Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Measurement and Information Systems

Attila Szarvas

Efficiency testing of active noise control by acoustic field modeling

Advisor

László Sujbert, Ph.D.

BUDAPEST, 2013

Contents

HALLGATÓI NYILATKOZAT

Alulírott Szarvas Attila, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelmûen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerzõ(k), cím, angol és magyar nyelvû tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhetõ elektronikus formában, a munka teljes szövegét pedig az egyetem belsõ hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetõvé.

Budapest, 2013. december 22.

Szarvas Attila

hallgató

Abstract

Active noise cancelling (ANC) of acoustic noises is becoming a common feature of consumer headphones. The active suppression of unwanted noises in special environments, such as the air conditioning vents of concert halls can be more efficient than passive isolation. However all of the market available solutions operate under strictly controlled acoustic conditions, such as the specially designed earmuffs of headphones.

This thesis presents an approach and a software simulation tool for investigating the behaviour and operating efficiency of active noise cancelling systems under arbitrarily chosen conditions. Special attention is given to environments described by typical conditions of room acoustics, which are generally unfavourable for active noise cancelling efforts. The efficiency of the feedforward noise cancelling structure utilizing multiple reference microphones is investigated under such conditions using finite-difference time-domain simulations. The results are evaluated in comparison with real-life measurement data. Based on the simulated and experimental results best practices and the outline of future investigations are identified.

The thesis briefly investigates the active noise cancelling problem, and presents the feedforward structure capable of suppressing wideband stochastic noises. The ELMS noise cancelling algorithm is also reviewed, which has been implemented on both the real-life and simulated ANC systems.

A brief overview of various acoustic field modelling techniques, and a detailed discussion of the finite-difference time-domain (FDTD) method is given. The techniques of locally reacting surfaces (LRS) and perfectly matched layer (PML) are also presented, which allow the simulation of arbitrary frequency dependent surfaces and anechoic conditions. As part of a simulation software these methods are implemented using the OpenCL language, allowing the efficient utilization of high-performance general purpose graphical processing units (GPGPU). The software has been integrated with a 3D modeller tool and Matlab to allow the easy description of acoustic environments and simple evaluation of simulation data.

Using the implemented simulation software a set of cornerstone ANC situations in reflective environments are simulated and the behaviour of the ANC system is investigated in comparison with real-life experiences.

Kivonat

Az akusztikus zajok aktív zajcsökkentése (ANC) manapság már hétköznapi használatra szánt fejhallgatókban is megtalálható. Különleges körülmények között – mint például koncerttermek légkondicionáló berendezéseiben – az aktív zajcsökkentés hatékonyabb megoldást jelent a passzív izolációnál. A jelenlegi piaci megoldások ugyanakkor mind szigorúan kontrollált akusztikai viszonyokat igényelnek. A fejhallgatók nagy gonddal megtervezett kosarai is ezt a célt szolgálják.

A diplomaterv az aktív zajcsökkentő eszközök általános körülmények között nyújtott teljesítményének vizsgálatával foglalkozik és bemutat egy a vizsgálatokhoz használható szimulációs szoftvert. A szoftver kialakítása során különös figyelmet szenteltem a teremakusztika területén tapasztalható tipikus viszonyoknak, amelyek jellemzően nehéz feladat elé állítják az ANC rendszereket. A szoftver az időtartománybeli végesdifferencia-módszer felhasználásával szimulálja a reflexiós környezetekre jellemző viszonyokat. A szimulációs és gyakorlati kísérletek eredményeit összevetve megvizsgálom a több referenciamikrofont használó, előrecsatolt struktúrájú zajcsökkentő rendszerek hatékonyságát.

A dolgozatban röviden ismertetem az aktív zajcsökkentési probléma lényegét, és bemutatom az előrecsatolt zajcsökkentő struktúrát, mely sztochasztikus zajok elnyomására is alkalmas. Ismertetem a szimulációs szoftverben és a gyakorlati kísérleti rendszerben egyaránt implementált ELMS zajcsökkentő algoritmust.

Röviden áttekintem a különböző akusztikai térszámítási módszereket, és részletesen ismertetem az időtartománybeli végesdifferencia-módszert (FDTD). Bemutatom a frekvenciafüggő reflexiós felületeket modellező *locally reacting surfaces* (LRS) és a szimulációs térrész reflexiómentes csonkolását lehetővé tévő *perfectly matched layer* (PML) technikákat is. A bemutatott eljárásokat OpenCL nyelven implementáltam, amely lehetővé teszi az általános célú grafikus processzorok (GPGPU) teljesítményének hatékony kiaknázását. A szoftvert integráltam egy 3D-s modellező eszközzel és a Matlabbal, lehetővé téve az akusztikai feladatok egyszerű leírását és a szimulációs eredmények könnyű feldolgozhatóságát.

Az elkészült szoftverrel szimuláltam két elméleti jelentőségű reflexiós környezetben végrehajtott kísérletsorozatot. Az eredményeket a gyakorlati kísérletekkel összevetve értékeltem.

Preface

The investigation of active noise cancelling systems has played a central role in my university studies. During the project laboratory courses of my BSc and MSc tuition I have been examining the properties of ANC systems regarding the achievable level of noise suppression depending on various qualitative and quantitative properties. Such properties are, e.g. the order of adaptive filters used to model the acoustic paths, the number of reference signals used to approximate the opposite of the unwanted noise, and the selection of sources from which these reference signals originate.

My BSc thesis covered the working principles, design aspects and usage possibilities of portable noise cancelling devices. I have examined market available compact solutions utilizing primarily analogue electronics. My focus however was and has been ever since on discrete systems utilizing digital signal processors. While lacking in certain areas – mostly processing latency – digital devices offer far more flexibility, reconfigurability, and given enough processing power they are able to solve acoustically more complex problems than their analogue counterparts. They allow for this by having a lot more degrees of freedom both algorithmically and structurally.

The multitude of parametric and structural arrangement options also means, that finding the optimal settings of such systems requires a great deal of experimentation, measurement and adjustment. During my BSc thesis work I have created a simulation tool to gain a deeper insight into the conditions and behaviour of the acoustic field. This tool allowed for the simulation of acoustically open, i.e. reflection free conditions, where periodic noises were present and suppressed by a feed forward ANC system. The simulations helped with the testing of algorithmic parameters and by plotting the field conditions during suppression helped in finding the ideal placement of the error microphones and actuators.

During my MSc project laboratory courses I have investigated the performance of ANC systems utilizing a large number of reference signals. These signals were recorded by microphones in the noise cancellation area's vicinity. In order to better understand the effect of the placement of these microphones relative to the noise sources and the cancellation region, I have improved on the previously developed simulation software by allowing arbitrary types of acoustic sources. This enabled

us to analyse the behaviour of the system in the presence of stochastic noises. A series of systematic measurements taken in an anechoic chamber were carried out, and a number of best and worse case scenarios were identified backed up by both measurements and simulation data.

The next series of measurements were taken in closed spaces governed by the physical typicalities of room acoustics. Compared to the results in the anechoic environment a significant decrease in noise cancelling performance was observed. Such a performance drop was expected since a large portion of the incoming noise is reaching the cancellation area after numerous reflections and reaching it from multiple angles. In these environments one must take special care during the placement of reference microphones being mindful not just of the direction of noise sources but the direction of the main reflection paths as well. Although good placement practices were identified and the results were consistent with our assumptions about the nature of reflections, we lacked the numerical tools to simulate the behaviour of active noise cancellation systems in these conditions.

Consequently we have decided to either improve or replace the simulation program, and acquire a tool that allows us to observe the behaviour of ANC systems in reflective environments. Such a program can provide theoretical verification for experimental results and makes it possible to test new ideas without the tedious process of setting up physical experiments. After considering the alternatives we have decided to create a completely new simulation program based on the iterative, numerical solution of the acoustic wave equation, thus giving an accurate representation of all phenomena present in room acoustics. The computationally heavy nature of this approach also demanded that we abandon the previously used Matlab language and adopt new technologies from the field of high performance computing.

1 Introduction

Active noise cancellation (ANC) today is a widely used and easily available technology. The first working solutions have been created in the 1950s when they were used to cancel noises in the cockpit of helicopters. Today it can be found in headphones from the low-end of the price range to the most expensive models.

Special requirements may also necessitate the usage of active noise cancelling solutions. In certain conditions passive noise isolation may either be too expensive, cumbersome to handle or simply impractical. It's hard to imagine for example the alternative for an active noise cancelling system working in the air ducts of a concert hall where the flow of air may not be impeded.

On the other hand active noise cancelling is not as much a substitution as an extension for passive isolation. Indeed present solutions all rely on strictly controlled acoustic conditions and their efficiency depends on proper passive isolation. Noise cancelling headphones are built on the assumption that the path between the speakers and the ears remains unchanged and can be largely described by a fixed delay and frequency independent amplification. Given this the noise cancelling problem can be solved with analogue electronics that generate anti-noise by delaying the signal of an outer reference microphone by a preset value and applying automatic gain control with the help of an inner error microphone. Such headphones can be found at a price from \$50 to \$500. The Sennheiser PXC 450 – one of the most expensive and most highly regarded ANC headphones – can provide 10 dB active noise suppression on top of the passive isolation.

One of the observations of my BSc thesis was that the simple ANC structure contained in headphones looses much of its efficiency when it's removed from the closed acoustic environment of the earmuffs. In an environment characterised by reflective surfaces – which is the majority of modern built in environments – the acoustic paths become more complex, and signals picked up by microphones at different locations of the acoustic field are not simply delayed and attenuated versions of each other. Even when assuming the availability of filters with arbitrarily high order, significant noise cancelling can not be guaranteed using a single reference microphone.

During the MSc project laboratory courses I have been investigating the be-

haviour of acoustically open noise cancelling systems using multiple reference microphones. [17] Real-life experiments with such systems suggested that increasing the number of reference microphones will often lead to better results than increasing the order of modelling filters, while consuming the same amount of computational resources.

Since these experimental systems are complex and require a lot of fine tuning it is useful to investigate our assumptions using simulations as well, since they provide more parameters to tweak and verification for the experimental results. While creating an anechoic simulation in Matlab was a straightforward task and was mainly concerned with the implementation details of the noise cancelling functionality, creating a simulation that models room acoustics required more consideration.

Simulating the wideband acoustic properties of large rooms requires more computational power and different technologies than Matlab, and creating such a simulation takes a lot more effort than the implementation of the noise cancelling part itself. While there are readily available tools for auralization purposes in room acoustics, we needed a solution that allows us to also simulate the effect of the noise cancelling system. Thus we have decided to create an own solution implementing acoustic and noise cancelling simulations with regard to each other.

The aim of this thesis' work was to create a high performance acoustic simulation tool that allows us to ascertain the effectiveness of active noise cancelling systems, while offering a large degree of freedom in the parameters of both the ANC system and its acoustic environment. My goal was to create a general framework that offers a transparent interface to interactive modules through which they are connected to the acoustic simulation. This means that any algorithm that would be implemented on a digital signal processor can be attached to the field simulation with minimal changes. Using the same algorithm that ran on the DSP in the physical experiments we executed simulations focusing on typical situations in reflective environments.

2 Theoretical background for active noise cancellation

2.1 Statement of the noise cancelling problem

This section gives an overview of the noise cancelling problem that is the main focus of our simulation efforts.

Active noise cancellation – referred to as ANC later on – aims to supress unwanted noises in a selected region of space by generating waveforms equal in amplitude but opposite in phase to the noise [10]. The waveforms are superposed in the target area and destructive interference results in a useful reduction of noise power.

Throughout the MSc courses my studies were directed towards the suppression of stochastic noises in acoustically complex environments. Systems that allow noise suppression in such situations are utilizing the *feedforward ANC structure* that relies on reference signals that are correlated with the noise [2]. Figure 1 shows the layout of such a system.

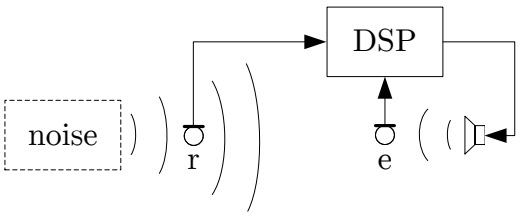

Figure 1: Feedforward structure in acoustic noise cancelling

The microphone labelled with r is the *reference microphone* from which the reference signal originates. This signal is then filtered with an adaptive FIR filter in the digital signal processor, and the resulting signal drives the speaker on the right, which is the actuator of the system. The microphone labelled with e is the *error microphone* since it picks up the remainder of the noise coming from the left direction, after it meets the anti-noise waves coming from the right. The error signal originating from this microphone is used to adapt the FIR filter realized on the DSP. The spatial vicinity of the error microphone will be henceforth referred to

as the *suppression region*.

Applying an FIR filter on the reference signal suggests that we expect that the error and reference signals are correlated as stated by the below equation where n is the incoming noise, x is the reference signal and H is the transfer function describing the correlation.

$$n = Hx \tag{1}$$

The feedforward structure is built on the assumption that the correlation between the noise and the reference signal is such that the H transfer function can be realized by causal systems.

A simple consequence of this requirement is that the reference microphone must be placed in such a manner that the noise front reaches it well before it would arrive in the suppression region. Thus the signal processing system has enough time to execute the filtering and actuation.

The time required for these actions to be carried out are affected by the various delays inherent in real life signal processing systems, e.g. delays caused by the analogue-digital conversion, software latency, digital-analogue conversion and signal propagation in the actuator's electronics. Another component of this delay can be attributed to the time of wave propagation from the actuator speaker to the suppression region. The effects of these events during information propagation are reduced to a single notion, i.e. the *secondary path* [18]. In other words the delay of the secondary path represents a minimal time interval by which events in the reference signal must precede the corresponding events in the noise reaching the suppression region. To ensure this condition the reference microphone – noise source distance has to be smaller than the suppression region – noise source distance.

In the case of a single noise source in an anechoic environment the occurrence of the previously mentioned condition is sufficient for the solution of the noise cancelling problem, i.e. there is no upper limit for the achievable level of noise suppression. However in the presence of multiple noise sources and or the occurrence of reflective surfaces in the environment complete noise cancellation may not be achievable.

Figure 2 shows a situation with two noise sources and two reference microphones. The red mark denotes the suppression point.

The arrows signify the transfer functions between the various points in the acous-

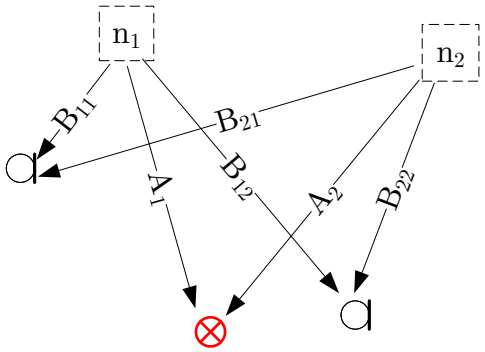

Figure 2: Multiple noise sources and reference microphones in an ANC context

tic space. Transfer functions between the noise sources n_1 and n_2 and the suppression point are labelled by A. The transfer functions between the noise sources and reference microphones are labelled by B.

Let's assume for now that the secondary paths have a transfer function of 1. Applying W filters on the reference signals the noise cancelling problem can be stated as follows.

$$W_1 \cdot \begin{bmatrix} B_{11} & B_{21} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} + W_2 \cdot \begin{bmatrix} B_{12} & B_{22} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \tag{2}$$

Rearranging the equation for the ANC system's adaptive filters we get

$$\begin{bmatrix} W_1 \\ W_2 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}^{-1} \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \tag{3}$$

.

According to (3) the adaptive filters must realise the inverse of the reference microphones' **B** transfer matrix multiplied by **A**. Under anechoic conditions the invertability of the **B** matrix can be guaranteed by observing simple rules regarding the placement of reference microphones. These generally concern the non-symmetrical placement of these microphones in relation to the noise sources.

Reflecting surfaces however can greatly increase the complexity of the transfer functions. Such conditions cover the majority of use cases for ANC systems. In this case there are no simple rules that would ensure the exact solution of (3). Even if **B** was invertible, the $\mathbf{B}^{-1}\mathbf{A}$ expression may not be causal [3]. Thus only an approximate solution can be given to the problem and the noise cannot be completely cancelled out.

In real life situations the exact nature of these transfer functions are unknown. They are determined by an extraordinarily large number of successive reflections between the environment's surfaces and an approach to analytically approximate them would be futile. The only course of action one can follow is to find a generally good arrangement of reference microphones that cover the main noise propagation paths. The microphones should encompass the suppression region so that reflected waves will pass a reference microphone before they would arrive in the suppression zone.

2.2 Noise cancelling algorithms

The presented feedforward structure can be controlled by a digital signal processor realising some extension of the LMS algorithm. It is a suitable choice since its iterative behaviour allows the system to be constantly adapted to the observed acoustic transfer relations, and it can match slow changes in the environment while keeping the error small. In a static situation it ensures that the power of the remainder noise is minimized.

The two most widely used variants of the LMS algorithm for noise cancelling purposes are the XLMS and ELMS algorithms [4]. The difference between the algorithms is how they utilize the information about the secondary paths. The former filters the reference signals with the approximated transfer function of the secondary path, and the latter filters the error signals with the approximated delayed inverse of the secondary path.

Consequently the ELMS algorithm presented by [13] is much more suitable for systems with many reference signals regarding both the computational and memory requirements [7]. For this reason I am only introducing the ELMS algorithm in a detailed fashion, which we have used in the majority of our experiments and the anechoic simulations. This is also the specific algorithm that has been implemented in the new field simulation.

2.2.1 The ELMS algorithm

Figure 3 shows the signal processing block diagram of the ELMS algorithm. The reference signal x originates from the reference microphones placed around the sup-

pression region. The correlation between the reference signal and the noise d reaching the suppression region is represented by the A_1 transfer function. This is usually referred to as the *primary path* and is determined by the acoustic properties of the noise propagating environment.

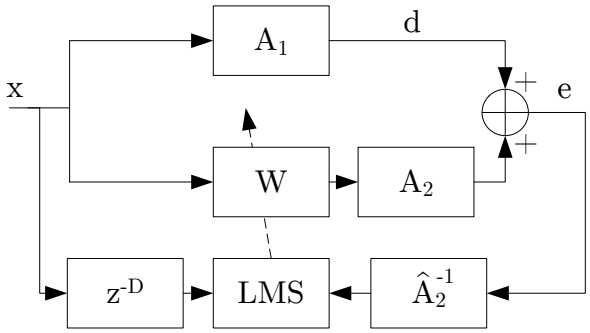

Figure 3: Block diagram of the ELMS method

The reference signal is filtered by the ANC system's W adaptive filter. The signal is also subjected to the effects of the A_2 secondary path's transfer function. As previously explained the effects of the secondary path are distributed among various parts of the system but they can be reduced to a single transfer function connected in series with the adaptive filter.

The superposition of the ANC system's output – the output of A_2 – and the incoming noise d is the e error of the control system.

The $\hat{A}_2^{-1}$ filter is the estimated delayed inverse of the secondary path. It is constructed in the system's identification mode prior to noise cancelling. It is estimated with the LMS method such that

$$A_2 \hat{A}_2^{-1} = z^{-D} \tag{4}$$

i.e. signals passing through the secondary path and then the filter should be delayed by D compared to the original signal. This signal is then fed to the LMS update algorithm. The algorithm also receives the reference signal delayed by the same D amount of discrete steps. The amount of delay is chosen by practical considerations and the correctness of its value has to be experimentally verified.

The adaptive filter's update step is given by (5) where e_f and $\mathbf{x}$ are the filtered error and reference signals respectively, and μ is an experimentally chosen step constant.

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \mu \cdot e_f(k) \cdot \mathbf{x}(k-D) \tag{5}$$

2.2.2 Identifying the secondary paths

In order to execute the ELMS algorithm we need to acquire an estimation of the secondary path's inverse. Since the secondary path itself is delaying the signal the exact inverse would have to undo that delay, which is a non-causal effect. Identifying the *delayed inverse* of this path however is possible as long as the chosen delay is larger than the delay inherent in the secondary path.

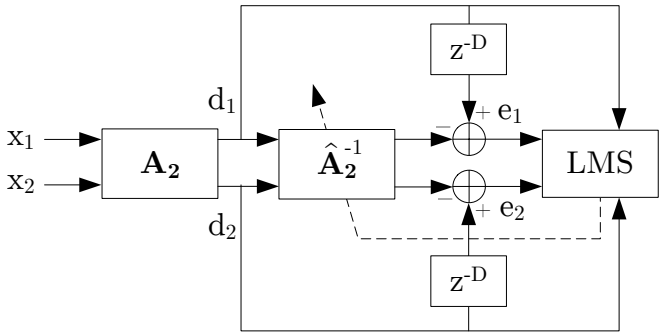

Figure 4: Block diagram of the identification of the secondary path's inverse

Figure 4 shows the identification process for the secondary filters. During this phase the ANC system generates white noise and sends it to the actuators. It needs to generate as many uncorrelated noise signals as the number of actuators. The method presented here works as long as the number of actuators and error microphones are equal.

The signals generated by the system are labelled x_1 and x_2 on the block diagram. The acoustic systems transfer function is represented by the $\mathbf{A_2}$ block. The signal of the error microphones are labelled d_1 and d_2. These signals are then filtered by the $\hat{\mathbf{A}}_{\mathbf{2}}^{-\mathbf{1}}$ filter and the results are compared with the delayed x identification signals. D is the delay determined by practical considerations and should be significantly

greater than the delay of $\mathbf{A_2}$. The error microphones' signals and the estimation's error is used to adapt $\hat{\mathbf{A}}_{\mathbf{2}}^{-\mathbf{1}}$ according to the LMS rule shown in (6)

$$\hat{A}_{2_{i,j}}^{-1}[k+1] = \hat{A}_{2_{i,j}}^{-1}[k] + \mu e_j[k] \cdot d_i[k] \tag{6}$$

where i and j are indices corresponding to the actuators and error microphones respectively.

3 Field simulation techniques

This section gives a brief summary of the capabilities and usage scenarios of the anechoic simulation software that I have been using during the MSc project laboratory courses. Then it will review the different candidate techniques that I have been considering to use in order to enable the simulation software to include reflections into its model.

3.1 Anechoic simulation

The simulation software used to investigate the behaviour of ANC systems in anechoic situations has been written in object oriented Matlab language. The object oriented features were included first in the 2010b version, and thus it is the minimum requirement for running it. The simulation program consists of a collection of classes that encapsulate the mathematical operations in a domain specific manner. To describe situations and run simulations one can use classes such as `XLMS`, `ELMS`, `PointSource` or functions such as `bandnoise`.

The parameters for these classes and functions are typical domain specific quantities, e.g. the order of modelling filters, the spatial position of the sources or the bandwidth of random noise.

Simulations can be run by calling the `ELMS.Simulate` function with the simulation length as an argument. The function would then iterate through all sources – including noises and actuators – and superpose their effects at the current simulation step at all points of interest. These points are the positions of reference and error microphones during the runtime of the noise cancelling algorithm, or all points of a spatial region during the plotting of suppression fields.

The simulation's acoustic modelling belongs to the *geometric methods* since the propagation of sound is modelled by straight lines instead of wavefronts. These lines are referred to as *acoustic rays*.

The effect of a single source is calculated by attenuating and delaying the signal values contained within. The simulation represents field values proportional to sound pressure, thus the attenuation coefficient is proportional to the distance from the source. The amount of delay is calculated by the assumption of radial wave

propagation with a velocity of 340 $\frac{m}{s}$. The delaying operation is carried out by a Thiran allpass fractional delay filter.

Since the program has no notion of acoustic barriers or reflecting surfaces it is only adequate for modelling the conditions of large, open spaces or anechoic chambers. In order to increase the complexity of certain ANC tasks we have placed many noise sources in directions from which we expected incoming reflections. While in case of low order modelling filters this is a reasonable attempt to approximate situations with reflections, it is not an accurate model of the conditions in reflective environments, and thus the need for improving the simulation software has arisen.

3.2 Image-source method

A straightforward improvement of the previously presented geometric method would have been the implementation of image sources. Calculating field values would have been carried out in the exact same manner as before, but the algorithm wouldn't just be applied to the actual sources but also to their mirror images across reflecting boundaries.

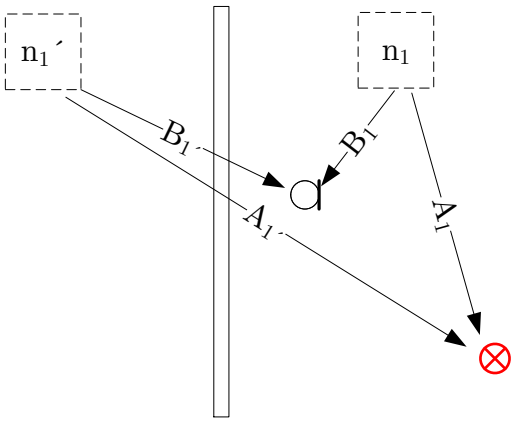

Figure 5: Image-source method – one source, single reflection

Figure 5 depicts a situation with a noise source in the presence of a reflecting surface. The noise source is duplicated on the other side of the wall according to the rules of geometrical optics. The signal's amplitude and phase contained within the source have to be delayed according to the reflective properties of the modelled boundary.

This method can be simply implemented for a single reflection, however in the case of multiple reflections, the placement and visibility of image sources have to be carefully considered. Using edge-sources the model can also account for diffraction effects. However it has a shortcoming regarding the manner of reflections, since they are treated completely in a specular way. In practice only a portion of incoming waves are reflected from the surfaces specularly and the rest is reflected in a diffuse fashion [9].

Since basic building materials have a reflectance above 95% in the frequency region of active noise cancellation, a large number of mirror sources would have to be placed for an accurate model. This would largely increase the computational cost of the simulation. By raising the simulated number of reflections by one, every mirror source previously created has to be mirrored again, thus the complexity of this approach increases exponentially. This makes the method feasible for the simulation of early reflections only, which aren't sufficient for accurate studies of ANC behaviour.

3.3 Raytracing

Raytracing is another geometric method but its different approach would have required the implementation of a completely new program.

Raytracing operates with a model where a finite number of rays are radiating away from each source in all directions. The path of these rays are observed, and whenever they intersect a reflecting boundary a specular rule of reflection is applied to the ray. Figure 6 illustrates this approach.

The sources effects at the investigated points are given by the sum of rays intersecting a finite region around that point. The accuracy of the results are affected by the number of rays leaving each source, and finding all wave propagation paths is not guaranteed.

Raytracing is much better suited to the simulation of multiple reflections than the image-source method, since the computational complexity increases linearly with the number of reflections.

Unlike the image-source method it can also model diffuse reflections by applying different angles of reflection to waves of different frequencies, however this re-

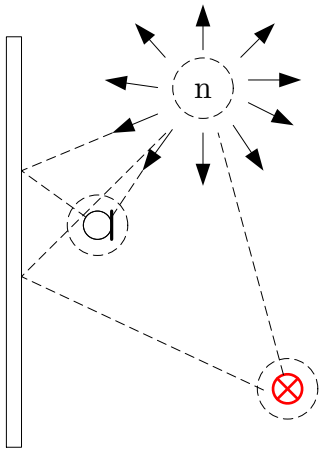

Figure 6: Raytracing method

quires running the simulation once for each octave band increasing the computational cost [9].

On the other hand this method is incapable of accounting for effects of wave diffraction. This shortcoming is acceptable only if the wavelengths are significantly shorter than the typical feature size of reflecting surfaces. That is however not the case with active noise cancellation. ANC systems' typical working range is under 2000 Hz where diffraction plays a significant role in the resulting field conditions.

3.4 Numerical solution of the wave equation

The approach that resembles the nature of the acoustic problem most closely is solving the governing physical equations themselves. This method will inherently account for all effects that are contained by the chosen formulation of physical laws. E.g. wave diffraction and diffuse reflection can be explained by the second order wave equation for acoustic pressure, and the numerical solution of this equation will exhibit these phenomena.

The main alternative methods based on the wave equation's solution are the finite element method (FEM), the boundary element method (BEM) and the finite difference time domain method (FDTD). A concise comparison of these methods is given by [9].

The FEM and FDTD methods are based on volume discretization, while BEM

is based on surface discretization. The latter method can be computationally more efficient when the problem's surface to volume ratio is low but can be significantly more expensive otherwise.

The FEM uses non-uniform discretization, and the creation of the computational grid requires extensive pre-calculations. In typical room acoustics problems both the FEM and BEM methods have significantly higher computational complexity and cost than the FDTD method.

The FDTD method can be treated as a special case of the FEM in its time domain formulation and can be realised by a simpler algorithm that approximates the time and space derivatives with finite differences. It relies on a uniform grid in time and space, and consequently has the advantage of being able to handle moving sources and observers. Its spatial and temporal grid sizes are bound by stability requirements, and their values can be determined by the maximal frequency of waves that can be accurately simulated.

The FDTD method is related to the popular digital waveguide mesh (DWM) method, which is a special case of FDTD and their algorithms inside simulated volumes are equivalent.

3.5 FDTD as a method for simulating ANC systems

After considering the alternatives the FDTD method was deemed the best for our purposes.

Since the reflectance of building materials is well above 90% in the relevant frequency range, a large amount of the noise power reaching the suppression region can be amounted to multiple reflections. The image-source method's exponential complexity regarding the number of reflections renders it unable to account for this effect.

The raytracing method's inability to simply handle diffuse reflections or wave diffraction cannot be neglected at the low frequencies at which ANC systems operate.

The FEM's larger computational cost and more complex algorithms make it unappealing for our purposes, while its large precision in handling complex geometries are not required in this case.

Although the FDTD method is much more demanding computationally than my

previous simulation, its simple equations clearly lend themselves to massive parallelization. With the recent technological level and availability of general purpose graphics processors (GPGPU) it presents a feasible approach to our simulation requirements.

4 The finite difference time domain method

The finite difference time domain method (FDTD) is a widely used numerical method for the solution of differential equations. It can be viewed as a generalization of the digital waveguide mesh (DWM) method or a special case of the finite element method (FEM).

Unlike the FEM it uses a uniform grid along which function values are discretized hence its unable to effectively handle complex geometries. On the other hand it is capable of handling moving sources and spatial objects – an element which would require the recreation of the FEM grid at each step – and it is much cheaper computationally and simpler to implement.

Its advantage compared to the DWM is its more accurate treatment of 3-dimensional corners and edges. While it requires more multiplication operations than the DWM, in practice it can be computationally just as effective on GPUs, since time domain field simulation techniques tend to be memory bandwidth bound, while the GPUs have unused computational capacities.

4.1 The method of finite differences

The FDTD method revolves around the iterative solution of differential equations in the time domain. Variables in space and time are discretized along a uniform grid and function derivatives are substituted with finite differences. A numerical approximation of the function solution at time step 0 is calculated using the initial conditions. The calculations are then repeated for the subsequent time steps.

The method handles functions using their discretized approximations on a uniform grid in both time and space.

$$f(x,t) \longrightarrow f(i \cdot \Delta x, n \cdot \Delta t) \tag{7}$$

where Δx is the spatial step size and Δt is the temporal step size. The discretization of spatial functions in two dimensions is illustrated by Figure 7.

The derivative terms of the functions are substituted by their finite difference approximations. Such an approximation is, e.g. the ordinary differential quotient

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{8}$$

which is a first order approximation for the first derivative of $f(x)$. The importance of the approximation order will be explained later on.

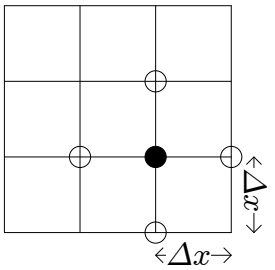

Figure 7: Function values on a 2D uniform grid

4.1.1 Approximating function derivatives

Various approximations of f' are exhibited on Figure 8. Looking at the figure one can qualitatively evaluate the approximations for f'. Our aim is to give a numerical substitution for the slope of f at point x. Approximations that use the subsequent function value to approximate the derivative at a given point are called *forward differences*, approximations using the previous value are called *backward differences*, and approximations using both are called *centered differences*.

By observing the different approximations it is apparent that while the forward and backward differences appear to be equally bad (or good) approximations, the centered difference appears to be much more accurate. It is a general rule that centered differences are more accurate, since they represent a higher order approximation of the derivatives than one sided differences using a similar number of function values.

The order of accuracy thus is determined by the number of function values used for the approximation as well as the placement of these points around the place of approximation. This accuracy is described by the *local truncation error*.

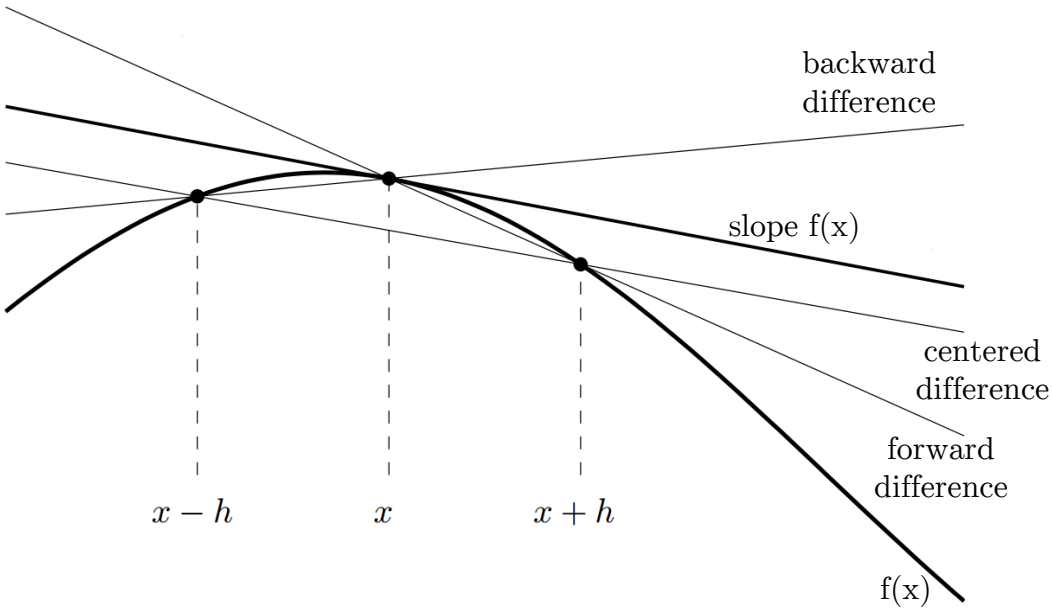

Figure 8: Approximating the derivative with differences

4.1.2 The local truncation error

A finite difference formula's local truncation error is an upper limit by which the difference may differ from the value of the continuous function derivative. It can be obtained by observing the function's Taylor series. To demonstrate the technique let's take the first order approximation presented in 4.1.

Assuming $f(x)$ is twice differentiable its first order Taylor expansion in $x + h$ is

$$f(x+h) = f(x) + f'(x)h + f''(x)(\xi)h^2 \tag{9}$$

where $f''(x)(\xi)h^2$ is the Taylor remainder term. Rearranging the equation yields the following expression.

$$\frac{f(x+h) - f(x)}{h} - f'(x) = f''(\xi)h \tag{10}$$

Which is a statement about the error of the examined finite difference (8). Considering this according to Taylor's theorem there exists a value $x \leq \xi \leq x + h$ such that the finite difference formula's error is less than the remainder term $f''(\xi)h$.

In other words the error of the formula is proportional to the step size h, where $|h| \ll 1$. Since the error is proportional to the first power of the step size the formula is called a *first order accurate finite difference approximation*. The error $f''(\xi)h$ is also called the local truncation error since it originates from the fact that the finite difference can formally be obtained by truncating the derivative's Taylor series.

4.1.3 The method of undetermined coefficients

Finite difference formulae approximate function derivatives with the linear combination of discretized function values. Such combinations can be formulated using an arbitrary number of function values chosen from arbitrary positions.

The method of undetermined coefficients is a tool for determining the coefficients of function values in order the maximize the formula's accuracy. The method is demonstrated on the derivation of a finite difference formula for the second derivative.

Let us obtain a finite difference formula for $\dfrac{\partial^2 f}{\partial x^2}$ in the form

$$a \cdot f(x_0 - h) + b \cdot f(x_0) + c \cdot f(x_0 + h) \tag{11}$$

. The described form contains our expectations regarding the included function values. We are looking for a formula that approximates $f''(x_0)$ by using the function values at the discrete points $x_0 - h$, x_0, $x_0 + h$. Since we are taking values in a symmetrical fashion from the vicinity of x_0 this is considered to be a *central difference*.

Let us observe the Taylor series of (11).

$$\begin{aligned} &a \cdot \left[f(x_0) - h \cdot f'(x_0) + h^2 \cdot \frac{f''(x_0)}{2} - h^3 \cdot \frac{f'''(x_0)}{6} + \dots \right] + b \cdot f(x_0) + \dots \\ &c \cdot \left[f(x_0) + h \cdot f'(x_0) + h^2 \cdot \frac{f''(x_0)}{2} + h^3 \cdot \frac{f'''(x_0)}{6} + \dots \right] \end{aligned} \tag{12}$$

Rearranging the equation for the derivatives yields

$$(a+b+c) \cdot f(x_0) + (c-a) \cdot h f'(x_0) + \frac{1}{2}(a+c) \cdot h^2 f''(x_0) + \frac{1}{6}(c-a) \cdot h^3 f'''(x_0) \dots \tag{13}$$

Since $|h| \ll 1$ the best approximation for f'' can be obtained by minimizing the terms with the lowest order multipliers of h.

$$a + b + c = 0 \tag{14}$$

$$c - a = 0 \tag{15}$$

$$a + c = \frac{2}{h^2} \tag{16}$$

Solving the system of linear equations we get $a = c = \frac{1}{h^2}$ and $b = -\frac{2}{h^2}$. With the coefficients we have obtained a centered finite difference formula for f''

$$\left.\frac{\partial^2 f}{\partial x^2}\right|_{x=x_0} \approx \frac{f(x_0 - h) - 2f(x_0) + f(x_0 + h)}{h^2} \tag{17}$$

Let us examine the accuracy of (17). Substituting the coefficients into (13) yields

$$D_2 f(x_0) = f''(x_0) + \frac{1}{12} f''''(x_0) h^2 + \dots \tag{18}$$

Where $D_2 f(x_0)$ is a notation for the finite difference approximation of $f''(x_0)$. After substituting $f''(x_0)$ we get

$$D_2 f(x_0) - f''(x_0) = \frac{1}{12} f''''(x_0) h^2 + \dots \tag{19}$$

According to Taylor's theorem an upper bound for the right side of the equation can be given.

$$D_2 f(x_0) - f''(x_0) = \frac{1}{12} f''''(\xi) h^2 \tag{20}$$

Thus the error of $D_2 f(x_0)$ is proportional to h^2 making the formula a *second order accurate* approximation of the second derivative.

It is worth noting that in (13) we get alternating coefficients for the successive terms of the Taylor series, i.e. $(c - a)$ and $(c + a)$, which results from the centered nature of our formula. As a consequence the term after the approximated expression – in this case f'' – will have a coefficient of 0, making the order of accuracy higher by one than it would be a case with a single sided difference. For this reason FDTD approximations use centered differences whenever applicable.

4.1.4 Finite difference schemes

The choice of selecting the discrete function values upon which the approximation of the derivative should depend describes a *finite difference scheme*. Using the method of undetermined coefficients this initial choice can be made arbitrarily. These numerical schemes differ in precision and guarantees of stability have to be determined separately for each one of them.

A scheme that is based on the substitution of second spatial derivatives with centered differences as shown in section 4.1.3 is called the *standard leapfrog scheme* (SLF). The time domain method involves arranging equations to a form, where the next temporal step can be calculated as a linear combination of values at previous iterations. The grid points involved in the calculation of the next iteration can be represented by a *stencil* that covers the points that are included in the calculation. Such stencils in three dimensions are shown on Figure 9.

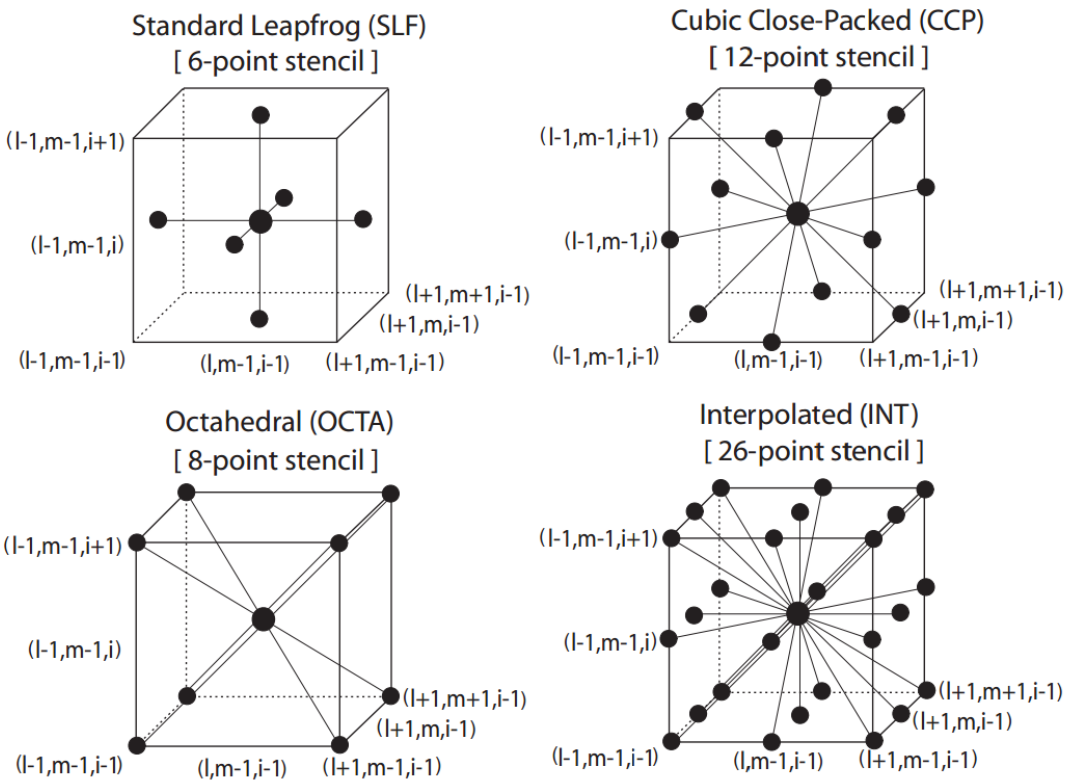

Figure 9: Compact 3D FDTD schemes [9]

Schemes using a larger number of neighbouring values to approximate the next iteration yield accurate simulation across a wider frequency range, especially when

the neighbours are chosen in a „spherical manner“ that attempts to cover all spatial directions uniformly. E.g. the SLF scheme used in this thesis' implementation strongly favors the axial directions compared to the space diagonals. The cubic close-packed (CCP) scheme however has a much more balanced approach in this regard. As a consequence the SLF scheme will exhibit significant error when comparing the wave propagation speed in axial and space diagonal directions, while the CCP scheme will have very similar propagation speeds in all directions. More on this phenomenon will be explained in the next section.

On the other hand larger schemes require more computations per grid points in each iteration and are more complicated to implement and fit to other techniques of the FDTD method. All schemes are able to provide the same accuracy at a chosen frequency range but they will do so at the cost of differing amounts of computing power. The CCP scheme is about 5 times as efficient in this regard as the SLF, but computing hardware architecture will determine how much of this theoretical advantage can be achieved in practice.

The simulation presented in this this thesis is utilizing the SLF scheme for its simplicity and easy adaptability to other techniques presented later on. The performance achieved in the typical simulations was more than satisfactory and thus the investigation of more complex schemes was not necessary. Since the typical operating range of ANC systems is in the low-frequency region below 2 kHz the SLF scheme's low cutoff frequency was not a problematic restriction.

4.1.5 Dispersion error

The varying numerical phase velocities in different directions mentioned in section 4.1.4 do not originate from the wave equation's formulation itself but are a result of its discretization.

This error is direction and frequency dependent causing the dispersion of wave packets, and thus it is referred to as *dispersion error*. A detailed analysis of the phenomenon is given by [9].

Different FDTD schemes have different numerical dispersion characteristics. The standard leapfrog scheme (SLF) has zero error in the space diagonal direction, i.e. the phase velocity has the theoretical c value at all frequencies. The magnitude of the

error is a continuous function of the direction and it is maximal in axial directions. The error in a given direction is a strictly increasing function of frequency and it is zero at the frequency 0. The dispersion of error various schemes is shown by Figure 10. Besides the schemes presented in the previous section the error of the interpolated digital waveguide mesh (IDWM), interpolated wideband (IWB), interpolated isotropic (IISO) and fourth order accurate (FOA) schemes are also given.

A concise way of describing a scheme's accuracy is the *isotropy error*, which is the absolute value of the difference between the relative phase velocity in two extreme directions of the scheme. This is also a strictly increasing function of frequency, thus it can be limited by setting an upper cutoff frequency above which signals are prohibited from entering the simulation. To determine this frequency an admissible level of isotropy error is 2% chosen as a rule of thumb.

For the 3D SLF scheme the cutoff frequency is 0.075 fs. Observing the stability criterion (28) with a spatial resolution of 0.02 m yields a practical sampling frequency of 32 kHz, thus the cutoff frequency for this case is 2.4 kHz, which is sufficient to model ANC solutions.

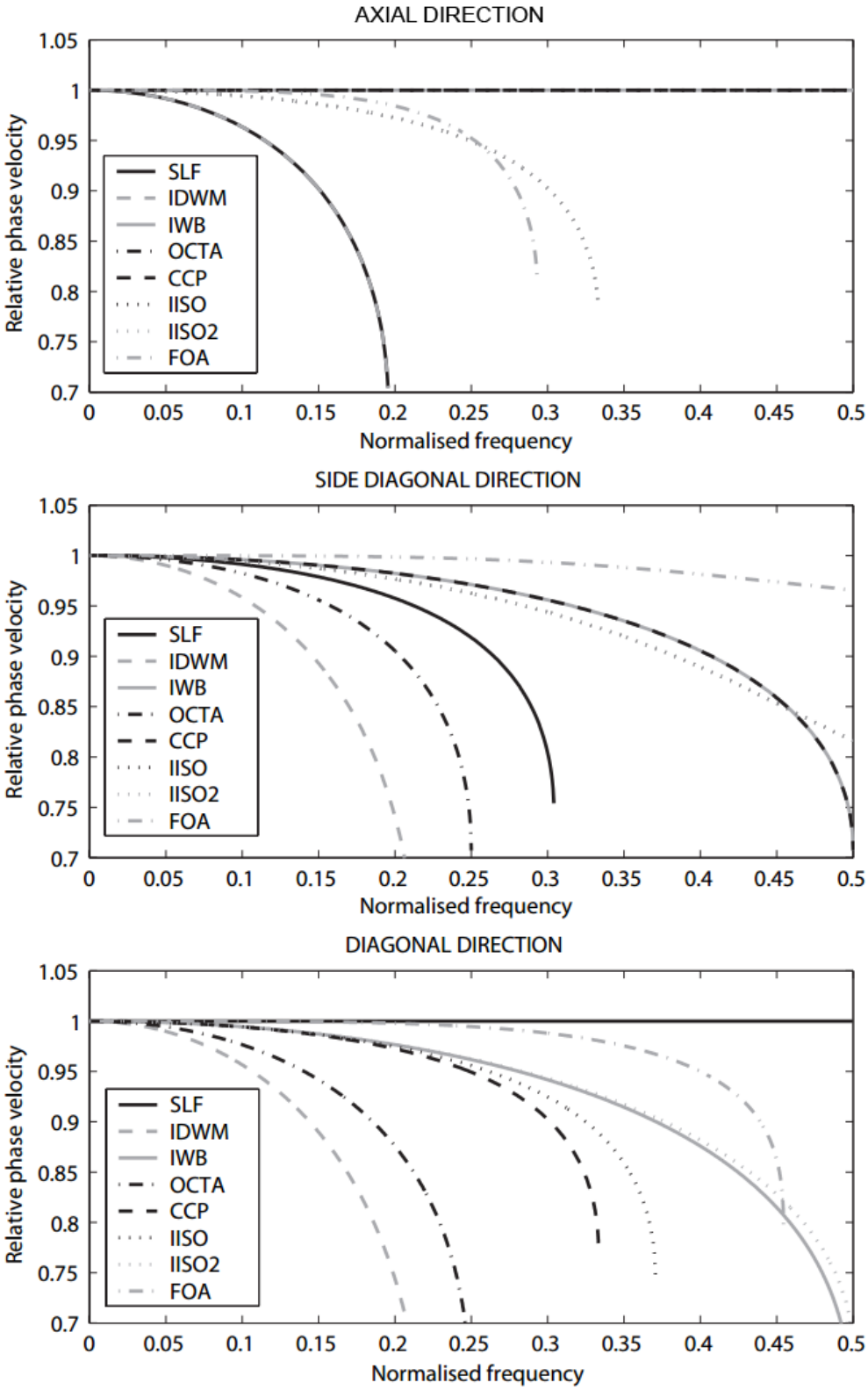

Figure 10: The dispersion error of various 3D FDTD schemes [9]

4.2 Solving the acoustic wave equation

The choice of the exact form of the physical equations will determine the correctness of an FDTD simulation. Discretization and approximation details can influence numerical precision but the existence of decisive phenomena such as diffraction and interference depend on the continuous equations.

For our purposes we have chosen the second order pressure wave equation governing *room acoustics*.

4.2.1 Waves in room acoustics

A general form of the acoustic wave equation in room acoustics is obtained from equations on particle velocity $\mathbf{u}$ and pressure p while observing the conservation of mass and inertia. It is represented by a pair of coupled differential equations.

$$\nabla p + \rho \frac{\partial \mathbf{u}}{\partial t} = 0 \tag{21}$$

$$\frac{\partial p}{\partial t} + \kappa \nabla \mathbf{u} = 0 \tag{22}$$

where ρ is the density of air and κ is its adiabatic exponent given by $\kappa = \rho c^2$. These first order equations often form the basics of numerical solutions on a *staggered grid*. In our case the second order wave equation is more useful from computing resource considerations. It can be obtained from the coupled equations by eliminating the particle velocity.

$$\frac{\partial^2 p}{\partial t^2} = c^2 \nabla^2 p \tag{23}$$

Expanding the ∇ expression in 3 dimensions yields

$$\frac{\partial^2 p}{\partial t^2} = c^2 \left[\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} \right] \tag{24}$$

which is the *second order wave equation* for the acoustic pressure where c is the velocity of sound propagation. This will be the basis of our numerical solution.

This form of the wave equation is based on the assumptions that time dependent changes in particle velocity are small compared to the static values and that particle velocity is substantially smaller than the sound velocity.

It accurately describes phenomena occurring in room acoustics, such as interference and wave diffraction. It does not account for phenomena happening at extraordinary sound pressure levels (> 120 dB) or when convection is present.

4.2.2 Discretizing the wave equation

Let us substitute both the temporal and spatial derivatives in (24) by the centered difference (17). We get

$$\frac{p^{n-1}_{x,y,z} - 2p^{n}_{x,y,z} + p^{n+1}_{x,y,z}}{\Delta t^2} = c^2 \cdot \frac{p^{n}_{x+1,y,z} + p^{n}_{x-1,y,z} + p^{n}_{x,y+1,z} + p^{n}_{x,y-1,z} + p^{n}_{x,y,z+1} + p^{n}_{x,y,z-1} - 6p^{n}_{x,y,z}}{\Delta x^2} \tag{25}$$

. Arranging the equation so that only values corresponding to the step $n+1$ are on the left we obtain the following expression.

$$\begin{aligned} p^{n+1}_{x,y,z} =& \frac{c^2 \cdot \Delta t^2}{\Delta x^2} \left[p^{n}_{x+1,y,z} + p^{n}_{x-1,y,z} + p^{n}_{x,y+1,z} + p^{n}_{x,y-1,z} + p^{n}_{x,y,z+1} + p^{n}_{x,y,z-1} - 6p^{n}_{x,y,z}\right] \\ &+ 2p^{n}_{x,y,z} - p^{n-1}_{x,y,z} \end{aligned} \tag{26}$$

Rearranging terms and substituting a single coefficient in place of the physical variables yields

$$\begin{aligned} p^{n+1}_{x,y,z} =& \lambda^2 \left(p^{n}_{x+1,y,z} + p^{n}_{x-1,y,z} + p^{n}_{x,y+1,z} + p^{n}_{x,y-1,z} + p^{n}_{x,y,z+1} + p^{n}_{x,y,z-1}\right) \\ &+ 2\left(1 - 3\lambda^2\right) p^{n}_{x,y,z} - p^{n-1}_{x,y,z} \end{aligned} \tag{27}$$

where $\lambda = \dfrac{c\Delta t}{\Delta x}$ denotes the Courant number.

The formula contains a rule originating from the wave equation, that describes how the pressure values in a given timestep should relate to previous values. This rule can be observed by the iterative solution of $p^{n+1}_{x,y,z}$ that gives the numerical approximation of the original equation.

The stencil covered by the update rule contains the midpoints of a cube's sides and the midpoint of the cube's volume, which is called the *3-dimensional leapfrog* scheme. The stencil is shown on Figure 11.

An important property of this recursive expression is that field values at a given time step are spatially independent from each other. Each one of them depends

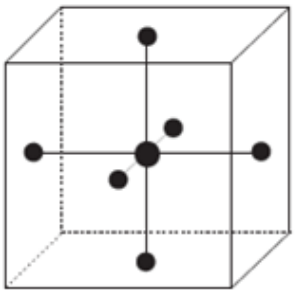

Figure 11: Stencil of the 3D leapfrog scheme

only on field values in the previous iteration and the one before that. This means that these values can be calculated in parallel that allows us to harness the power of modern multi-core processors (CPU) and graphical processing units (GPU).

4.3 Stability of the iterative formula

Although the second order wave equation describes a finite physical phenomena, the stability of derived numerical formulae is not guaranteed. Stability is a necessary condition in order to accept the discrete formula as a reasonable model for a physical process.

The *numerical stability* of a system means that no growing solution of it exists. The stability of schemes are determined by Fourier-analysis which is referred to as *von Neumann analysis* in the context of FDTD [11]. An equivalent procedure is presented by [9] where single frequency, plane wave solutions are assumed and substituted into the FDTD formula. It is investigated then for which values of the Courant number these solutions can grow in time.

The analysis of the 3D leapfrog scheme by [9] yields that it is numerically stable if the Courant number is bound as

$$\lambda^2 = \frac{c^2 \cdot \Delta t^2}{\Delta x^2} \leq \frac{1}{3} \tag{28}$$

.

4.4 Locally reacting surfaces

The physically correct simulation of wave reflection would require solving the wave equation inside the reflecting bodies – taking into account the different propagation

speed and acoustic impedance of its material. For most purposes however reflecting bodies can be sufficiently described by their geometries and a single frequency dependent parameter of their material. This parameter is usually either the *absorption* or *reflectance* coefficient of given material.

The $0 < \alpha < 1$ absorption coefficient is a real value, and it is defined as the ratio of absorbed energy and incident energy contained in an incoming wave.

$$\alpha = \frac{\text{absorbed energy}}{\text{incident energy}} \tag{29}$$

The absorption is measured at octave bands typically starting at 125 Hz. Tables containing the absorption coefficient are available for a wide range of materials.

Related to the absorption coefficient, the $0 < |R| < 1$ reflection factor or *reflectance* is a complex value, and it is defined as the ratio of the reflected and incoming pressure waves.

$$R = \frac{\text{sound pressure of the reflected wave}}{\text{sound pressure of the incoming wave}} \tag{30}$$

The two coefficients are thus related, for plane waves the following equation holds.

$$\alpha = 1 - |R|^2 \tag{31}$$

Although the absorption coefficient lacks the phase information regarding the reflected wave, it is widely used, because for most applications concerned with the simulation of complete rooms, e.g. auralisation don't require such accuracy. The results experienced in such complex environments are determined mostly by the geometrical properties, hence the absorption coefficient contains sufficient information for these purposes.

4.4.1 Theoretical background

The technique of *locally reacting surfaces* (LRS) ignores wave propagation phenomena occurring inside reflecting objects and focuses only on the observed properties of reflection, i.e. its model treats every surface point as an element with a locally defined impedance.

A formulation of LRS for the 3D wave equation is presented by [9] that is implemented in the simulation. Surface points are locations of the computational grid

that have a neighbouring point inside the volume of a reflecting object. The FDTD stencil is modified by replacing these neighbouring points by *ghost points*, that have a value defined by the impedance assigned to the surface point. The occurrence of boundaries is demonstrated by Figure 12.

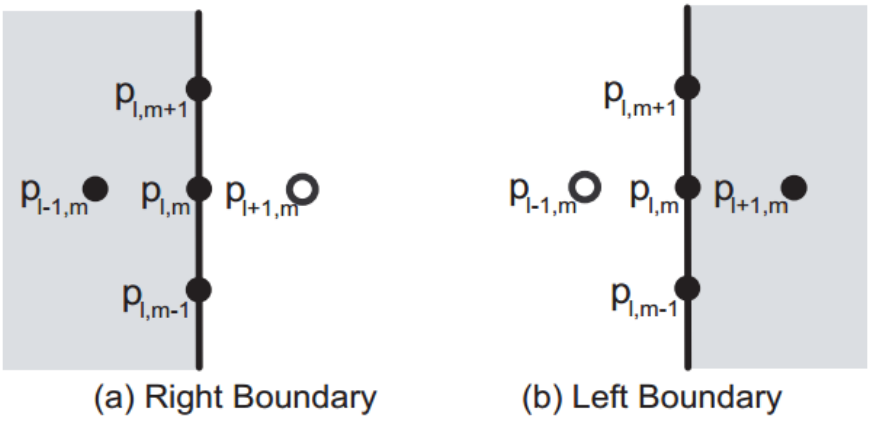

Figure 12: Left and right boundaries in the standard leapfrog scheme – the simulation region is shaded with grey and the reflecting material is shown in white [9]

In case of a sound wave travelling in the positive x direction the acoustic pressure is related to the incoming wave's particle velocity by the boundary impedance Z_ω.

$$p = Z_\omega u_x \tag{32}$$

Substituting this expression into the conservation of momentum and eliminating the particle velocity yields the *boundary condition* for the right boundary.

$$\frac{\partial p}{\partial t} = -c\xi_\omega \frac{\partial p}{\partial x} \tag{33}$$

where $\xi_\omega = \dfrac{Z_\omega}{\rho c}$ is the normalised wall impedance known as the *specific acoustic impedance*. The relation between the reflection coefficient $R(\theta)$ and ξ_ω is given by

$$R(\theta) = \frac{\xi_\omega cos\theta - 1}{\xi_\omega cos\theta + 1}. \tag{34}$$

According to this relation the choice of $\xi_\omega = 1$ will result in a surface that has zero reflection for planar waves of normal incidence. This model is sufficiently accurate for modelling specular reflections for most materials, however the angle dependence makes it unsuitable for creating anechoic conditions. Such situations can be modelled with the PML technique presented in section 4.5.

Substituting the derivatives in (33) with centered differences yields

$$\frac{p^{n+1}_{x,y,z} - p^{n-1}_{x,y,z}}{2\Delta t} = -c\xi_\omega \left(\frac{p^{n}_{x+1,y,z} - p^{n}_{x-1,y,z}}{2\Delta x} \right). \tag{35}$$

which provides the basis for calculating the values of ghost points. Depending on the treatment of ξ_ω the resulting method can model frequency independent or – in the case of more elaborate substitutions – frequency dependent surfaces as well. Real materials behave in a frequency dependent way and accurate models of acoustic environments require frequency dependent models. A highly flexible way of modelling arbitrary frequency dependent reflection properties is introduced in the next section.

4.4.2 Locally reacting surfaces with digital impedance filters

Real-life surfaces exhibit frequency dependent reflexive behaviour and wideband models of acoustic environments need to be able to model them appropriately. A highly flexible method is presented by [9] that has been implemented in the discussed simulation.

In the presented approach the impedance coefficient in (33) is generalized to utilize digital filters thus providing a precisely adjustable frequency dependent model. The central element of the surface model in this case is termed as a *digital impedance filter*.

The filter is treated as a direct form IIR filter implementation. A practical way of obtaining the filter is using the widely available absorption coefficients of surfaces and designing a corresponding reflectance filter. The reflectance values for octave bands can be calculated from the absorption values using the $|R| = \sqrt{(1-\alpha)}$ relation. From this data a normal-incidence digital filter $R(z)$ is designed. In order to represent a physical material the filter must be passive, i.e. $|R(z)| \leq 1$ must hold for all frequencies. The impedance filter $\xi_\omega(z)$ can then be obtained by

$$\xi_\omega(z) = \frac{1 + R(z)}{1 - R(z)}. \tag{36}$$

After this step the stability of the resulting impedance filter should be checked.

In its direct implementation the filter is represented by the coefficients of its transfer function.

$$\xi_\omega(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_N z^{-N}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}} \tag{37}$$

This specific implementation used in the presented software relies on transfer functions where the numerator and denominator are of the same order N.

The ghost point values are calculated using the filter's output through a g_n intermediate value. The update formula in case of the SLF scheme for a right boundary is as follows.

$$p^{n+1}_{x,y,z} = [\lambda^2 \left(2p^n_{x-1,y,z} + p^n_{x-1,y,z} + p^n_{x,y+1,z} + p^n_{x,y-1,z} + p^n_{x,y,z+1} + p^n_{x,y,z-1}\right) \\ +2\left(1-3\lambda^2\right)p^n_{x,y,z} + \frac{\lambda^2}{b_0}g^n + \left(\frac{\lambda a_0}{b_0} - 1\right)p^{n-1}_{x,y,z}]/\left(1 + \frac{\lambda a_0}{b_0}\right) \tag{38}$$

where the intermediate value is given by

$$g^n = \sum_{i=1}^{N}\left(b_i x^{n-i} - a_i y^{n-i}\right) \tag{39}$$

which is calculated using the impedance filter's input and output equations.

$$x^n = \frac{a_0}{\lambda b_0}\left(p^{n+1}_{x,y,z} - p^{n-1}_{x,y,z} - \frac{g^n}{b_0}\right) \tag{40}$$

$$y^n = \frac{1}{a_0}\left(b_0 x^n + g^n\right). \tag{41}$$

In case of outer edges and corners – from the viewpoint of the simulation region – two and three points of the SLF stencil will be inside the reflecting material thus two and three points have to be substituted with ghost points in the same manner as presented above. Each ghost point requires a separate filter, i.e. input and output values have to be stored for them separately. The appearance of corners in the scheme is demonstrated by Figure 13.

Inner corners don't require special treatment, since none of the stencil points are placed inside objects in their case.

4.5 Truncating the simulation domain – perfectly matched layer

4.5.1 Theoretical background

Simple acoustic phenomena, such as interference or diffraction, are best investigated under anechoic conditions. In this case typical wavefront formations can be easily observed or measured, and mathematical models can work with very simple transfer

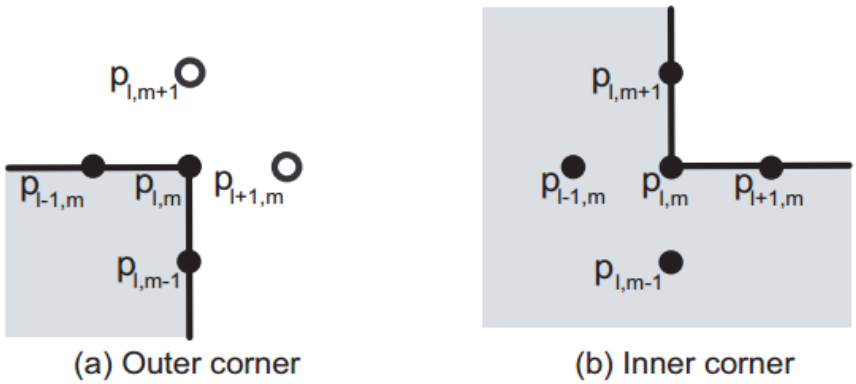

Figure 13: Inner and outer corners in the standard leapfrog scheme [9]

functions. Such conditions were assumed in the simulation software presented in section 3.1.

While trivial with other techniques, simulating an anechoic environment with the numerical solution of the wave equation can be a challenging task. The computational grid has to be truncated in a way that does not introduce unwanted phenomena into the simulation such as reflections. The simplest form of grid truncation is the *Dirichlet boundary condition* where the outermost layer of the simulated volume has a fixed value of 0. This condition however creates an envelope around the simulation region with a reflectance of -1.

The LRS technique presented in section 4.4 is useful for most practical situations modelling specular reflections. However its modelled reflectance is only accurate for waves that reach the surface at normal incidence. Thus in order to model anechoic conditions other techniques are required.

The initial investigations for the truncation of the computational grid were looking for effective *absorbing boundary conditions* (ABC). Boundary conditions in general are rules that determine the value of the outermost layer of the grid. ABCs are aiming to extrapolate these boundary values from the inner grid values in such a way that they represent the natural continuation of the wavefront so that no reflection occurs.

While perfectly absorbing conditions have been found in one dimension, finding such a condition for two and three dimensions has so far eluded researchers. Looking for a solution in these cases is very hard due to the infinite number of directions in

which waves can propagate in multiple dimensions, and the existence of a perfect ABC in three dimensions seems unlikely [6].

The lack of a good ABC solution in higher dimensions directed attention to *absorbing boundary layers* instead. A method of constructing such a layer for electromagnetic waves was published by Berenger [1] in 1994, which has been successfully adopted in all other areas dealing with numerical wave simulations. The approach was termed the *perfectly matched layer* (PML) which refers to the layer's matched property, i.e. the wave impedance of the layer and the simulation region are perfectly matched, hence no reflections will occur at this boundary. Inside the layer the wave equation is modified such that waves propagating outwards will be exponentially attenuated. The best implementation of the PML in the time domain is still being debated [5] but the theoretical background is common for them.

The technique is based on *complex coordinate stretching* explained by [6]. The solutions and equations are analytically continued along complex spatial variables, e.g. a complex x contour. In the region of interest – where the results are meaningful to us – this contour is the same real variable as before, but outside the region it takes a growing imaginary part that causes oscillating waves to decay exponentially. The theoretical formulation investigates the propagation of plane wave solutions in infinite space, but the resulting technique will apply to all practical waveforms occurring in the simulations.

Let $\mathbf{w}(\mathbf{x}, t)$ be a solution of the wave equation in infinite space. Assuming that the space far from the region of interest is homogeneous, linear and time invariant, this solution can be given as a superposition of planewaves.

$$\mathbf{w}(\mathbf{x}, t) = \sum_{\mathbf{k},\omega} \mathbf{W}_{\mathbf{k},\omega} e^{j(\mathbf{k}\cdot\mathbf{x}-\omega t)} \tag{42}$$

where $\mathbf{W}_{\mathbf{k},\omega}$ are the amplitudes corresponding to the solutions with $\mathbf{k}$ wavevector and ω angular frequency. These radiating solutions can be decomposed into functions of the form

$$\mathbf{W}(y, z) e^{j(kx-\omega t)}. \tag{43}$$

Since (43) is an analytical function of x it can be analytically continued for complex x values. In order to achieve the intended decaying behaviour we will evaluate it for an x contour, which remains unchanged in the region of interest but

gains an imaginary part outside it as seen on Figure 14.

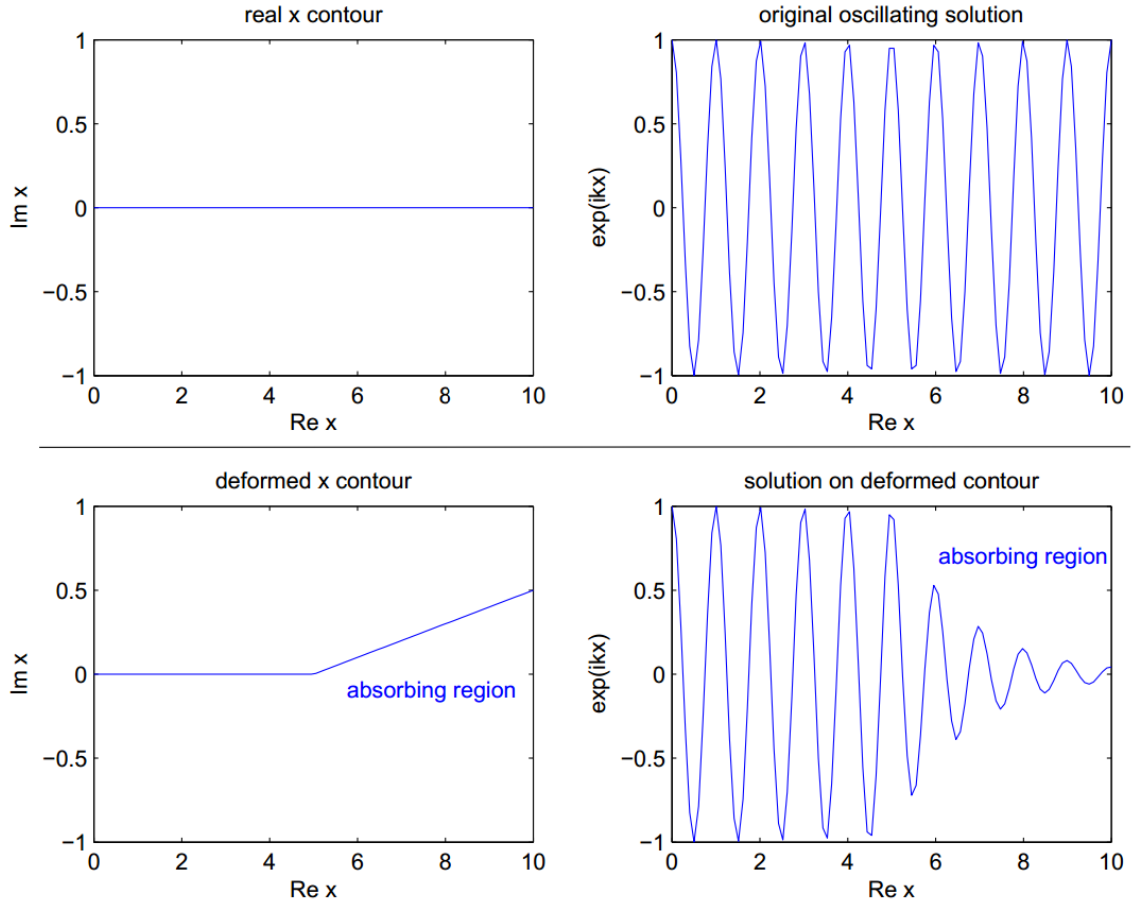

Figure 14: The x contour and corresponding wave solutions inside and outside the region of interest [6]

With the complex contour the solutions can also be expressed as

$$\mathbf{W}(y,z)e^{jk(\Re(x)+j\Im(x))} \cdot e^{-j\omega t} \tag{44}$$

where the second exponential can be rewritten as

$$e^{jk\Re(x)} \cdot e^{-k\Im(x)} \tag{45}$$

which is exponentially decaying where $\Im(x) > 0$. Thus this region acts as an absorbing material. However compared to absorbing materials the important feature of this technique is that the solutions are unchanged where $\Im(x) = 0$, thus the absorbing region causes no reflections, hence it is perfectly matched to the region of interest.

To be easily solved by traditional FDTD methods the results are transformed back into real coordinate variables. Let us denote the complex contour by $\bar{x}$. The real part of $\bar{x}$ equals to x and the imaginary part is some f function of it, thus $\bar{x} = x + jf(x)$. Changing the PML equation back to real variables is carried out by substituting the $\partial\bar{x}$ differentials along the deformed contour with $\partial\bar{x} = 1 + j\frac{df}{dx}\partial x$.

The entire process of the PML method can be carried out by replacing the derivatives of the original equation according to the following rule

$$\frac{\partial}{\partial x} \rightarrow \frac{1}{1 + j\frac{\sigma_x(x)}{\omega}} \frac{\partial}{\partial x} \tag{46}$$

where $\frac{\sigma(x)}{\omega}$ is a choice for $\frac{\partial f}{\partial x}$ that guarantees the frequency independent attenuation of wave solutions. Observing (45) we can see that the attenuation factor is proportional to the wavenumber k. Applying (46) transforms the wave solution to

$$e^{jkx} \cdot e^{-\frac{k}{\omega}\int^x \sigma_x(x)dx} \tag{47}$$

where $\frac{k}{\omega} = \frac{1}{c_x}$ is the inverse of the phase velocity for waves of all frequencies.

The transformed differential equation thus contains a frequency dependent term, which can be substituted by a constant for single frequency solutions, or simply eliminated in 1D cases. In multiple dimensions the *auxiliary differential equation* approach is used to obtain expressions for the time domain. In short, expressions containing a $\frac{j}{\omega}$ factor are substituted with an auxiliary field variable ψ. The equation describing the substitution is then multiplied with $-j\omega$, which results in an equation for $-j\omega\psi$. For the exponential wave solution this expression is equivalent to $\frac{\partial\psi}{\partial t}$ in the time domain, hence we gain an additional differential equation coupled to our PML modified equation. The two equations can be solved with the presented FDTD methods in parallel.

Creating a perfectly matched region for any equation can be done by substituting function derivatives according to (46). The σ_x function determines the absorbing conditions occurring at various regions and in various directions of the simulation. In the region of interest $\sigma_R = 0$ should be observed where R is the direction along which the substitution acts. In the absorbing region $\sigma_R(r) > 0$ should be chosen as a smoothly increasing function of the spatial variable.

The PML technique attenuates planar waves propagating in the R axial directions where $\sigma_R(r) > 0$. In practice this means that normal wave components arriving

at the boundary of the region will be attenuated. However since the absorbing region has a thickness of multiple grid layers even waves coming in at an angle will suffer significant attenuations and parallel components will eventually reach another absorbing region attenuating in a normal direction. The configuration of PML layers around the region of interest and the absorbing directions are shown on Figure 15.

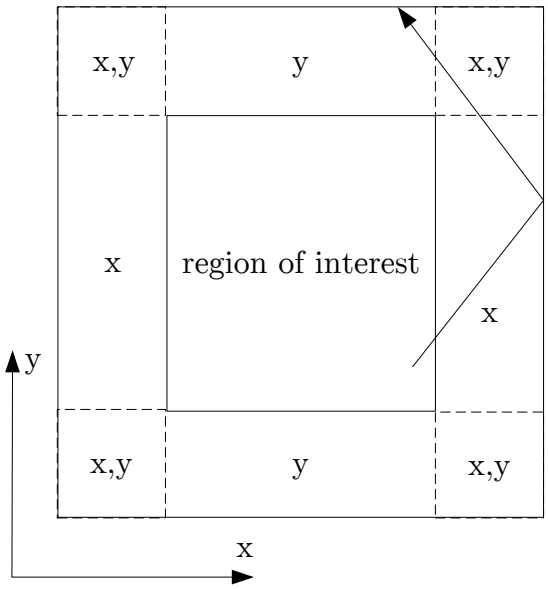

Figure 15: Perfectly matched layer attenuating waves along the given directions

Since the waves entering the PML region are exponentially attenuated we can now truncate the simulating region. The simplest and most common solution is simply applying the Dirichlet boundary condition at the outer edge of the PML. Waves reaching this reflective boundary will already be attenuated and their decay will continue on their way back to the region of interest, thus the reflected waves will be exponentially small.

4.5.2 Practical solutions

While the PML technique guarantees a reflection free boundary between the inner and absorbing regions, the effect of discretization results in reflections depending on the changes in $\sigma_R(r)$. As a consequence one cannot choose an arbitrarily narrow PML with a very large $\sigma_R(r)$, but has to implement an absorbing region several layers wide and use a $\sigma_R(r)$ that changes smoothly and increases not too quickly

outside the region of interest.

Another task for creating practical implementation of PMLs is creating an FDTD scheme that can be applied to wideband signals like the original schemes for the wave equation such as (27). The substitution dictated by (46) contains a frequency dependent term $j \cdot \frac{\sigma_x(x)}{\omega}$, which is a necessary choice if one wants the PML region's absorption to be frequency independent. This substitution however will not result in schemes that are applicable to wideband signals.

In order to create practical PML solutions the frequency dependent term is eliminated by using *auxiliary variables*. These variables are calculated by solving equations coupled to the original wave equation. These variables can be chosen in infinitely many ways and as a result there are many different PML implementations in the time domain. At this point there is no clear consensus on the best one, but the superiority of some implementations compared to others can be determined based on stability and computational requirements.

Many implementations require the statement of the acoustic problem in the form of coupled hyperbolic equations, while there are others formulated for the second order wave equation, and are evidently more practical in our case.

The implementation used for my simulation is presented by [5]. Among the various implementations I have investigated this is clearly the most practical for my purposes, since it is formulated for the second order wave equation, has good stability, and requires only four auxiliary variables in three dimensions. The coupled PML modified equations are given as follows.

$$\frac{\partial^2 p}{\partial t^2} + (\sigma_x + \sigma_y + \sigma_z)\frac{\partial p}{\partial t} + (\sigma_x\sigma_y + \sigma_y\sigma_z + \sigma_z\sigma_x)p = \nabla \cdot (c^2 \nabla p) + \nabla \mathbf{\Phi} - \sigma_x\sigma_y\sigma_z\psi \quad (48)$$

$$\mathbf{\Phi}_t = \Gamma_1 \mathbf{\Phi} + c^2 \Gamma_2 \nabla p + c^2 \Gamma_3 \nabla \psi \quad (49)$$

$$\psi_t = p \quad (50)$$

where

$$\Gamma_1 = \begin{bmatrix} -\sigma_x & 0 & 0 \\ 0 & -\sigma_y & 0 \\ 0 & 0 & -\sigma_z \end{bmatrix}, \quad (51)$$

$$\Gamma_2 = \begin{bmatrix} \sigma_y + \sigma_z - \sigma_x & 0 & 0 \\ 0 & \sigma_z + \sigma_x - \sigma_y & 0 \\ 0 & 0 & \sigma_x + \sigma_y - \sigma_z \end{bmatrix} \tag{52}$$

and

$$\Gamma_3 = \begin{bmatrix} \sigma_y \sigma_z & 0 & 0 \\ 0 & \sigma_z \sigma_x & 0 \\ 0 & 0 & \sigma_x \sigma_y \end{bmatrix}. \tag{53}$$

Inside the simulation region the σ damping profiles and thus the auxiliary variables $\mathbf{\Phi}$ and ψ vanish.

The equations are discretized on the standard grid for p, a *space staggered grid* for $\mathbf{\Phi}$ and a *time staggered* grid for ψ. Staggered grids have points that are shifted by a half grid constant along each axes compared the other, i.e. a grid point of one grid is exactly halfway between two points on the other grid. This technique has the advantage that using only two neighbouring values from one grid a centered difference can be calculated in respect to the other's values. The function derivatives are substituted by standard second order centered differences. The modified, discretized wave equation is as follows.

$$\begin{aligned} &\frac{p^{n+1}_{x,y,z} - 2p^n_{x,y,z} + p^{n-1}_{x,y,z}}{\Delta t^2} + (\sigma_{x[x]} + \sigma_{y[y]} + \sigma_{z[z]}) \frac{p^{n+1}_{x,y,z} - p^{n-1}_{x,y,z}}{2\Delta t} + \\ &(\sigma_{x[x]}\sigma_{y[y]} + \sigma_{y[y]}\sigma_{z[z]} + \sigma_{z[z]}\sigma_{x[x]}) p^n_{x,y,z} = \\ &\frac{c^2}{\Delta x^2} \frac{p^n_{x+1,y,z} + p^n_{x-1,y,z} + p^n_{x,y+1,z} + p^n_{x,y-1,z} + p^n_{x,y,z+1} + p^n_{x,y,z-1} - 6p^n_{x,y,z}}{\Delta x^2} \\ &+ \frac{\bar{\Phi}^n_{1x+\frac{1}{2},y,z} - \bar{\Phi}^n_{1x-\frac{1}{2},y,z} + \bar{\Phi}^n_{1x,y+\frac{1}{2},z} - \bar{\Phi}^n_{1x,y-\frac{1}{2},z} + \bar{\Phi}^n_{1x,y,z+\frac{1}{2}} - \bar{\Phi}^n_{1x,y,z-\frac{1}{2}}}{\Delta x} \\ &- \sigma_{x[x]}\sigma_{y[y]}\sigma_{z[z]} \frac{\psi^{n+\frac{1}{2}}_{x,y,z} - \psi^{n-\frac{1}{2}}_{x,y,z}}{2} \end{aligned} \tag{54}$$

where the cell averages of the auxiliary functions Φ_1, Φ_2 and Φ_3 are defined as

$$\hat{\Phi}^n_{1x+\frac{1}{2},y,z} = \frac{1}{4}\left(\Phi^n_{1x+\frac{1}{2},y-\frac{1}{2},z-\frac{1}{2}} \Phi^n_{1x+\frac{1}{2},y-\frac{1}{2},z+\frac{1}{2}} + \Phi^n_{1x+\frac{1}{2},y+\frac{1}{2},z-\frac{1}{2}} + \Phi^n_{1x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}}\right), \tag{55}$$

$$\hat{\Phi}^n_{2x,y+\frac{1}{2},z} = \frac{1}{4}\left(\Phi^n_{2x-\frac{1}{2},y+\frac{1}{2},z-\frac{1}{2}} \Phi^n_{2x-\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}} + \Phi^n_{2x+\frac{1}{2},y+\frac{1}{2},z-\frac{1}{2}} + \Phi^n_{2x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}}\right), \tag{56}$$

$$\hat{\Phi}^n_{3x,y+\frac{1}{2},z} = \frac{1}{4}\left(\Phi^n_{3x-\frac{1}{2},y-\frac{1}{2},z+\frac{1}{2}}\Phi^n_{3x-\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}} + \Phi^n_{3x+\frac{1}{2},y-\frac{1}{2},z+\frac{1}{2}} + \Phi^n_{3x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}}\right). \tag{57}$$

The discretized equations for the auxiliary variables are as follows. These have to be concurrently updated during the iterative solution of the wave equation.

$$\frac{\psi^{n+\frac{1}{2}}_{x,y,z} - \psi^{n-\frac{1}{2}}_{x,y,z}}{\Delta t} = p^n_{x,y,z}, \tag{58}$$

whereas for Φ_1 we use

$$\begin{aligned}\frac{\Phi^{n+1}_{1x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}} - \Phi^{n-1}_{1x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}}}{\Delta t} \\ = -\sigma_{x[x+\frac{1}{2}]}\frac{\Phi^{n+1}_{1x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}} + \Phi^{n-1}_{1x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}}}{2} \\ + \left(\sigma_{y[y+\frac{1}{2}]} + \sigma_{z[z+\frac{1}{2}]} - \sigma_{x[x+\frac{1}{2}]}\right) D^h_x p^{n+\frac{1}{2}}_{x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}} \\ + \sigma_{y[y+\frac{1}{2}]}\sigma_{z[z+\frac{1}{2}]} D^h_x \psi^{n+\frac{1}{2}}_{1x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}},\end{aligned} \tag{59}$$

where

$$D^h_x p^{n+\frac{1}{2}}_{x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}} = \frac{1}{2}\left(\frac{\bar{p}^{n+1}_{x+1,y+\frac{1}{2},z+\frac{1}{2}} - \bar{p}^{n+1}_{x,y+\frac{1}{2},z+\frac{1}{2}}\bar{p}^{n}_{x+1,y+\frac{1}{2},z+\frac{1}{2}} - \bar{p}^{n}_{x,y+\frac{1}{2},z+\frac{1}{2}}}{\Delta x}\right), \tag{60}$$

$$D^h_x \psi^{n+\frac{1}{2}}_{1x+\frac{1}{2},y+\frac{1}{2},z+\frac{1}{2}} = \frac{\bar{\psi}^{n+\frac{1}{2}}_{x+1,y+\frac{1}{2},z+\frac{1}{2}} - \bar{\psi}^{n+\frac{1}{2}}_{x,y+\frac{1}{2},z+\frac{1}{2}}}{\Delta x}. \tag{61}$$

The cell averages for p and ψ are defined as

$$\bar{p}^n_{x,y+\frac{1}{2},z+\frac{1}{2}} = \frac{1}{4}\left(p^n_{x,y,z} + p^n_{x,y,z+1} + p^n_{x,y+1,z} + p^n_{x,y+1,z+1}\right), \tag{62}$$

$$\bar{\psi}^n_{x,y+\frac{1}{2},z+\frac{1}{2}} = \frac{1}{4}\left(\psi^{n+\frac{1}{2}}_{x,y,z} + \psi^{n+\frac{1}{2}}_{x,y,z+1} + \psi^{n+\frac{1}{2}}_{x,y+1,z} + \psi^{n+\frac{1}{2}}_{x,y+1,z+1}\right). \tag{63}$$

The finite difference approximations for Φ_2 and Φ_3 are analogous.

As previously mentioned as a result of the discretization „sudden changes“ in the value of σ_R cause reflections. Thus its values are chosen to represent a smoothly increasing, twice differentiable function given in one direction as

$$\sigma_R(r) = \bar{\sigma}\left(\frac{|x_R - a_R|}{L_R} - \frac{sin\left(\frac{2\pi|x_R - a_R|}{L_R}\right)}{2\pi}\right) \tag{64}$$

for $a_i \leq |x_R| \leq a_R + L_R$ and 0 otherwise, where a_R is the size of the computation region and L_R is the length of the PML. The $\sigma_R(r)$ function is demonstrated by Figure 16 for a case of $\hat{\sigma} = 8000$.

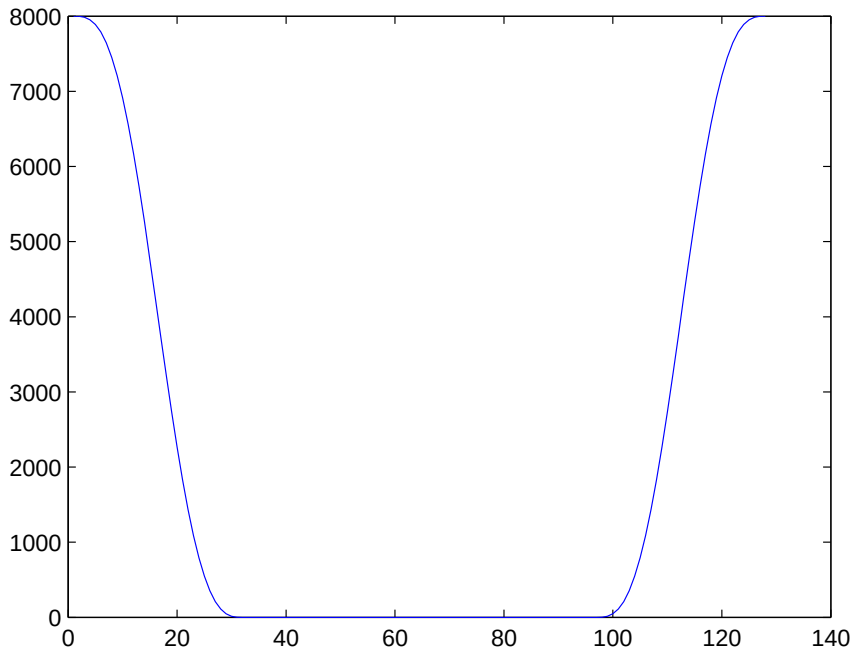

Figure 16: Profile of the 31 cell wide dampening factor in a 128 cell wide simulation

5 Simulation software based on the numerical solution of the acoustic wave equation

This section gives a detailed description of the implemented simulation software covering the theoretical considerations and practical solutions. Discarding the previous simulation framework it has been implemented bottom up using technologies –, e.g. C++, OpenCL, massive parallelism – that allow the exploitation of the application's high performance nature.

An overview of the system's structure and its interaction with other components is shown by Figure 17.

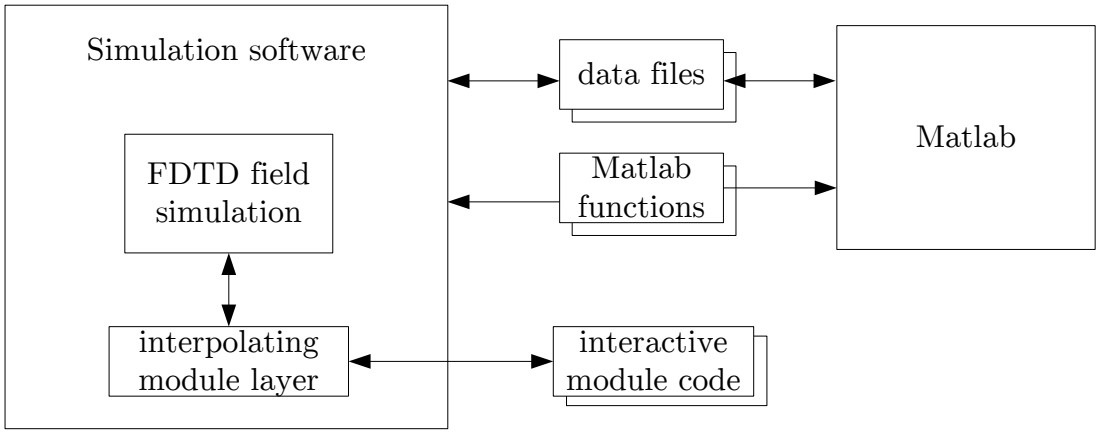

Figure 17: Overall system design

The software is used to execute acoustic field simulations taking into account various parameters read from data files, most of which will have been created using Matlab.

Certain parameters need to be calculated based on the user's commands according to complex mathematical formulae. These formulae are evaluated by Matlab functions that are called by the software in runtime utilizing user given parameters.

A central role – and the primary reason for developing this software – is played by interactive modules written in OpenCL language. The language is a subset of the C99 language, and due to the module layer they can resemble code developed for digital signal processors very closely. These modules are loaded and compiled in

runtime and can be easily modified by a text editor when module parameters have to be tweaked.

During the ANC simulations a noise cancelling module will be used implementing the ELMS algorithm. The module's code has been carried over from the DSP project used in the real-life experiments. With the exception of the `process` function's header the complete code can be copied to the DSP without modifications.

5.1 Initial specifications

In order to appropriately simulate the investigated noise cancelling situations, the implemented software should

- model acoustic wave propagation in a physically correct manner using the FDTD method
- allow the simulation of volumes up to 200 m^3
- allow the single pass simulation of all wideband phenomena up to 2 kHz including the simulation of frequency dependent reflective surfaces
- allow the description of the environment in an easy to use, graphical 3D modelling tool
- enable the usage of various materials for different reflecting surfaces
- allow the simulation of anechoic situations
- allow the importing of simulation variables from and the exporting of simulation results to Matlab
- have an online visualisation tool for diagnostic purposes
- contain a massively parallel description of the algorithms enabling the utilization of GPU devices
- provide a transparent mechanism through which programmable modules – such as the ANC system – can interact with the field conditions as if they were interacting with the physical environment.

The last two requirements present the main novelty of the implemented program.

The second to last requirement was met by implementing the core algorithms of the program in OpenCL language while exposing all parallelism inherent in the equations of the FDTD method. The program can run on a wide range of supported devices including CPUs and GPUs of most major manufacturers. The parallel description of the algorithms allow the device drivers to compile binaries that can utilize all computational resources of the selected hardware. This means that in case of a modern multicore processor, e.g. Intel i7 all cores will be utilized including the vector capabilities such as AVX as well.

The last requirement is a natural consequence of the noise cancelling problem domain. The inner workings of complex systems such as an ANC system should be clearly separated from the simulation of physical conditions. In order to ensure the reusability of this simulation software and the testability of signal processing modules, the physical simulation should be transparent to the investigated module. This means that the module code should not contain anything that is dependent on the implementation specifics of the field simulation.

In other words the module code should resemble the code running on a real world digital signal processor as closely as possible.

5.2 Technology

The software has been implemented in C++ and OpenCL languages and it relies on some additional functions written in Matlab. Some features of the C++11 standard have been used, and consequently the minimum requirement for compiling it is the 2012 version of Microsoft Visual Studio.

5.2.1 Computations on GPGPUs

The choice of language was dictated by the application's high performance nature and its adaptability to modern general purpose graphical processors (GPGPU). The architecture and implementation was designed so that the software is well adapted to the efficient execution on modern multi-core processors and GPUs of the major manufacturers.

The paradigm upon which GPGPU computations are based is called *massive*

parallelism. It is based on the data parallel execution of thousands of threads simultaneously. These threads are very lightweight compared to CPU threads due to the assumptions upon which GPU architectures are built.

The data parallelism refers to the fact that these threads are executing the same operations but on different pieces of data. This is a sort of SIMD approach but is usually referred to as superscalar architecture. The difference between them is that SIMD devices – such as DSPs, or CPUs with vector extensions – are executing vectorized commands explicitly expressing parallelism on the level of the command set architecture. Superscalar devices on the other hand are fed by a large number of scalar operations and their parallel execution will be determined by the device's scheduler in runtime. This also means that the order of execution is non-deterministic, and in case of more complex programs the achieved throughput of the system cannot be exactly predicted.

Another characteristic property of GPGPU architectures that makes the execution of thousands of threads feasible, is that unlike CPUs they don't provide cache coherence. This makes them suitable only for algorithms that require minimal or no synchronization between threads at all. Synchronization is only explicitly possible through the local memory between small batches of threads or the global memory in general.

Not providing cache coherence frees the GPU from the significant burden of synchronization, and allows a larger section of the die to be dedicated to I/O and arithmetic units. As a result GPUs offer about $10 - 20$ times higher raw computational power than similarly priced CPUs. With the availability of large bus width (256–512 bit) high-frequency GDDR5 memory they can have a comparable advantage in memory bandwidth. This makes the execution of both compute and memory bound computations more efficient on GPUs, given that they can exhibit massively parallel nature.

Explicit FDTD schemes are trivially executable using a large number of independent threads and thus are very well suited to GPGPU computations.

5.2.2 OpenCL

OpenCL [8] is an open standard for computing languages aimed primarily at data parallel applications and computations carried out by GPUs. It is supported by a large number of manufacturers and architectures, e.g. Intel, AMD, Nvidia, ARM. There are working implementations for running OpenCL code on CPUs, GPUs and embedded processors, while DSPs and FPGAs are among the targeted devices as well.

OpenCL shows a very close similarity to Nvidia's proprietary solution, CUDA, almost all features of the languages have their counterpart in the other. Unlike CUDA, the development and capabilities of OpenCL is largely focused on multi-platform availability and adaptability.

Platforms supporting OpenCL have a driver, containing an OpenCL compiler that can compile code in a way that is most efficient for the underlying hardware. Programs using OpenCL have to communicate with this driver through the OpenCL API in order to run computations. The lifecycle of the process is shown by Figure 18.

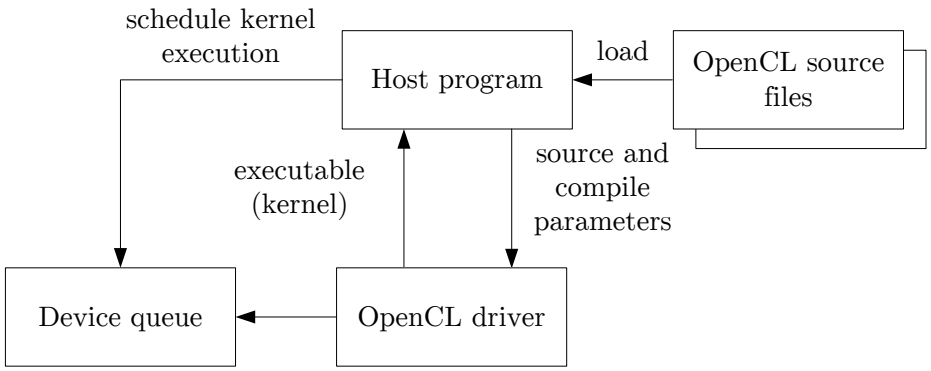

Figure 18: The lifecycle of an OpenCL computation

Although OpenCL programs can be executed on various kinds of devices, in order to avoid ambiguity the terms *host* and *device* will be henceforth used for the CPU and GPU of the same system respectively, and they will be treated as synonyms.

It is the responsibility of the host to initiate the compilation of OpenCL source files. The driver then returns handles to the compiled binaries called *kernels*. It is

also the hosts responsibility to provide input data to the kernels. Kernels can only work with data present on the device's memory. The host requests handles from the OpenCL driver to the device's memory. Using these handles the host needs to explicitly synchronize data between the host and device memory. E.g. the CPU asks for a handle to a section of the GPU's onboard memory and copies initial data to it. Once the computations are done it copies the results from the GPU's memory to the host memory (system RAM).

Synchronization tasks are considered to be costly in terms of efficiency since they are carried out through the PCI-E bus, which has a very limited bandwidth compared to the device's memory. Thus synchronization tasks generally have to be kept at a minimum, and only used before and after lengthier computational sessions. E.g. the CPU has no knowledge of the acoustic field variables during the runtime of the simulation, even measurement tasks – such as integrating acoustic pressure values – are carried out by the GPU and the results are stored in the GPU's memory. Only after the simulation are these results synchronized to the CPU.

5.3 Software architecture

The top level architecture of the simulation software is shown on Figure 19.

The `CommandHandler` class is the top level class of the solution. It interprets user commands given either through the standard input, or loaded from a command file passed as the first argument of the program. It also manages the lifecycle of objects depending on the simulation criteria prescribed by the commands.

First it creates an instance of the `PlatformManager` class, which handles the communication with the OpenCL driver. It can be used to query the available platforms and devices. The first command of the user has to determine whether they want to use the primary CPU or GPU device as the simulation's target. This information is passed on to the `PlatformManager`, which sets up an OpenCL context pertaining to the selected device. This context is necessary for the creation of all device memory objects or device executables, however these actions will also be carried out by the `PlatformManager` class sparing much of the boilerplate code.

The next user command has to establish the spatial size of the computations. Based on this information a `FieldSolver` object is created containing an accordingly

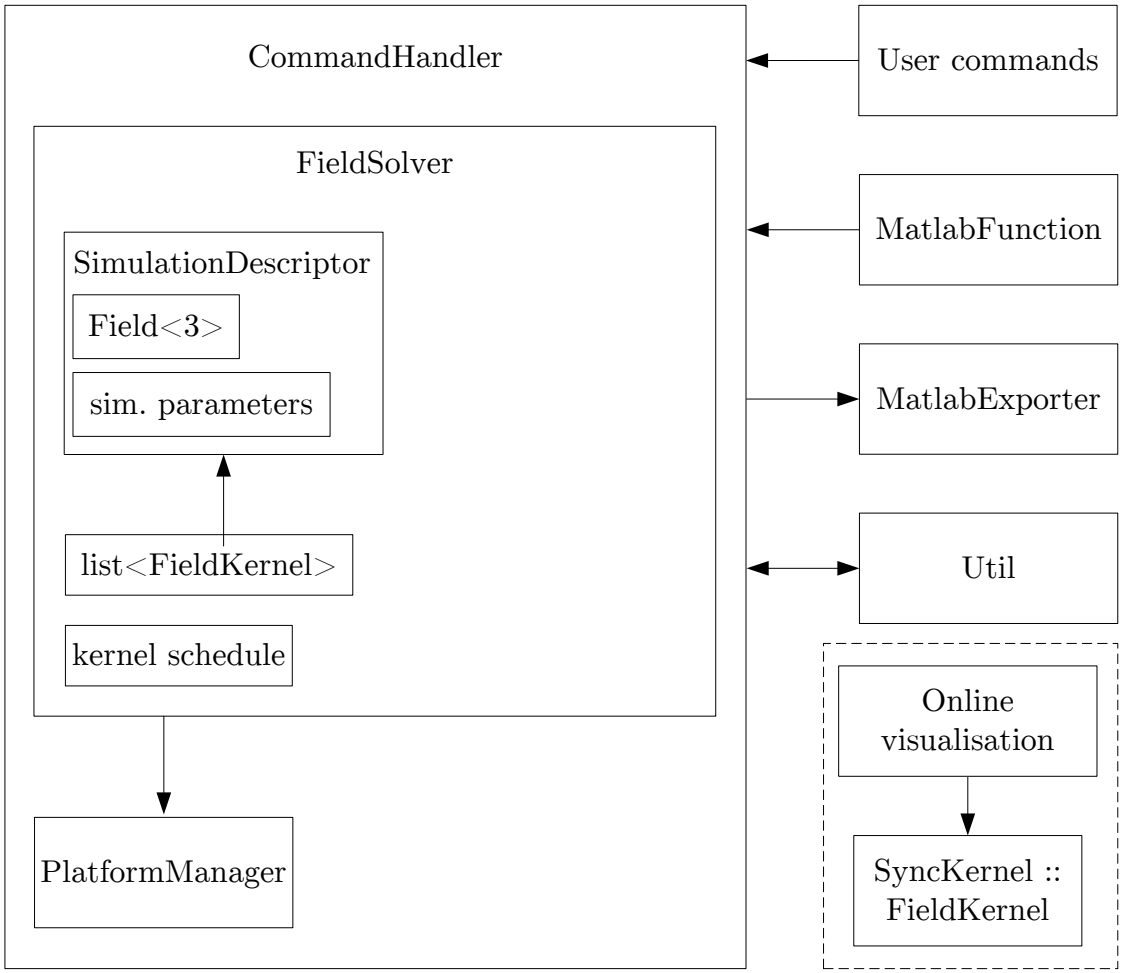

Figure 19: The top level architecture of the simulation software

sized `Field<3>` object. The template number 3 means that this object will contain three temporal stages of the field variables, which is dictated by the numerical formula for the wave equation (27). This is also an object that automatically retains a copy of its data contents in the device memory. All kernels working with the field variables will reference this object, and so the consistency of the three time steps will be ensured.

The `Field<3>` object is contained within a `SimulationDescriptor`, which bundles the various parameters of the FDTD method together checking the stability requirement, and ensuring that all kernels will interpret the numerical field values in the same physical sense.

The `FieldSolver` object keeps a list of `FieldKernels` that are distinct modules executing different operations on the field variables. Such kernels are responsible for iterating the general field equations, applying source effects or modelling reflective surfaces.

The kernel schedule is a list of references to the `FieldKernels` in the order of execution. Each kernel referenced in this list will be executed in every iteration. The list can contain multiple references to a single kernel, thus some kernels can be executed more often than others.

Using the `MatlabFunction` class the program can evaluate Matlab functions referenced as `.m` files. This facility is used for designing filters or generating source signals. Simulation results are exported into `.m` files by the `MatlabExporter` class. Members of the `Util` static class allow the software to read and write pure text data files among other things.

While the computations are run on the main thread, a separate thread is created for the online visualisation facility. It creates a window and uses the OpenGL language to create near realtime plots based on the data provided by the `SyncKernel`. The `SyncKernel` is scheduled by the FieldSolver and synchronizes field values along a selected plane after every 100 iterations.

Interactive modules such as the ANC module can access the simulation through an interpolating module layer, which in practice is just another `FieldKernel` scheduled at every iteration. The modules however are only run for every 4th or 8th iteration depending on the simulation parameters. The difference in sampling frequencies is managed by the module layer's interpolating filter.

5.4 Implementation

This section introduces the development choices that went into the realization of the FDTD techniques presented in section 4. Some of the solutions, such as the 3D modelling support, are providing the essential tools to describe detailed simulation situations. The section will mention classes and methods in order to provide insight into the mechanics of the program, but it does not give an exhaustive list of all implemented classes and functions. Almost all classes mentioned herein are template classes, that require a type template, determining the numerical representation of

the simulation. This template is substituted with the `float` type in the current implementation, but future simulations might require using the `double` type instead.

5.4.1 Field simulation

The field simulation is realized by subsequent calls to kernels acting on the field variables. This process is supervised by the `FieldSolver` class. It keeps a consistent record of simulation parameters in the form of a `SimulationDescriptor` object. It also contains a list of derived classes of the `FieldKernel` abstract class. These `FieldKernel` instances are created by the `CommandHandler` class depending on the prescribed simulation criteria.

The `FieldKernel` instances are responsible for realizing the iterative forms of the equations presented in section 4. They contain an actual OpenCL kernel that executes those calculations when the `FieldSolver` schedules them. These objects carry out the following tasks.

5.4.1.1 Solving the wave equation The `FdtdKernel` is responsible for applying the general SLF scheme to the field values as prescribed by the update equation (27). At scheduling a separate kernel is launched for every field value on the FDTD grid. Kernels located at the outermost layer of the simulation volume will not execute the update operation. Hence the grid here will retain a value of 0, and thus a Dirichlet condition will be observed.

Effects inherent in the second order wave equation are used to demonstrate the `FdtdKernel` on Figure 21 and 20.

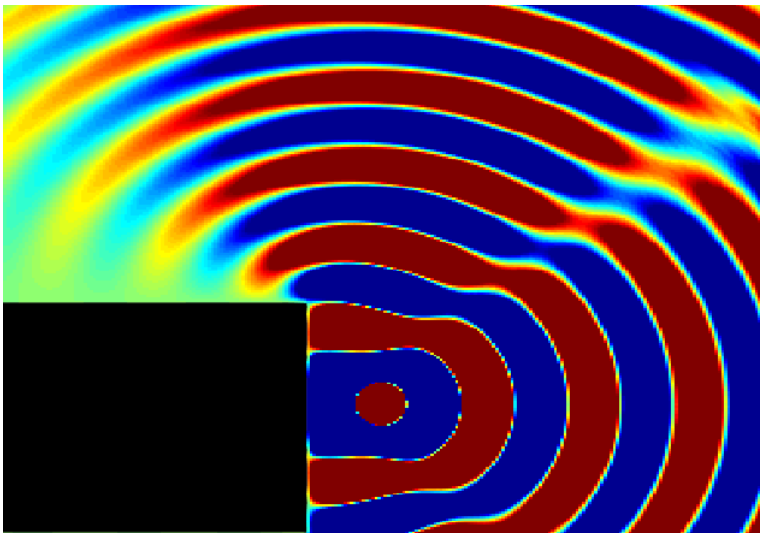

Figure 20: Wave diffraction at 500 Hz

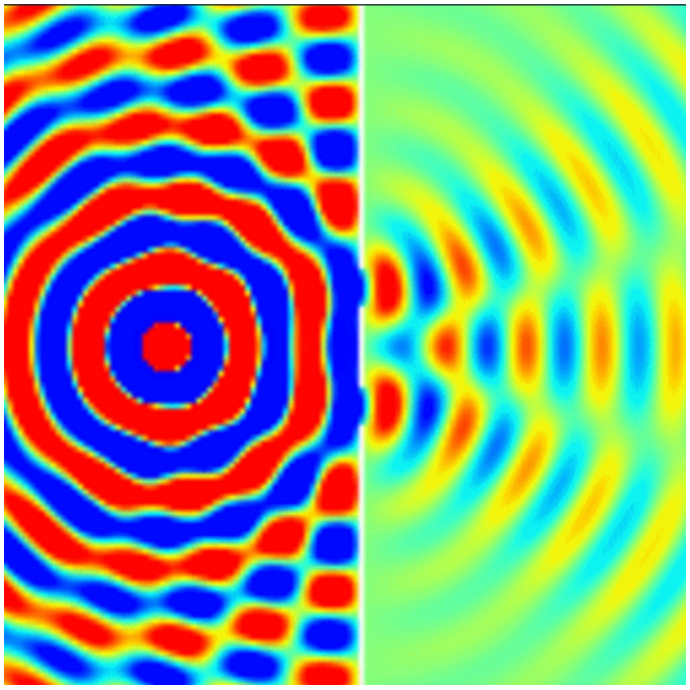

Figure 21: Simulation of the double slit experiment at 500 Hz

5.4.1.2 Calculating surface reflections Surface reflections are calculated by the `DifKernel` class. It implements the update formulae presented in section 4.4.2.

First the g^n intermediate value is calculated according to (39). Then the pressure field values are updated according to (38). Finally using the new pressure value the input and output equations of the impedance filter can be evaluated according to (40) and (41). The input and output values are then shifted into the buffer corresponding to the surface element's filter, ready to be used in the next iteration.

The calculations carried out by the class and its kernels are rather complex due to the large number of possible surface configurations. Surface elements of a planar section, an edge or a corner require one, two and three impedance filters respectively, and the values of these have to be taken into account accordingly when substituting for the ghost points. There are three separate OpenCL kernels for the separate surface types and they all have to be parametrized according to the orientation of the surface element. Counting the possible variations by going through the faces, edges and corners of a cube, we can conclude that there are 26 different configurations accounting for all surface elements. They are implemented by differently parametrized versions of one of the three kernels.

Enumerating all the surface elements in a scene, determining their type and orientation, and assigning the proper material filter is carried out by member functions of the `DifKernel` class during the import process of the 3D model describing the scene. More of this process is explained in section 5.4.3.

During the simulation one kernel is launched for every surface element in every iteration. The exact type of the kernel already determines whether it calculates reflections off of a surface, edge or corner. The kernel's arguments are the surface orientation, the previous input and output values of the surface filter, the id of the surface's material and the impedance filter coefficients ordered by material ids.

Materials are described by their impedance filter coefficients, which are stored in the `materials` directory of the software in the form of `material_1_b.dat` and `material_1_a.dat` files corresponding to the numerators and denominators of the filter's transfer function.

5.4.1.3 Applying PML conditions The PML conditions are applied by the `PmlKernel` class. The class is storing the auxiliary variable fields used for the solution of the PML equations given in section 4.5.2.

Each of the four auxiliary variable fields are separated into 6 `ClArrays` folded around the cuboid shaped simulation region. This way field values inside the region of interest – where they would remain 0 – are not stored. During the iterations one kernel is launched for each element of the outer layers.

Since the FDTD stencils depend on neighbouring values at every grid point, the values along the surfaces of these 6 field arrays have to be mutually copied into the

neighbouring sections of the absorbing layer.

The effect of a 31 layer thick PML is demonstrated by Figure 22.

5.4.2 Memory management

Memory management in OpenCL applications consists of area allocation on both the host and the device and synchronizing data between the two memory spaces. These actions are handled by the `ClArray` class.

At creation the class requires the size of the area in each of the three dimensions, and a `PlatformManager` object that allows it to access the previously selected compute device. The class then allocates an appropriate chunk of memory on both the host and the device. A `cl::Buffer` reference is retained inside the class that allows access for the device side data.

The class's setter and getter functions provide direct access to the host side memory area. The class keeps track whether new values have been set since the last synchronization with the device.

Using the `d_data` getter function the `cl::Buffer` reference can be requested from the class. Before returning the reference the class will synchronize data between the host and device if the relevant state indicates its necessity.

The `SyncDevice` and `SyncHost` member functions can be used for explicit synchronization. A single host synchronization will be initiated after each subsequent call of the `d_data` and `h_data` functions, but an explicit call of the `SyncHost` function is always used for safety reasons before accessing the host side data after a compute session.

5.4.3 Describing acoustic environments

Acoustic environments can be described by creating 3D models parametrized by material types. Surface elements of the DIF technique are identified by analysing the 3D model, and materials are described by previously designed digital impedance filters.

5.4.3.1 Creating and loading models The 3D models of the environments are created by the Wings3D polygon modeller program. [19] It provides an easy to

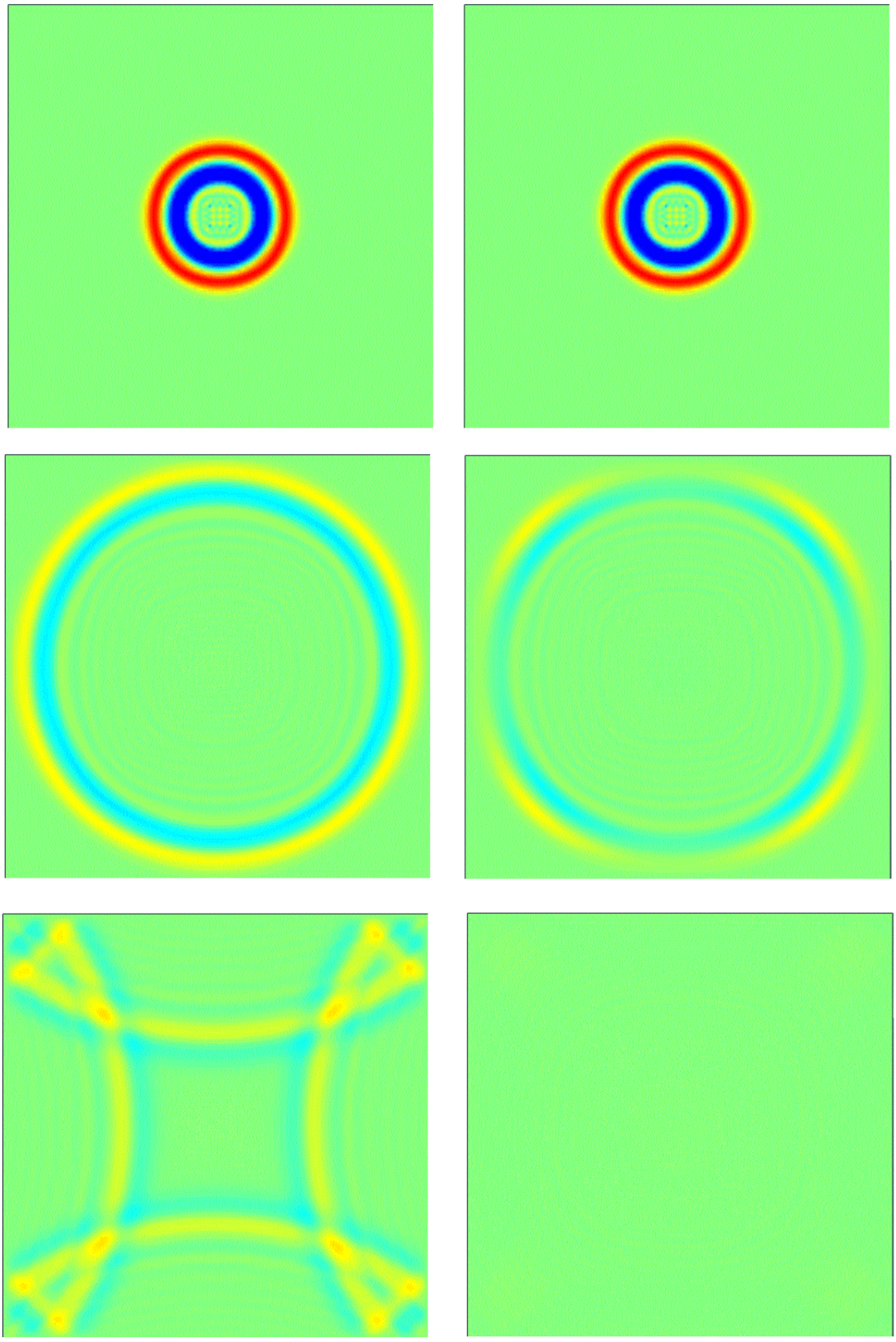

Figure 22: The effect of the perfectly matched layer (on the right)

use graphical interface for creating and editing 3D geometry. Figure 23 shows the user interface of the program.

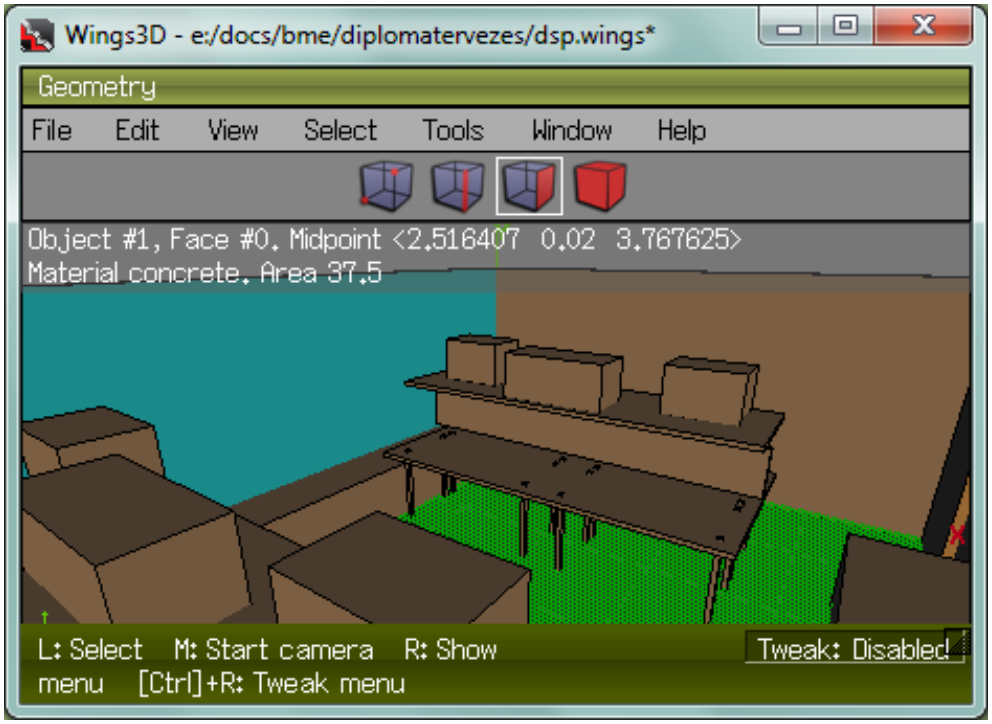

Figure 23: Acoustic environment in the Wings3D polygon modeller tool

Every surface in the model can have an assigned material property. Materials have to be named so that there are matching files in the `materials` directory containing the impedance filter coefficients. E.g. if a surface was named *concrete* the simulation software would be looking for files named `concrete_a.dat` and `concrete_b.dat` when analysing the scene.

The models can only contain cuboids aligned with the axial directions, i.e. each face of a cuboid has to be parallel with one of the coordinate planes. Although the simulation software is generally capable of handling arbitrarily oriented objects, the function interpreting the 3D models will only identify such shapes correctly. Meanwhile FDTD techniques generally require surfaces to be aligned with the computational grid, unless the simulated waves have significantly larger wavelengths than the grid resolution.

The 3D model has to be exported to a Kerkythea file and saved in the `scenes` directory. The scene files are loaded by the `DifKernel` class's `LoadGeometry` function. Interpreting the scene information is done in three phases.

First the cuboids contained within the file are identified. Cuboids outside the simulation boundaries are ignored and a warning is generated by the function. The material indices are also determined based on the material name corresponding to the cuboid.

Then a volumetric map is created. All grid points contained within the volume of a cuboid are identified, and the material index of the cuboid is stored in them. Grid points not contained within any body receive an index of -1.

Finally every grid point in the volumetric map is analysed and a surface map is generated. If one or more neighbours of the investigated point is within the volume of a body, the point is identified as a surface point. The program creates impedance filter variables for all such points, and determines their orientation.

The resulting surface data can be directly utilized by the kernels calculating the field values.

5.4.3.2 Designing digital impedance filters The digital impedance filters describing material properties are designed in Matlab and are stored in the simulation's `materials` directory.

The filters are designed using real-life measurement data regarding the absorption coefficient of materials. The measured values used in the presented simulations are given by Table 1

Material	absorption [%]				
	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz
concrete	4	4	5	6	6
glass	8	4	3	3	2
plywood	14	10	6	8	10
double plasterboard	14	10	6	8	10
door	14	10	8	8	8

Table 1: Absorption coefficients of various materials [12]

The absorption values given for octave bands were converted to reflectivity coefficients using the relation

$$R = \sqrt{1 - \alpha}. \tag{65}$$

Our investigation concerns the range of low frequencies in which these materials exhibit simple absorption characteristics. The reflectance values were approximated with a first order reflectance filter $R(z)$ using the `invfreqz` function and scaled such that their transmission would be accurate at the frequency of 500 Hz, since this is the center frequency of the 1 kHz noise used during ANC experiments and simulations. The reflexivity filter designed for plywood can be seen on Figure 24.

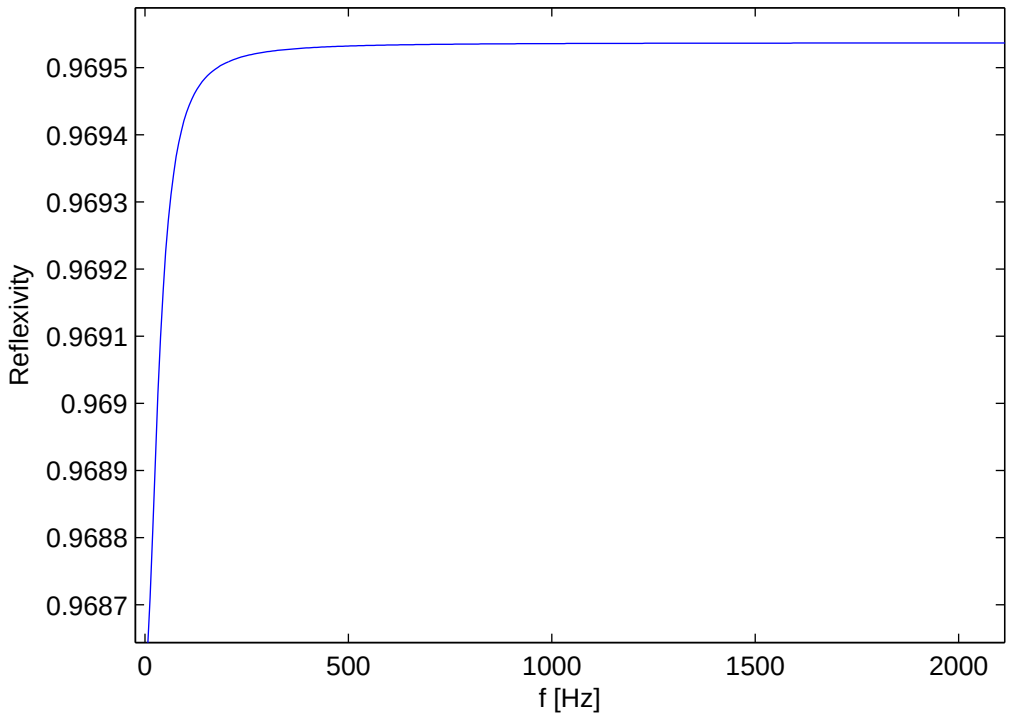

Figure 24: The reflexivity filter designed for the plywood material

Then the reflexivity filter was converted into the impedance filter using (36), and the stability of the resulting filter was checked by observing the pole-zero map. Finally the numerator's and denominator's coefficients were exported into the corresponding `.dat` files in the `materials` directory.

The filter's phase properties were ignored, since previous studies indicate that the phase change caused by reflecting bodies does not significantly affect the nature of acoustic fields in room acoustics [9].

5.4.4 Handling interactive modules

One of the most important design decisions was that interactive modules should interact with the simulation as if they were functioning in real-life environments. Thus the implementation details of the field simulation should be hidden from them.

The `InoutKernel`, sometimes referred to as interpolating module layer, provides input and output variables for modules corresponding to in-field speakers and microphones. Using the `inout`, `speaker` and `mic` commands the user can specify the module source code, and the speaker and microphone positions with which the module will interact. E.g. in the case of the noise cancelling module the following commands can be used.

```
inout elms.cl
speaker 197 50 178
mic 187 50 178
floatbuffer 8200    1   1
floatbuffer 8000    1   1
floatbuffer 1       1   1
floatbuffer 8000    1   1
floatbuffer 8000    1   1
intbuffer   2       1   1
```

These commands will compile an instance of the `elms.cl` kernel which is the interactive module implementation of the ANC system. Observing the function header of the module helps with understanding the module layer's mechanism.

```
__kernel void process(
    __global float* inputToUse,
    int num_inputs,
    __global float* outputChannel,
    int num_outputs,
    __global int*   buf_size,
    __global float* x_ref,
    __global float* x_err,
    __global float* reference_average,
    __global float* coefS,
```

```
__global float* coefW,
__global int* gvi
)
```

The first five arguments are compulsory for all module functions. The `inputToUse` array has `num_inputs` elements corresponding to the microphones defined for the module layer. In this case `inputToUse` wil have one element and will always represent the current pressure value at the microphone location $(187, 50, 178)$. Similarly the `outputChannel` array will have elements – one in this case – corresponding to the speakers specified with the `speaker` command.

The next six arguments in this case are required by the internals of the module code. They are also provided by the module layer in a persistent manner, such that they will retain their values across iterations. To access such additional buffers the `floatbuffer` and `intbuffer` commands can be used.

The physical simulation's sampling frequency can and in practical cases will differ from the module's internal sampling frequency, since limiting isotropy error requires simulation frequencies much higher than the maximum allowed signal bandwidth. The interpolation between the differing frequencies is handled by the module layer, which is transparent to the modules. In the current implementation the module layer can handle whole number ratios between the simulation and module frequencies.

In the initialization phase of the simulation software the module layer will design a minimum order FIR filter using Matlab. In order to preserve the simulation's accuracy the filter's cutoff frequency is always 2 kHz regardless of the interpolation ratio.

During operation either the module will be scheduled and its outputs will be inserted into the filter buffer, or the module layer will insert a 0 instead. The filtering is carried out by an efficient, parallel GPU implementation.

5.4.5 Matlab integration

5.4.5.1 Calling Matlab functions The software requires the simulation dependent calculation of numerical parameters – such as interpolating filter coefficients or signal values – for which Matlab offers easy to use, efficient solutions.

The `MatlabFunction` class provides an easy to use facility to initiate the evalu-

ation of Matlab functions from C++ code and receive the results in C++ variables. The usage of the class is demonstrated by the following snippet.

```
MatlabFunction<float> mfunc("interpolating_filter.m");
    mfunc.SetArg(0, 2000.0f);
    mfunc.SetArg(1, 3000.0f);
    mfunc.SetArg(2, 32000.0f);
    std::vector<float> result;
    mfunc.SetRes(0, &result);
    mfunc.Eval();
```

The function header of the executed Matlab function is as follows.

```
function b = interpolating_filter(Fpass, Fstop, Fs)
```

According to the function name, and the arguments set by the `SetArg` method the class creates a Matlab script, that contains the parametrized function call of `interpolating_filter` and another Matlab function call that will export the results into a data file. The Matlab script is executed through a call to the Matlab executable. Then the resulting data file is parsed, and the results are returned in the vector specified by the `SetRes` method.

5.4.5.2 Exporting data Simulation data is exported to Matlab using the `MatlabExporter` class. The following snippet demonstrates the usage of the class.

```
MatlabExporter<float> exporter;
exporter.AddArray(*(inout_kernel_->out_positions_), "spk_positions");
exporter.AddArray(*(inout_kernel_->in_positions_), "mic_positions");
exporter.AddArray(*(inout_kernel_->buffers_[0]), "speaker_signal");
exporter.AddArray(*(inout_kernel_->buffers_[1]), "mic_signal");
exporter.Write("simulation_results.m");
```

Using the `AddArray` function we can add `ClArray` object used in the simulation to the list of variables to export. The second argument of the function is the variable name through which the data will be accessible in Matlab.

The `Write` function writes all variables into the specified Matlab script file.

5.4.6 Online visualisation

An online visualisation is provided by the `Plotter` and `SyncKernel` classes. The `Plotter` class creates an OpenGL window on a separate thread which is updated after every 40 ms. The `SyncKernel` is scheduled by the `FieldSolver` class after every 100 iterations and it synchronizes a plane of the simulation volume to the host. The OpenGL functions access this host variable to create a map of the current pressure conditions. The visualisation's window is shown on Figure 25. The red lines mark the boundary of the PML region.

Figure 25: Online visualisation window

5.4.7 Interpreting user commands

In addition to the 3D models and material filter coefficients, the simulations can be described by textual commands. These can be executed one-by-one through the program's commandline, or they can be saved in a text file, and passed as a first argument to the program at launch.

The first command after launch always have to be the `device` command telling the program whether to execute the calculations on the first available cpu or gpu device.

A complete list of commands is given below. Parameters that have a fixed set of valid values are enumerated between [brackets], separated by vertical lines. Optional arguments are given in <chevrons>.

```
device          [cpu|gpu]
field           size_x      size_y      size_z
visualisation   [x|y|z]     level
scene           scene_file
pml             length
inout           kernel_file
speaker         pos_x       pos_y       pos_z
mic             pos_x       pos_y       pos_z
floatbuffer     size_x      size_y      size_z      <data_file>
floatbuffer     size_x      size_y      size_z
run             iterations
exportinout     file_name
source          pos_x       pos_y       pos_z   <data_file>|<m_func>
quit
```

5.5 Runtime characteristics

The runtime comparisons are based on running the second simulation series presented in section 6.2.2. The simulation was run on an Intel i7 3770 CPU and an AMD R9 290 video card.

Device	iteration time [ms]	real-time factor	specific computational cost second/m^3/real-life second
Intel i7 3770	51.5	1640	10.79
AMD R9 290	3.3	105	0.69

Table 2: Runtime performance for the simulation of 7.5×5 m room

Due to the parallel nature of the problem it is worth using a GPU, which proved

to be 15 times faster than the CPU. The result is not surprising given that the current implementation is memory bandwidth bound and the video card's theoretical memory bandwidth is exactly 15 times higher.

With these iteration times the simulation of 1 real-life second took 105 seconds, which is expressed by the real-time factor.

Running the first simulation series, which has a problem size 1.76 times higher, the iteration time increased tenfold to 33 ms in case of the GPU. This suggest some sort of memory or thread management bottleneck yet to be explored. Since the simulation times were still tolerable at this level, this phenomenon hasn't been further investigated. Due to the current architecture of the software and the dual memory management on host and device side, the larger simulation would not run on the CPU. This limitation may be possible to overcome by more complicated memory management schemes separating the management of host and device side copies.

6 Simulation and verification

6.1 Simulating ANC systems in reflective environments

This section describes the methodology for creating ANC simulations approximating real-life conditions. Simulations presented in the next section are emulating typical conditions that can put ANC systems to the test by presenting acoustic paths characterized by a large number of reflections. Under these conditions ANC systems with a single reference microphone provide poor performance. The experimentally tested method for increasing noise cancelling efficiency in these conditions is surrounding the suppression region with a large number of reference microphones.

The simulation environments have been modelled after real-life locations where the experiments took place. Material properties were chosen to resemble the objects in that particular environment. However all simulations are inherently simplifications and the typical behaviour of ANC systems is expected to present itself regardless of the slight differences between the real and simulated conditions.

The simulations were carried out with a spatial resolution of 0.02 m and a sampling frequency of 32 kHz. In order to limit the isotropy error an upper cutoff frequency of 2 kHz was observed. The simulations were run in three phases.

1. First an 8 second field simulation was carried out for each source, i.e. the actuator and noise source. During the simulation the selected source would emit white noise with a bandwidth of 2 kHz. The noise signal and the signal perceived by the microphones were exported by the simulation software. The inputs and responses were sent and received through the interpolating module layer working with an interpolation ratio of 4, thus the sampling frequency of the imported and exported signals was 8 kHz.

2. The signals then were loaded into Matlab and the impulse responses between the points of interest were identified using the LMS method. After the identification the remainder error of the LMS algorithm were checked for sufficient accuracy. Initially the transmission paths were represented by 16000 step long impulses, however inspecting them I found, that 8000 is sufficient for an accurate representation even in the case of the large room.

Using the information contained within the impulse responses about the acoustic environment a model of the ANC system is identified. The model is implemented as a Matlab class and its elements realize the signal processing task described by the ELMS noise cancelling and identification algorithms. The class generates a random noise sequence and filters it with the impulse responses, thus creates the x, d and e signals presented on Figure 3. Then iteratively applying the update formulae (6) and (5) it synthesizes the model components, i.e. the $\hat{A}_2^{-1}$ and W secondary and primary filter coefficients. The adaptation phase of the system is thus simulated outside the field model using only the information relevant to the signal processing task.

Simulating the adaptation phase based only on the impulse responses proved to be a good approximation – verified by the third phase – and was running 10 times faster than the field simulation, allowing me to experiment with the ANC system parameters and running 1 minute long adaptation phases for reaching maximum suppression.

The ANC system – both in Matlab and the field simulation – used primary and secondary filters with an order of 8000 and an inverse delay value of 200 for the secondary filter. Normalization of the reference signals was enabled and the μ step constant was chosen to provide maximum suppression by the end of the 1 minute adaptation phase.

3. During the third phase the coefficients of the ANC system's filters' were exported from Matlab and loaded into the field simulation. Since the adaptation phase was already simulated with a Matlab model, the filters should already contain the optimal coefficients in order to provide the maximum level of suppression. Thus the transient states of the ANC system are omitted from the field simulated phase.

   The field simulation was then run for another 2 seconds. During the first second no measurements were taken, it was executed only to ensure that the acoustic power distribution in the room would reach a static state. During the last second the integrating module would sum the square of the field values in the plane of the ANC devices. The integrated field values were then exported

to Matlab, where the suppression was verified and the images were created.

The acoustic environments, the ANC system's configuration and the placement of its devices were chosen so that they would resemble conditions observed during a set of cornerstone measurements carried out over the past two years. A detailed description of these measurements and a systematic collection of experimental data can be found in [16], [14] and [15]. By verifying a set of characteristic experiments through simulation results a stronger basis for previously assumed relations can be provided.

6.2 Verifying experimental results using field simulation

The presented simulation descriptions are close approximations of real-life experiments documented in [16] and [15].

6.2.1 Large, lightly furnished room

One of the first and more general experiments that investigated the usage of multiple reference microphones took place in a large, lightly furnished room – the university's IE224 room.

It has an 8×11 m floor, walls on three sides modelled with double layered plasterboard, a glass window from wall-to-wall on one side, and a floor and a ceiling of slab (concrete). Along one side there are metal lockers next to the wall and a smaller one in the corner of the opposite side. Under the window there is a rectangular cover for the heating modelled by plywood. Opposite the window there is a door and moved to the sides of the room there are a few plywood desks. The 3D model of the room is shown by Figure 26.

The positions of the ANC system's actuator, error and reference microphones as well as the single noise source are located on the figure. The arrangement of these devices is also shown in a clear manner by Figure 27. The noise cancelling arrangement is placed on the far-away end of the room from the window in the middle along the short axis. Most reference microphones were placed along a circle with a 1 m radius around the error microphone. The devices were placed at a height of 1 m.

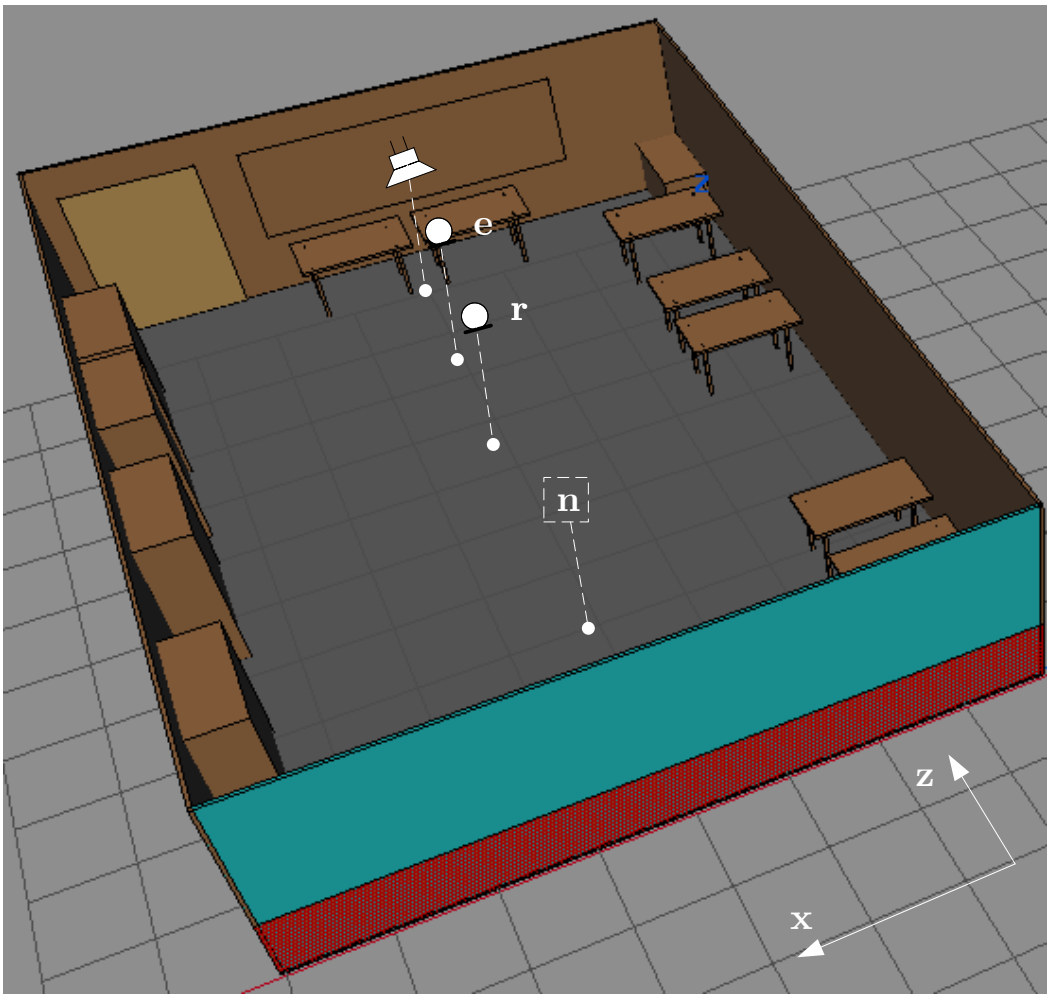

Figure 26: 3D model of the large room

The various arrangement of reference microphones is illustrated by Figure 28. The top microphone was placed 1 m above the error microphone, while the bottom one was placed 0.7 m below it. The tetrahedron arrangement – shown by the green circles – was situated so that the error microphone would be at the center of mass 1 m away from the forward corner.

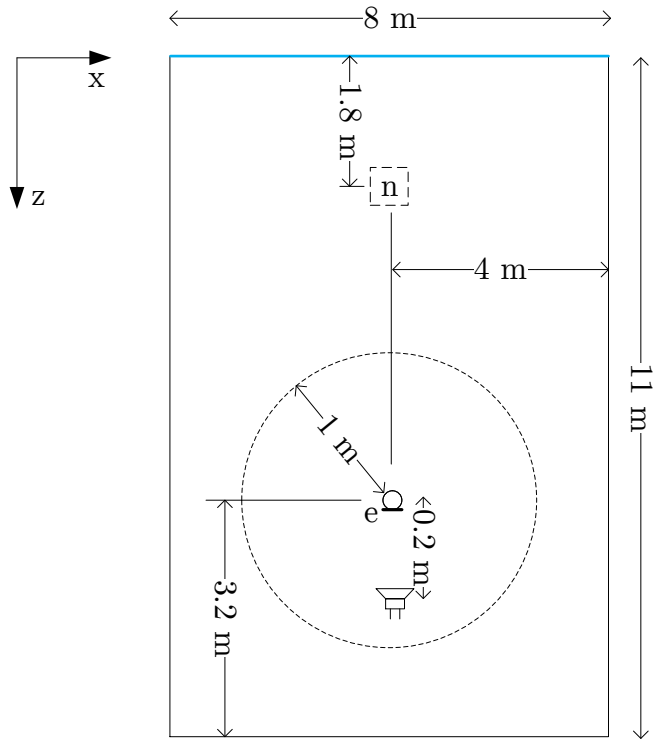

Figure 27: Spatial arrangement for the simulation – large room

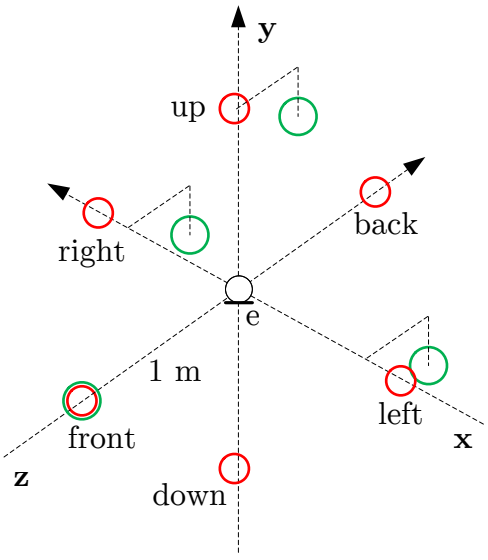

Figure 28: Arrangement of the reference microphones – large room

The nature of the acoustic paths in the room can be evaluated by the qualitative analysis of the identified impulse responses. Figure 29 shows the impulse response between the noise source and the front reference microphone placed 7 m apart. At this distance reflected wavefronts will have a similar amplitude to the waves arriving through the direct path.

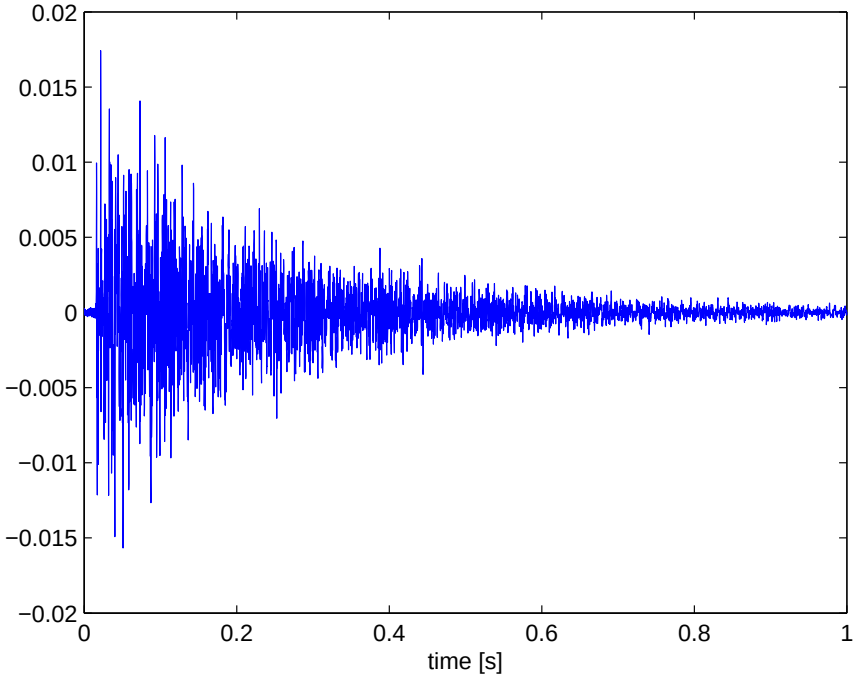

Figure 29: Impulse response between distant points – large room (11 x 8 m)

The impulse vanishes after 1 second and an exponential envelope can be clearly observed that is characteristic of wave amplitudes undergoing consecutive reflections. Figure 30 shows the initial section of the impulse. Observing the impulse front no significant peak is standing out that would set the direct path clearly apart from reflections. Wavefronts from multiple paths are arriving in quick succession and their amplitude is comparable. It shows that the original signal is not separated from the reflections, which represents large difficulty in the noise cancelling problem.

Figure 40 shows the impulse between the actuator and the error microphone placed only 0.2 m apart. A significant peak can be observed at the beginning of the impulse that corresponds to the wave arriving through the direct path.

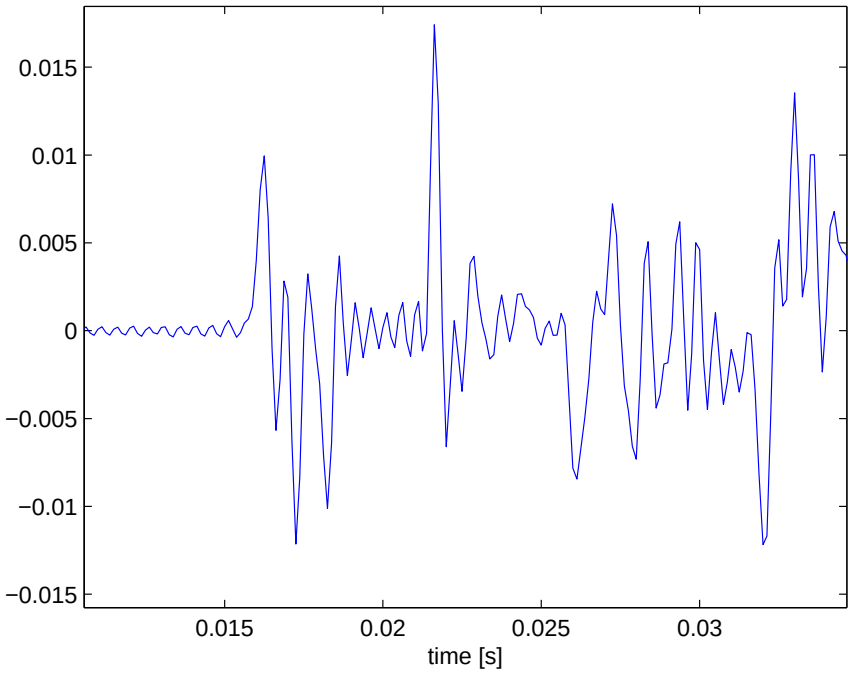

Figure 30: Impulse response between distant points – early phase, large room

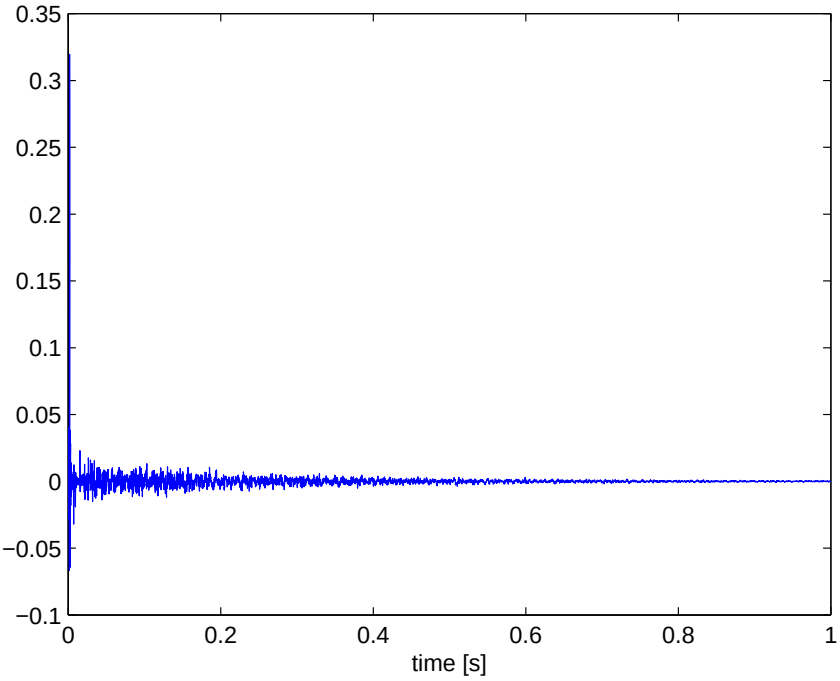

Figure 31: Impulse response between near points – large room

Figure 41 shows the early section of the impulse where the typical sections of an impulse response in room acoustics can be identified. After the peak correspond-

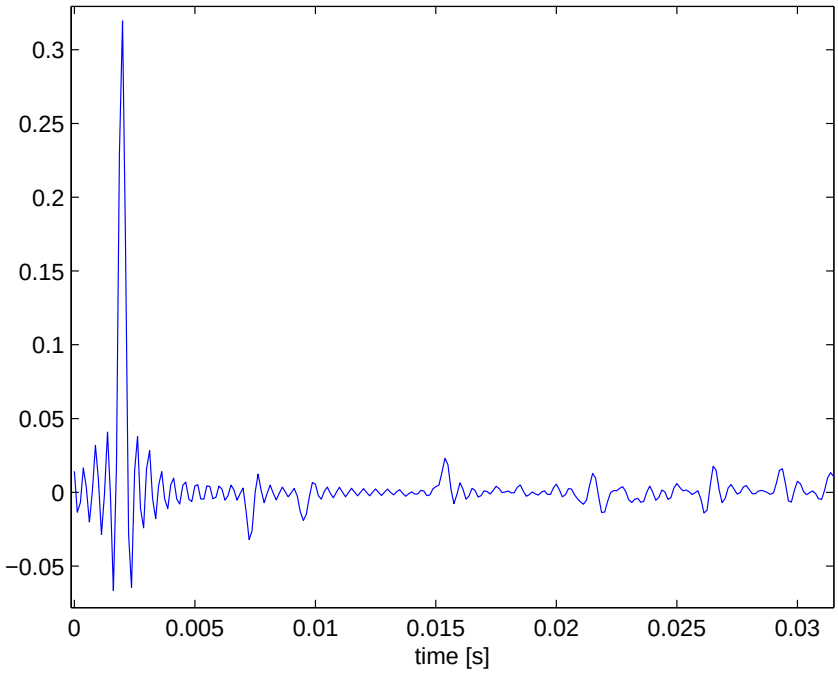

Figure 32: Room 8 x 11 - actuator to error microphone early reflections

ing to the direct path a number of smaller distinct peaks can be observed. These belong to the section of the impulse termed *early reflections*. These are caused by the closest reflecting surfaces in the observation point's vicinity. After about 0.03 seconds the peaks cannot be distinguished anymore, and the *diffuse section* of the impulse response begins. This section's transfer function is very complex and it still represents a non-negligible amount of wave power reaching the area.

Figure 33 shows an acoustic power distribution map of the room evaluated for 1 second after the first second of initial transients. A colourbar provides numerical values proportional to the square sum of the acoustic pressure and hence average power. These values' main purpose is to allow for the comparison with the next simulation's values. Observing the power distribution map the power appears to be evenly distributed in the volume of the room, no particular hotspots can be identified.

The suppression values for the various reference microphone arrangements are given by Table 3. Next to the simulated results measurement data of a comparable real-life experiment is given. Simulation results are given in unweighted (dBC) values, while the measured suppression is given in A-weighted (dBA) values. For

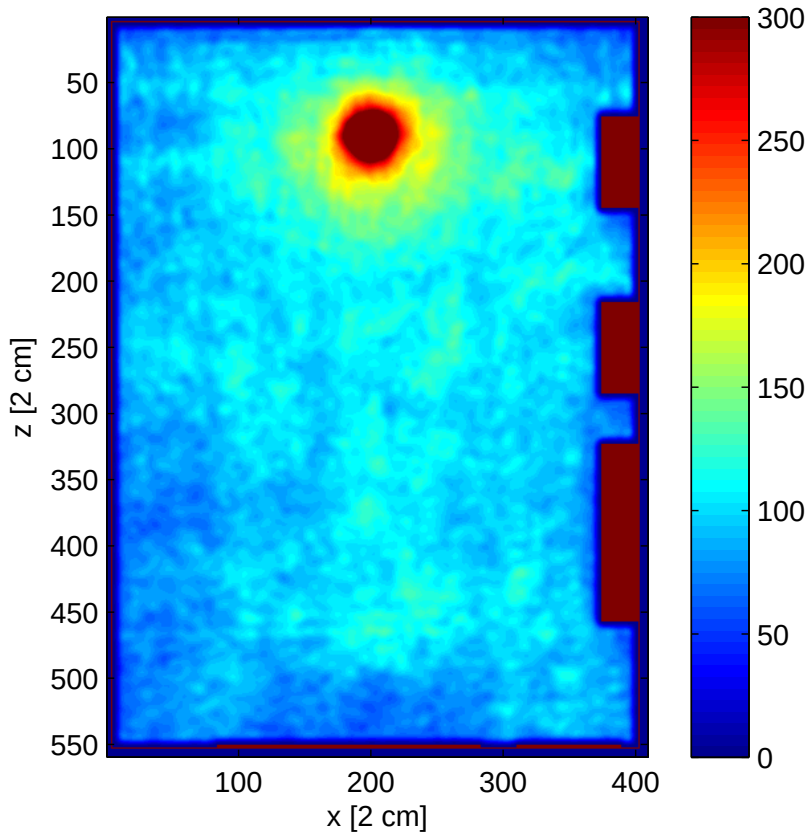

Figure 33: Acoustic power distribution in large room

this reason the absolute level of suppression values is not strictly comparable, but it does not affect the tendency regarding the number of reference microphones.

Arrangement	suppression	
	Simulation [dBC]	Experiment [dBA]
front	3.2	3
front, back	5.1	8
front, back, right, left	8.5	14
tetrahedron	8.5	15
front, back, right, left, top, bottom	10.3	16

Table 3: suppression of 1 kHz stochastic noise in a large room

The simulated results exhibit the same behaviour of ANC systems that has been observed during the experiments. With a single forward looking reference microphone only about half of the noise power can be eliminated. This is consistent with the diffuse nature of the impulse response observed on Figure 29.

Adding additional reference microphones in opposite facing directions increases noise suppression by closely the same amount. suppression values measured in the field simulation were 2.8 dB for one reference, and 9.5 dB for six. The lower values may be a consequence of the finite impulse length approximation in the Matlab simulation and could have gone slightly higher as the field simulated system was affected by the infinite responses. Unfortunately the settling rate becomes very slow as the system approaches maximum suppression and the field simulation would have to be run for longer periods to make a difference. During the real-life experiments the terminal phase of the settling proved to be very slow indeed, and occasionally 1.5 – 2 minutes elapsed before the results were registered.

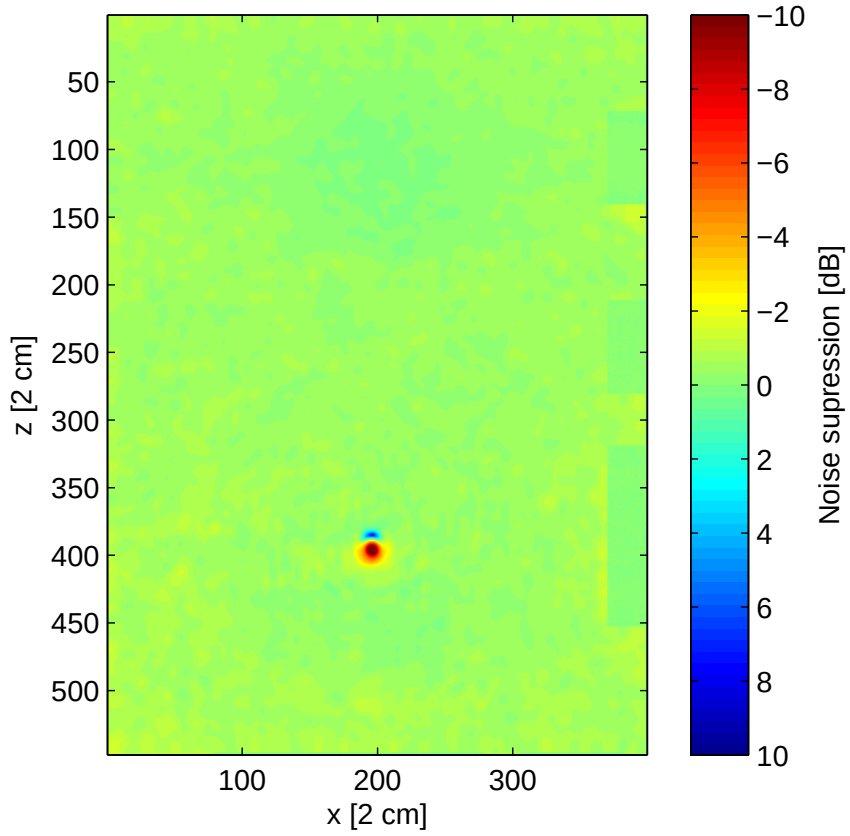

Figure 34: Noise suppression in large room with 6 reference microphones

Figure 34 shows the suppression values along the ANC system's plane. The effects of the actuator are limited to the close vicinity of the arrangement as expected.

At a distance of 0.2 m the actuator can achieve suppression with a significantly lower power than that of the noise source, thus the majority of the room remains unaffected.

Figure 35 shows the close vicinity of the actuator.

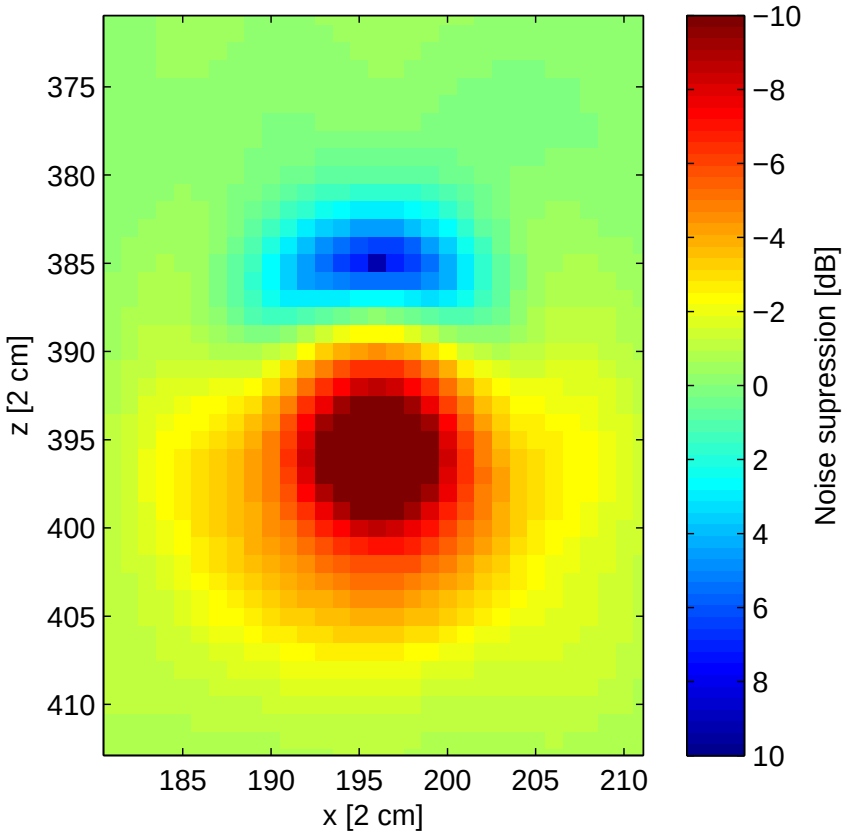

Figure 35: Closeup of the suppression region

The suppression region extends to a volume with a typical size of 10 cm and is limited to the region around the error microphone. This is also expected since the matching amplitude and phase of the anti-noise can only be guaranteed at the error microphone, moreover at different locations the various reflections will be superposed in different phase.

6.2.2 Medium sized, moderately furnished room

The second series of simulations has been set up in a 7.5×5 m room furnished with desks, metal lockers and a book shelf. One side of the room is covered by a window from end-to-end.

The noise cancelling arrangement was placed near the wall opposite the window. The 3D model and the floorplan for the simulation can be seen on Figure 36 and Figure 37. The top microphone is placed 1 m above the error microphone.

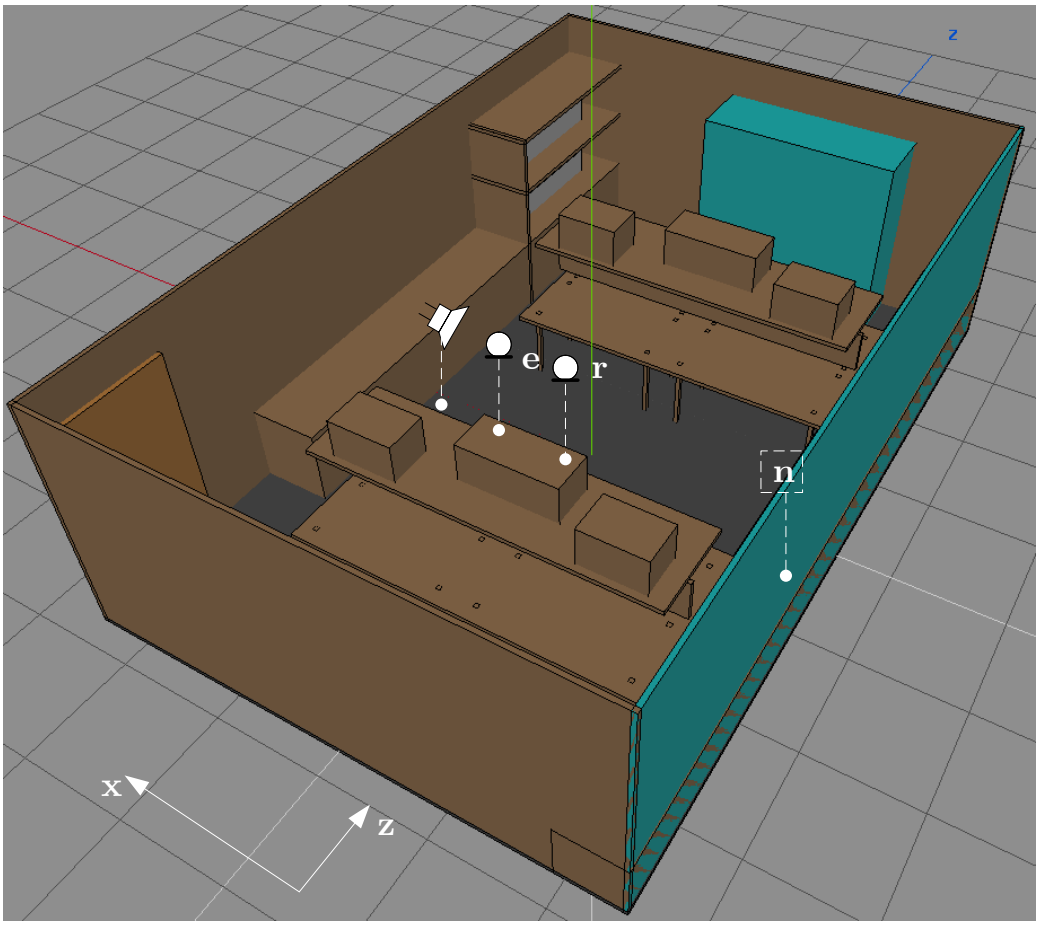

Figure 36: 3D model of the medium sized room

Since the room has a smaller volume and is more densely furnished the noise cancelling problem can be expected to be quite different than the previous case. The surface to volume ratio is much larger, and the reflecting objects are closer to the sources and microphones. The close proximity of the wall behind the actuator prevents the placement of a „back“ reference microphone giving an asymmetrical sample of the acoustic field. The noises reflected from this wall are still reaching the

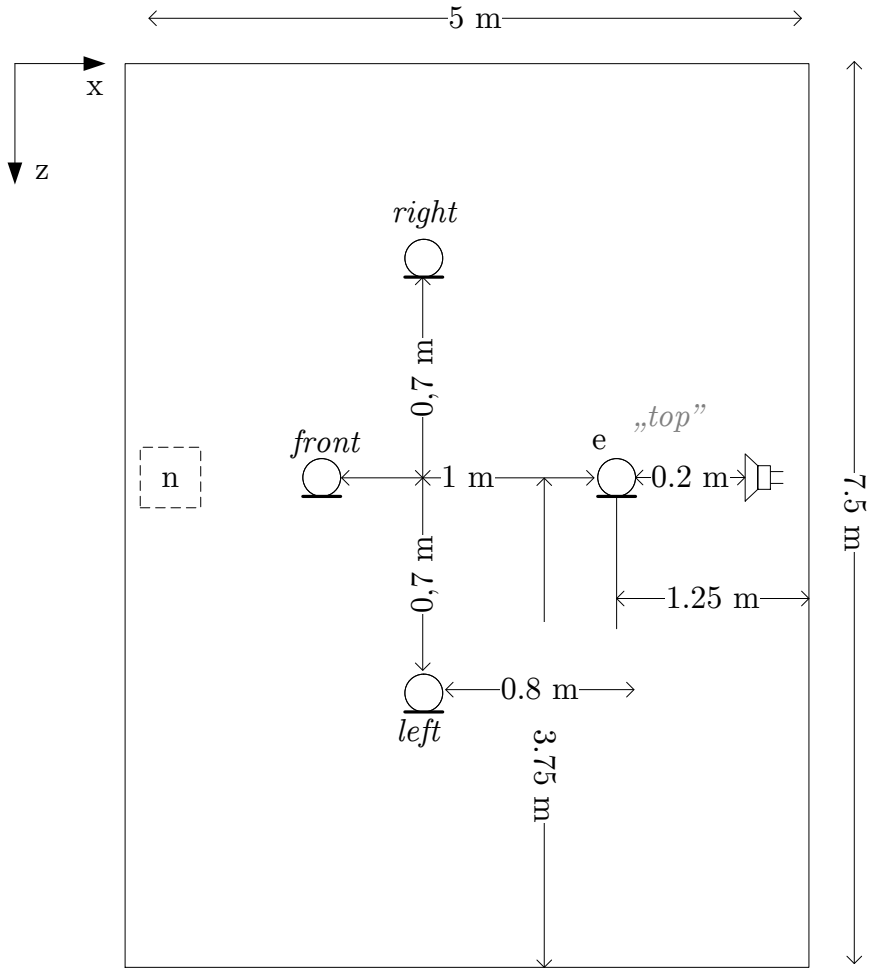

Figure 37: Spatial arrangement for the simulation – medium sized room

ANC system and with smaller attenuation than in the previous case. All in all this situation appears to be less favourable for noise cancelling.

Figure 38 shows the impulse response between the noise source and the front reference microphone. Its quality is similar to the corresponding impulse response of the previous simulation, but it vanishes faster, by 0.5 s only a small portion of the impulse is still perceivable. Its early phase seen on Figure 39 is also very similar, the reflected wave components may be even more accented though.

The impulse response between the actuator and error microphone is also very similar to the previous simulation, but the early reflections aren't so easily distinguished from the diffuse phase. This is consistent with the initial observations about the room's geometry.

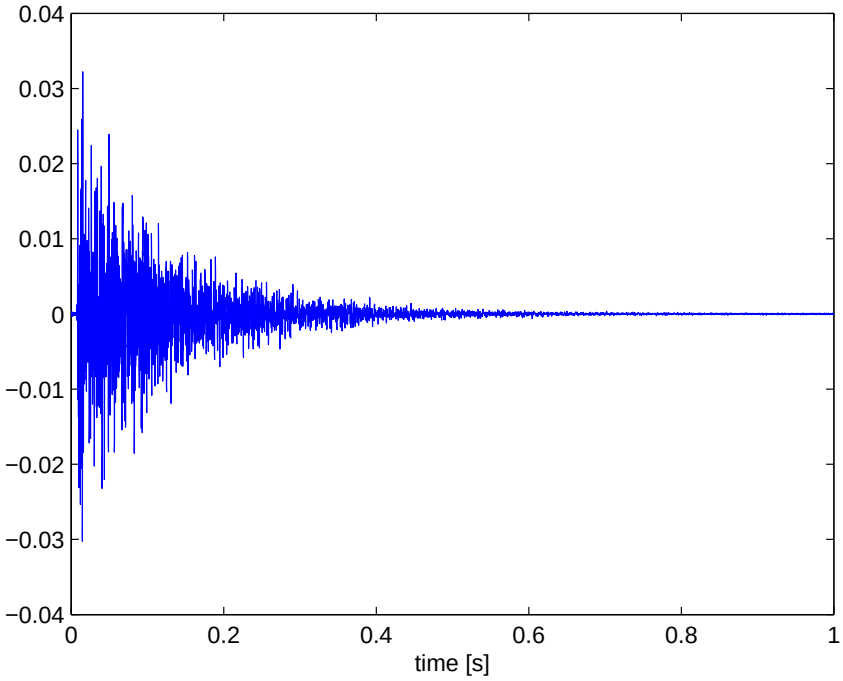

Figure 38: Impulse response between distant points – medium sized room

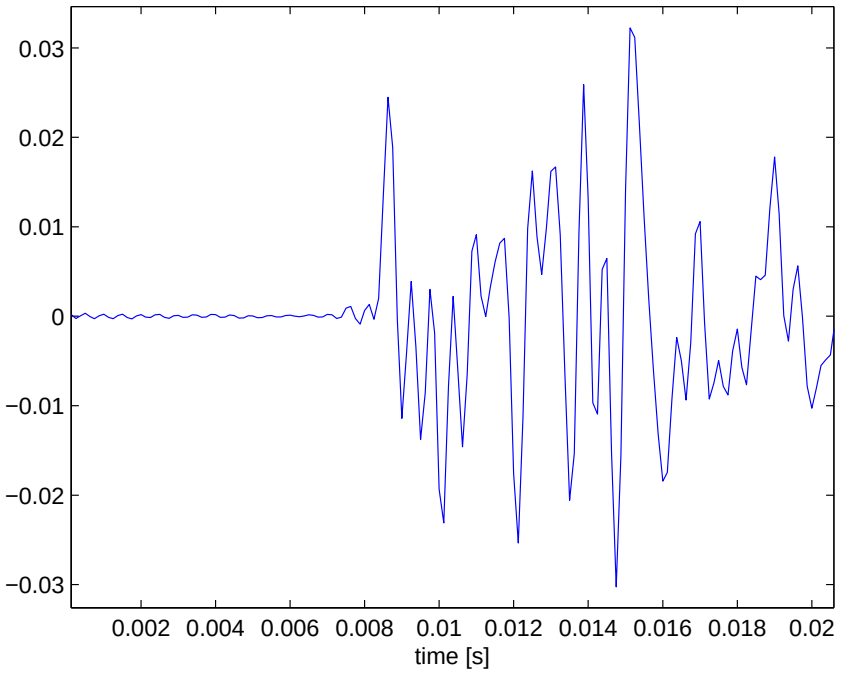

Figure 39: Impulse response between distant points – early phase, medium sized room

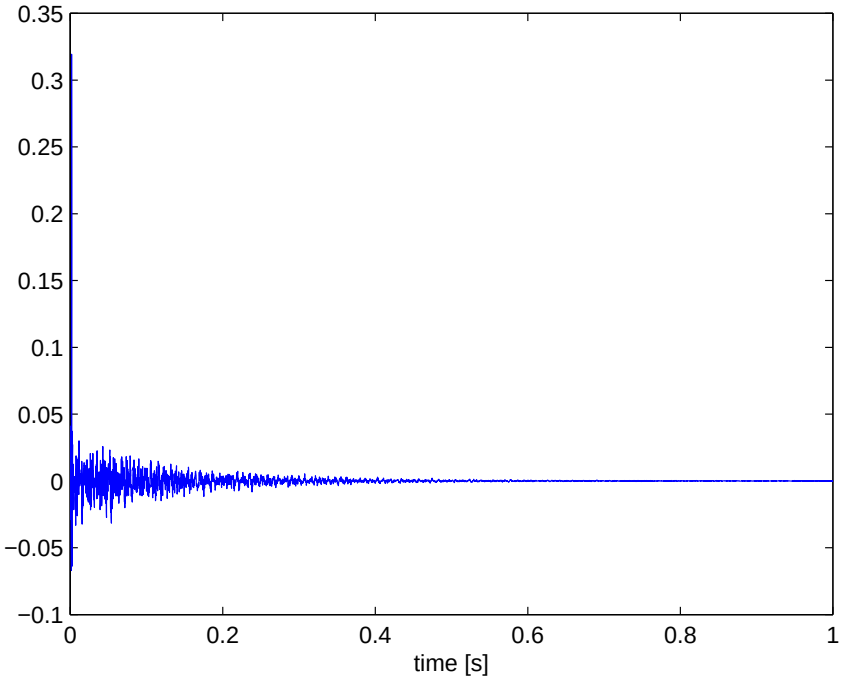

Figure 40: Impulse response between near points – medium sized room

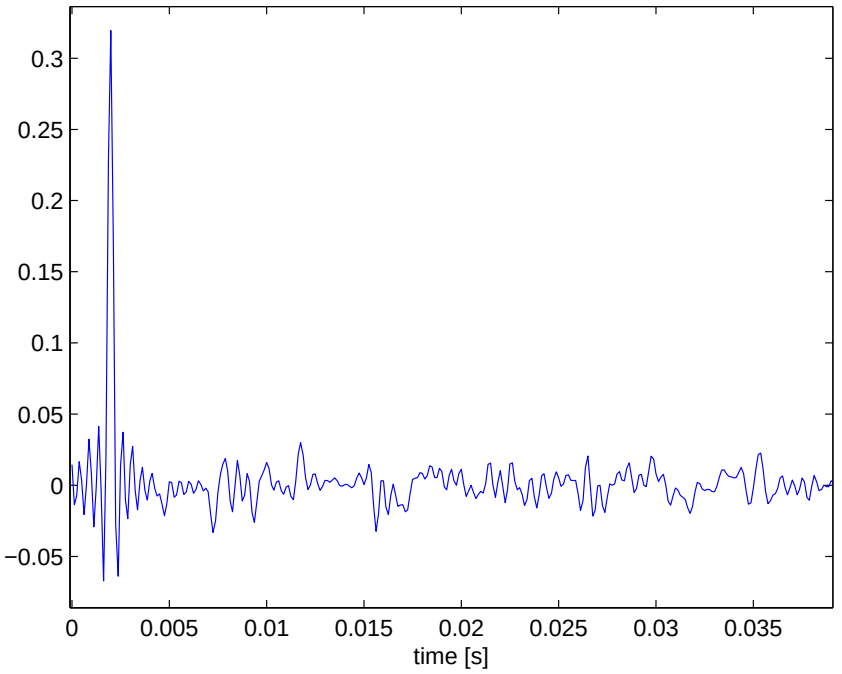

Figure 41: Impulse response between near points – early phase, medium sized room

The acoustic power distribution of the noise in the room is shown by Figure 42. The distribution is strongly inhomogeneous which will play an important role in

forming the field conditions during the noise cancelling phase. Unlike in the previous case there is a very significant hotspot along the line dividing the longer side of the room, which extends to the positions of the reference and error microphones.

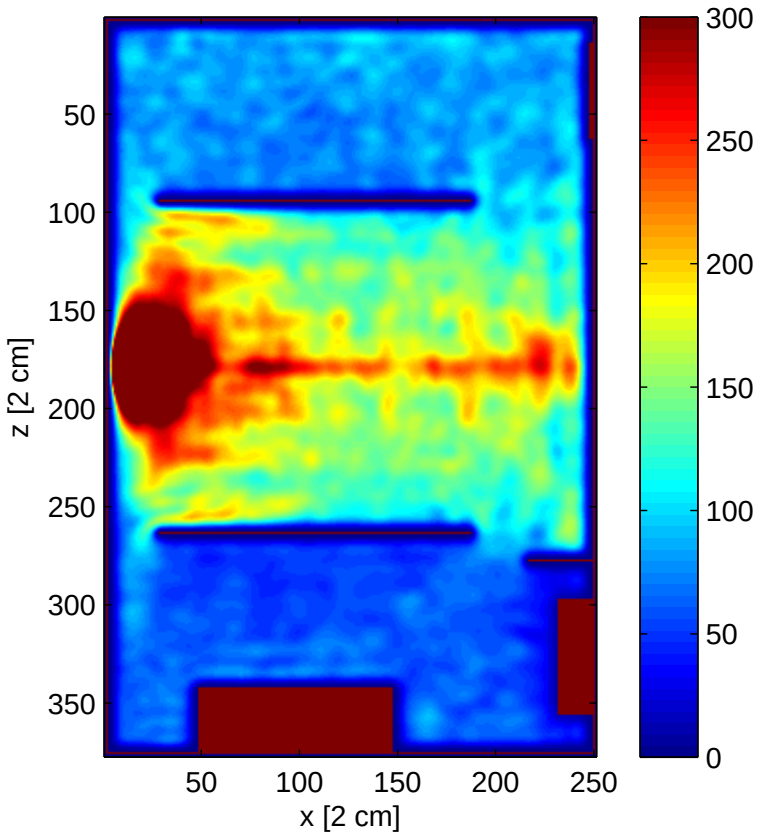

Figure 42: Acoustic power distribution in the medium sized room

The suppression values for the various reference microphone arrangements are given by Table 4. Next to the simulated results measurement data of a comparable real-life experiment is given.

The noise suppression scales well with the addition of the right and left microphones but – similarly to the experimental case – is almost unaffected by placing yet another microphone at the top. This suggests that only a small portion of the noise power arrives from the direction given by the error – top microphones' axis. This may be due to the fact that that this microphone is placed in the direction of an edge from which very few reflections are originating.

suppression values measured in the field simulation were 2.1 dB for a single reference microphone, and 5.6 dB in case of four.

Arrangement	suppression [dB]	
	Simulation	Experiment
front	2.9	3.6
front, right, left	6.6	8.5
front, right, left, top	6.8	8.0

Table 4: suppression of 1 kHz stochastic noise in a medium sized room

Figure 43 shows the suppression values along the ANC system's plane in the case of 4 reference microphones. Unlike in the previous simulation, the actuator is significantly affecting field values across the whole room.

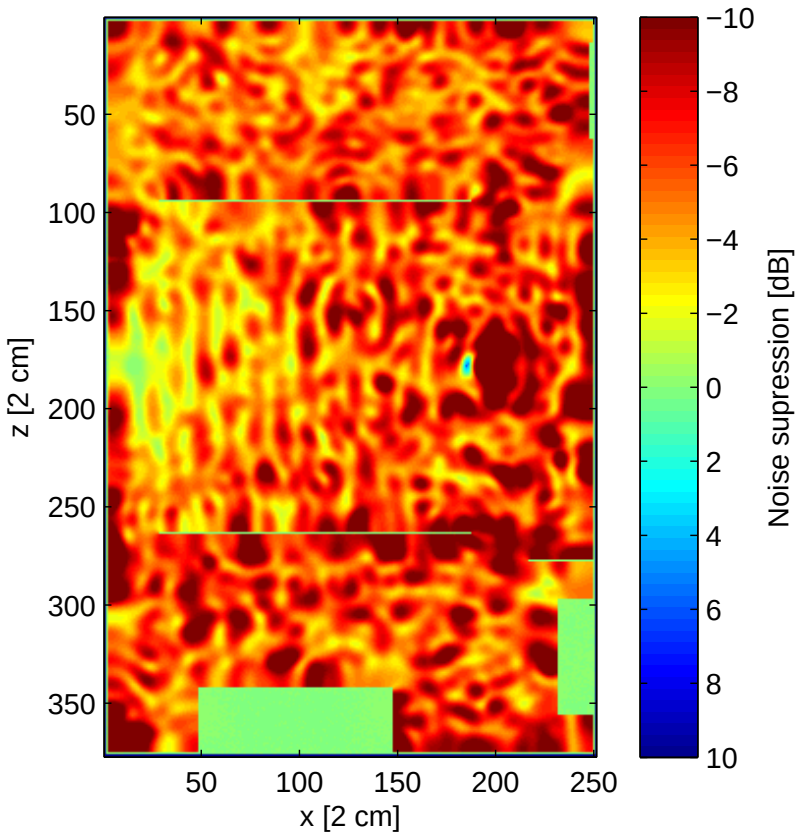

Figure 43: Noise suppression in medium sized room with 4 reference microphones

This is a consequence of a hotspot overlapping the error microphone's position.

The hotspot is mainly due to the vertical separating boards in the middle of the double sided desks. Such a board is visible on Figure 44 and it is cross-cutting the the ANC system's plane at 1 m height.

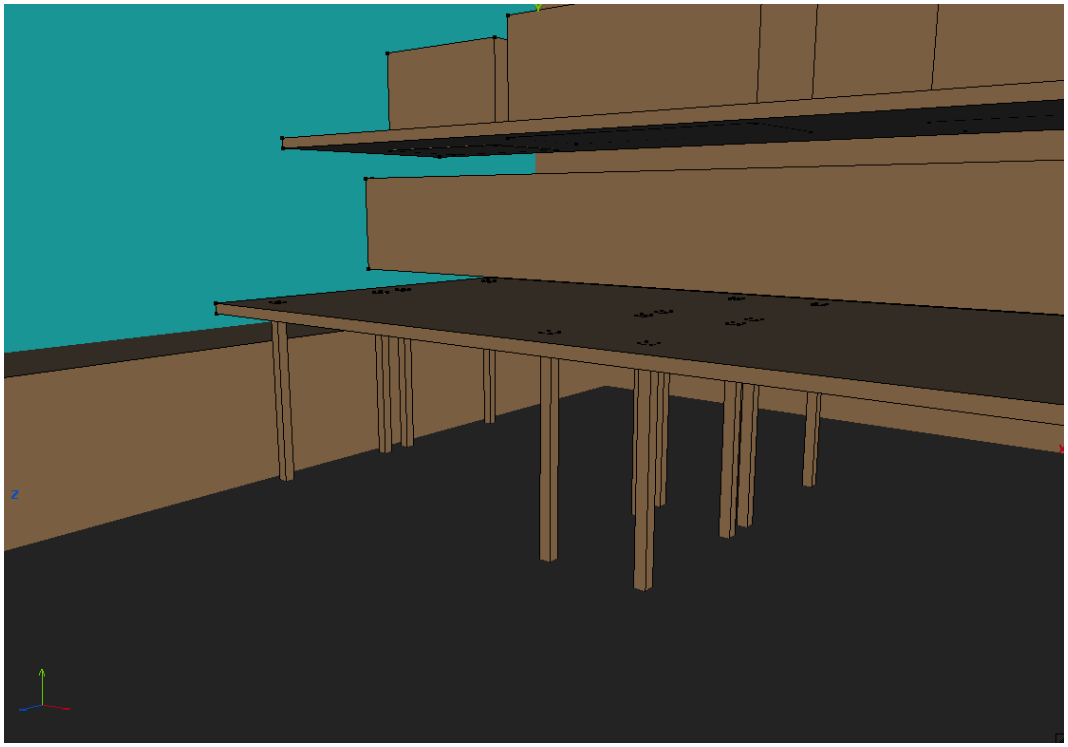

Figure 44: Closeup of the desk models in the medium sized room

These acoustic conditions are highly unfavourable, but the simulated system can still cope with the noise cancelling problem. Figure 45 shows the suppression region which has a typical size of 10 cm, but suppression values exceeding 5 dB are limited to the immediate vicinity of the error microphone. The cost of achieving this localized noise reduction is the significantly higher noise in other areas. This is not an optimal outcome but is not unobserved in real-life situations.

It should also be noted that a real-life ANC system will have natural limits regarding the maximum level of actuation, i.e. the outputs are saturated above a certain level. This saturation behaviour is not modelled in the simulations. In most cases this lack of realism can be considered an advantage of simulated systems, since they enable the investigation of behavioural aspects without being limited by a multitude of imperfections that would make the evaluation of a single property difficult in practical cases. E.g. in this example the simulated ANC system was capable of achieving significant noise cancelling results without being affected by

the absolute level of the noise.

The results are consistent with both the assumptions made about the difficulty of the noise cancelling problem, and the results of similar real-life experiments.

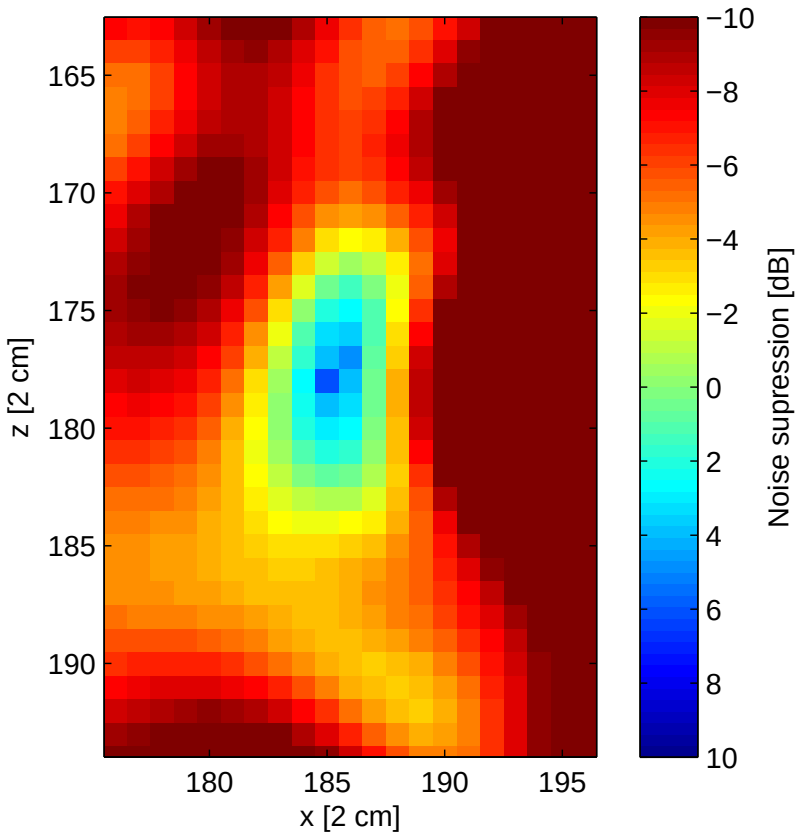

Figure 45: Closeup of the suppression region

A lesson has also been demonstrated by this simulation series, that the overall difficulty of the noise cancelling problem is not directly related to the length of the impulse responses, when the adaptive filter order is sufficiently high. The conditions of the second series proved to be an unfavourable situation compared to the first one, but the ANC system's performance still scaled well with the number of reference microphones.

7 Summary

The physics-based simulation of active noise cancelling systems in reflecting environments seems a promising avenue for testing and investigating the behaviour of ANC systems with multiple reference microphones. In the presented simulations the same typical behaviour of noise cancelling solutions emerged that were observed during previous experiments. This suggests that the simulations can be used both as an alternative and as a form of verification for real-life experiments.

The numerical solution of the second-order wave equation using FDTD method proved to be a well adjustable technique to the ANC problem domain. The chosen SLF scheme was easily adapted to the complementary techniques enabling the simulation of frequency dependent reflective surfaces and anechoic conditions as well. The presented simulation software allows the easy description of acoustic environments through importing graphically edited 3D models. The module layer component provides a level of abstraction that enables the testing of interactive signal processing modules without knowing the implementation details or numerical parameters of the physical simulation.

The presented software is integrated with the Wings3D modeller and the Matlab numerical computing environment. Using these tools simulation parameters are easier to describe than setting up a real-life ANC system, and the simulation results are more accessible than the measurement data of experiments.

The program can efficiently utilize high-power GPUs offering a 15 times speedup compared to a high-end CPU. The simulation runtime for rooms not larger than 150 m^3 is less than a second for a real-time second per cubic-meter.

The investigation of active noise cancelling systems in reflective environments both experimentally and using simulations has yielded valuable lessons that are hard to quantify.

In typical conditions governed by room acoustics the transfer functions between various locations are complicated, and an analytical approach to determine the efficiency of ANC systems is unfeasible. Similarly determining the optimal arrangement for the placement of reference microphones cannot be carried out in a mathematically exact manner. Our aim was to identify best practices based on the overall behaviour of ANC systems in such conditions and collect rules of thumb for achieving

reasonable noise cancelling efficiency with moderate effort.

Carrying out a large number of real-life experiments and verifying a set of cornerstone results using simulations based on physical principles has provided us with a valuable database upon which our considerations can be based. The main lessons of the investigations can be summed up as follows.

The main factor determining the efficiency of an ANC system in a reflective environment is the number and placement strategy of reference microphones. The complex and differing superposition of various reflected wave fronts at differing locations in the diffuse section makes them less valuable than the waves corresponding to the initial section of the impulse responses. Thus, e.g. an ANC system relying on two reference microphones and having adaptive filters with an order of just 1000 can consistently outperform a system relying on a single reference microphone and utilizing an adaptive filter with 8000 taps. The simulations verified that even when the length of the adaptive filter is as long as the impulse responses of the acoustic environment, the systems efficiency will be bound by the number and placement of the reference microphones. The second set of simulations also demonstrated that merely the length of the impulse responses does not determine the difficulty of the noise cancelling task.

A consequence of the large influence of the structure of the acoustic field is that a reasonable approach for creating efficient ANC solutions would not only be involved with the parameters of the ANC system itself, but would also aim to affect the environment's properties. Choosing smaller rooms for the installation of ANC systems will shorten the impulse responses describing the transfer paths. Absorbing materials such as thick drapes, and additional furniture will further reduce the vanishing time of echos. The irregular placement of reflecting bodies will help avoid hotspots where noise power may be exceptionally high.

The overall lesson is that in order to create efficient noise cancelling solutions one can not separate the signal processing aspects from careful acoustic considerations. Modern DSPs provide the power and flexibility to achieve satisfactory noise cancelling performance in most conditions and acoustic studies of these conditions can provide the most defining parameter settings.

The presented simulation software provided results consistent with experimental

measurements even when using crude approximations of the real-life environments. The simulations discussed sufficiently demonstrated the important characteristics of ANC systems in reflective environments and were accurate enough for the purposes of this thesis. If necessary however further efforts could be taken to provide a more accurate model of real-life environments, and the absolute convergence of simulated and measured results could be investigated.

Using a high-performance GPU the current implementation proved to be a convenient solution for the simulation of rooms no larger than 7.5×5 m and an upper cutoff frequency of 2 kHz with a runtime of 105 s per simulated real-life second. The observed drop in performance in the case of the 11×8 m room implies that the implementation may have to be optimized for larger problem sizes, but the 16 minute runtime per real-life second was still tolerable for the presented simulations.

Finally the import – export procedures between the simulation software and Matlab are somewhat cumbersome and an interface providing complete Matlab command line integration would be a useful and straightforward improvement of the simulation tool.

With the collected experimental and simulation data, the lessons of the investigations and the help of the to be improved simulation software, future investigations regarding the efficiency of a given ANC system in a given acoustic environment can be carried out more effectively. Using the simulation software the experimental database can be further expanded and existing results can be verified.

8 References

[1] J. P. Berenger. "A perfectly matched layer for the absorption of electromagnetic waves". In: *Journal of Computational Physics* 114 (1994), pp. 185–200.

[2] S. J. Elliot. "Active Noise Control". In: *IEEE Signal Processing Magazine* (Oct. 1993).

[3] S. J. Elliot. "Optimal controllers and adaptive controllers for multichannel feedoforward control of stochastic disturbances". In: *IEEE Transactions on Signal Processing* SP-48 (Apr. 2000), pp. 1053–1060.

[4] S. J. Elliot, I. M. Stothers, and P. A. Nelson. "A multiple error LMS algorithm and its application to the active control of sound and vibration". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP-35 (Oct. 1987), pp. 1423–1434.

[5] Marcus J. Grote and Imbo Sim. *Efficient PML for the wave equation*. 2010. URL: `http://arxiv.org/abs/1001.0319v1`.

[6] Steven G. Johnson. *Notes on Perfectly Matched Layers (PMLs)*. Massachusetts Institute of Technology, 2010. URL: `http://math.mit.edu/~stevenj/18.369/pml.pdf` (visited on 12/01/2013).

[7] Yoshinobu Kajikawa. "Recent advances on active noise control: open issues and innovative applications". In: *APSIPA Transactions on Signal and Information Processing* 1 (2012), pp. 1–21.

[8] *Khronos Group OpenCL reference site*. URL: `http://www.khronos.org/opencl/` (visited on 12/01/2013).

[9] Konrad Kowalczyk. "Boundary and medium modelling using compact finite difference schemes in simulations of room acoustics for audio and architectural design applications". PhD thesis. Queen's University Belfast, 2008.

[10] S. M. Kuo and D. R. Morgan. "Active noise control: a tutorial review". In: *Proceedings of the IEEE*. Vol. 87. June 1999, pp. 943–973.

[11] Randall J. LeVeque. *Finite Difference Methods for Differential Equations*. University of Washington, 2005. URL: `http://www.nhcue.edu.tw/~jinnliu/proj/Device/LeVeque-FDM-2005.pdf`.

[12] *Physikalisch-Technische Bundesanstalt, alpha-Databank*. URL: `http://www.ptb.de/cms/en/fachabteilungen/abt1/fb-16/ag-1630/room-acoustics/absorption-grade-data-collection.html` (visited on 12/12/2013).

[13] László Sujbert. "A new filtered lms algorithm for active noise control". In: *Proceedings of the Active '99*. The International EAA Symposium on Active Control of Sound and Vibration. 1999, pp. 1101–1110.

[14] László Sujbert et al. *Fejhallgatóval történő aktív zajcsökkentés call-centerben*. 2013.

[15] László Sujbert et al. *Médiaszékre épülő aktív zajcsökkentő rendszer*. 2013.

[16] László Sujbert et al. *Több referenciajelet felhasználó aktív zajcsökkentő rendszerek vizsgálata*. 2012.

[17] Y. Tu and C. R. Fuller. "Multiple reference feedforward active noise control, part I-II." In: *Journal of Sound and Vibration* 233 (5 June 2000), pp. 745–774.

[18] B. Widrow and E. Walach. *Adaptive Inverse Control*. Prentice Hall, Inc., 1996.

[19] *Wings3D polygon modeler homepage*. URL: `http://www.wings3d.com/` (visited on 12/01/2013).