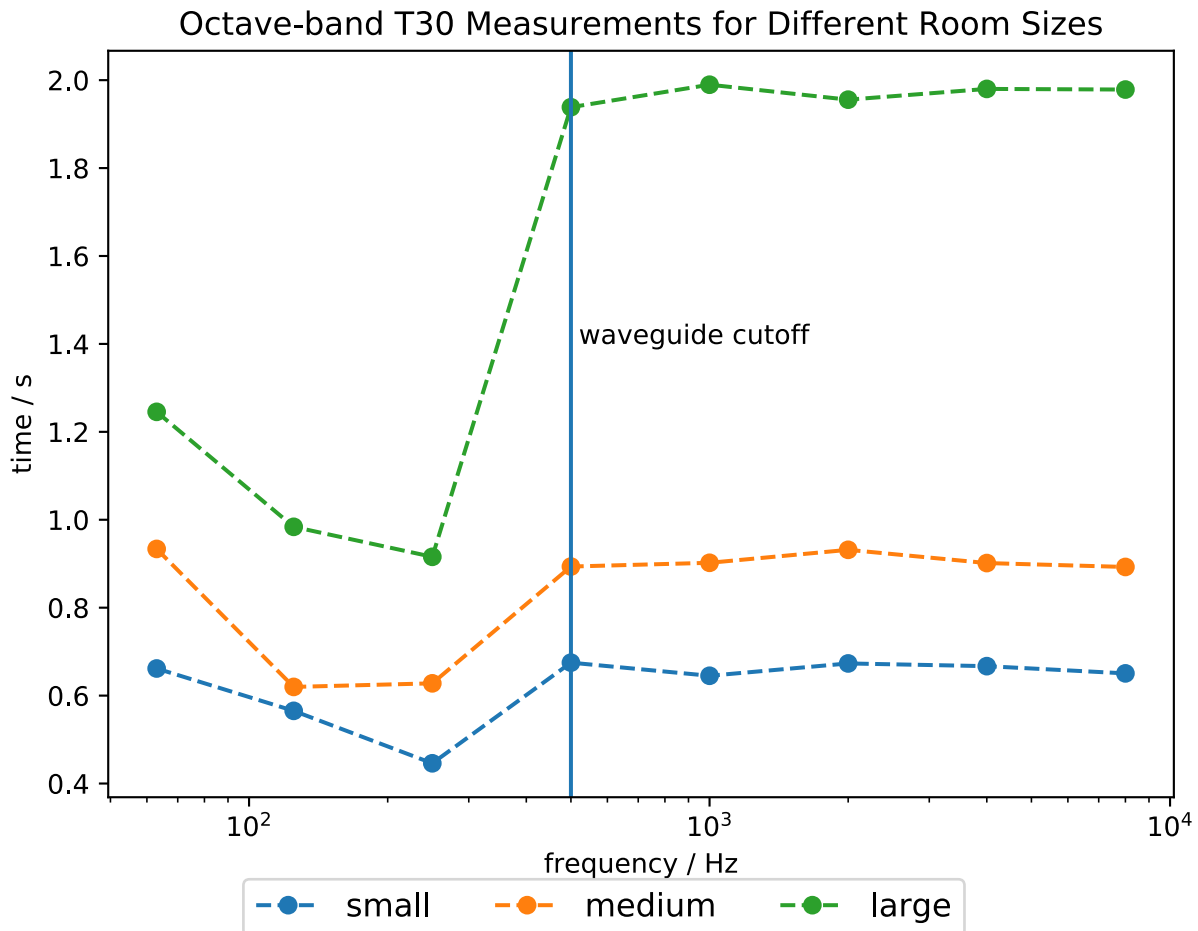

Figure 9.2: T30 in octave bands, for simulated rooms of different sizes, calculated from the measured impulse responses.

This result is difficult to explain. A shorter reverb time indicates that energy is removed from the model at a greater rate than expected. Energy in the waveguide model is lost only at boundaries, so the most likely explanation is that these boundaries are too absorbent. It is also possible that the microphone model causes additional unexpected attenuation.

Further tests (not shown) of the three rooms were carried out to check possible causes of error. In one test, the Yule-Walker-generated boundary filters were replaced with filters representing a constant real-valued impedance across the spectrum, to check whether the boundary filters had been generated incorrectly. In a second test, the modelled omnidirectional microphone at the receiver was removed, and the raw pressure value at the output node was used instead, to check that the microphone was not introducing undesired additional attenuation. However, in both tests, similar results were produced, with reverb times significantly lower than the Sabine prediction. The boundary and microphone models do not appear to be the cause of the problem.
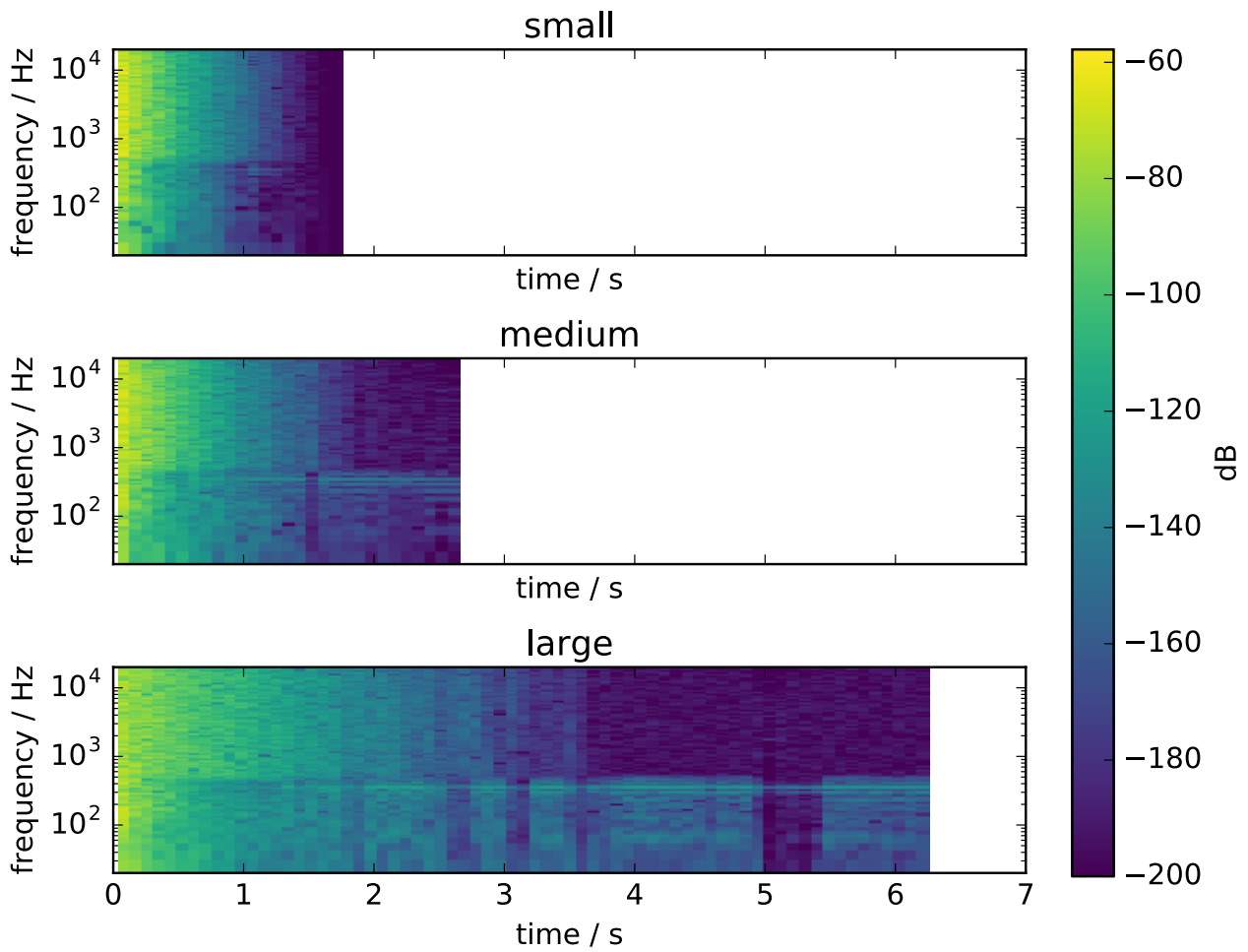
Figure 9.3: Spectrograms of impulse responses obtained from different room sizes.

The reverb-time test at the end of the Hybrid Model section shows that the waveguide reverb times match the reverb times of the exact image-source model, which will be close to the analytical solution in a cuboid room. The close match to the almost-exact image-source model suggests that the waveguide and boundary model have been implemented correctly. Additionally, the tests in the Boundary Modelling section show that wall reflectances generally match predicted values to within 1dB for three material types and angles of incidence, at least in the band below 0.15 of the mesh sampling rate.

Given that in all previous tests the waveguide behaves as expected, it is likely that the Sabine equation is simply a poor predictor of low-frequency reverb times in these tests. This is a reasonable suggestion: the Sabine equation assumes that the sound field is diffuse, which in turn requires that at any position within the room, reverberant sound has equal intensity in all directions, and random phase relations [84]. This is obviously untrue in a cuboid at low frequencies, where the non-random phase of reflected waves causes strong modal behaviour due to waves resonating between the parallel walls of the enclosure.

If multiple simulations were run with randomised source and receiver locations, the low-frequency diffuse-field reverb time could be approximated by averaging the results. It may be that the low reverb time in the test above is entirely due to the particular placement of the source and receiver, and that the *average-case* waveguide output would match the Sabine estimate more closely. If the waveguide *does* match the predicted reverb times in the average case, then this would mean that further research should focus on reducing the impact of the mismatch between the outputs of different models, rather than on improving the waveguide model itself. However, there was not time to run such a test in the course of this project.

Further testing is also required to locate the exact cause of the low-amplitude resonance in the waveguide. Although low-frequency resonant behaviour is to be expected in the tests presented here, it is surprising that all room-sizes tested displayed some localised resonance at around 400Hz (see fig. 9.3). The fact that the resonant frequency is the same across all rooms suggests that this is not caused by constructive interference of room modes, but rather some implementation deficiency in the waveguide. Perhaps the first component to check would be the waveguide boundaries: the results in fig. 8.4 showed that the boundary implementation can introduce unpredictable artefacts at the top end of the valid bandwidth. Therefore, it may be that the artefacts present in these results can be removed simply by increasing the waveguide sampling rate relative to the crossover frequency.

9.2.2 Reverb Times for Varying Absorptions

This test simulates the same room with three different absorption coefficients. The "medium" room from the above test is simulated, again with the source and receiver placed 1 metre apart in the centre of the room. Scattering is set to 0.1 in all bands. The absorption coefficients are set to 0.02, 0.04, and 0.08, corresponding to Sabine predictions of 4.43, 2.22, and 1.11 seconds. The results are summarised in table 9.3, with octave-band T30 values shown in fig. 9.4, and spectrograms of the outputs are shown in fig. 9.5.

Table 9.3: Comparison of Sabine reverb time against measured reverb time for an enclosure with varying boundary absorptions.

absorption	Sabine RT / s	measured T20 / s	measured T30 / s
0.02	4.433	4.295 (3.113% error)	4.283 (3.384% error)
0.04	2.217	2.210 (0.3157% error)	2.219 (0.09021% error)
0.08	1.108	1.126 (1.625% error)	1.156 (4.322% error)

It can be seen in fig. 9.4 that low-frequency bands have shorter reverb times than high-frequency bands, as in the previous test. The minimum and maximum reverb times from each test are shown in table 9.4, alongside percentage differences. In all cases, the difference in reverb time between the bands with longest and shortest decay is more than 9 times the JND. However, the broadband reverb time responds correctly to the change in absorption coefficients. All broadband results are within the 5% JND for reverb time.

Table 9.4: Minimum and maximum reverb times taken from the octave-band results in fig. 9.4.

absorption	min T30 / s	max T30 / s	percentage difference
0.02	2.604	4.338	49.98
0.04	1.395	2.255	47.16
0.08	0.7093	1.179	49.72

The spectrograms in fig. 9.5 do not show the same resonance at 400Hz as the previous test results. Given that models of different sizes exhibited resonance at the same frequency, but that changing the surface absorption causes the resonant frequency to move, it seems very likely that the resonant artefact is caused by the boundary model. However, further tests are required in order to be certain.

9.2.3 Direct Response Time

The "large" room above is simulated again, but with the source and receiver positioned in diagonally opposite corners, both 1 metre away from the closest walls in all directions. The generated impulse response is compared to the previous impulse response, in which the source and receiver are placed 1 metre apart in the centre of the room. Broadband reverb time statistics are computed with the Room EQ Wizard, and displayed in table 9.5.

Table 9.5: Comparison of measured T20, T30, and EDT for a large room with near and far microphone spacings.

test	T20 / s	T30 / s	EDT / s
near (1.00m spacing)	1.907	1.963	1.593
far (11.8m spacing)	1.887	1.962	1.763
percentage difference	**1.054**	**0.05100**	**10.13**

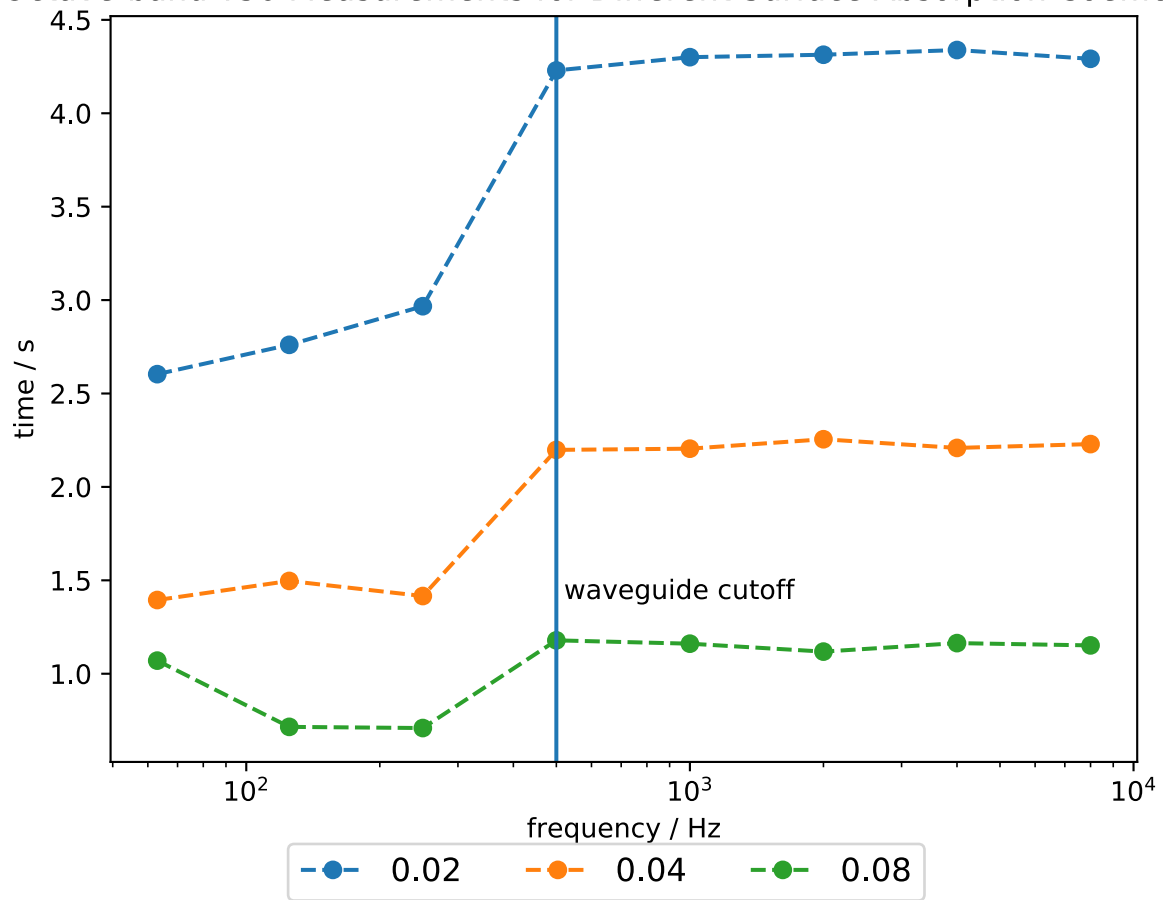Figure 9.4: T30 in octave bands, for simulated rooms with different surface absorption coefficients, calculated from the measured impulse responses.
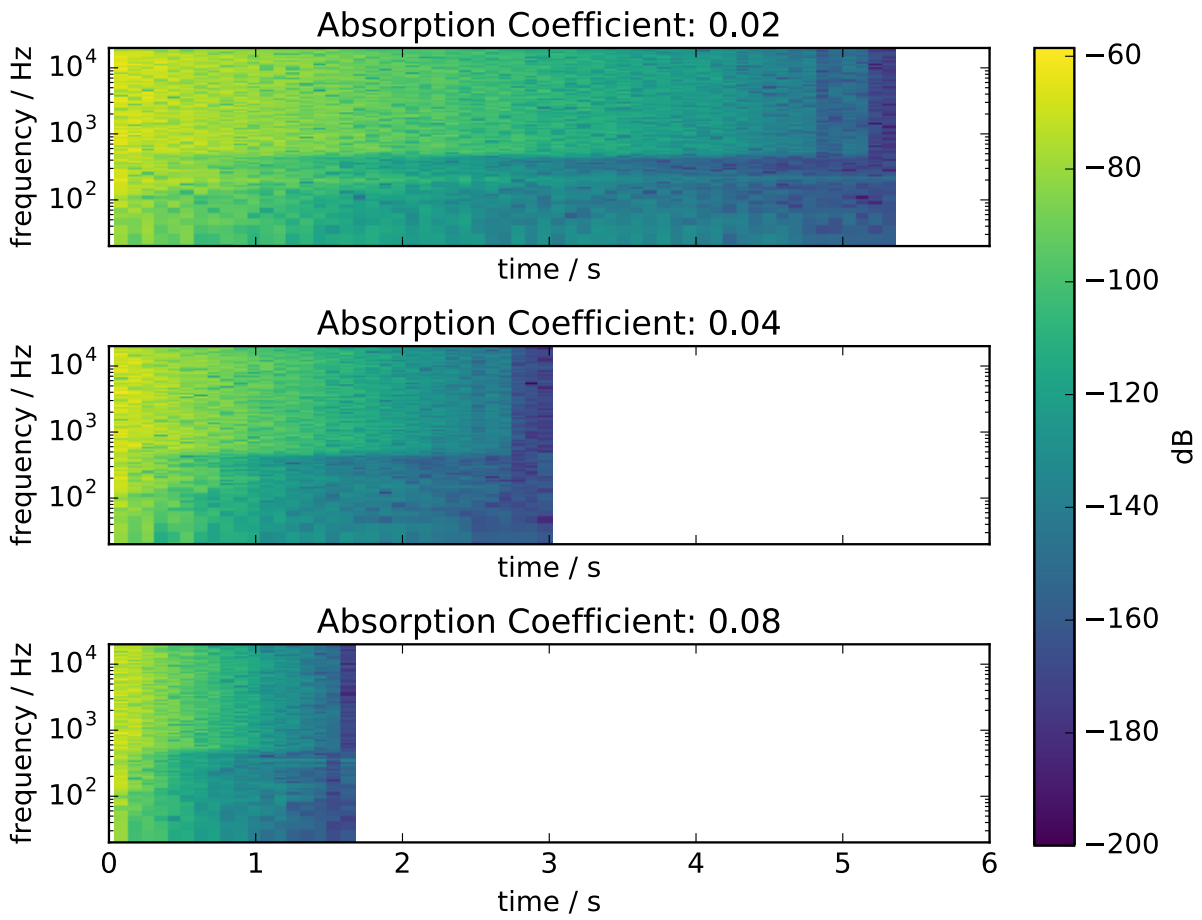
Figure 9.5: Spectrograms of impulse responses obtained from simulating the same room with different absorption coefficients. Note that the low-frequency content has a shorter decay time than the high-frequency content.

According to Kuttruff [33, p. 237], early decay time will be strongly influenced by early reflections, and so will depend on the measurement position. Meanwhile, the overall reverb time should not be affected by the observer's position [33, p. 229]. This appears to be true of the results in table 9.5. The differences between the reverb time measurements for the two source-receiver spacings are below the 5% JND, but the difference in early decay time is more than twice the JND. This suggests that the relative levels of the early and late reflections in the simulation change depending on receiver position, as expected.

Sound is simulated to travel at 340m/s, so when the source and receiver are placed 1m apart, a strong impulse is expected after $1/340 = 0.00294$ seconds. In the new simulation, the source and receiver are placed $\sqrt{10^2 + 2^2 + 6^2} = 11.8$m apart, corresponding to a direct contribution after 0.0348 seconds.

When the source is further away, the direct contribution may not be the loudest part of the impulse. As the distance from the source r increases, the energy density of the direct component decreases proportionally to $1/r^2$. However, the energy density in an ideally-diffuse room is independent of r. At a certain distance, known as the *critical distance* r_c, the energy densities of the direct component and reverberant field will match. Beyond the critical distance, the energy of the direct component will continue to decrease relative to the reverberant field [33, pp. 146–147].

The initial contributions of the two simulated impulse responses are shown in detail in fig. 9.6. The predicted effect of a larger separating distance is observed: The first and second order early reflections arrive very shortly after the direct response. Many of these reflection paths cover the same distance, and so arrive at the same time. These contributions are added, giving an instantaneous energy which is greater than that of the direct contribution. When the source and receiver are placed close together, the direct contribution has the greatest magnitude, and the early contributions occur at a lower rate (have greater temporal spacing) than when the source-receiver spacing is larger. Intuitively, it appears that increasing the distance between source and receiver has the effect of lowering the ratio of energy densities between direct and reverberant contributions.

9.2.4 Obstructions

Early reflection behaviour seems to be correct in simple cuboid models, where there is always line-of-sight between the source and receiver. The behaviour in more complex models, in which the source and receiver are not directly visible, must be checked.

The simulated space is a simple vault-like model, similar to a small hall, but broken up by regularly repeating pillars. The source and receiver are positioned seven metres apart, with their view obstructed by two pillars. If there were no obstruction, a strong direct impulse would be expected after 0.0206 seconds. However, the pillars should block this direct contribution. The model is shown in fig. 9.7.

In the real world, objects with areas of a similar or greater order to the incident wavelength cause diffraction effects [33, p. 59]. The result of diffraction is that an additional "diffraction wave" is created at the edge of the object, which radiates in all directions. In the vault model, the edges of the pillars should cause diffraction, and in this way, some energy should be scattered from the source to the receiver. This energy will arrive slightly after the direct contribution would have, but before the first early reflection. The shortest possible path from source to receiver which travels around the pillars
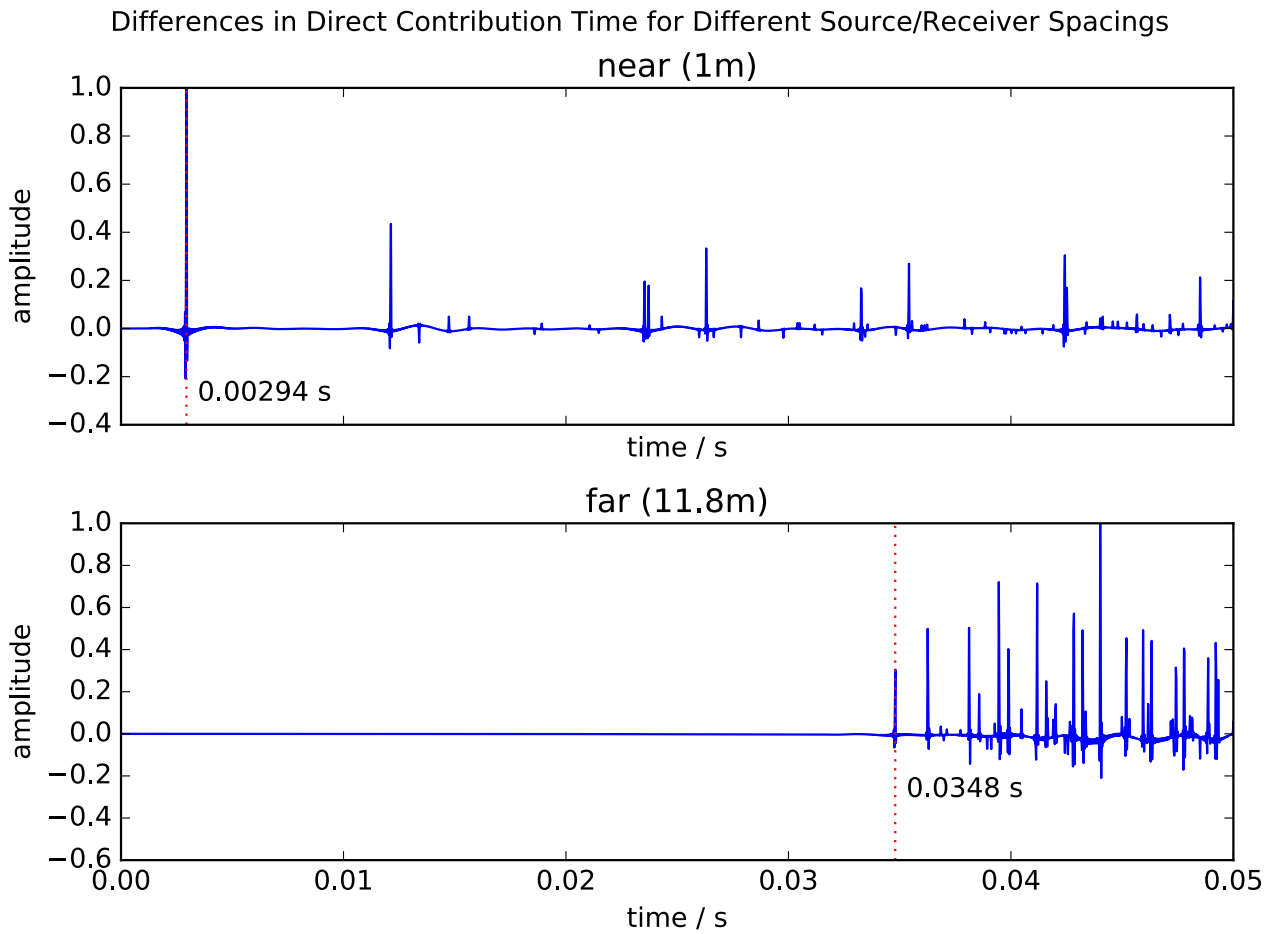
Figure 9.6: Larger distances between the source and receiver result in delayed initial contributions. This figure shows the first 0.05s of the outputs of two simulations in which the source and receiver are separated by 1m and 11.8m. The dotted red lines show the expected time of the initial response.
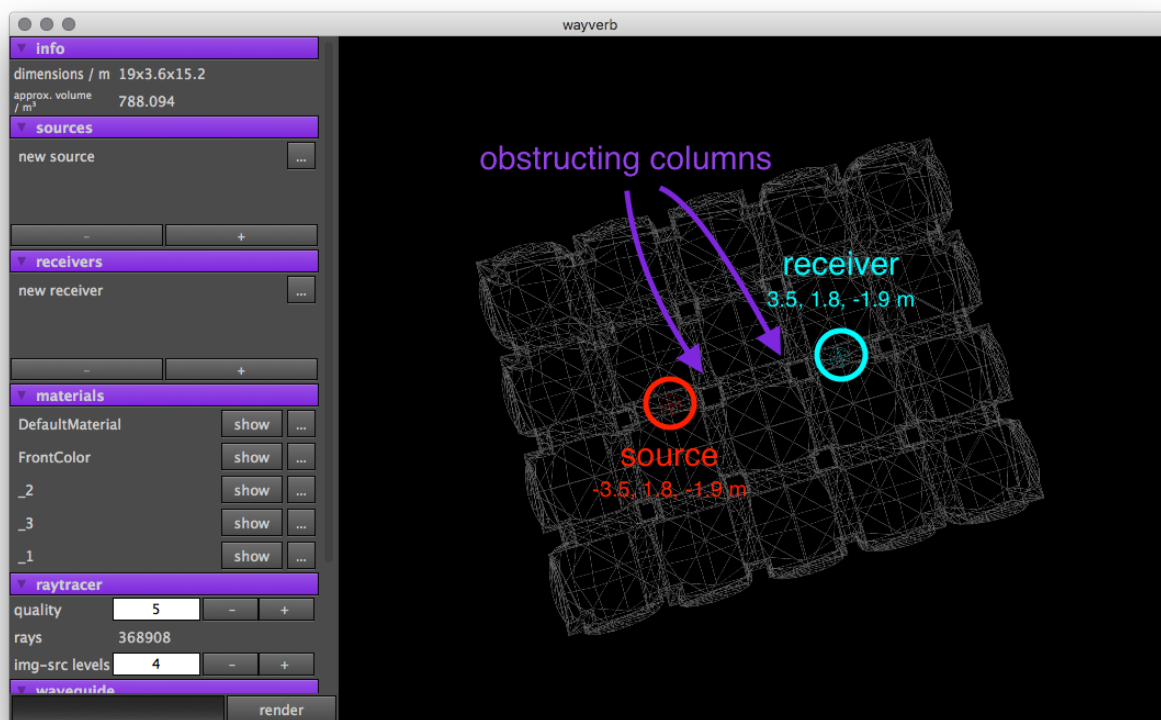
Figure 9.7: The testing set-up, showing the pillars blocking the line-of-sight between the source and receiver.

has a length of 7.12m, corresponding to a time of 0.0209s. Though the image-source and ray tracing models are not capable of modelling diffraction effects, the waveguide model inherently models this phenomenon. Therefore, the impulse response should record a low-frequency excitation at around 0.0209s.

The impulse response graph (fig. 9.8) shows that low frequency diffraction is in fact recorded. The waveform shows a low-frequency ripple starting at around 0.02 seconds, which occurs before the first impulsive contribution from the geometric models. This is mirrored in the spectrogram, which shows that the low-frequency waveguide contribution (up to 500Hz) has more energy than the geometric contribution at the very beginning of the impulse response. Though the behaviour of the waveguide is physically correct, it highlights the main shortcoming of the hybrid algorithm. For simulations such as this, which rely on the effects of wave phenomena, the physical modelling of the waveguide conflicts with the approximate nature of the geometric algorithms, causing an obvious divide or disconnect between the low and high frequency regions in the output. The impulse response shown here is physically implausible, making it unsuitable for realistic, high-quality reverb effects.

9.2.5 Late Reflection Details

Having checked the behaviour of early reflections, now the late-reflection performance must be checked. The nature of the ray tracing process means that fine detail (below 1ms precision) is not captured. However, it should be possible to observe reverb features on a larger scale, such as distinct echoes from a long tunnel.
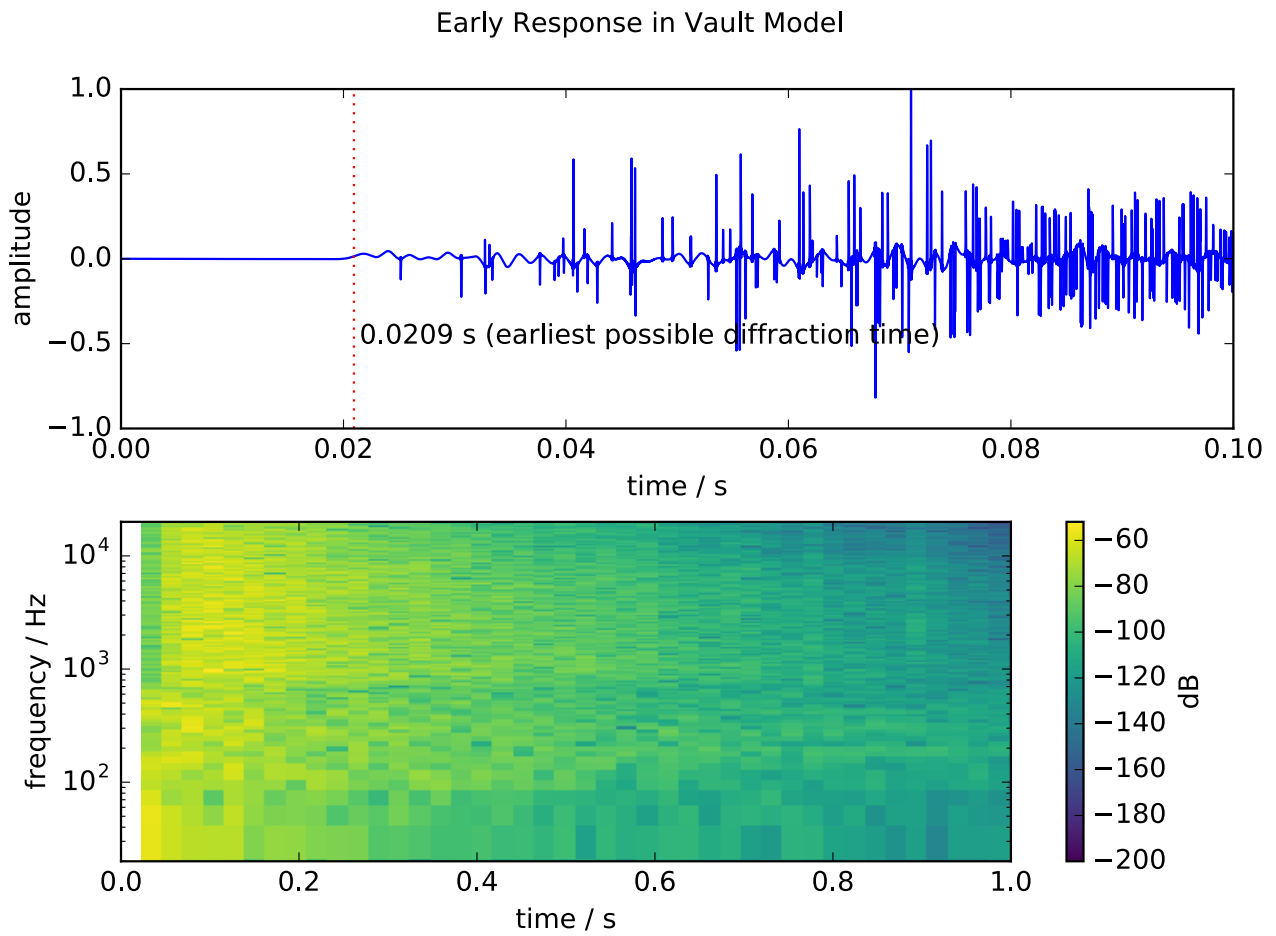
Figure 9.8: The early part of the vault impulse response. Low frequency diffraction from the waveguide is detected before the first image-source contribution.

A cuboid with dimensions $4 \times 7 \times 100$ metres is simulated. The receiver is placed exactly at the centre of the model, with the source positioned two metres away along the z direction. The output should contain a direct contribution at 0.00588s, and some early reflections from the nearby walls. The reverb tail should contain strong echoes every 0.294s, as the initial wave-front reflects back-and-forth between the two end walls.

The tunnel is modelled using absorption coefficients of 0.03 in the bottom five bands, then 0.04, and 0.07 in the highest two bands. The scattering coefficients are set to 0.1 in all bands. This scattering should cause echoes in the reverb tail to be "smeared" in time. To check the effect of the scattering coefficients, the same test is also run using scattering coefficients of 0 in all bands.
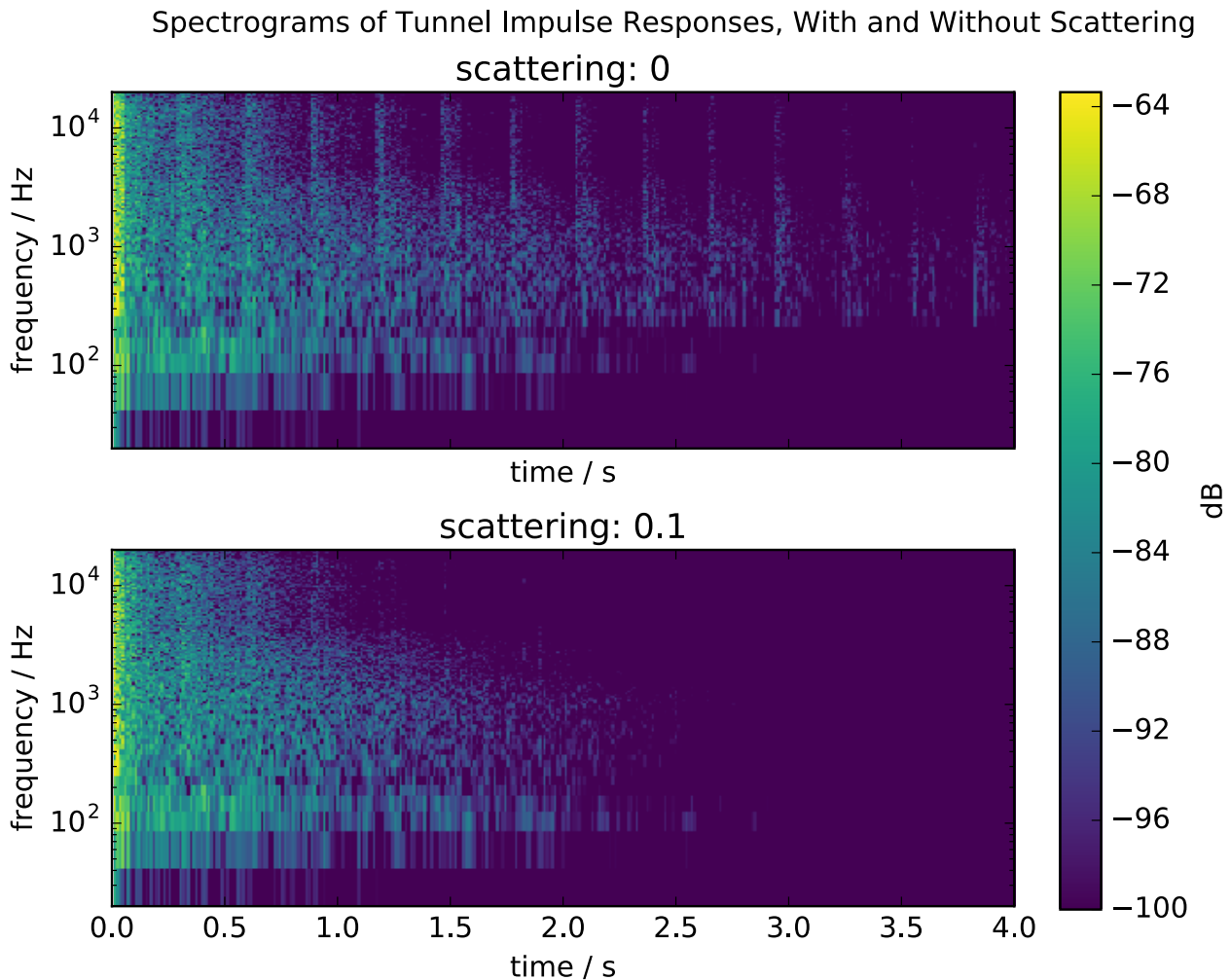
Figure 9.9: Spectrograms of the tunnel impulse responses with different scattering levels. Note that the first 3 echoes are clear in both responses, but the scattered response quickly becomes less distinct, while the response with no scattering has clear echoes which die away more slowly.

The spectrograms show that there are clear increases in recorded energy, occurring approximately every 0.3 seconds after the initial onset. Each echo is followed by its own decay tail. In the case of the non-scattering simulation, the echoes are clearly separated, with very short tails. When the scattering is increased, the tails become longer, and the individual echoes become less defined. This is expected: when there is no scattering, rays travelling in any direction other than along the length of the tunnel will bounce between the walls many times, attenuated each time. These contributions may arrive at any time at the receiver, however, their amplitude will be greatly reduced. The rays travelling directly

between the ends of the tunnel will be reflected fewer times, will lose less energy, and will produce louder regular contributions. When scattering is introduced, the ray paths are less regular, giving less correlation between the number of reflections and the time at which the ray is recorded. This in turn leads to a more even distribution of recorded energy over time.

9.2.6 Directional Contributions

To test that microphone modelling behaves as expected, two cardioid microphones are placed in the exact centre of the "large" cuboid room, facing towards and away from the source, which is positioned at a distance of 3m along the z-axis.

An ideal cardioid microphone has unity gain in its forward direction, and completely rejects waves incident from behind. In this test, the direct contribution from the source originates directly in front of one receiver and behind the other, so it should be present only in the signal measured at the capsule facing the source. The next-shortest path from the source to the receivers is caused by reflections from the floor and ceiling, both with path lengths of 5m, corresponding to 0.0147s. These paths should cause a strong contribution in the capsule facing the source, and a quieter contribution in the capsule facing away. There is a path with a length of 11m which should strike the receivers from the opposite direction to the source. At this time, there should be a large contribution in the away-facing capsule, and a small contribution in the towards-facing capsule.

These expectations are reflected in the results, shown in fig. 9.10. The impulse response is silent in the away-facing capsule at the time of the direct contribution, 0.00882s. The first significant level is recorded at 0.0147s, with a loud contribution seen at 0.0324s. In the capsule facing towards the source, the first two contributions are loud, with no contribution at 0.0324s. These results are consistent across the spectrum, indicating that the microphone model is matched between both simulation methods.

9.2.7 Binaural Modelling

Finally, the binaural model is tested. A concert hall measuring approximately $33 \times 15 \times 50$ metres is simulated (shown in fig. 9.11), with wall absorptions increasing from 0.25 in the lowest band to 0.67 in the highest band, and scattering at 0.1 in all bands. The source is placed in the centre of the stage area, and the receiver is placed 10m along both the x- and z-axes relative to the source. The receiver is oriented so that it is facing directly down the z axis, meaning that the source is 14.1m away, on the left of the receiver.

The simulation produces output impulse responses for both the left and right ears. All of Wayverb's simulation methods (image-source, ray-tracing and waveguide) use HRTF data to produce stereo effects based on interaural level difference. The image-source method additionally offsets the receiver position to produce interaural time difference effects, so in the outputs, slight time differences in the early reflections between channels are expected, and small level differences should be seen throughout both files.
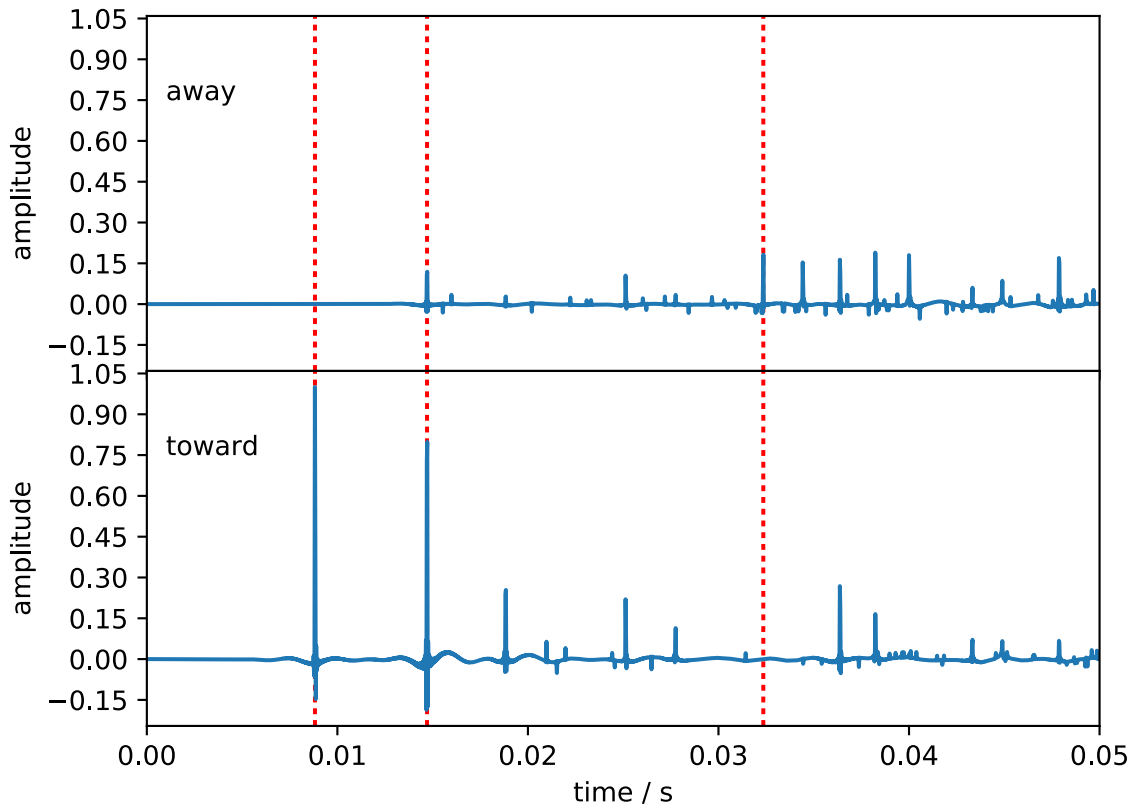
Figure 9.10: The response measured at cardioid microphones pointing towards and away from the source. The times marked with dotted lines, from left to right, are the direct contribution time, the first reflection time, and the time of the first contribution from the opposite direction to the source.
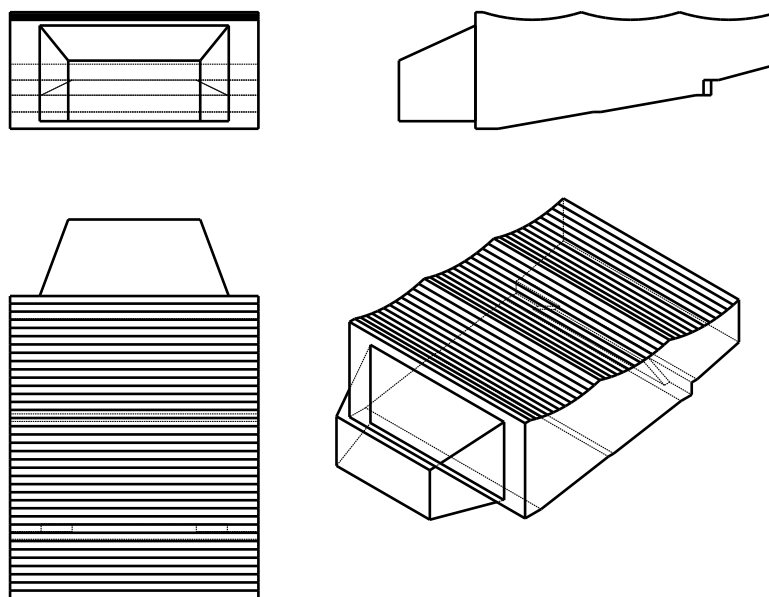

Figure 9.11: The concert hall model.

In particular, the direct contribution would normally arrive at 14.1/340=0.0416s. However, the left "ear" is actually slightly closer to the source, and the right ear is slightly further away, which means that the first contribution should be at 0.0414s in the left channel, and 0.0418s in the right. The right ear is obstructed by the listener's head, and should therefore have a reduced level relative to the left ear.

The expected behaviour is observed in the outputs, which are shown in fig. 9.12. The earliest contribution in the left channel occurs at 0.0414s, and has a greater level than the right channel's contribution at 0.148s. The left-channel early reflections have an overall higher level than the early reflections in the right channel. However, as the impulse response progresses and becomes more diffuse, the energy levels equalise between channels.

Comparison of Left and Right Ear Responses
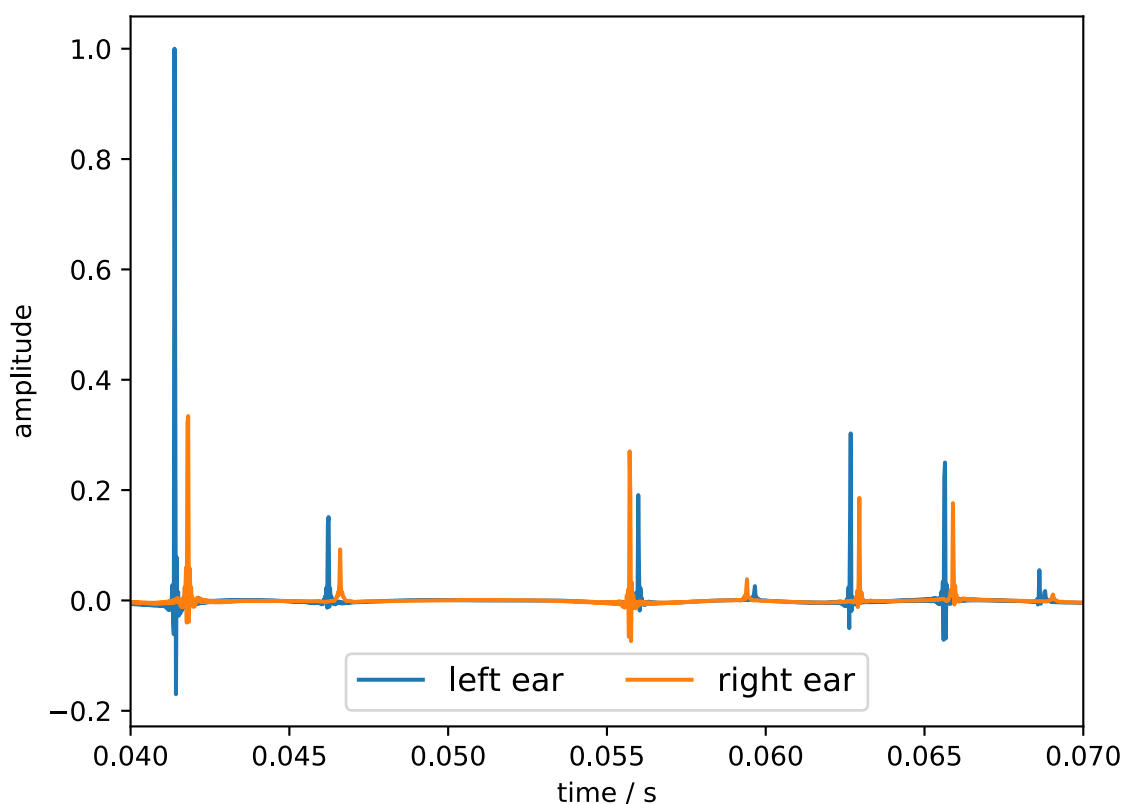
Figure 9.12: Comparison of left- and right-ear responses, when the source is placed to the left of the receiver. Note the amplitude and time differences between the early reflections. The first contribution, in particular, is quieter in the right ear because it is occluded by the listener's virtual head.

9.3 Summary

A simulation program, Wayverb, has been implemented and its feature set described. Tests have been run to gauge how closely Wayverb's simulation matches predicted behaviour. These tests have been designed in accordance with the primary research goal, which was physical plausibility. The results show that changes in parameters cause appropriate changes in the outputs:

- Larger rooms have longer reverb times than smaller rooms.
- Rooms with high absorption produce shorter reverb times than more reflective rooms.
- When the source and receiver are moved apart, the ratio of direct to reverberant sound decreases.
- Obstructions in the room cause diffraction.
- Long reflective rooms produce distinct echoes.
- Receiver modelling produces appropriate direction-dependent attenuation.

The main downfall of Wayverb's simulation method is an obvious mismatch between the low- and high-frequency regions of the output. The room-size and absorption tests showed that reverb times above and below the model crossover frequency differ by significantly more than the JND. Also, the simulation as a whole is not accurate: in the largest room, the broadband reverb time differed from the Sabine prediction by more than the JND. Although the effects of *changing* parameters are plausible, the overall simulation results are not.

Tests have not been presented regarding the two lesser research goals of efficiency and accessibility. Ideally, benchmarks and usability tests would have been conducted. The shortcomings of the test procedure, and the implications thereof, are analysed in the following chapter.

10 Conclusion

10.1 Analysis

10.1.1 Testing Procedure

Due to time constraints, the tests of Wayverb, presented in the Evaluation, are not exhaustive. Only one of the research goals, namely plausibility, is examined. However, the examination does not consider all aspects of this goal. Ideally a more thorough testing procedure, such as that used in [26], would have been followed. That paper presents test results for the boundary performance, modal content, diffraction, and reverb times of acoustic models including FDTD, FEM, image-source model and *beam-tree acoustic radiance transfer*. A wide range of results for each test is shown: the modal response of the FDTD model is compared to that of the image source model for six different reflection coefficient values, and the reverb time predicted by each model is shown for a range of ten different reflection coefficient values. This level of detail gives a clear view of the models' performance. Due to the narrow range of parameters tested with Wayverb, it is possible that the implementation may exhibit issues for some combination of parameters not presented here. It would be desirable to test a wider range of parameters to find out whether Wayverb performs appropriately under all circumstances.

The summaries of the chapters concerning microphone and boundary models both describe how the components in question work in isolation. However, the models have not been tested in the context of a long-running simulation. In particular, it has not been shown how either model affects the overall frequency response and reverb time in the waveguide. The lack of tests in these areas may hide latent issues in Wayverb's implementation.

All tests presented throughout this paper use approximations to find the desired or target results. For instance, the test of the waveguide modal response uses the image-source method as a target, and the reverb time tests in the Evaluation compare the simulation results against the Sabine reverb time. These approximations have shortcomings, as discussed in the relevant sections. It would be more useful to directly compare the results obtained against experimentally-obtained IRs of real locations, which would have no such shortcomings. If the simulated results were shown to be within a "just noticeable difference" of the real results, this would be strong evidence of the simulation's suitability for prediction of architectural acoustics. These tests would be time consuming, not just because impulse responses would have to be recorded in a physical location, but also because that same location would need to be reconstructed virtually, using the same dimensions and surface characteristics. In particular, measuring the absorption and scattering coefficients for each material in the room would require access to a specialised reverberation chamber, or these coefficients would have to be approximated using published values.

As an extension to modelling physical locations, it may be instructive to conduct listening tests. Such tests might aim to test whether listeners can tell apart simulation results and experimentally-obtained IRs. If the simulation results were shown to be indistinguishable from real IRs, this might indicate that Wayverb is suitable for realistic sound-design applications. It might also be useful to conduct listening tests to gauge the (subjective) perceived sound quality of simulation outputs. A simulation program that sounds realistic but "bad" is not useful when producing music that is designed to be enjoyed. If it could be shown that listeners perceived the sound quality to be at least as good as real IRs, this would make a good case for Wayverb's utility in a sound designer's toolbox.

Wayverb's results could be compared to the outputs of other similar modelling programs, both through acoustic parameters like T30, and using listening tests. As noted in the Context chapter, Wayverb is, at time of writing, the only publicly-available acoustics-modelling tool that uses both geometric and waveguide models. Without comparing against other tools, it is difficult to say whether this hybrid approach yields tangible sound quality gains over a more traditional single-model approach.

The goals of efficiency and usability were not tested. In terms of efficiency, it would be useful to show how simulation time varies with parameters such as room size, average surface absorption, and model crossover frequency. The simulation speed could also be compared to competing software packages. Usability is more difficult to test, but a sensible starting point would be to simply observe some creative professionals using the software (in order to find common points of confusion or frustration) and to ask for feedback.

As it stands, the tests presented in the Evaluation do not have sufficient scope in order to make claims about Wayverb's overall "usefulness" as a software tool. Further tests which would help to gauge Wayverb subjectively, and in the context of competing packages, have been suggested.

10.1.2 Evaluation of Project Goals

10.1.2.1 Plausibility

All models presented in the Evaluation perform as expected with regards to changes in room size and shape, material coefficients, source and receiver positions, and receiver microphone type. Reverb features such as distinct late echoes can be generated, and stereo effects, relying on both interaural time- and level-difference, can be created. The simulation responds appropriately to changes in parameters. For example, increasing the absorption coefficients of surfaces leads to a decrease in overall reverb time, and increasing the volume of the simulated room increases the reverb time. In these respects, the Wayverb program achieves the goal of physical plausibility.

Although parameter changes have the expected effects, the simulation is not entirely successful. The main drawback of the presented implementation, evident in several of the tests, is that the geometric and wave-based models have distinct behaviours. In the room-size and material tests, there are markedly different reverb times between the ray-tracer and waveguide outputs; and in the obstruction test, the waveguide exhibits diffraction behaviour which is not mirrored in the geometric output. These differences lead to obvious discontinuities in the frequency response, which persist despite calibrating the models to produce the same sound level at the same distance, and implementing matching boundary models in all models.

Some differences are to be expected. The primary reason for implementing multiple simulation methods is the relative accuracy of the different methods. Geometric algorithms are known to be inaccurate at low frequencies, a shortcoming that the waveguide was designed to overcome. However, the magnitude of the difference in reverb time between the waveguide and ray-tracer outputs (evident in fig. 9.2) suggests that there are errors or false assumptions in the implementation. Even though the different components of the waveguide (such as the boundary model, the source injection method, and the microphone model) appear to work in isolation, the interactions between these components are complex. Further testing of the waveguide as a whole, with all these components enabled, is required in order to find the cause of its lower-than-expected reverb times.

Even if further testing were somehow to reveal that the implementation is correct, the current simulation results are useless if there is an obvious disconnect between high- and low-frequency outputs. For applications like music and sound design, where it is important that the generated responses are believable, the current algorithm is unusable. Even if the low-frequency response by itself accurately represents the modelled space, responses will sound *more* artificial if there is a rapid change in the frequency response when combined with the geometric output. Indeed, it is preferable that the frequency response not contain obvious discontinuities, even if this necessitates a reduction in overall accuracy.

Practical solutions to this problem are unclear. Ideally, the entire simulation would be run with the waveguide method, but this is impractical for all but the smallest simulations. Another option is to reduce the audible impact of the crossover between the waveguide and geometric outputs, simply by increasing its width. This has the dual drawbacks of decreasing the low-frequency accuracy, while also requiring a higher waveguide sample-rate, which is computationally expensive. Alternatively, the geometric algorithms may be extended, to account for effects such as diffraction, with the goal of minimising the differences between the wave and ray-based methods. This solution would maintain (or even improve) the accuracy of the simulation, but would again increase its cost. Finally, the goal of low-frequency accuracy could be abandoned, and the entire spectrum modelled using geometric methods. However, this would prevent important characteristics such as modal behaviour from being recorded.

10.1.2.2 Efficiency

The efficiency goal put forward in the project aims stated that simulations should, in general, take less than ten minutes to run. This held true for the test cases presented in the Evaluation. However, the render time is still long enough to be distracting, and the user experience could be greatly improved by decreasing its duration. The reason for the long render times is simple: the majority of the development time on Wayverb was spent trying to ensure that each model component functioned correctly. By the time the components were "complete" there was little time left for optimising.

The length of the render time may be especially problematic in creative contexts. When producing music, it is important to be able to audition and tweak reverbs in order to produce the most appropriate effect. With long rendering times, this auditioning process becomes protracted, which impacts the productivity of the musician. In addition, if the program uses the majority of the machine's resources while it is running, then this prevents the user from running it in the background and continuing with other tasks in the meantime. Usability tests would be required to find how problematic the render times are in practice.

The simulation methods have been implemented to minimise average-case computational complexity wherever possible, but both the ray-tracing and waveguide processes are limited to a complexity of $O(n)$ where n refers to the number of rays or nodes required in the simulation. Any further performance improvements may only be gained by improving the per-ray or per-node processing speed. This is certainly possible, but would yield relatively small improvements to the simulation speed. It may be equally valid to simply wait for hardware with increased parallelism support: a machine with twice as many graphics cores will run the program twice as fast. Such machines are likely to be commonplace in two or three years. Therefore, a better use of time would be to spend those two years focusing on the algorithm's functional problems rather than optimisation. Although it is disappointing to fall back on the argument that hardware improvements will obviate the need for software efficiency improvements, note that Wayverb is uniquely well-placed to benefit from hardware improvements going forward. Newer computing platforms are generally more powerful due to an increase in the number of processor cores rather than an increase in overall clock speed, and Wayverb by design can scale across all available GPU cores.

10.1.2.3 Accessibility

Considerations regarding the accessibility of the program were limited to the design of the graphical interface. The target demographic of creative users was considered in several ways.

The workflow is designed to be intuitive. When the app is opened, it asks for a CAD file to load. Once a file has been selected, a list of settings will open on the left side of the window, along with a graphical view of the 3D space. The user can simply work down the list of settings, starting with the most important parameters: source and receiver placement, and surface settings. At the bottom of the list is a large "render" button, which will open an exporter dialog box when clicked. There are no unlabelled icons or options hidden in menus; the entire simulation is controlled from a single window. This makes it simple to get started with the app.

The controls themselves are simple and self-explanatory. The number of rays is controlled by a single "quality" parameter, as opposed to allowing the user to specify an exact number of rays. The allows the user to think in terms of the output quality. If the ray number was directly exposed as a control, it may not be clear how this parameter would affect the output. Similarly, the surface settings dialog box is set out like a multiband EQ (with which users will be familiar), which provides hints as to the effect of the absorption in each band. Presets are also provided to give users a starting point when designing their own materials.

The UI of the program was not only developed with users in mind. In the early stages of the project, the 3D display of the room was used for debugging aspects of the simulation algorithms, such as ray scattering, boundary node placement, and waveguide node pressures over time. It was later extended to allow positioning the sources and receivers through mouse interaction, and all of the debugging options were removed. Now, other than displaying source and receiver positions, and the different surface regions, this display is of limited use. The focus on debugging meant that too little time was spent polishing the end-user experience in this part of the interface. In particular, the material and source/receiver controls might be considerably more intuitive if they could be accessed from the 3D view. For example, specific surfaces could be modified by right-clicking them on the 3D model. Also, a receiver's direction and polar pattern could be adjusted from controls which appeared upon hovering. This would provide a more unified experience than the current "menus and dialogs"-based approach.

As mentioned above, user tests would be required in order to evaluate the accessibility of the current interface. Although the simulation behaves plausibly in many respects, this may not translate into an intuitive control scheme. It is unknown whether a user with limited acoustics experience would be able to create a reverb with the characteristics they desire. A potential issue for such users is that familiar controls from parametric reverbs, such as reverb time and EQ, are not directly available in Wayverb. This is not only disorienting, it also necessitates a time-consuming workflow for some common tasks. If the user desires a longer reverb time, they cannot simply move a slider: they must open the 3D model in a CAD program, adjust its scale, and then export it and reconfigure it in Wayverb. Similarly, if a particular "EQ" is desired, there is no way to adjust the frequency-dependent absorption of all surfaces simultaneously. If the user is trying to simulate the sound of a particular space this is not a problem. However, if the user is trying to work backwards, tweaking a space to produce a particular sound, Wayverb's controls are unquestionably awkward.

A final usability issue is the dependency upon 3D models. Wayverb requires that 3D model files are solid and watertight, which in practice means that models must be specially constructed, or validated before use. It is unreasonable to expect musicians and sound designers to be proficient with professional CAD software, so in order to be useful, a library of suitable 3D models should be supplied alongside an installation of Wayverb. In the Context chapter, the dependency upon libraries of IRs was described as a drawback of convolution reverb tools, as it is impossible for a library to cover all possible usage scenarios. However, it appears that Wayverb relies upon libraries of 3D models. A possible solution to this problem would be to integrate a custom 3D modelling tool into Wayverb. This tool would only allow the creation of valid watertight spaces, and could be much simpler and more minimal than professional CAD tools. Then, users could quickly create spaces directly from the software, without having to worry about whether or not the model is valid for simulation.

10.1.3 Future Work

Shortcomings of the testing procedure have already been discussed, and it should go without saying that future work on the project should begin by testing existing features, rather than implementing new ones. However, there are many possibilities for features which might be considered for inclusion in a future version of Wayverb. These features can broadly be separated into two categories: extensions to the simulation algorithm, and usability improvements to the program interface.

10.1.3.1 Algorithm

As noted in the Digital Waveguide Mesh section, the input signal used to excite the waveguide is not optimal. Its frequency response extends up to the Nyquist frequency, which means that the mesh has energy above the desired output frequency. As shown in the Boundary Modelling section, the performance of the boundary filters is often erratic above 0.15 of the mesh sampling rate, sometimes increasing rather than reducing gain of incident signals. In combination, the broadband input signal sometimes causes the reflection filters to repeatedly amplify high-frequency content in the mesh. This is not audible in the final results, as the high frequency content is filtered out. However, it still leads to loss of precision, and sometimes numeric overflow. This might be solved by high-pass filtering the input signal, and then deconvolving the mesh output. However, it is not clear how such a process would interact with the microphone simulation. For example, it would probably be necessary to record and deconvolve the signals at all nodes surrounding the output node in order to produce a correct intensity vector. This would require further development and testing, for which there was insufficient time during the Wayverb project.

A similar drawback is to do with the low-frequency response of the mesh. Most input signals cause an increasing DC offset when the mesh uses a "soft" input node. To solve this, Wayverb's mesh is excited using a "hard" node, which introduces low-frequency oscillations and reflection artefacts. An important area for future research is the development of an input signal which can be used in conjunction with a soft source, which does not cause any DC offset.

Wayverb's waveguide topology is currently rectilinear, which is suboptimal both in terms of efficiency and accuracy. This topology was chosen because it was the only one which supported the LRS boundary model. The tetrahedral topology is faster and requires less memory. Therefore, future work may seek to formulate the LRS boundary equations such that they can be used with a tetrahedral mesh.

An important feature which is not implemented in Wayverb is the modelling of directional sources. Currently, all modelled sources emit a single spherical wave-front, which has equal energy in all directions. Real-world sources such as musical instruments and the human voice are directional. The ability to model directional sources would allow musicians and sound designers to create much more realistic and immersive acoustic simulations.

As well as directional sources, it might be useful to make the implementation of directional receivers more generic. Specifically, an ambisonic receiver would be useful, so that simulation results could be exported for directional processing in dedicated software. This could be achieved without modifying the geometric microphone model, in which coincident capsules are well-justified and lead to performance improvements (the simulation is run once for the entire capsule group, instead of once per individual capsule). However, the approach is not strictly justified in combination with the current waveguide microphone model [75]. Ambisonic output would therefore require further research into waveguide microphone modelling, in order to find a model which better supports coincident capsules.

10.1.3.2 User Interface

The user interface is less fully-featured than the interfaces of similar simulation programs (as listed in the Context section, table 1.1). The reason for this is simple: the entire application was developed by a single developer, over sixteen months, whereas all the software in table 1.1 was either developed by several collaborators, or was constructed over several years. To ensure that Wayverb would reach a usable state, its scope had to be limited. In its first release, the application is only capable of loading, saving, configuring, and running simulations. Examples of features generally found in acoustic simulation apps but missing in Wayverb follow.

Originally, the app was designed to include built-in convolution support, so that generated impulse responses could be previewed without leaving the application. This feature would greatly improve the usability of the program. However, it would not contribute to the main goal of the program, which is the accurate and fast acoustic simulation of virtual environments. Convolution reverb tools already exist, and many users will have their own favourite programs and plug-ins, like Altiverb [29] and Reverberate 2 [30], for this purpose. The time that would have been spent replicating this functionality was better spent working on the unique and novel features of the program.

Similarly, the ability to edit the virtual spaces themselves from within the app was not implemented. Writing an intuitive editor for 3D objects would be a large undertaking, even for a team of developers. Instead, the ability to load a variety of 3D file formats was included, and users are advised to use dedicated software such as Blender or Sketchup to create their simulated spaces.

Some further usability features which are missing, which would ideally be included in a future release, include:

- **Undo and redo**: If the user accidentally moves a source or receiver, or makes some other unwanted change, they must revert the change manually. There is no way of automatically reverting to the previous program state.
- **Load and save of capsule and material presets**: If the model contains several surfaces with different materials, and the user wants to apply a specific set of coefficients to all surfaces, each material must be configured by hand. There is no way to copy coefficients between surfaces, or to save and load materials from disk. Similarly, there is no way to save and load complex receiver set-ups from disk.
- **Improved visualisation**: Currently, ray energies are not communicated via the visualisation. There is also no way of adjusting the gain of the waveguide visualisation, which means that often the waveguide energy dies away quickly, becoming invisible, and giving the false impression that the simulation is not progressing.
- **Command-line interface**: For scripting or batch-processing of simulations, it would be useful to be able to run simulations from the command-line. Currently, this is only made possible by writing custom wrapper programs for the Wayverb library. It would be more useful to integrate command-line options directly into the Wayverb program.

Finally, it was not possible to test the program extensively for crashes and bugs. The program was tested on a 15-inch MacBook Pro running OS 10.11, and a little on earlier models of 15- and 13-inch Macbook Pros running OS 10.10. On 10.11, and on the 13-inch laptop running 10.10, no crashes were evident, although on the 15-inch 10.10 machine there were a few crashes within OpenCL framework code. These crashes were reported by a user, from their personal machine. There was not sufficient time to fix these bugs during the project.

Extended access to this machine was not possible, and debugging OpenCL code without access to the problematic hardware is difficult. Depending on the drivers supplied by the GPU vendor, the kernel may be compiled in subtly different ways. For most (non-OpenCL) software, there will be a single binary image for a given processor architecture, and if the program crashes then a stack trace can be used to find the location of the bug. However, for OpenCL code, the executed binary is generated at runtime, and will vary depending on the specification and driver of the GPU. Also, crashes inside the OpenCL kernel do not emit a stack trace. Therefore, it is almost impossible to debug OpenCL code without access to the specific machine configuration which causes the issue.

A future release could fix these problems, but only with access to any problematic hardware and software configurations. As the program is open-source it would also be possible for third-parties experiencing bugs to contribute fixes.

10.2 Summary

The goal of the Wayverb project was to create a program which was capable of simulating the acoustics of arbitrary enclosed spaces. For the program to be useful to its target audience of musicians and sound designers, it must be simultaneously plausible, efficient, and accessible.

The aims of plausibility and efficiency would be met by combining wave-modelling and geometric simulation methods, benefiting from both the physical realism of wave-modelling, and the computational performance of geometric simulation. This technique is not used by any other publicly available simulation package, so it was thought that a program implementing both models would be both faster, and produce higher-quality results, than competing programs. To further improve performance, the simulation would be implemented to run in parallel on graphics hardware. The program would be free and open-source, with a graphical interface, to ensure accessibility and encourage adoption.

Testing shows that the individual modelling methods are individually reasonably plausible. The ray-tracing and image-source methods respond appropriately to changes in room size, material, source/receiver spacing and receiver type. This is also true of the waveguide, which additionally is capable of modelling low-frequency modal responses, taking wave-effects such as diffraction into account. However, when waveguide outputs are combined with geometric outputs, the blended spectrum shows obvious discontinuities. Although individual aspects of the waveguide model (such as the boundary and receiver implementations) perform as expected, the waveguide model as a whole does not. The audible artefacts produced by combining the models mean that generated results are not fit for the purposes of high-quality music production or sound design. That is, the project fails to achieve its principal design goal. Future work may seek to locate errors in the implementation of the waveguide, and to improve the match between the outputs of the different models, perhaps sacrificing some low-frequency accuracy in the interests of sound quality.

In terms of efficiency, simulations generally complete within minutes, rather than hours or days, meeting the project's efficiency target. It is also possible to observe the progression of the simulation, and to retry it with less intensive parameters, if it is progressing too slowly. Unfortunately, the time taken to generate outputs is not necessarily reflected in the quality of the results. For example, it is disappointing to wait for ten minutes for an impulse response, only to find that the output has markedly different reverb times at the top and bottom of the spectrum. Good user experience relies on users being able to generate results with acceptable quality as quickly as possible. If the user has to tweak and re-render, waiting for several minutes each time, before eventually finding appropriate settings, this translates to a poor user experience. This may be solved in two ways: by improving the quality of the outputs; and/or by further optimisation of the simulation algorithms.

The application has an accessible graphical interface. Although some desirable features (such as built-in convolution and 3D editing) are missing, the interface is focused and functional. It is possible to install and use without specialist training. Additionally, all code is open-source, allowing collaboration and contribution from interested third-parties.

On reflection, it appears that the scope of the project was too large, which in turn meant that none of the research goals were satisfactorily achieved. The majority of the time spent on the project was given to implementing the program, which meant that when functional issues (like the reverb time mismatch) were discovered during testing, there was no time to attempt a fix. Additionally, much more testing would be necessary in order for a comprehensive view of the program to be given, but again this was not possible.

In its current state, Wayverb is unsuitable for use in creative contexts. However, it provides open-source implementations of three different acoustic models, along with an intuitive graphical interface. As such, Wayverb may serve as a solid starting point for further research in this field. It is conceivable that a future version of Wayverb, with more comprehensive testing and subsequent adjustment, could be a genuinely helpful tool for creative users.

References

[1] A. Southern, S. Siltanen, and L. Savioja, "Spatial room impulse responses with a hybrid modeling method," in *Audio Engineering Society Convention 130*, 2011.

[2] L. Savioja and U. P. Svensson, "Overview of geometrical room acoustic modeling techniques," *The Journal of the Acoustical Society of America*, vol. 138, no. 2, pp. 708–730, 2015.

[3] J. O. Smith, "Physical modeling using digital waveguides," *Computer music journal*, vol. 16, no. 4, pp. 74–91, 1992.

[4] M. Aretz, R. Nöthen, M. Vorländer, and D. Schröder, "Combined broadband impulse responses using FEM and hybrid ray-based methods," in *EAA Symposium on Auralization*, 2009.

[5] S. B. Shelley, *Diffuse boundary modelling in the digital waveguide mesh*. University of York, 2007.

[6] D. T. Murphy and D. M. Howard, "Digital waveguide mesh topologies in room acoustics modelling," PhD thesis, Citeseer, 2000.

[7] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[8] G. R. Campos and D. M. Howard, "On the computational efficiency of different waveguide mesh topologies for room acoustic simulation," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1063–1072, 2005.

[9] L. Savioja and V. Välimäki, "Reduction of the dispersion error in the interpolated digital waveguide mesh using frequency warping," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, 1999, vol. 2, pp. 973–976.

[10] S. A. Van Duyne and J. O. Smith, "The tetrahedral digital waveguide mesh," in *Applications of Signal Processing to Audio and Acoustics, 1995., IEEE ASSP Workshop on*, 1995, pp. 234–237.

[11] L. Savioja and V. Välimäki, "Interpolated 3-D digital waveguide mesh with frequency warping," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, 2001, vol. 5, pp. 3345–3348.

[12] "Odeon." 2016 [Online]. Available: http://www.odeon.dk/. [Accessed: 05-Dec-2016]

[13] "CATT-Acoustic." 2016 [Online]. Available: http://www.catt.se/. [Accessed: 05-Dec-2016]

[14] "OTL." 2016 [Online]. Available: http://www.olivetreelab.com/Room. [Accessed: 05-Dec-2016]

[15] "EASE." 2016 [Online]. Available: http://ease.afmg.eu/index.php/features.html. [Accessed: 05-Dec-2016]

[16] "Audioborn." 2016 [Online]. Available: http://www.audioborn.com. [Accessed: 05-Dec-2016]

[17] D. Schröder and M. Vorländer, "RAVEN: A real-time framework for the auralization of interactive virtual environments," in *Forum Acusticum*, 2011.

[18] M. J. Beeson and D. T. Murphy, "RoomWeaver: A digital waveguide mesh based room acoustics research tool," in *Proc. COST G6 Conf. Digital Audio Effects (Naples, Italy, October 2004)*, 2004, pp. 268–73.

[19] "Ear," *GitHub*. 2016 [Online]. Available: https://github.com/aothms/ear. [Accessed: 05-Dec-2016]

[20] "Pachyderm Acoustic," *GitHub*. 2016 [Online]. Available: https://github.com/PachydermAcoustic. [Accessed: 05-Dec-2016]

[21] "ParallelFDTD," *GitHub*. 2016 [Online]. Available: https://github.com/juuli/ParallelFDTD. [Accessed: 05-Dec-2016]

[22] "I-Simpa," *I-Simpa*. 2016 [Online]. Available: http://i-simpa.ifsttar.fr/. [Accessed: 05-Dec-2016]

[23] "Odeon FAQ." 2016 [Online]. Available: http://www.odeon.dk/faq-page#t16n151. [Accessed: 08-Dec-2016]

[24] A. Southern and S. Siltanen, "A hybrid acoustic model for room impulse response synthesis," in *Proceedings of Meetings on Acoustics*, 2013, vol. 19, p. 015113.

[25] D. Murphy, M. Beeson, S. Shelley, A. Moore, and A. Southern, "Hybrid room impulse response synthesis in digital waveguide mesh based room acoustics simulation," in *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, 2008, pp. 129–136.

[26] A. Southern, S. Siltanen, D. T. Murphy, and L. Savioja, "Room impulse response synthesis and validation using a hybrid acoustic model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 9, pp. 1940–1952, 2013.

[27] M. Vorländer, "Simulation and auralization of broadband room impulse responses," in *Tecniacústica 2009*, 2009.

[28] "Waves iR1 convolution reverb." 2017 [Online]. Available: http://www.waves.com/plugins/ir1-convolution-reverb. [Accessed: 22-Jul-2017]

[29] "Altiverb." 2017 [Online]. Available: https://www.audioease.com/altiverb/. [Accessed: 22-Jul-2017]

[30] "Reverberate 2." 2017 [Online]. Available: https://www.liquidsonics.com/software/reverberate-2/. [Accessed: 22-Jul-2017]

[31] "Our philosophy." 2017 [Online]. Available: http://www.mediterraneanacoustics.com/philosophy.html. [Accessed: 05-Jul-2017]

[32] A. Krokstad, S. Strom, and S. Sørsdal, "Calculating the acoustical room response by the use of a ray tracing technique," *Journal of Sound and Vibration*, vol. 8, no. 1, pp. 118–125, 1968.

[33] H. Kuttruff, *Room Acoustics, Fifth Edition*. CRC Press, 2009.

[34] M. Vorländer, *Auralization: Fundamentals of acoustics, modelling, simulation, algorithms and acoustic virtual reality*. Springer Science & Business Media, 2007.

[35] D. Schröder, *Physically based real-time auralization of interactive virtual environments*, vol. 11. Logos Verlag Berlin GmbH, 2011.

[36] A. Alpkocak and M. Sis, "Computing impulse response of room acoustics using the ray-tracing method in time domain," *Archives of Acoustics*, vol. 35, no. 4, pp. 505–519, 2010.

[37] S. A. Van Duyne and J. O. Smith III, "The 3D tetrahedral digital waveguide mesh with musical applications," in *Proceedings of the 1996 International Computer Music Conference*, 1996, pp. 9–16.

[38] L. Savioja, T. Lokki, and V. Välimäki, "The interpolated 3-D digital waveguide mesh method for room acoustic simulation and auralization," *Ultragarsas" Ultrasound"*, vol. 48, no. 3, pp. 48–52, 2014.

[39] K. Kowalczyk and M. van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 34–46, 2011.

[40] "Visual Studio support for C++ language features." 2016 [Online]. Available: https://msdn.microsoft.com/en-us/library/hh567368.aspx. [Accessed: 09-Dec-2016]

[41] "Clang support for C++ language features." 2016 [Online]. Available: http://clang.llvm.org/cxx_status.html. [Accessed: 09-Dec-2016]

[42] "Download LLVM releases." 2016 [Online]. Available: http://llvm.org/releases/. [Accessed: 09-Dec-2016]

[43] C. L. Christensen and J. H. Rindel, "A new scattering method that combines roughness and diffraction effects," in *Forum Acousticum, Budapest, Hungary*, 2005.

[44] J. Durany, T. Mateos, and A. Garriga, "Analytical Computation of Acoustic Bidirectional Reflectance Distribution Functions," *Open Journal of Acoustics*, vol. 5, no. 04, p. 207, 2015.

[45] International Organization for Standardization, Geneva, Switzerland, "ISO 3382-1:2009 acoustics – measurement of room acoustic parameters – part 1: Performance spaces." 2009.

[46] T. Möller and B. Trumbore, "Fast, minimum storage ray/triangle intersection," in *ACM SIGGRAPH 2005 courses*, 2005, p. 7.

[47] J. Amanatides, A. Woo, and others, "A fast voxel traversal algorithm for ray tracing," in *Eurographics*, 1987, vol. 87, pp. 3–10.

[48] Z.-h. Fu and J.-w. Li, "GPU-based image method for room impulse response calculation," *Multimedia Tools and Applications*, pp. 1–17, 2016.

[49] R. Heinz, "Binaural room simulation based on an image source model with addition of statistical methods to include the diffuse sound scattering of walls and to predict the reverberant tail," *Applied Acoustics*, vol. 38, no. 2, pp. 145–159, 1993.

[50] S. H. Linkwitz, "Active Crossover Networks for Noncoincident Drivers," *Journal of the Audio Engineering Society*, vol. 24, no. 1, pp. 2–8, Feb. 1976.

[51] F. Gustafsson, "Determining the initial states in forward-backward filtering," 1994.

[52] J. Antoni, "Orthogonal-like fractional-octave-band filters," *The Journal of the Acoustical Society of America*, vol. 127, no. 2, pp. 884–895, 2010.

[53] J. O. Smith III, "On the equivalence of the digital waveguide and finite difference time domain schemes," *arXiv preprint physics/0407032*, 2004.

[54] M. Karjalainen and C. Erkut, "Digital waveguides versus finite difference structures: Equivalence and mixed modeling," *EURASIP Journal on Applied Signal Processing*, vol. 2004, pp. 978–989, 2004.

[55] J. Botts and L. Savioja, "Integrating finite difference schemes for scalar and vector wave equations," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 171–175.

[56] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, "Acoustic modeling using the digital waveguide mesh," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 55–66, 2007.

[57] J. Sheaffer and B. M. Fazenda, "FDTD/K-DWM simulation of 3D room acoustics on general purpose graphics hardware using compute unified device architecture (CUDA)," *Proc. Institute of Acoustics*, vol. 32, no. 5, 2010.

[58] H. Jeong and Y. W. Lam, "Source implementation to eliminate low-frequency artifacts in finite difference time domain room acoustic simulation," *The Journal of the Acoustical Society of America*, vol. 131, no. 1, pp. 258–268, 2012.

[59] S. Sakamoto, "Phase-error analysis of high-order finite difference time domain scheme and its influence on calculation results of impulse response in closed sound field," *Acoustical Science and Technology*, vol. 28, no. 5, pp. 295–309, 2007.

[60] Y. W. Lam and J. A. Hargreaves, "Time domain modelling of room acoustics," in *Proceedings of the Institute of Acoustics*, 2012.

[61] J. Sheaffer, M. van Walstijn, and B. Fazenda, "Physical and numerical constraints in source modeling for finite difference simulation of room acoustics," *The Journal of the Acoustical Society of America*, vol. 135, no. 1, pp. 251–261, 2014.

[62] D. T. Murphy, A. Southern, and L. Savioja, "Source excitation strategies for obtaining impulse responses in finite difference time domain room acoustics simulation," *Applied Acoustics*, vol. 82, pp. 6–14, 2014.

[63] B. Dimitrijevic, B. Nikolic, S. Aleksic, and N. Raicevic, "Optimization of Excitation in FDTD Method and Corresponding Source Modeling," *RADIOENGINEERING*, vol. 24, no. 1, p. 11, 2015.

[64] J. B. Schneider, C. L. Wagner, and S. L. Broschat, "Implementation of transparent sources embedded in acoustic finite-difference time-domain grids," *The Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 136–142, 1998.

[65] J. Sheaffer, M. V. Walstijn, and B. M. Fazenda, "A physically-constrained source model for FDTD acoustic simulation," in *Proc. of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, 2012.

[66] J. Botts and L. Savioja, "Spectral and pseudospectral properties of finite difference models used in audio and room acoustics," *IEEE transactions on audio, speech, and language processing*, vol. 22, no. 9, pp. 1403–1412, 2014.

[67] K. Kowalczyk and M. van Walstijn, "Modeling frequency-dependent boundaries as digital impedance filters in FDTD and K-DWM room acoustics simulations," *Journal of the Audio Engineering Society*, vol. 56, no. 7/8, pp. 569–583, 2008.

[68] S. Siltanen, A. Southern, and L. Savioja, "Finite-difference time domain method source calibration for hybrid acoustics modeling," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 166–170.

[69] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[70] M. Noisternig, T. Musil, A. Sontacchi, and R. Holdrich, "3d binaural sound reproduction using a virtual ambisonic approach," in *Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2003. VECIMS'03. 2003 IEEE International Symposium on*, 2003, pp. 174–178.

[71] W. Oliver, "Listen HRTF database." 2003 [Online]. Available: http://recherche.ircam.fr/equipes/salles/listen/. [Accessed: 04-Jan-2017]

[72] A. Southern and D. Murphy, "Methods for 2nd Order Spherical Harmonic Spatial Encoding in Digital Waveguide Mesh Virtual Acoustic Simulations," in *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2007, pp. 203–206.

[73] A. Southern and D. T. Murphy, "2nd order spherical harmonic spatial encoding of digital waveguide mesh room acoustic models," in *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx07), Bordeaux, France*, 2007, pp. 101–108.

[74] A. Southern, D. T. Murphy, and L. Savioja, "Spatial encoding of finite difference time domain acoustic models for auralization," *IEEE transactions on audio, speech, and language processing*, vol. 20, no. 9, pp. 2420–2432, 2012.

[75] H. Hacıhabiboglu, B. Günel, and Z. Cvetkovic, "Simulation of directional microphones in digital waveguide mesh-based models of room acoustics," *IEEE Trans. on Audio, Speech and Language Process*, vol. 18, no. 2, pp. 213–223, 2010.

[76] J. H. Rindel, "The use of computer modeling in room acoustics," *Journal of vibroengineering*, vol. 3, no. 4, pp. 41–72, 2000.

[77] D. T. Murphy and M. Beeson, "The KW-boundary hybrid digital waveguide mesh for room acoustics applications," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 2, pp. 552–564, 2007.

[78] A. Kelloniemi, "Frequency-dependent boundary condition for the 3-D digital waveguide mesh," in *Proc. Int. Conf. Digital Audio Effects (DAFx'06)*, 2006, pp. 161–164.

[79] K. Kowalczyk and M. van Walstijn, "Modelling Frequency-Dependent Boundaries as Digital Impedance Filters in FDTD Room Acoustic Simulations," in *Audio Engineering Society Convention 124*, 2008.

[80] S. Oxnard, D. O'Brien, J. van Mourik, and D. Murphy, "Frequency-Dependent Absorbing Boundary Implementations in 3D Finite Difference Time Domain Room Acoustics Simulations," in *Proc. EuroNoise 2015*, 2015.

[81] "ITPP homepage." 2013 [Online]. Available: http://itpp.sourceforge.net/4.3.1/. [Accessed: 13-Dec-2016]

[82] "Assimp Supported Formats." 2017 [Online]. Available: http://www.assimp.org/main_features_formats.html. [Accessed: 13-Jan-2017]

[83] "Room EQ Wizard Room Acoustics Software." 2017 [Online]. Available: https://www.roomeqwizard.com/. [Accessed: 12-Jan-2017]

[84] M. Hodgson, "When is diffuse-field theory accurate?" *Canadian Acoustics*, vol. 22, no. 3, pp. 41–42, 1994.