DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Simulation of diffraction effects of sound waves using the ADER-DG method

Henri Johannes Rößler

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Simulation of diffraction effects of sound waves using the ADER-DG method

Simulation von Beugungseffekten bei akustischen Wellen mithilfe der ADER-DG Methode

Author: Henri Johannes Rößler
Supervisor: Univ.-Prof. Dr. Michael Bader
Advisor: M.Sc. Carsten Uphoff
Submission Date: September 16, 2019

I would like to thank my parents for always keeping me motivated and enabling my studies, as well as my advisor, Carsten Uphoff, for his great support and for the interesting insights beyond the scope of this thesis he gave me.

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, _____ Henri Johannes Rößler

Abstract

The objective of this thesis is to simulate the propagation of sound waves in virtual environments, in order to enable the auralization of diffraction effects. To accomplish this efficiently, without having to simulate each source signal individually, the acoustic domain can be treated as a linear, time-invariant (LTI) system. As such, it is fully describable by its impulse response, which embodies the reverberation characteristics of the simulated environment. These can then be transferred to arbitrary source signals via a convolution operation, once the impulse response has been simulated using the ADER Discontinuous Galerkin (DG) method. This approach not only significantly accelerates the auralization, it also incorporates important physical effects, such as refraction and diffraction. Furthermore, to feature the construction of rudimentary environments, we implement reflecting and absorbing boundaries. Associated with this, our implementation adapts the domain grid in such a way, that simple planar boundaries can be defined everywhere within the domain. The results of our method are validated by convergence tests, which base on simulations for which the analytical solution is known and can be compared to the numerical measurements. Altogether, our experiments produced reasonably realistic sounding results, which suggests that our method can be advanced to an applicable auralization system.

Contents

1 Introduction

Simulating the propagation of acoustic waves, which is characterized by a hyperbolic partial differential equation (PDE), is computational very expensive, especially in three-dimensional space. This makes it impractical for applications that require or desire real-time performance, like sound engines in video games or auralization systems.

Thus, the wave propagation is often expressed as a geometrical problem. As we will conclude in chapter 6, acoustic waves propagate spherically and their intensities decrease proportionally to the travelled distance. These properties follow directly from the analytical solution of the PDE, and can therefore be exploited to simulate acoustics in rather simple virtual environments, where only reflection effects occur.

More specifically, it is often sufficient to model sound waves as beams that get reflected and partially attenuated at surfaces in the environment. This eliminates the need to simulate the waves via a numerical PDE solver, as one can simply backtrace the reflections of the waves from the receiver to the sound source. One popular auralization method, which accomplishes this, utilizes the so-called *image source model*. There, the sound source is repeatedly mirrored along the surface planes. These mirrored points can then, in simplified terms, be treated as virtual sources, such that the line segments between those and the receiver intersect with the surfaces exactly at the reflection points. Such a method is capable of producing highly realistic results in real-time, provided that the number of considered reflections from the source to the receiver is kept small enough. [1, sec. 11.1.3.1]

However, the main disadvantage with this approach is, that it cannot make diffraction effects audible, or is at least not easily extensible to do so. Thus, we utilize the ADER-DG method, with which hyperbolic PDEs such as the acoustic wave equation can be solved numerically, including refraction and especially diffraction effects, if appropriate boundary conditions are applied. This enables the auralization of more complex environments, which cannot be modeled appropriately with the abovementioned geometrical approach. The accomplishment of our method in particular is, that we do not have to simulate each source term, like a piece of music, individually, which would require a lot of computational power. In fact, the acoustic domain behaves like a LTI system, such that the received signal $y(t)$ is exactly characterized by the convolution

$$y(t) = x(t) * h(t),$$

where $x(t)$ is an arbitrary source signal and $h(t)$ is the measured signal generated by the Dirac delta impulse. In acoustics, this procedure of determining the acoustic signal is referred to as *convolution reverberation*, which is the key concept of our method. So first, we simulate an impulse, using the ADER-DG method, in order to retrieve $h(t)$ numerically. Once we know the impulse response, we do not need to expensively invest in another simulation, as the received signal can then be determined using discrete convolution, for which highly performant algorithms already exist. Altogether, this makes our method not only physically highly accurate, but also relatively efficient.

This thesis is organized as follows: In the following chapter, the basic concepts involved in a Finite Volume Method (FVM) will be outlined, such as the PDE that describes the propagation

of acoustic waves, and the Riemann problem, which is fundamental for FVMs. In chapter 3, the implementation of boundary conditions, namely absorption, total and partial reflection, is discussed. In order to impose aforementioned boundary conditions on arbitrary cuboidal cell interfaces, the necessary spatial domain construction is discussed in chapter 4. The predominant focus of this thesis is on the earlier mentioned convolution reverberation in chapter 5, where we describe how an impulse response can be measured and used to transfer the simulated reverberation to arbitrary acoustic signals. This is followed by a discussion in the sixth chapter, where the impact of certain settings on the accuracy of the simulation is illustrated. In chapter 7, we will present a collection of experiments to showcase diffraction effects and certain capabilities of our simulation software. Finally, a general conclusion wraps up this thesis.

All implementations regarding the simulation are integrated into the existing ADER-DG code for three-dimensional linear acoustics (*LinA*) from https://github.com/TUM-I5/LinA/tree/3D. The convolution reverberation is implemented as a post-processing step in Python.

2 Foundations

This chapter explores the basic concepts which are fundamental for any issue that has been implemented in the course of this thesis. The methods proposed for imposing boundary conditions (chapter 3), refining the simulation domain grid (chapter 4) and measuring an impulse response for convolution reverberation (chapter 5) are mainly based on the following considerations. All concepts will be described for the one-dimensional case, but can be generalized for higher dimensions as well. For additional information on that topic, the reader is directed to [3], which has been the main source for this chapter.

2.1 Linear acoustics equation

From a physical point of view, acoustic waves arise due to the mechanical oscillation of an elastic medium. In air, sound emerges from small disturbances in the atmospheric pressure that propagate through the medium and balance out, both in space and time [4, p. 25] [3, sec. 2.7]. This behavior is described in the linear acoustics equation, for which we will derive a solution in the following. Throughout this section, we will refer to equations and conclusions that can be found in sections 2.7, 2.8 and 2.9 of [3].

Let $q : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^2$ be a function that maps a point x and time t to the pressure (p) and velocity (u) perturbations of the medium:

$$q(x,t) = \begin{bmatrix} p(x,t) \\ u(x,t) \end{bmatrix}. \tag{2.1}$$

In a stationary gas, i.e. with no constant background velocity, the propagation of acoustic waves can be expressed by the linear PDE

$$\frac{\partial}{\partial t} q(x,t) + A \frac{\partial}{\partial x} q(x,t) = 0, \tag{2.2}$$

where

$$A = \begin{bmatrix} 0 & K_0 \\ 1/\rho_0 & 0 \end{bmatrix}. \tag{2.3}$$

Here, K_0 denotes the bulk modulus of compressibility and ρ_0 the density of the gas.

Intuitively, we would expect that the waves propagate to the left and right with some constant speed s, such that $q(x,t)$ only depends on the point $x - st$. Thus, we can introduce a function $\bar{q} : \mathbb{R} \to \mathbb{R}^2$ and use it in the ansatz

$$q(x,t) = \bar{q}(x - st) \tag{2.4}$$

to solve the abovementioned PDE. It immediately follows that

$$\frac{\partial}{\partial t}q(x,t) = -s\vec{q}'(x-st), \tag{2.5}$$

$$\frac{\partial}{\partial x}q(x,t) = \vec{q}'(x-st), \tag{2.6}$$

hence, (2.2) can be transformed into an eigenvalue problem of the form

$$s\vec{q}'(x-st) = A\vec{q}'(x-st). \tag{2.7}$$

We see that the speed s has to be one of the eigenvalues of A, which can be derived by solving

$$\det\left(A-\lambda I_2\right) = \det\left(\begin{bmatrix}-\lambda & K_0\\ 1/\rho_0 & -\lambda\end{bmatrix}\right) = \lambda^2 - \frac{K_0}{\rho_0} = 0, \tag{2.8}$$

where I_2 denotes the 2-by-2 identity matrix. This leads to a left- and right-going wave with speeds

$$\lambda^- = -c_0 \quad \text{and} \quad \lambda^+ = +c_0, \tag{2.9}$$

where

$$c_0 = \sqrt{K_0/\rho_0} \tag{2.10}$$

is the speed of sound. Given the ansatz from (2.4), we can thus assume that the solution for q can be split into a superposition of those two waves with functions $q^-, q^+ : \mathbb{R}\times\mathbb{R} \to \mathbb{R}^2$, i.e.

$$q(x,t) = q^-(x,t) + q^+(x,t). \tag{2.11}$$

Furthermore, we know from (2.7) that $\vec{q}'(x-st)$ has to be an eigenvector of A. And as the ansatz (2.4) decouples into a left- and right-going wave (2.11), the same must hold for $q^-(x,t)$ and $q^+(x,t)$.

One can derive that any scalar multiple of

$$r^- = \begin{bmatrix}-Z_0\\ 1\end{bmatrix} \quad \text{and} \quad r^+ = \begin{bmatrix}+Z_0\\ 1\end{bmatrix} \tag{2.12}$$

is a corresponding eigenvector for λ^- and λ^+, where $Z_0 = \rho_0 c_0$ denotes an important acoustic property called the impedance. Consequently, there have to be scalar functions $w^-, w^+ : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$, called characteristic variables, such that

$$q^-(x,t) = w^-(x,t)r^- \quad \text{and} \quad q^+(x,t) = w^+(x,t)r^+. \tag{2.13}$$

Plugging (2.13) into (2.11) yields the linear system of equations (LSE)

$$q(x,t) = w^-(x,t)r^- + w^+(x,t)r^+ = Rw(x,t) \tag{2.14}$$

with

$$R = [r^- \mid r^+] = \begin{bmatrix}-Z_0 & +Z_0\\ 1 & 1\end{bmatrix} \quad \text{and} \quad w(x,t) = \begin{bmatrix}w^-(x,t)\\ w^+(x,t)\end{bmatrix}. \tag{2.15}$$

From (2.14) and (2.15), it immediately follows that

$$w(x,t) = R^{-1}q(x,t), \tag{2.16}$$

where the inverse of R can be determined, e.g. via Cramer's rule for a 2-by-2 matrix:

$$R^{-1} = -\frac{1}{\det(R)} \begin{bmatrix} -1 & Z_0 \\ 1 & Z_0 \end{bmatrix} = \frac{1}{2Z_0} \begin{bmatrix} -1 & Z_0 \\ 1 & Z_0 \end{bmatrix}. \tag{2.17}$$

We further exploit the hyperbolicity of the linear system (2.2). That is, we can apply a similarity transformation in order to diagonalize A with real eigenvalues, such that

$$R^{-1}AR = \Lambda \quad \text{and} \quad A = R\Lambda R^{-1}, \tag{2.18}$$

where $\Lambda = \text{diag}(\lambda^-, \lambda^+)$ is a matrix with the eigenvalues as diagonal components [3, p. 31, Def. 2.1]. This is of great importance, because by multiplying R^{-1} to (2.2), this results into a system of advection equations for each characteristic variable:

$$\begin{aligned} 0 &= R^{-1}q_t(x,t) + R^{-1}A(RR^{-1})q_x(x,t) \\ &= R^{-1}q_t(x,t) + (R^{-1}AR)R^{-1}q_x(x,t) \\ &= R^{-1}q_t(x,t) + \Lambda R^{-1}q_x(x,t) \\ &\overset{(2.16)}{=} w_t(x,t) + \Lambda w_x(x,t). \end{aligned} \tag{2.19}$$

From the general solution for advection equations given in [3, p. 18], we can conclude that

$$w^{\mp}(x,t) = w^{\mp}(x - \lambda^{\mp}t, 0). \tag{2.20}$$

That is, the characteristic variables, and thus q itself, only depend on the initial state

$$q_0(x) := q(x,0) = \begin{bmatrix} p_0(x) \\ u_0(x) \end{bmatrix} \tag{2.21}$$

at $t = 0$, which gives

$$\begin{bmatrix} w_0^-(x) \\ w_0^+(x) \end{bmatrix} := \begin{bmatrix} w^-(x,0) \\ w^+(x,0) \end{bmatrix} = R^{-1}q_0(x). \tag{2.22}$$

Inserting into (2.14) allows us to formulate a final solution, which is

$$\begin{aligned} q(x,t) &= w_0^-(x - \lambda^- t)r^- + w_0^+(x - \lambda^+ t)r^+ \\ &= \frac{1}{2Z_0}[-p_0(x + c_0t) + Z_0u_0(x + c_0t)] \begin{bmatrix} -Z_0 \\ 1 \end{bmatrix} \\ &\quad + \frac{1}{2Z_0}[p_0(x - c_0t) + Z_0u_0(x - c_0t)] \begin{bmatrix} Z_0 \\ 1 \end{bmatrix}. \end{aligned} \tag{2.23}$$

In conclusion, $q(x,t)$ only depends on the initial values at points $\{x \mp c_0t\}$, which is also called the domain of dependence (DOD) for (x,t). Notice, that if the initial velocity $u_0(x) = 0\ \forall x$, the pressure simplifies to a linear superposition of the initial pressures:

$$p(x,t) = \frac{1}{2}\left[p_0(x + c_0t) + p_0(x - c_0t)\right]. \tag{2.24}$$

2.2 Riemann problem

The issue with the analytical solution in (2.23) is, that it is only applicable for undisturbed wave propagations in an infinitely large domain. Therefore, a numerical approach, like the FVM, is

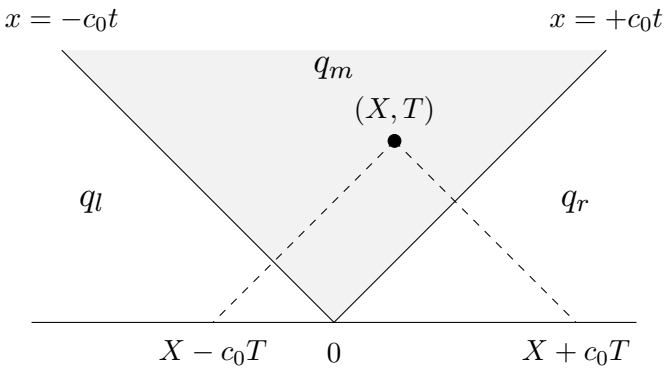


Figure 2.1: Solution of the Riemann problem for (X, T), where one point of its DOD (indicated by the dashed lines) is left and the other one right from the initial discontinuity at $x = 0$. The solution is then based on both initial states q_l and q_r, instead of just one of those. This establishes a new constant middle state q_m between the two wedges (gray area), which propagates in either direction with the speed of sound $\mp c_0$. That is, the solution at each point (X, T), for which $|X| < c_0 T$ holds, is q_m. This illustration is based on [3, p. 54, Fig. 3.3].

more appropriate in order to simulate waves in finite spaces, including their interactions with boundaries or other mediums. To understand how this is achieved, we first have to look at a special initial value problem and derive the solution to these so-called *Riemann problems*. These are pervasive in Finite Volume Methods, which will be discussed in the next section. The concepts stated in the following are based on sections 3.8 and 3.9 from [3].

Consider piecewise constant initial data of the form

$$q_0(x) = \begin{cases} q_l = \begin{bmatrix} p_l \\ u_l \end{bmatrix} & \text{if } x < 0 \\ \\ q_r = \begin{bmatrix} p_r \\ u_r \end{bmatrix} & \text{if } x > 0 \end{cases}. \tag{2.25}$$

Both, the left (q_l) and right (q_r) vectors can be decomposed as in (2.14) into left and right-going waves

$$q_l = w_l^- r^- + w_l^+ r^+ \quad \text{and} \quad q_r = w_r^- r^- + w_r^+ r^+. \tag{2.26}$$

Notice, that the characteristic variables $w_l^\mp, w_r^\mp \in \mathbb{R}$ are constant, just like $q_l, q_r \in \mathbb{R}^2$ are. Furthermore, the waves still advect to the left and right with speeds $\mp c_0$, and so does the discontinuity at $x = 0$. This immediately follows from the fact, that $q(x, t)$ only depends on its DOD, as displayed in the previous section. But now, either both points of the DOD are left or right from the discontinuity, or one point is left and the other one right from it (see Figure 2.1).

In the former case, the solution is just q_l or q_r, according to (2.26). In the latter case, however, the solution becomes a new middle state q_m. Thus, we have

$$q(x, t) = \begin{cases} q_l & \text{if } x + c_0 t < 0 \\ q_m & \text{if } |x| < c_0 t \\ q_r & \text{if } x - c_0 t > 0 \end{cases}, \tag{2.27}$$

where q_m still needs to be determined.

We know that

$$w_0^{\mp}(x) = \begin{cases} w_l^{\mp} & \text{if } x < 0 \\ w_r^{\mp} & \text{if } x > 0 \end{cases}, \tag{2.28}$$

and that the general solution for q according to (2.23) is

$$q(x,t) = w_0^-(x + c_0 t) r^- + w_0^+(x - c_0 t) r^+. \tag{2.29}$$

In the case that $x + c_0 t > 0$ and $x - c_0 t < 0$, respectively $|x| < c_0 t$, we get $w_0^-(x) = w_r^-$ and $w_0^+(x) = w_l^+$. Plugging into (2.29) leads to the middle state

$$\begin{aligned} q_m &= w_r^- r^- + w_l^+ r^+ \\ &= \frac{1}{2Z_0}(-p_r + Z_0 u_r) \begin{bmatrix} -Z_0 \\ 1 \end{bmatrix} + \frac{1}{2Z_0}(p_l + Z_0 u_l) \begin{bmatrix} Z_0 \\ 1 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} (p_l + p_r) - Z_0(u_r - u_l) \\ (u_l + u_r) - (p_r - p_l)/Z_0 \end{bmatrix}. \end{aligned} \tag{2.30}$$

In consequence, two waves arise from the discontinuity, traveling to the left and right with speeds $\mp c_0$. The jumps $\mathcal{W}^-, \mathcal{W}^+ \in \mathbb{R}^2$ across those waves are given as

$$\mathcal{W}^- = q_m - q_l = (w_r^- - w_l^-) r^- \quad \text{and} \quad \mathcal{W}^+ = q_r - q_m = (w_r^+ - w_l^+) r^+. \tag{2.31}$$

These can also be expressed by the so-called wave strengths $\alpha^{\mp} = (w_r^{\mp} - w_l^{\mp})$, which solve the LSE

$$R\alpha = \begin{bmatrix} \alpha^- \\ \alpha^+ \end{bmatrix} = q_r - q_l, \tag{2.32}$$

such that the above defined wave jumps simplify to

$$\mathcal{W}^{\mp} = \alpha^{\mp} r^{\mp}. \tag{2.33}$$

Finally, with (2.31), we can rewrite (2.29) to

$$q(x,t) = q_l + H(x + c_0 t)\mathcal{W}^- + H(x - c_0 t)\mathcal{W}^+, \tag{2.34}$$

where $H(\cdot)$ denotes the Heaviside step function. Furthermore, the middle state (2.30) can be written as

$$q_m = q_l + \mathcal{W}^- \quad \text{or} \quad q_m = q_r - \mathcal{W}^+. \tag{2.35}$$

2.3 Finite Volume Method

In order to solve the linear acoustics equation (2.2), we will utilize the ADER-DG method. However, as the numerical fluxes and boundary conditions, which are most relevant to us, are chosen equivalently as in the Finite Volume Method, it is sufficient and more comprehensible to discuss the latter instead. Hence, we are going to describe the simulation of acoustic waves using the one-dimensional FVM in this section.

For that, the domain is spatially discretized into, $N \in \mathbb{N}_{>0}$ cells, such that q is represented by piecewise polynomial functions with discontinuities at the cell interfaces. We will consider

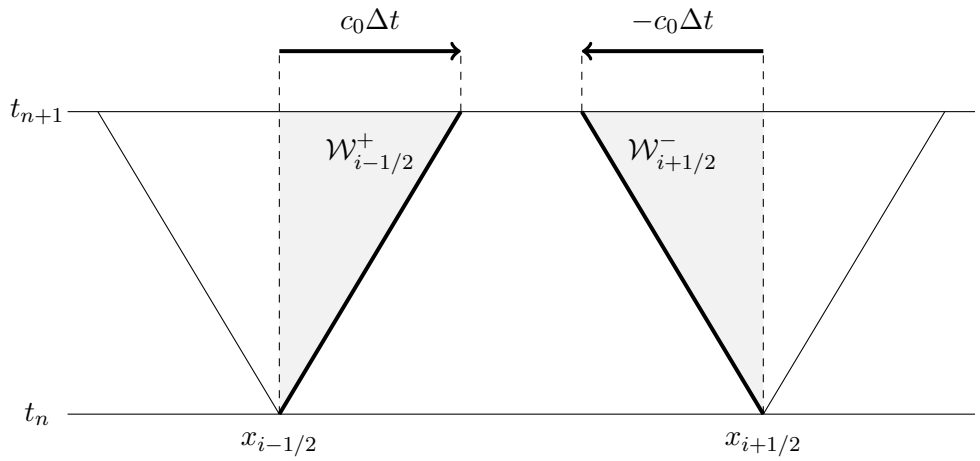


Figure 2.2: Waves propagating from the left ($x_{i-1/2}$) and right ($x_{i+1/2}$) cell interfaces. Q_i^n is updated by the waves $\mathcal{W}^+_{i-1/2}$ and $\mathcal{W}^-_{i+1/2}$ reaching into cell $\mathcal{C}_i$ after time Δt. This illustration is based on [3, p. 79, Fig. 4.7].

the simplest representation with a polynomial degree of 0, i.e. a piecewise constant function. This essentially results into Riemann problems that have to be solved at each cell interface. Throughout this section, we refer to sections 4.10 and 4.12 of [3].

Let Δx be the cell width and Δt be the timestep of the simulation. Then, $Q_i^n \in \mathbb{R}^2$ denotes the quantity vector in cell

$$\mathcal{C}_i := (x_{i-1/2}, x_{i+1/2}) = ((i-1/2)\Delta x, (i+1/2)\Delta x) \tag{2.36}$$

for $i \in [N]$ at time $t_n := n \cdot \Delta t$.

In general, the aim is to update the cell values gradually in each timestep ($Q_i^n \to Q_i^{n+1}$), which can be realized using the so-called reconstruct-evolve-average (REA) algorithm. It consists of the following three steps, which repeat themselves after each iteration:

1. *Reconstruct* a piecewise constant function $\tilde{q} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^2$ from the cell averages, i.e.

$$\tilde{q}^n(x, t_n) = Q_i^n \quad \forall x \in \mathcal{C}_i. \tag{2.37}$$

2. *Evolve* the hyperbolic PDE a time Δt further by solving the Riemann problems at each cell interface to obtain $\tilde{q}^n(x, t_{n+1})$.

3. *Average* the resulting solution within each cell to get the new cell averages

$$Q_i^{n+1} = \frac{1}{\Delta x} \int_{\mathcal{C}_i} \tilde{q}^n(x, t_{n+1}) \, \mathrm{d}x. \tag{2.38}$$

Suppose the cell averages, and thus $\tilde{q}^n(x, t_n)$, are known for the current timestep. In the previous section, we found out that the solution which arises from a discontinuity between two cells consists of two waves propagating to the left and right (2.34). To be more precise, at the left interface of a cell $\mathcal{C}_i$, the right-going wave $\mathcal{W}^+_{i-1/2}$ interfers with Q_i^n, as well as the left-going wave $\mathcal{W}^-_{i+1/2}$ from the right interface (see Figure 2.2). We also know, that the waves move with velocities $\mp c_0$, thus, after a time Δt, they travelled $\mp c_0 \Delta t$ far. This exactly describes how the piecewise constant function (2.37) evolves, so we can now elaborate on the last step of the REA

algorithm.

Following (2.35), the middle states that evolve from the cell interfaces are given as

$$Q_i^n - \mathcal{W}_{i-1/2}^+ \quad \text{and} \quad Q_i^n + \mathcal{W}_{i+1/2}^-, \tag{2.39}$$

but the waves only effect the cell's total average Q_i^n by the fraction $c_0\Delta t/\Delta x$. Thus, we can conclude that the updated cell value is

$$Q_i^{n+1} = Q_i^n - \frac{c_0\Delta t}{\Delta x}\left[\mathcal{W}_{i-1/2}^+ - \mathcal{W}_{i+1/2}^-\right]. \tag{2.40}$$

Note, that Q_i^{n+1} only depends on its own cell value and the values of its neighboring cells from the last timestep. The important consequence which follows from that is specified by the so-called CFL condition, which requires that the true solution for Q_i^{n+1} must not depend on cells other than the three mentioned ones. This can only be ensured if the waves propagate at most the distance equal to the cell widths during one timestep, i.e.

$$C := \frac{c_0\Delta t}{\Delta x} \leq 1, \tag{2.41}$$

where C is called the Courant or CFL number. [3, sec. 4.4]

Although these considerations are already sufficient to implement the REA algorithm, it will be convenient for the next chapter to investigate in an alternative formulation of (2.40). It is clear that $\mathcal{W}_{i-1/2}^-$ and $\mathcal{W}_{i+1/2}^+$ do not influence the cell $\mathcal{C}_i$ at all, because those waves move apart from it. So

$$Q_i^{n+1} = Q_i^n - \frac{c_0\Delta t}{\Delta x}\left[\left(0\cdot\mathcal{W}_{i-1/2}^- + \mathcal{W}_{i-1/2}^+\right) - \left(\mathcal{W}_{i+1/2}^- + 0\cdot\mathcal{W}_{i+1/2}^+\right)\right] \tag{2.42}$$

is equivalent to (2.40). If we split Λ from (2.18) into two diagonal matrices, for which one contains only the negative and the other one the positive eigenvalues

$$\Lambda^- = \operatorname{diag}(-c_0, 0) \quad \text{and} \quad \Lambda^+ = \operatorname{diag}(0, +c_0), \tag{2.43}$$

we also obtain a splitting of the coefficient matrix A into

$$A^- = R\Lambda^- R^{-1} \quad \text{and} \quad A^+ = R\Lambda^+ R^{-1}. \tag{2.44}$$

This is useful, as multiplying these matrices to the differences between two cell values $\Delta Q_{i-1/2} := Q_i - Q_{i-1}$ yields

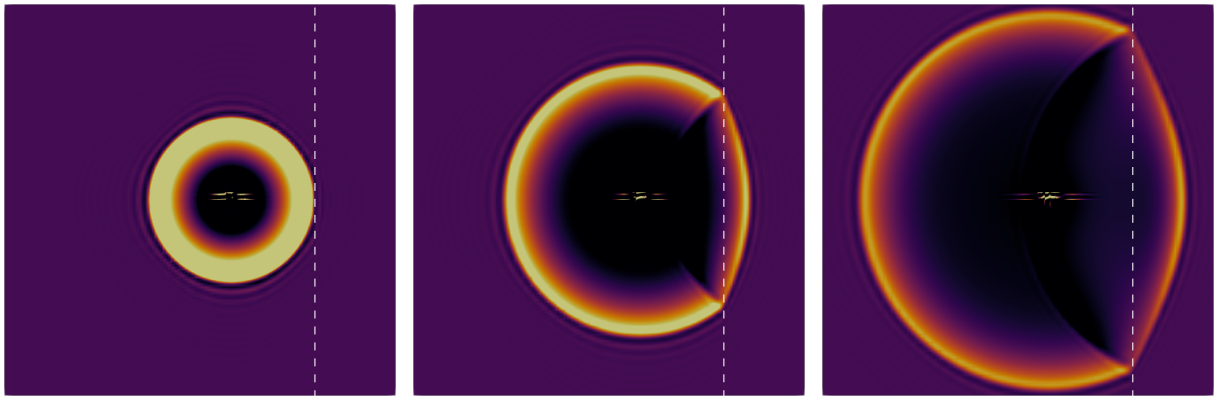
$$\begin{aligned} A^-\Delta Q_{i+1/2} &= R\Lambda^- R^{-1}(Q_{i+1} - Q_i) \\ &\overset{(2.32)}{=} R\Lambda^-\alpha_{i+1/2} \\ &= -c_0\alpha_{i+1/2}^- r^- + 0\cdot\alpha_{i+1/2}^+ r^+ \\ &\overset{(2.33)}{=} -c_0\mathcal{W}_{i+1/2}^- + 0\cdot\mathcal{W}_{i+1/2}^+ \end{aligned} \tag{2.45}$$

and analogously

$$A^+\Delta Q_{i-1/2} = 0\cdot\mathcal{W}_{i-1/2}^- + c_0\mathcal{W}_{i-1/2}^+. \tag{2.46}$$

Now, the idea behind the reformulation in (2.42) emerges, as we can simply write

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(A^+\Delta Q_{i-1/2} + A^-\Delta Q_{i+1/2}). \tag{2.47}$$



$t = 0.0006$ $t = 0.0010$ $t = 0.0014$

Figure 2.3: Simulation of a wave generated by a point source that is refracted at the interface between two mediums (indicated by the dashed line). The medium coefficient matrices A_i were initialized in such a way, that in the left medium, the speed of sound and impedance are $c_l \approx 343$ m/s and $Z_l \approx 413$, whereas the wave speed in the right medium is $c_r \approx 171$ m/s with impedance $Z_r \approx 205$. As can be seen, the wave propagating in the right medium has approximatively half the speed as in the left medium. Note also, that both, the reflected and transmitted wave, have less power than the original wave.

This transformation is more convenient to deal with in higher dimensions and different hyperbolic systems, as it decouples the solution from the waves $\mathcal{W}^{\pm}$ in (2.40) and instead only depends on the cell differences.

2.4 Variable impedance

A similar representation can be derived for a domain with varying medium coefficients in the cells, which is particularly useful to understand partially reflecting boundary conditions in section 3.3. Assume we have cellwise constant acoustic properties

$$A_i = A(x) = \begin{bmatrix} 0 & K_i \\ 1/\rho_i & 0 \end{bmatrix} \tag{2.48}$$

$$c_i = c(x) = \sqrt{K_i/\rho_i} \tag{2.49}$$

$$Z_i = Z(x) = \sqrt{K_i \rho_i} \tag{2.50}$$

$\forall x \in \mathcal{C}_i$. Now, consider the Riemann problem at the interface between two mediums with different impedances Z_l, Z_r. The eigenvectors (2.12) of the waves propagating into the left and right medium with speeds $-c_l$ and c_r become

$$r^- = \begin{bmatrix} -Z_l \\ 1 \end{bmatrix} \quad \text{and} \quad r^+ = \begin{bmatrix} +Z_r \\ 1 \end{bmatrix}. \tag{2.51}$$

Any futher derivation made throughout the previous sections can be generalized for variable impedances by substituting (2.12) with (2.51). [3, sec. 9.9]

The only difference is, that for the FVM, we obtain matrices (2.44) at each interface [3, sec.

9.11], i.e.

$$A^{\pm}_{i-1/2} = R_{i-1/2}\Lambda^{\pm}_{i-1/2}R^{-1}_{i-1/2} \tag{2.52}$$

with

$$R_{i-1/2} = \begin{bmatrix} -Z_{i-1} & +Z_i \\ 1 & 1 \end{bmatrix} \tag{2.53}$$

and

$$\Lambda^{-}_{i-1/2} = \mathrm{diag}(-c_{i-1}, 0) \quad \text{and} \quad \Lambda^{+}_{i-1/2} = \mathrm{diag}(0, +c_i). \tag{2.54}$$

Thus, for variable impedances, we update each cell similar to equation (2.47) as follows:

$$Q^{n+1}_i = Q^n_i - \frac{\Delta t}{\Delta x}(A^{+}_{i-1/2}\Delta Q_{i-1/2} + A^{-}_{i+1/2}\Delta Q_{i+1/2}). \tag{2.55}$$

Note, that with various wave speeds c_i, the CFL condition (2.41) is only then satisfied, if the fastest wave cannot travel further than the cell width within the timestep [3, p. 70], i.e.

$$C = \frac{\Delta t}{\Delta x}\max_i |c_i| \leq 1. \tag{2.56}$$

An interesting observation that can be made at the interface between two cells with different impedances is refraction, where the incident wave gets reflected and transmitted into the other medium to some extent (see Figure 2.3). As mentioned earlier, this phenomenon will be discussed in more detail in section 3.3, where we exploit this property in order to implement partial reflections.

3 Boundary conditions

The numerical approach of simulating acoustic waves, that has been developed in the previous chapter, updates each cell based on the differences to its neighboring cells. But those do not exist at the border of the simulation domain. For instance, the one-dimensional domain exists of a limited number of adjacent cells along the x-axis. Thus, the furthest left and right cells do not have a left or right neighbor. This necessitates the definition of how the simulation should behave at the domain borders. One possibility to solve this problem is to introduce so-called ghost cells. These are additional, adjoining cells at the border, which do not belong to the domain as such. In 1D, this is accomplished by appending one cell to either end of the domain. In order to realize a particular and stable behavior at the borders, it has to be defined how the values of these ghost cells need to be set. As they are not part of the domain, they do not get updated like the regular cells. Instead, after each timestep, their values will be reevaluated depending on the cells within the domain. In other words, by introducing ghost cells, it is ensured that all outer cells in the domain have neighboring cells in each direction and can thus be updated as usual.

In this chapter, we will implement absorbing (section 3.1), totally reflecting (section 3.2) and partially reflecting (section 3.3) boundary conditions by making use of the idea behind ghost cells. In fact, this concept can be applied to impose boundary conditions on any cell interface, and is not restricted to the outer border of the domain.

3.1 Absorption

At absorbing boundaries, the acoustic waves shall flow out of the domain. Ideally, no ingoing waves will be created and the incident waves hitting the boundary will not be reflected. This objective has been explored in more depth, e.g. in [5], where the Perfectly Matched Layer (PML) was first introduced. However, we will use an approximative solution, which works sufficiently in most cases.

In [3, sec. 7.3.1], it has been suggested to use zero-order extrapolation for the ghost cells, i.e.

$$Q_0^n = Q_1^n \quad \text{and} \quad Q_{N+1}^n = Q_N^n, \tag{3.1}$$

where Q_0^n is the left and Q_{N+1}^n the right ghost cell. The reasoning behind (3.1) is, that the difference between both adjacent cells at the domain border is zero, and thus the wave strengths $\alpha^{\mp}$ from (2.32) are zero too. Consequently, there is neither an outgoing nor an incoming wave (2.33). For higher dimensions, the same approach has been described in [3, sec. 21.8.5], which however led to unstable results in our experiments.

We will instead refer to the work from [6, sec. 4.1], where the ghost cells are initialized with zero, i.e.

$$Q_0^n = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{and} \quad Q_{N+1}^n = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{3.2}$$

However, we may also want to have absorbing interfaces in the interior of the domain, and not only at its outer border. That can be achieved in the exact same way, but now the ghost cells are rather conceptual than actual cells.

Suppose we want to implement an absorbing boundary (i, s) in cell $Q_L = Q_i$ on side $s \in \{-1, +1\}$, where $s = -1$ refers to the left and $s = +1$ to the right interface of the cell. We will henceforth refer to Q_L as *local* cell, whereas the adjacent cell $Q_N = Q_{i+s}$ is the artificial ghost or simply *neighboring* cell.

Often, the desired boundary conditions can be realized with ghost cells that are a linear transformation of the local cells. That is, we pretend that the neighboring cell of Q_L is

$$Q_N = TQ_L = T \begin{bmatrix} p_L \\ u_L \end{bmatrix}, \tag{3.3}$$

where $T \in \mathbb{R}^{D+1 \times D+1}$ is the transformation matrix. Note, that with higher dimensions, the size of the quantity vectors Q adapt accordingly in order to include the pressure, as well as the velocities for any of the D dimensions, hence $Q \in \mathbb{R}^{D+1}$.

In the case of $D = 1$, we are given equation (2.55), which can be equivalently rewritten to

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(L^+_{i-1/2}Q_i^n - N^+_{i-1/2}Q_{i-1}^n + N^-_{i+1/2}Q_{i+1}^n - L^-_{i+1/2}Q_i^n), \tag{3.4}$$

where the $A^\pm_{i\mp1/2}$ matrices are further split into identical local ($L^\pm_{i\mp1/2}$) and neighboring ($N^\pm_{i\mp1/2}$) matrices. The reasoning behind this reformulation is, that it enables us to impose boundary conditions on any cell interface and for each side independently.

In order to do so, we substitute the neighboring cell $Q_{i\mp1}$ in (3.4) for an interface $(i, \mp1)$ with (3.3). In fact, we do not need to calculate Q_N specifically for each cell and timestep, as applying the transformation matrix T on the corresponding neighboring matrices $N^\pm_{i\mp1/2}$ is of the same effect:

$$N^\pm_{i\mp1/2}Q_N \overset{(3.3)}{=} N^\pm_{i\mp1/2}(TQ_L) = (N^\pm_{i\mp1/2}T)Q_L. \tag{3.5}$$

Thus, for any interface (i, s), we establish artificial ghost cells by redefining the neighboring matrices from (3.4) as follows:

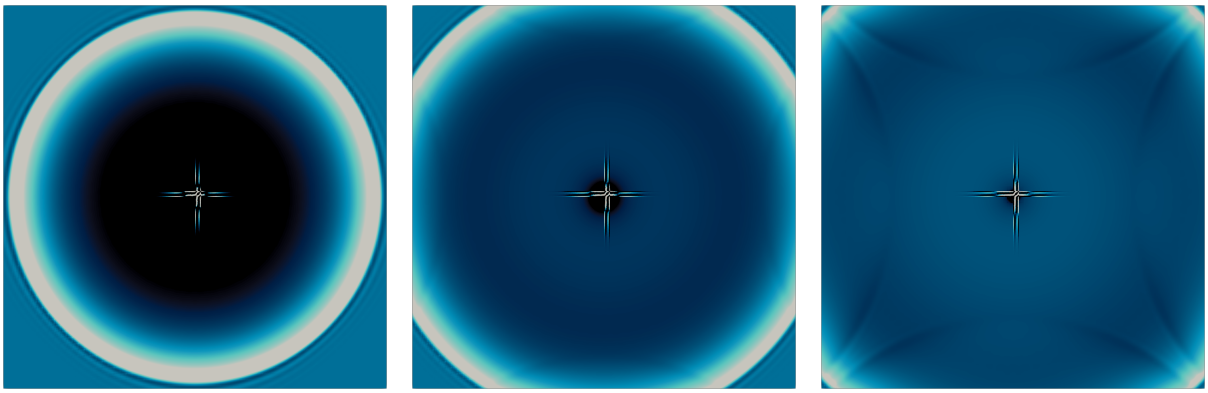
$$N^+_{i-1/2} \mapsto A^+_{i-1/2}T \quad \text{if } s = -1 \quad \text{or} \quad N^-_{i+1/2} \mapsto A^-_{i+1/2}T \quad \text{if } s = +1. \tag{3.6}$$

In the case of absorbing boundaries, the corresponding transformation matrix is a zero matrix

$$T = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \tag{3.7}$$

such that the ghost cell $Q_N = 0 \cdot Q_L = 0$ becomes a zero-vector as required by (3.2).

The huge advantage of this approach is, that we avoid to calculate the values of any ghost cell with a matrix-vector multiplication as described in (3.3) for each timestep. Instead, we simply equate Q_N with the local cell Q_L and perform the transformation once during the initialization of the $N^\pm$ matrices, which is equivalent, as (3.5) shows. In conclusion, this enables us to impose boundary conditions on any interface (i, s), as long as the corresponding ghost cell can be expressed as a linear transformation of Q_i.



$t = 0.0014$ $t = 0.0017$ $t = 0.0020$

Figure 3.1: Simulation of a wave in a 1×1×1-meter domain with absorbing boundary conditions imposed on all borders of the domain. A point source in the center radiates a wave with velocity $\approx$ 343 m/s, which thus hits the boundaries after $\approx$ 0.0015 seconds. These illustrations show a slice of the three-dimensional space in $x-y$-plane at $z = 0.5$. As can be seen in the middle and right image, the wave is mostly absorbed, but weak ingoing waves still arise. Note, that the strengths of those ingoing waves vary with the angle. A wave hitting the border frontally just advects along a single dimension and can thus be totally absorbed. But the steeper the angle of incidence is, the stronger is the wave that arises at the boundary, which can be seen in the right image.

In Figure 3.1, the propagation of a wave in a domain with absorbing borders is illustrated.

3.2 Total reflection

When acoustic waves hit an obstacle, i.e. another medium with different properties, they get reflected to some extent at the interface between those two mediums. In this section, we will consider ideal solid walls, where incident waves get reflected totally.

For a solid interface (i, s) at $x_{i+s\cdot 1/2}$, the physical condition requires

$$u(x_{i+s\cdot 1/2}, t) = 0 \quad \forall t \geq 0. \tag{3.8}$$

According to [3, sec. 7.3.3], this can again be achieved with a ghost cell of the form

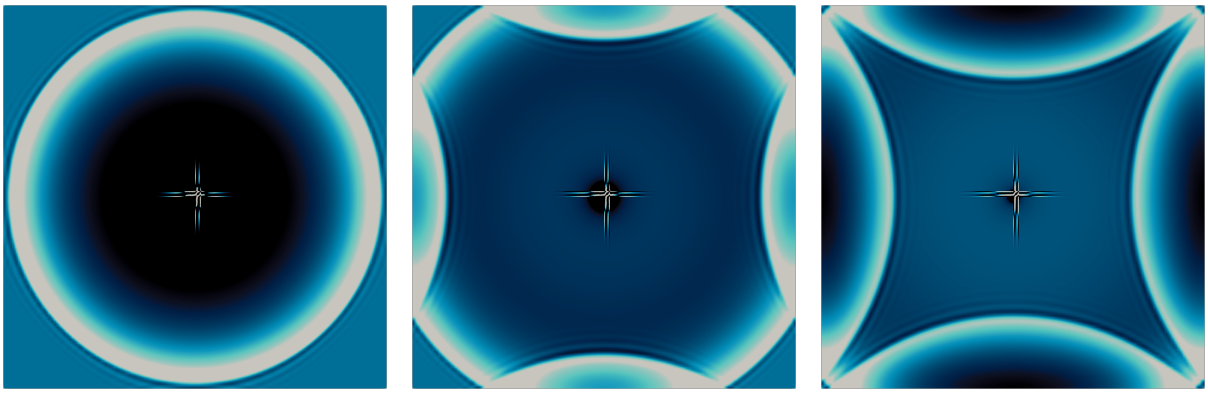
$$Q_N = \begin{bmatrix} p_L \\ -u_L \end{bmatrix}. \tag{3.9}$$

It can be easily proved that (3.9) ensures (3.8) by solving the Riemann problem introduced in section 2.2. We set

$$q_l = Q_N = \begin{bmatrix} p \\ -u \end{bmatrix} \quad \text{and} \quad q_r = Q_L = \begin{bmatrix} p \\ u \end{bmatrix} \tag{3.10}$$

and show that the velocity quantity of the arising middle state (2.30) is zero:

$$u_m = \frac{1}{2}\left[(-u + u) - (p - p)/Z_0\right] = 0 \quad \square \tag{3.11}$$



$t = 0.0014$ $t = 0.0017$ $t = 0.0020$

Figure 3.2: Simulation of the same wave and physical domain as in Figure 3.1, but with totally reflecting boundary conditions imposed on all borders. As can be seen in the images, the reflected waves have the exact same power as the incident waves, i.e. no energy is lost.

Thus, for $D = 1$, totally reflecting boundary conditions are characterized by (3.9), which yields

$$T = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{3.12}$$

In higher dimensions, this is analogous. Suppose we have $D = 3$ and the interface at which the boundary condition shall be imposed is orthogonal to dimension $d \in \{x, y, z\}$. Then, (3.8) must hold for the velocity quantity along the d-axis. So with

$$q = \begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix}, \tag{3.13}$$

where u, v, w denote the velocities in x, y and z directions, the transformation matrix is defined as

$$T = \begin{cases} \text{diag}(1, -1, 1, 1) & \text{if } d = x \\ \text{diag}(1, 1, -1, 1) & \text{if } d = y \\ \text{diag}(1, 1, 1, -1) & \text{if } d = z \end{cases}. \tag{3.14}$$

Hence, we can simply redefine the neighboring matrices in the exact same way as stated in (3.6), but with (3.12) or (3.14) as transformation matrix. The results for totally reflecting boundaries can be seen in Figure 3.2.

3.3 Impedance reflection

In reality, total reflection does not exist. Depending on the material of the obstacle, like concrete or glass, the acoustic waves will be absorbed to different extents (see section 5.3.2). The refraction effect that occurs at the interface between two different mediums has already been briefly mentioned in section 2.4. In fact, it can be exactly determined how strong the absorption, respectively the reflection is. Depending on the impedances Z_1, Z_2 of those mediums, an incident

wave from medium 1 will be reflected at the interface and its pressure magnitude changes by a factor

$$C_R = \frac{p_{\text{reflected}}}{p_{\text{incident}}} = \frac{Z_2 - Z_1}{Z_2 + Z_1}, \tag{3.15}$$

as derived in [3, sec. 9.10]. Hence, we can simulate an impedance mismatch between cells by specifying their medium coefficients accordingly in order to match a particular C_R. Assume the regular impedance in the domain is Z_1 and we want to change the impedance of cells at certain interfaces to Z_2, such that (3.15) is satisfied for a particular C_R. Solving for Z_2 yields

$$Z_2 = Z_1 \frac{1 + C_R}{1 - C_R}, \tag{3.16}$$

and as C_R does not depend on the speeds c_1, c_2, we could define

$$K_2 = Z_2 c_1, \tag{3.17}$$
$$\rho_2 = Z_2 / c_1, \tag{3.18}$$

such that the waves transmitted into medium 2 advect with the same speed

$$c_2 = \sqrt{K_2/\rho_2} = \sqrt{c_1^2} = c_1 \tag{3.19}$$

as in medium 1. This has the advantage of not having to decrease the timestep Δt for a fixed cell width Δx in order to satisfy the CFL condition (2.56).

Another approach is to incorporate the impedance mismatch directly into the cell interfaces by adjusting the $L^{\pm}$ and $N^{\pm}$ matrices, as done for the other boundary conditions. This allows us, although not being physically realistic, to implement infinitesimally small partially reflecting walls, where furthermore no wave is transmitted through the interface.

Suppose we want to simulate a change of impedance at interface (i, s) from Z_i in the local cell to an impedance $\tilde{Z}_{i+s}$ in the neighboring cell. To make clear that we are just mocking the impedance change without actually modifying the medium coefficients of the neighboring cell, we will in the following utilize the tilde $(\tilde{\cdot})$ notation to refer to values resulting from the artificial neighboring cell. The previous approach suggests to redefine the coefficient matrix in the neighboring cell to

$$A_{i+s} \mapsto \tilde{A}_{i+s} = \begin{bmatrix} 0 & \tilde{K}_{i+s} \\ 1/\tilde{\rho}_{i+s} & 0 \end{bmatrix}, \tag{3.20}$$

where $\tilde{K}_{i+s}$ and $\tilde{\rho}_{i+s}$ are simply derived from (3.17) and (3.18) with $Z_1 = Z_i$ and $Z_2 = \tilde{Z}_{i+s}$. According to (2.52), we then have interface matrices

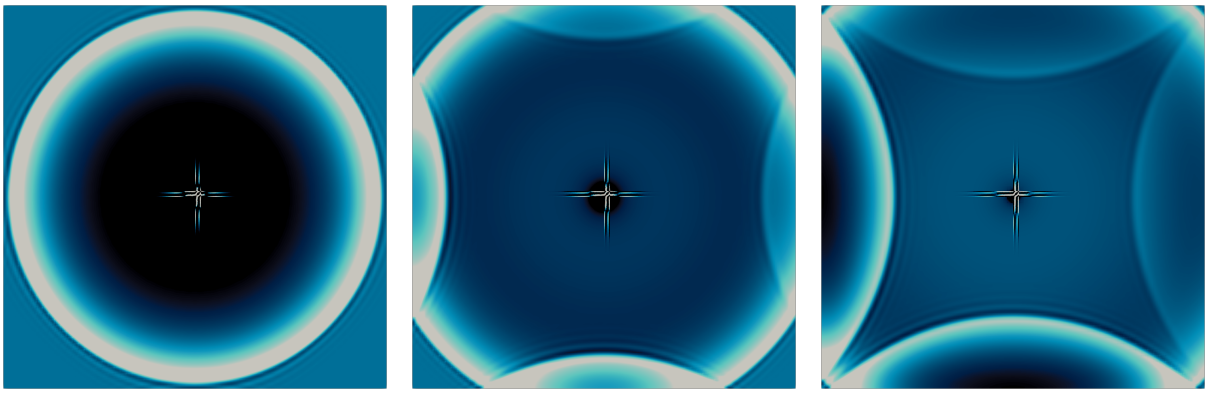
$$\tilde{A}^{\pm}_{i+s\cdot 1/2} = \tilde{R}_{i+s\cdot 1/2} \tilde{\Lambda}^{\pm}_{i+s\cdot 1/2} \tilde{R}^{-1}_{i+s\cdot 1/2}, \tag{3.21}$$

where the eigenvector matrix (2.53) becomes

$$\tilde{R}_{i-1/2} = \begin{bmatrix} -\tilde{Z}_{i+s} & +Z_i \\ 1 & 1 \end{bmatrix} \quad \text{if } s = -1 \quad \text{or} \quad \tilde{R}_{i+1/2} \begin{bmatrix} -Z_i & +\tilde{Z}_{i+s} \\ 1 & 1 \end{bmatrix} \quad \text{if } s = +1. \tag{3.22}$$

Moreover, as shown in (3.19), the wave speeds c_i and $\tilde{c}_{i+s}$ are equal, so we have diagonal matrices (2.54)

$$\tilde{\Lambda}^{+}_{i+s\cdot 1/2} = \operatorname{diag}(0, +c_i) \quad \text{and} \quad \tilde{\Lambda}^{-}_{i+s\cdot 1/2} = \operatorname{diag}(-c_i, 0). \tag{3.23}$$



$t = 0.0014$ $t = 0.0017$ $t = 0.0020$

Figure 3.3: Partially reflecting boundary conditions imposed on the upper and right boundaries. In order to see the difference, total reflection is imposed on the other boundaries. The local matrices (3.24) have been set to establish a reflection coefficient of $C_R = 0.2$. Note, that the wave front after the reflection does not change, i.e. the strength of the reflected wave is independent on the angle of incidence, as opposed to Figure 3.1.

Thus, having a neighboring cell with a different impedance as in the local cell eventually results into local and neighboring matrices from (3.4) which are equal to (3.21) at the interface $x_{i+s\cdot 1/2}$. However, we do not want any wave from the neighboring cell to flow into the local cell, thus, we simply set $Q_N = 0 \cdot Q_L = 0$, the same way we did for absorbing boundary conditions in section 3.1. Consequently, for an interface $(i, \mp 1)$, we redefine the matrices as follows:

$$L^{\pm}_{i\mp 1/2} \mapsto \tilde{A}^{\pm}_{i\mp 1/2} \tag{3.24}$$

and

$$N^{\pm}_{i\mp 1/2} \mapsto 0. \tag{3.25}$$

Note again, that we did not actually change the impedance in the neighboring cell. Instead, we manipulated the interface matrices accordingly *as if* there was an impedance discontinuity. In consequence, we are able to realize partial reflection at any interface, such that the incident wave gets reflected and attenuated pursuant to an arbitrarily chosen reflection coefficient C_R. The results are illustrated in Figure 3.3.

4 Grid refinement

In this chapter, it will be expounded how the domain can be discretized in variably sized cuboidal cells, in order to enable arbitrarily located boundaries, as well as global and local domain refinements. All methods will be explained in 3D, but most of it can be explained for each dimension independently. Thus, for ease of notation, we will throughout this chapter denote an arbitrary dimension as $d \in \{x, y, z\}$.

4.1 Timestep analysis

To perform numerical simulations, the domain has to be discretized both in space and time, which are closely intertwined. Here, the maximum timestep Δt is mainly limited by the cell sizes, provided that the wave speeds are equal in each cell. In this work, we will always simulate the propagation of acoustic waves in a single medium with constant coefficients K_0, ρ_0 and thus a constant wave speed c_0. According to the CFL condition in one dimension (2.41), where the neighboring cells contribute to the result for the next timestep, a wave must only propagate for at most the cell width Δx within one timestep in order to ensure stability.

Similarly, this also applies to three-dimensional space, but now we consider cells $\mathcal{C}_{i,j,k}$ with variable widths $\Delta x_i, \Delta y_j, \Delta z_k \in \mathbb{R}_{>0}$ for each dimension. The authors of [7] suggest that the timestep $\Delta t_{i,j,k}$ in each cell should be at most

$$\Delta t_{i,j,k} = \frac{C_O}{c_0 \cdot \left(\frac{1}{\Delta x_i} + \frac{1}{\Delta y_j} + \frac{1}{\Delta z_k}\right)}, \tag{4.1}$$

where C_O is a factor that further ensures stability for a simulation convergence order O [8, sec. 5.1].

In our implementation, however, we use a constant timestep Δt for the whole domain. That is, regardless of the other cells, the cell with the smallest volume-to-surface ratio is decisive for the timestep, which gives

$$\Delta t = \min_{i,j,k} \Delta t_{i,j,k}. \tag{4.2}$$

Of course, this is not optimal and there exist other approaches like [9] where local timesteps are used for each cell individually. But in our case, we elaborate on the optimal way to refine the grid within given criteria, such that the global timestep (4.2) gets maximized. According to equation (4.1), this is achieved, if the minimal cell widths

$$\Delta d_{\min} := \min_{n \in [N_d]} \Delta d_n \tag{4.3}$$

get maximized in each dimension individually. Here, $\vec{N} = (N_x, N_y, N_z) \in \mathbb{N}^3_{>0}$ denotes the number of cells per dimension. Consequently, all methods that are proposed in the following sections are designed to maximize (4.3), in order to get the best global timestep possible.

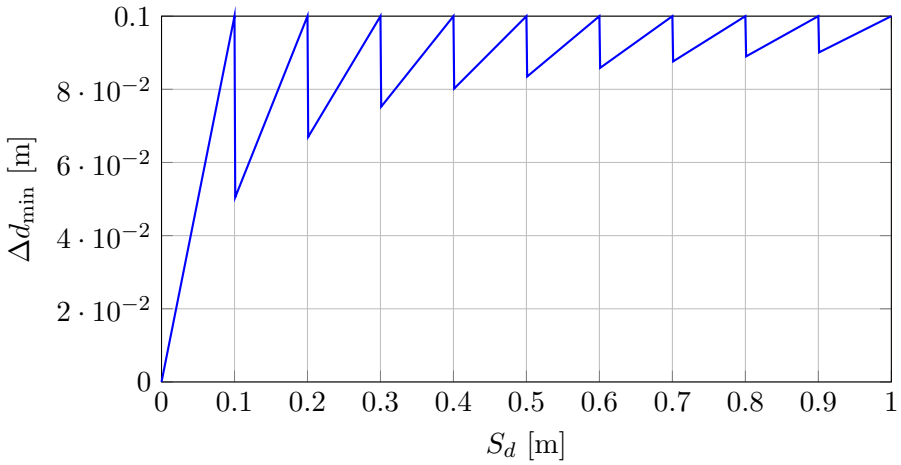


Figure 4.1: Cell widths in a uniformly discretized domain with resolution $R = 10$, depending on the length S_d of the domain in one dimension d.

4.2 Uniform domain

Let $\vec{S} = (S_x, S_y, S_z) \in \mathbb{R}^3_{>0}$ be the variable size of the spatial simulation domain, which is split into cuboidal cells. These generate a three-dimensional grid, which can be accurately represented by its grid points $G_d \subseteq [0, S_d]$. We introduce a global resolution $R \in \mathbb{R}_{>0}$, which limits the cell sizes, such that

$$\Delta d_n \leq 1/R \quad \forall n \in [N_d] \tag{4.4}$$

must hold $\forall d$. To maximize the cell widths, the domain clearly has to be split into equally sized cells. Thus, an arbitrary interval $[a, b]$ in one dimension would have to be split into

$$N_R(a, b) := \lceil (b - a) \cdot R \rceil \tag{4.5}$$

cells with a constant width of

$$\Delta_R(a, b) := \frac{b - a}{N_R(a, b)}. \tag{4.6}$$

Hence, the grid points within the interval are located at

$$G_R(a, b) := \{a + n \cdot \Delta_R(a, b) \mid n \in \{0, \ldots, N_R(a, b)\}\}. \tag{4.7}$$

Suppose we want to discretize the whole domain of size $\vec{S}$ with a global resolution R, then, according to (4.7), the domain's grid points are

$$G_d = G_R(0, S_d). \tag{4.8}$$

With that, all cells have the same size, such that the minimum cell widths (shown in Figure 4.1) are

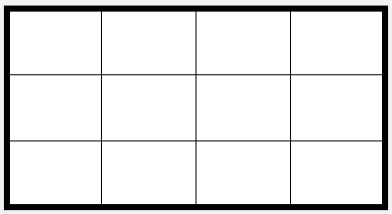
$$\Delta d_{\min} = \Delta_R(0, S_d). \tag{4.9}$$

This yields the following upper bound for the global timestep (4.2):

$$\Delta t \leq C_O \left[c_0 \cdot \left(\frac{1}{\Delta_R(0, S_x)} + \frac{1}{\Delta_R(0, S_y)} + \frac{1}{\Delta_R(0, S_z)} \right) \right]^{-1}. \tag{4.10}$$

Example Let $\vec{S} = (2, 1.05)$ be the size of a two-dimensional domain with global resolution $R = 2$. Then, according to (4.8), we get grid points

- $G_x = \{0, 0.5, 1, 1.5, 2\}$ with $\Delta x_{\min} = 0.5$ and
- $G_y = \{0, 0.35, 0.7, 1.05\}$ with $\Delta y_{\min} = 0.35$.



4.3 Interior boundary interfaces

Modeling rooms with obstacles like walls requires us to impose boundary conditions not only at the border of the domain, but at the interior as well. As discussed in chapter 3, we apply boundary conditions at the cell's interfaces. An arbitrary interface in 3D is a rectangle that is orthogonal to one dimension $d^\perp \in \{x, y, z\}$ with corner points $P_1 = (x_1, y_1, z_1)$ and $P_2 = (x_2, y_2, z_2)$, where $d_1^\perp = d_2^\perp$ and $d_1 \neq d_2$ for the other two dimensions. For instance, the left border of the domain is defined by the frontal lower left corner $P_1 = (0, 0, 0)$ and the rear upper left corner point $P_2 = (0, S_y, S_z)$, where $d^\perp = x$.

Given a set of corner points P_{BC} that define the interfaces on which boundary conditions shall be imposed, a coarse grid $\tilde{G}_d$ can be defined, that includes not more than those interfaces. That is, the coarse grid is the minimal grid required to represent all boundary interfaces and thus only consists of the interface and domain corner points

$$\tilde{G}_d = \{P_d \mid P \in P_{\text{BC}}\} \cup \{0, S_d\}, \tag{4.11}$$

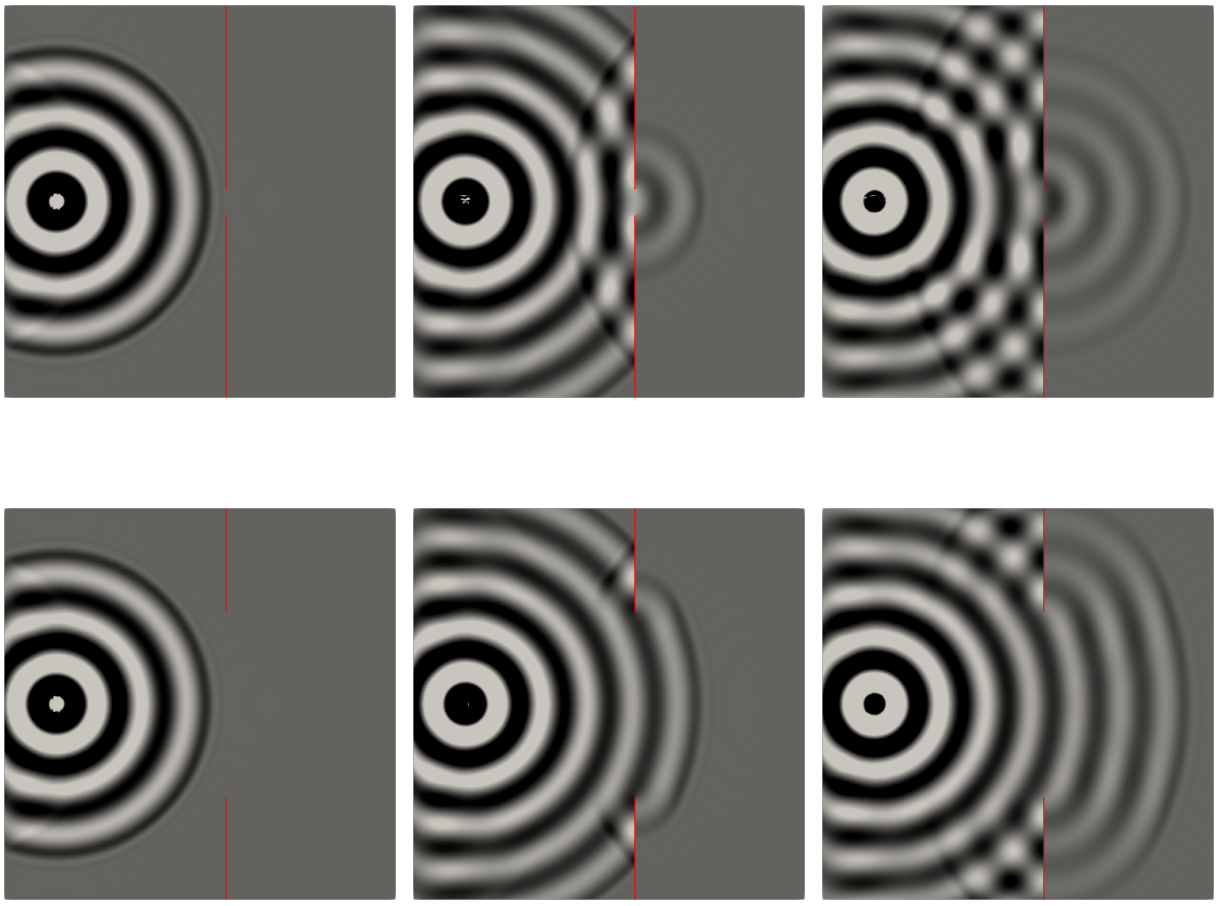
where P_d denotes the d-th coordinate of a point P. This grid is the basis for any further refinement. For ease of notation, we will from now on consider each set of grid points $\tilde{G}_d, G_d$ an ascendingly sorted and indexed family, such that the n-th component of a grid G is defined as

$$(G)_n := [\text{sortedlist}(G)]_n \quad \forall n \in [|G|].$$

Although the coarse grid (4.11) provides all specified boundary interfaces, it may not yet satisfy the condition for a global resolution (4.4). Thus, we still need to uniformly refine every slice of the coarse grid, as defined in (4.7). This yields

$$G_d = \bigcup_{1 \leq n < |\tilde{G}_d|} G_R\Big((\tilde{G}_d)_n, (\tilde{G}_d)_{n+1}\Big). \tag{4.12}$$

In Figure 4.2, it is illustrated how interior boundaries cause diffraction effects.

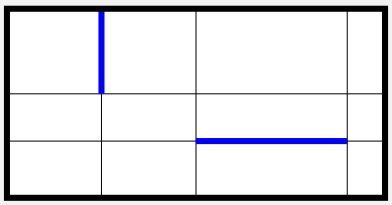


$t = 0.0017$ $t = 0.0026$ $t = 0.0035$

Figure 4.2: Simulation of sine waves that pass an aperture of width 0.1 m in the upper row and an aperture of width 0.6 m in the lower row. The domain has size $\vec{S} = (1.5, 1.5, 1.5)$, with absorbing boundary conditions imposed on all borders and a totally reflecting wall indicated by the red lines at $x = 0.85$. In both scenarios, the point source is located at $P_S = (0.2, 0.75, 0.75)$ and emits a sinusoidal signal with a frequency of 2000 Hz. The wave speed is ≈ 343 m/s. These illustrations show a two-dimensional slice of the domain at $z = 0.75$. As can be seen in both simulations, the waves hitting the aperture get diffracted around the corners into the so-called acoustic shadow [1, sec. 7.2.3].

Example Imagine the following domain with size $\vec{S} = (2, 1)$ and interface points $P_{\text{BC}} = \{(0.5, 0.55), (0.5, 1.0)\} \cup \{(1.0, 0.3), (1.8, 0.3)\}$. According to equation (4.11), the coarse grid is defined as

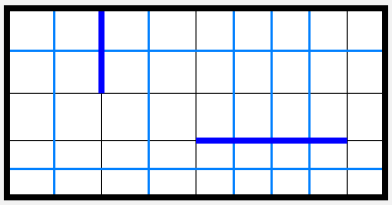
- $\tilde{G}_x = \{0, 0.5, 1.0, 1.8, 2\}$ and
- $\tilde{G}_y = \{0, 0.3, 0.55, 1\}$.



Here, the blue lines represent the interfaces defined in P_{BC}. Notice, that the coarse grid is indeed as minimal as required to depict all specified interfaces in P_{BC}. One can especially see, that the grid consists of cells with very diverse sizes.

Applying a global resolution of $R = 4$ according to (4.12) gives the following grid:

- $G_x = \tilde{G}_x \cup \{0.25, 0.75, 1.2, 1.4, 1.6\}$ and
- $G_y = \tilde{G}_y \cup \{0.15, 0.775\}$.



Here, the light blue lines indicate the additional grid points. Compared to the coarse grid, the domain now is more equally refined, but still exactly integrates all specified boundaries.

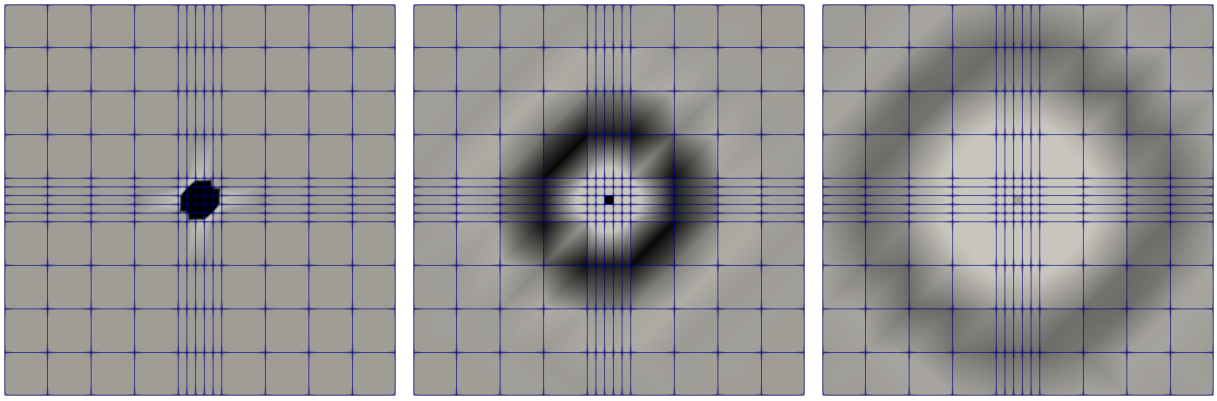
4.4 Local refinement

Going one step further, we may want to define parts in our domain which have a higher resolution than R. This gives us more control over the grid and enables us to refine specific parts of the domain, which may be more prone to inaccuracies compared to the rest of the domain.

Let R_P be the local resolution that shall be applied at a point $P = (P_x, P_y, P_z)$. First, we need to determine the cells which contain P and then again uniformly split the corresponding intervals into equally sized smaller slices based on the specified local resolution. The locally refined grid G_d^* thus becomes:

$$G_d^* = G_d \cup \left[\bigcup_{P_d \in [(G_d)_n, (G_d)_{n+1}]} G_{R_P}\Big((G_d)_n, (G_d)_{n+1}\Big) \right]. \tag{4.13}$$

Of course, this can also be applied iteratively in order to refine the domain locally at multiple points with individual resolutions. An examplary simulation with a single local refinement at the point source is shown in Figure 4.3.



$t = 0.0001$ $t = 0.0006$ $t = 0.0011$

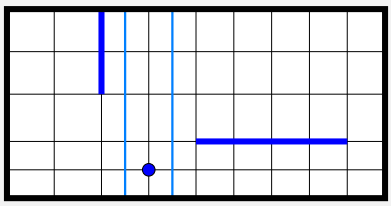
Figure 4.3: Simulation of a wave in a domain of size $\vec{S} = (1, 1, 1)$, which has an higher resolution around the point source $P_S = (0.5, 0.5, 0.5)$. For better visibility of the grid, this simulation has been executed with relatively small resolutions $R = 9$ and $R_{P_S} = 5R = 45$. The grid is visualized using ParaView.

Example Suppose we would like to locally refine the grid in the example from section 4.3 at point $P = (0.75, 0.15)$ with a local resolution of $R_P = 6$. There are four cells containing that point between grid points

- x: $\{0.5, 0.75, 1.0\}$ with $\Delta x = 0.25$ and
- y: $\{0.0, 0.15, 0.3\}$ with $\Delta y = 0.15$.

As $\Delta y \leq 1/6$, the grid is already resolved high enough in y-dimension and does not need any additional grid points, thus $G_y^* = G_y$. But $\Delta x > 1/6$, so (4.13) gives two additional grid points, to wit

$$\begin{aligned} G_x^* &= G_x \cup G_6(0.5, 0.75) \cup G_6(0.75, 1.0) \\ &= G_x \cup \{0.625, 0.875\} \end{aligned}$$



The advantage compared to uniform refinement with a higher global resolution in general is, that the domain will consist of fewer cells. Thus, the computing time is noticeably better. On the other hand, there will also be significantly fewer support points distributed over the whole domain, which is expected to produce a worse accuracy. As the timestep would be similar for both approaches, we can quantify the performance difference by comparing the number of cells for each method.

(i) If only the global resolution R is increased without applying any local refinement, the number of cells is given by (4.5).

(ii) The other approach is to define a constant global resolution $\tilde{R}$ and refine the domain locally at a single point with an increasing local resolution R. Let $I \in [2]$ be the number slices in

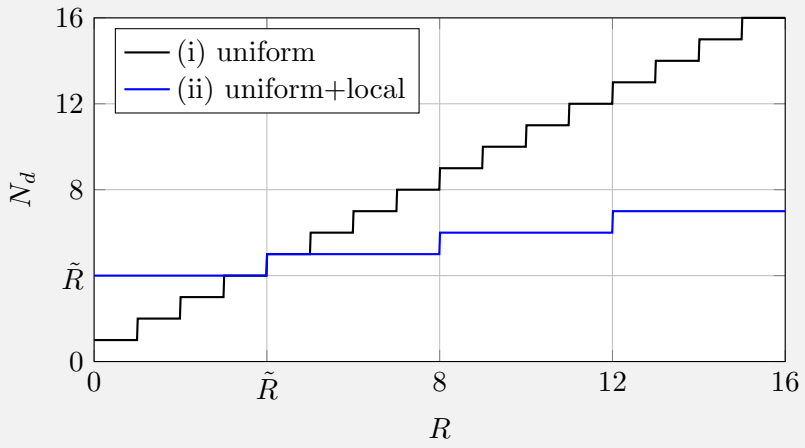
which the refinement point is located. This is typically 1, but can be 2 if the refinement point lies exactly on a grid interface. We know that the domain without local refinement consists of $N_{\tilde{R}}(0, S_d)$ slices, each of width $\Delta_{\tilde{R}}(0, S_d)$. I of them will be split further into $N_R(0, \Delta_{\tilde{R}}(0, S_d))$ locally refined slices. Thus, according to equations (4.5) and (4.6), we get a total number of

$$N_d = \lceil S_d \cdot \tilde{R} \rceil + I \cdot \left\lceil R \cdot \frac{S_d}{\lceil S_d \cdot \tilde{R} \rceil} \right\rceil - I \tag{4.14}$$

cells per dimension.

In section 6.2.2, we will compare both refinement methods regarding their efficiency.

Example Given a domain of size $\vec{S} = (1, 1, 1)$, a constant resolution of $\tilde{R} = 4$ and $I = 1$ slices that shall be refined, we obtain the following numbers of cells in each dimension, depending on R:



Note, that the numbers exponentiate accordingly in 3D.

5 Convolution reverberation

In audio engineering, the determination of a room's reverberation characteristics is of great interest. For this purpose, the propagation of acoustic signals can be described as a LTI system. The acoustics in a room or any environment can thus be exactly characterized by the impulse response, which is typically gathered analogously from microphones. This enables a variety of useful applications. For instance, if a signal gets measured at a point for which the impulse response is known, the original signal can be extracted through an inverse filter of the transfer function. This is also called *deconvolution*. The other way around is to transfer the reverberation ambience of a room like a concert hall to an arbitrary input signal. This effect is referred to as *convolution reverberation*. In this chapter, we will discuss both methods in order to obtain an impulse response numerically, which we will use for the auralization of virtual environments. [1, p. 44] [2, sec. 6.5]

5.1 Theoretical procedure

For any LTI system, its response $y(t)$ to an input signal $s(t)$ can be calculated exactly, if the impulse response $h_\delta(t)$ of the system is known:

$$y(t) = h_\delta(t) * s(t) = \int_0^{+\infty} h_\delta(\tau) \cdot s(t-\tau)\,\mathrm{d}\tau. \tag{5.1}$$

In the frequency domain, the convolution becomes a multiplication

$$Y(s) = H_\delta(s) \cdot S(s), \tag{5.2}$$

where capitalized function names represent the corresponding Laplace transformations. Furthermore, we will utilize the notation $y(t) \circ\!\!-\!\!\bullet\, Y(s)$ to express the correspondence $Y(s) = \mathcal{L}\{y(t)\}$.

However, we are not able to simulate a perfect Dirac delta impulse $\delta(t)$ in practice, so we have to use another source function $f(t)$ instead to obtain the impulse response. We know that the system's response to f is given by (5.1), i.e.

$$h_f(t) = h_\delta(t) * f(t), \tag{5.3}$$

which we will henceforth refer to as approximative impulse response (AIR). To retrieve h_δ, deconvolution can be performed in the frequency domain [2, p. 254], which yields

$$h_\delta(t) \circ\!\!-\!\!\bullet\, H_f(s) \cdot \frac{1}{F(s)}. \tag{5.4}$$

Thus, if the inverse

$$f^{-1}(t) \circ\!\!-\!\!\bullet\, 1/F(s) \tag{5.5}$$

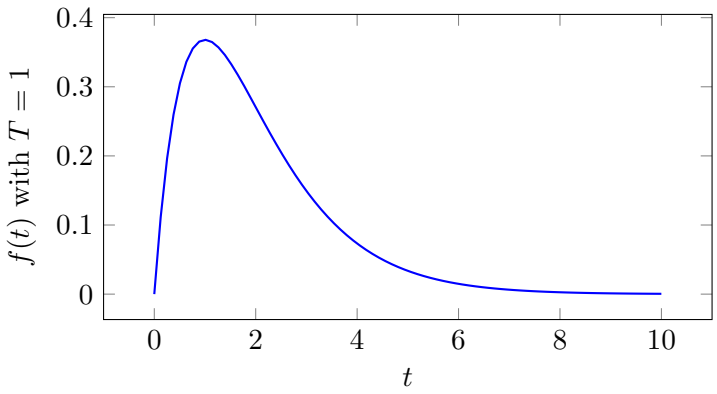
is known, the system output is

$$y(t) = [h_f(t) * f^{-1}(t)] * s(t). \tag{5.6}$$

As proposed by [10], we will use

$$f(t) = H(t) \cdot \frac{t}{T^2} e^{-t/T} \tag{5.7}$$

as our source function, where $T \in \mathbb{R}_{>0}$ is a constant.



Note, that $f(t)$ is a good approximation of the Dirac delta impulse itself. On one hand, it has a strong peak at $t = T$, which quickly attenuates against 0 afterwards. On the other hand, it has the same measure of $\int_{-\infty}^{\infty} f(t) = 1$, regardless of T. But foremost, its inverse according to (5.5) is very simple, which is desirable for the deconvolution. Using partial integration, we obtain the following Laplacian of $f(t)$:

$$\begin{aligned}
f(t) \circ\!\!-\!\!\bullet\ & \int_0^\infty f(t) \cdot e^{-st}\,\mathrm{d}t = \int_0^\infty \frac{t}{T^2} e^{-t/T} \cdot e^{-st}\,\mathrm{d}t = \int_0^\infty \frac{t}{T^2} e^{-t(s+T^{-1})}\,\mathrm{d}t \\
&= \left[-\frac{t}{T^2} \cdot \frac{1}{s+T^{-1}} e^{-t(s+T^{-1})}\right]_0^\infty - \int_0^\infty -\frac{1}{T^2} \cdot \frac{1}{s+T^{-1}} e^{-t(s+T^{-1})}\,\mathrm{d}t \\
&= \left[-\frac{1}{T^2} \cdot \frac{t}{s+T^{-1}} e^{-t(s+T^{-1})}\right]_0^\infty - \left[\frac{1}{T^2} \cdot \frac{1}{(s+T^{-1})^2} e^{-t(s+T^{-1})}\right]_0^\infty \\
&= \frac{1}{T^2 \cdot (s+T^{-1})^2} \\
&= \frac{1}{s^2T^2 + 2sT + 1} \\
&= (1+sT)^{-2}.
\end{aligned} \tag{5.8}$$

Now, inserting (5.8) into (5.5) leads to

$$\begin{aligned}
f^{-1}(t) \circ\!\!-\!\!\bullet\ & \frac{1}{(1+sT)^{-2}} \\
&= 1 + 2T \cdot s + T^2 \cdot s^2.
\end{aligned} \tag{5.9}$$

We can exploit the linearity of the Inverse Laplace Transform and the correspondence [11, p. 262]

$$\frac{\mathrm{d}^n}{\mathrm{d}t^n}\delta(t) \circ\!\!-\!\!\bullet\ s^n \tag{5.10}$$

to retrieve the inverse

$$f^{-1}(t) = \delta(t) + 2T \cdot \frac{\mathrm{d}}{\mathrm{d}t}\delta(t) + T^2 \cdot \frac{\mathrm{d}^2}{\mathrm{d}t^2}\delta(t) = \left(1 + T\frac{\mathrm{d}}{\mathrm{d}t}\right)^2 \delta(t). \tag{5.11}$$

Furthermore, with the differentiation rule [12, p. 156], we get

$$\frac{\mathrm{d}^n}{\mathrm{d}t^n}\delta(t) * s(t) = \delta(t) * \frac{\mathrm{d}^n}{\mathrm{d}t^n}s(t) = \frac{\mathrm{d}^n}{\mathrm{d}t^n}s(t), \tag{5.12}$$

such that (5.6) can be simplified to

$$\begin{aligned} y(t) &\overset{(5.11)}{=} h_f(t) * \left[\left(1 + T\frac{\mathrm{d}}{\mathrm{d}t}\right)^2 \delta(t) * s(t)\right] \\ &\overset{(5.12)}{=} h_f(t) * \left[\left(1 + T\frac{\mathrm{d}}{\mathrm{d}t}\right)^2 s(t)\right], \end{aligned} \tag{5.13}$$

as stated in [10]. Note, that we changed the convolution order compared to (5.6), which is allowed due to the associativity of the convolution operator. The reasoning behind this is, that we know the source signal $s(t)$ exactly, as opposed to the AIR, because the simulation is not absolutely accurate. Hence, we convolve the inverse with the source function instead, to avoid intensifying the error of the AIR.

Now, we are able to calculate the system response for any arbitrary source function $s(t) \in \mathcal{C}^2$, like a piece of music, once we know the AIR. That means, we only need to invest once in simulating a point source with function $f(t)$, and can refer to (5.13) afterwards, which drastically reduces the computing time.

However, as we are dealing with discrete signals, we also need to perform the abovementioned convolution in the discrete domain. Suppose we have signals $s^{(k)}[n] = s^{(k)}(n/f_s)$ and $h_f[n] = h_f(n/f_s)$ with sample frequency $f_s \in \mathbb{R}_{>0}$. Here, $s^{(k)}(t)$ denotes the k-th derivative of $s(t)$. The discrete representation of (5.13) then becomes

$$y[n] = \frac{1}{f_s}\sum_{i=0}^{\infty} h_f[i] \cdot \left(s[n-i] + 2T \cdot s^{(1)}[n-i] + T^2 \cdot s^{(2)}[n-i]\right) \tag{5.14}$$

where $y[n]$ is the discrete output signal [11, p. 47].

We will discuss how to retrieve the AIR with a fixed sample frequency in the next section, as well as how to obtain the first and second derivatives of a discrete source signal in section 5.4.

5.2 Approximative impulse response

In chapter 4, it has been discussed how cell sizes determine the simulation timestep. Furthermore, a method for global and local grid refinement has been proposed, from which the support points for the Gauss-Lobatto quadrature [13, p. 431 f.] of degree O can be derived. With that, the simulation is discretized in both the temporal and spatial domain.

Let $P_R \in [0, S_x] \times [0, S_y] \times [0, S_z]$ be the receiver point where the AIR shall be measured. In order to obtain the quantites q exactly at P_R with timesteps $\Delta t_{\mathrm{AIR}} = 1/f_s$, one approach would be to adjust the grid points and Courant number $C \in (0, 1]$ accordingly, such that P_R becomes a support point in the domain and Δt_{AIR} is a multiple of $C \cdot \Delta t$.

Inserting a new grid point to establish a support point exactly at P_R can lead to infinitesimally small cell sizes (e.g. if the receiver point is arbitrarily close to the domain border) and thus to infinitesimally small timesteps as well. But even if we do not change the grid and use the support point which is closest to the actual receiver point P_R instead, it might be necessary to decrease the simulation timestep Δt excessively in order to obtain measurements at the right times.

For that, the loss of efficiency is indicated by the maximal required Courant number C. For ease of notation, we define

$$\beta := \frac{\Delta t_{\mathrm{AIR}}}{\Delta t} > 0. \tag{5.15}$$

The minimal number of timesteps between two AIR measurements is $n = \lceil \beta \rceil$. Thus,

$$C \cdot \Delta t = \frac{\Delta t_{\mathrm{AIR}}}{n}, \tag{5.16}$$

which bounds the Courant number to

$$0 < C = \frac{\Delta t_{\mathrm{AIR}}}{\Delta t} \cdot \frac{1}{n} = \frac{\beta}{\lceil \beta \rceil} \leq 1. \tag{5.17}$$

That is, in order to retrieve the AIR at an exact sample frequency of f_s, the simulation might need to be slowed down intentionally by the factor $1/C$, which can be very high, depending on β.

Due to these considerations, it is justifiable to instead interpolate the values both in time and space. This enables us to use the maximal timestep, i.e. $C = 1$, which is more efficient than the abovementioned approach. Here, the spatial interpolation will be based on Lagrange polynomials, whereas for the interpolation in time, we will utilize the Taylor expansion.

The first step is to determine a cell

$$\mathcal{C}_{i,j,k} = [(G_x)_i, (G_x)_{i+1}] \times [(G_y)_j, (G_y)_{j+1}] \times [(G_z)_k, (G_z)_{k+1}] \tag{5.18}$$

within the domain grid G, such that $P_R \in \mathcal{C}_{i,j,k}$. For that cell, the quantity matrix $Q^n_{i,j,k} \in \mathbb{R}^{O \times O \times O \times 4}$ is given at time $t_n = n \cdot \Delta t$ and contains the pressure and velocity values at O support points for each dimension. These support points span over an interval of $[0, 1]$, hence, the receiver point within the cell has to be projected onto that interval, resulting into its relative coordinates:

$$\xi = \frac{(P_R)_x - (G_x)_i}{(G_x)_{i+1} - (G_x)_i}, \tag{5.19a}$$

$$\eta = \frac{(P_R)_y - (G_y)_j}{(G_y)_{j+1} - (G_y)_j}, \tag{5.19b}$$

$$\zeta = \frac{(P_R)_z - (G_z)_k}{(G_z)_{k+1} - (G_z)_k}. \tag{5.19c}$$

The Lagrange polynomials [13, p. 334] are

$$L_m(p) = \prod_{n:n \neq m} \frac{p - p_n}{p_m - p_n}, \tag{5.20}$$

where $m, n \in [O]$ and p_n are the Gauss-Lobatto points in the interval $[0, 1]$. Due to the property

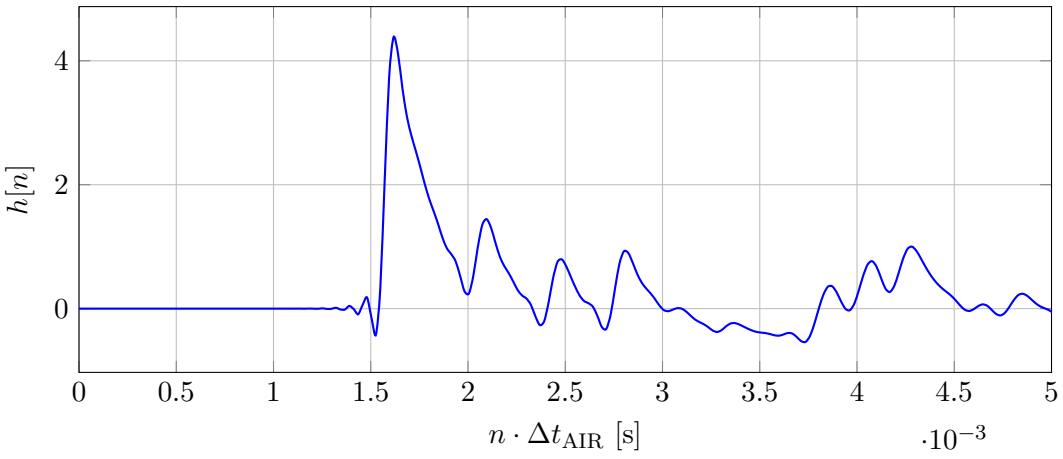


Figure 5.1: AIR for a domain of size $\vec{S} = (1, 1, 1)$ with sample frequency $f_s = 44100$ Hz, receiver point $P_R = (0.5, 0.5, 0.5)$, source point $P_S = (0.1, 0.2, 0.3)$, speed of sound $c_0 \approx 343$ m/s and convergence order $O = 4$. Impedance boundary conditions were imposed on all domain borders with reflection factor $C_R = 0.5$. The source function $f(t)$ has been simulated with $T = 0.0005$. Note, that the distance between receiver and source is $\|P_R - P_S\|_2 \approx 0.54$ m, such that the first incidence of the wave is correctly expected after $0.54/343 \approx 1.57 \cdot 10^{-3}$ seconds.

of Lagrange polynomials, that

$$L_m(p_n) = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{otherwise} \end{cases}, \tag{5.21}$$

a polynomial

$$q_l(Q, 0) = \sum_{m_1} \sum_{m_2} \sum_{m_3} Q_{m_1,m_2,m_3,l} \cdot L_{m_1}(\xi) \cdot L_{m_2}(\eta) \cdot L_{m_3}(\zeta) \tag{5.22}$$

can be constructed, which exactly interpolates the values given in the quantity matrix Q, where $m_1, m_2, m_3 \in [O]$ and $l \in [4]$. Here, $q_l(Q, t)$ denotes the quantity l, interpolated at time t and point (ξ, η, ζ) with given support values Q at time 0.

In order to form the Taylor expansion in the time-domain, we also need the derivatives of q_l with respect to the time. We do not want to investigate that further and instead rely on the implementation that has already been provided in the *LinA* code. So in the following, we consider the derivatives $\frac{\partial^n}{\partial t^n} Q$ given for $n \in [O]$ at the support points. Using the spatial interpolation from (5.22), the derivatives can be determined at the receiver point too, which gives

$$q_l(Q, t) = \sum_{n=1}^{O} \frac{t^n}{n!} q_l \left(\frac{\partial^n}{\partial t^n} Q, 0 \right). \tag{5.23}$$

Of course, t cannot be arbitrarily high, as stability is only guarenteed with a limited simulation timestep, i.e. $0 \leq t \leq \Delta t$ is required. Thus, the final definition for the AIR is

$$h_f[n] = q_1 \left(Q_{i,j,k}^{\lfloor n\beta \rfloor}, n\Delta t_{\text{AIR}} - \lfloor n\beta \rfloor \Delta t \right), \tag{5.24}$$

with the quantity $l = 1$ being the pressure. An example is shown in Figure 5.1

Note, that we are able to measure the AIR at multiple receiver points simultaneously. This could

for instance be used to mimic human ears that perceive acoustics in stereo, which is crucial for us to estimate the distance or angle to a source via triangulation. Thus, we hereby mention that a stereo signal certainly also contributes to a more realistic sound.

5.3 Acoustic properties

As mentioned at the beginning of this chapter, the auralization of specific environments can be accomplished using convolution reverberation. In doing so, the impulse response (or rather an approximation of such) is usually retrieved by recording the sound at a certain point in the environment with microphones. Our approach is to simulate the AIR instead, as described in the previous sections. This, however, requires us to specify acoustic parameters like the speed of sound or boundary conditions appropriately to imitate real-world acoustics. In this section, we will briefly mention the properties that have been used in our experiments.

5.3.1 Medium coefficients

As propagation medium, we used air at $20°C$. Depending on the literature, there exist slightly different coefficients K_0 and ρ_0. We chose the values in [4, p. 161, Table 13.3] with

$$K_0 \approx 140 \text{ kPa} = 140{,}000 \ \frac{\text{kg}}{\text{m} \cdot \text{s}^2}, \tag{5.25}$$

$$\rho_0 \approx 1.189 \ \frac{\text{kg}}{\text{m}^3}, \tag{5.26}$$

which yields the speed of sound

$$c_0 = \sqrt{\frac{K_0}{\rho_0}} = \sqrt{\frac{140{,}000 \text{ kg} \cdot \text{m}^3}{1.189 \text{ kg} \cdot \text{m} \cdot \text{s}^2}} \approx 343 \frac{\text{m}}{\text{s}} \tag{5.27}$$

and an impedance of

$$Z_0 = \rho_0 c_0 \approx 408 \frac{\text{kg}}{\text{m}^2 \cdot \text{s}}. \tag{5.28}$$

5.3.2 Boundary conditions

Besides the propagation medium, we also need to characterize the objects in the environment with which the sound waves interact. This depends on various circumstances, like the material and texture of the objects. For instance, a plain surface reflects waves rather sharply, whereas a rough surface scatters the waves, which is also referred to as *diffuse reflection* [1, sec. 11.1.3.2].

But also the frequency of the waves have an influence on how strong the reflection or absorption is. In the following table, some examplary reflection coefficients C_R from [1, p. 346, Table 11.2] are cited for different materials and frequencies:

Material	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz
Concrete	0.98	0.98	0.97	0.96	0.95	0.95
Carpet ($\approx$ 5 mm)	0.97	0.96	0.94	0.80	0.70	0.60
Window	0.88	0.92	0.94	0.95	0.95	0.95
Linoleum covering on a felt layer	0.98	0.95	0.90	0.85	0.93	0.95

It is apparent that some materials reflect waves more evenly across all frequency bands, and others vary strongly. This is certainly a critical limitation of our current system, as we are only able to simulate surfaces with a frequency-independent reflection coefficient, as described in section 3.3. In that regard, we will experiment with different parameters until the results are adequate enough.

5.4 Implementation

After having described the theoretical foundations for convolution reverberation, its implementation is defined by the following steps:

1. Define the domain that shall be simulated, e.g. its size, resolution and boundaries with the properties given in section 5.3.
2. Specify coordinates for the point source P_S and receiver P_R. Where appropriate, refine the domain locally, e.g. at the point source (see section 6.2.2 for a considered decision).
3. Choose (5.7) as source function and specify an elaborate value for the parameter T. In section 6.2.3, the choice of T will be argued in detail. To simulate this function, we yet have to provide its antiderivative for the code, which can be determined using partial integration:

$$\begin{aligned}\int f(t)\,\mathrm{d}t &= \int \frac{t}{T^2} e^{-t/T}\,\mathrm{d}t \\ &= \left[-\frac{t}{T}e^{-t/T}\right] - \int -\frac{1}{T}e^{-t/T}\,\mathrm{d}t \\ &= -e^{-t/T}\left(1+\frac{t}{T}\right) + c. \end{aligned} \tag{5.29}$$

4. Run the simulation and measure the AIR according to (5.24).
5. Numerically convolve the AIR with the desired source signal as described in (5.14). We use cubic splines as interpolation for the discrete source signal in order to obtain its first and second derivatives. This can be, for instance, easily implemented with NumPy and SciPy:

```
1  from scipy.interpolate import splrep, spalde
2  from scipy.io.wavefile import read
3  from numpy import linspace, convolve
4
5  fs, source = read(audioPath)     # sample frequency, source signal
6  N = len(source)                  # number of sample points
7  t = linspace(0.0, (N-1)/fs, N)   # sample time points
8
9  tck = splrep(t, source, k=3)     # cubic spline interpolation of source,
10                                  # twice differentiable at time points
11 deriv = spalde(t, tck)           # derivatives of polynomial at time points
12 deconvSource = [                 # deconvolved source signal
13     deriv[i][0] + 2*T*deriv[i][1] + T**2*deriv[i][2] for i in range(N)
14 ]
15
16 # discrete convolution, 'response' = AIR
17 output = convolve(response, deconvSource) / fs
```

6 Validation

The approach described in the previous chapter involves many numerical solutions, that make the convolution reverberation prone to errors. Especially the AIR certainly contributes to an inaccurate output signal, if itself deviates from the expected response. In order to quantify that error, we will derive the analytical solution for the AIR in a simple setting and compare it with the actual measurements obtained through the simulation.

6.1 Analytical solution using Green's function

Let $s : \mathbb{R} \to \mathbb{R}$ be an arbitrary function for the pressure perturbation that is being generated from a single point source $\hat{s} : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ located at $\boldsymbol{x} = (0, 0, 0)$, i.e.

$$\hat{s}(\boldsymbol{x}, t) := \delta(\boldsymbol{x}) s(t). \tag{6.1}$$

Using this as the pressure quantity in the three-dimensional PDE of (2.2), which follows from the conservation laws [14, sec. 1.1], gives

$$\hat{s}(\boldsymbol{x}, t) = p_t + K_0(u_x + v_y + w_z), \tag{6.2a}$$
$$0 = u_t + (1/\rho_0)p_x, \tag{6.2b}$$
$$0 = v_t + (1/\rho_0)p_y, \tag{6.2c}$$
$$0 = w_t + (1/\rho_0)p_z. \tag{6.2d}$$

Differentiating with respect to t gives

$$\frac{\partial}{\partial t}\hat{s}(\boldsymbol{x}, t) = \delta(\boldsymbol{x}) s'(t) = p_{tt} + K_0(u_{xt} + v_{yt} + w_{zt}), \tag{6.3}$$

where

$$u_{xt} = (u_t)_x = -(1/\rho_0)p_{xx}, \tag{6.4a}$$
$$v_{yt} = (v_t)_y = -(1/\rho_0)p_{yy}, \tag{6.4b}$$
$$w_{zt} = (w_t)_z = -(1/\rho_0)p_{zz}. \tag{6.4c}$$

Plugging equations (6.4a-6.4c) into (6.3) yields

$$p_{tt} - \frac{K_0}{\rho_0}\left(p_{xx} + p_{yy} + p_{zz}\right) = \left(\partial_t^2 - c_0^2\Delta\right) p = \delta(\boldsymbol{x}) s'(t), \tag{6.5}$$

where $\Delta = \partial_x^2 + \partial_y^2 + \partial_z^2$ denotes the Laplacian operator. With the differential operator for the 3D wave equation given as

$$L = \partial_t^2 - c_0^2\Delta, \tag{6.6}$$

equations (6.2a-6.2d) can be rewritten as a PDE of second order:

$$Lp(\boldsymbol{x}, t) = \delta(\boldsymbol{x}) s'(t). \tag{6.7}$$

These types of PDEs can be solved using Green's functions [14, sec. 3.1]. The basic idea is to find a function G for a given linear differential operator L, such that

$$LG(\boldsymbol{x}, t) = \delta(\boldsymbol{x})\delta(t). \tag{6.8}$$

Then, any equation of the form

$$Ly(\boldsymbol{x}, t) = f(\boldsymbol{x}, t) \tag{6.9}$$

can be transformed into

$$Ly(\boldsymbol{x}, t) = L[G(\boldsymbol{x}, t) * f(\boldsymbol{x}, t)] = [LG(\boldsymbol{x}, t)] * f(\boldsymbol{x}, t) \overset{(6.8)}{=} f(\boldsymbol{x}, t) \tag{6.10}$$

to show that

$$y(\boldsymbol{x}, t) = G(\boldsymbol{x}, t) * f(\boldsymbol{x}, t). \tag{6.11}$$

The Green's function for (6.6) is given as

$$G(\boldsymbol{x}, t) = \frac{1}{4\pi r c_0^2} \delta\left(t - \frac{r}{c_0}\right), \tag{6.12}$$

where $r = \|\boldsymbol{x}\|_2$, i.e. the distance between receiver and source point [14, p. 141]. In the following, we can substitute r for $\boldsymbol{x}$ in the other functions as well. With

$$\int_{-\infty}^{\infty} \delta(x - x_0) f(x)\, \mathrm{d}x = f(x_0) \tag{6.13}$$

[11, p. 73] and (6.11), the analytical solution for p in (6.7) then becomes

$$\begin{aligned} p(r, t) = G(r, t) * \delta(r) s'(t) &= \int_0^{\infty} \int_0^{\infty} \frac{\delta(\tau - \xi/c_0)}{4\pi\xi c_0^2} \delta(r - \xi) s'(t - \tau)\, \mathrm{d}\xi\, \mathrm{d}\tau \\ &= \int_0^{\infty} \frac{\delta(\tau - r/c_0)}{4\pi r c_0^2} s'(t - \tau)\, \mathrm{d}\tau \\ &= H\left(t - r/c_0\right) \cdot \frac{1}{4\pi r c_0^2} s'\left(t - r/c_0\right). \end{aligned} \tag{6.14}$$

Notice, that this equation follows our intuition of how acoustic waves propagate. On one hand, the sound intensity decreases proportional to r. On the other hand, the waves arrive at the receiver with a delay of r/c_0, which is the time the waves with speed c_0 need to travel r meters. Moreover, the waves propagate equally in each direction.

In conclusion, the analytical solution (6.14) is known for any source function for which the derivative exists. For instance, the approximation of the Dirac delta impulse (5.7) has the derivative

$$s'(t) = f'(t) = e^{-t/T} \left(\frac{1}{T^2} - \frac{t}{T^3}\right). \tag{6.15}$$

For the analysis in the next section, we will evaluate the error terms with this function, as we are foremost interested in the accuracy for convolution reverberation.

6.2 Convergence tests

In this section, we will present various experiments to assess the impact of different parameters on the numerical error. As the equations derived in the previous section describe the analytical solution for an undisturbed spherical wave, we will design the experiments in such a way, that the wave generated by a point source will never interact with any boundary. We will especially ensure that the domain is always big enough, such that the waves do not hit the borders, in order to avoid any undesirable interference.

If not stated otherwise, each experiment has been executed with the following settings:

- Domain size: $\vec{S} = (3, 3, 3)$
- Source point: $P_S = (1.5, 1.5, 1.5)$
- Speed of sound: $c_0 \approx 343$ m/s (according to section 5.3.1)
- Simulation time: $t_{\text{sim}} = 0.0036$, thus the wave propagates ≈ 1.235 meters
- Impulse parameter: $T = 0.001$
- Convergence order: $O = 4$
- Cluster: CoolMUC 3 (mpp3)
- Number of cores: 64

6.2.1 Uniform refinement

First, we are going to analyze the impact of the uniform domain resolution on the pressure error. Moreover, the two approaches mentioned in section 4.4, i.e. a constant uniform resolution and local refinement at the point source will be compared in section 6.2.2.

The error is calculated as follows: We measure the AIR, according to section 5.2, at ten randomly generated points within the domain (for comparability, we always use the same ten points in each experiment). Those measurement points were intentionally generated in such a way, that they are at least 0.1 meters apart from the source. The reason for this is, that the point source produces strong local discontinuities, which also yields to visible artefacts in the simulations. Furthermore, the analytical solution (6.14) diverges to infintity for $r = 0$, which makes the area in the close proximity of the point source even more sensitive to errors. Hence, and also because we are foremost interested in the accuracy after longer distances, we do not consider points with a distance smaller than 0.1 meters to the source when evaluating the numerical error.

Now, with an AIR sample frequency of $f_s = 50000$ Hz, we get $t_{\text{sim}} \cdot f_s = 180$ pressure measurements for each point, i.e. time series $\tilde{y}_i[n]$ with $i \in [10]$ and $n \in [180]$. The absolute error e_a is then defined as the L2 norm of all deviations from the analytical solutions $y_i[n]$:

$$e_a := \sqrt{\sum_{i=1}^{10} \sum_{n=1}^{180} (\tilde{y}_i[n] - y_i[n])^2}. \tag{6.16}$$

Figure 6.1 shows, that with increasing global resolution R, the error decreases expectedly. In consequence, the measured AIR is more accurate, which can be seen in Figure 6.2, where an AIR is displayed for various resolutions.

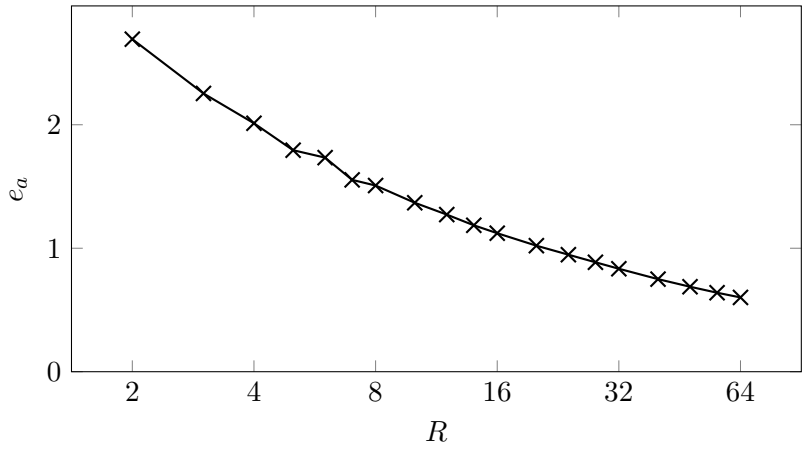


Figure 6.1: Absolute pressure errors depending on the global resolution R for uniform refinement.

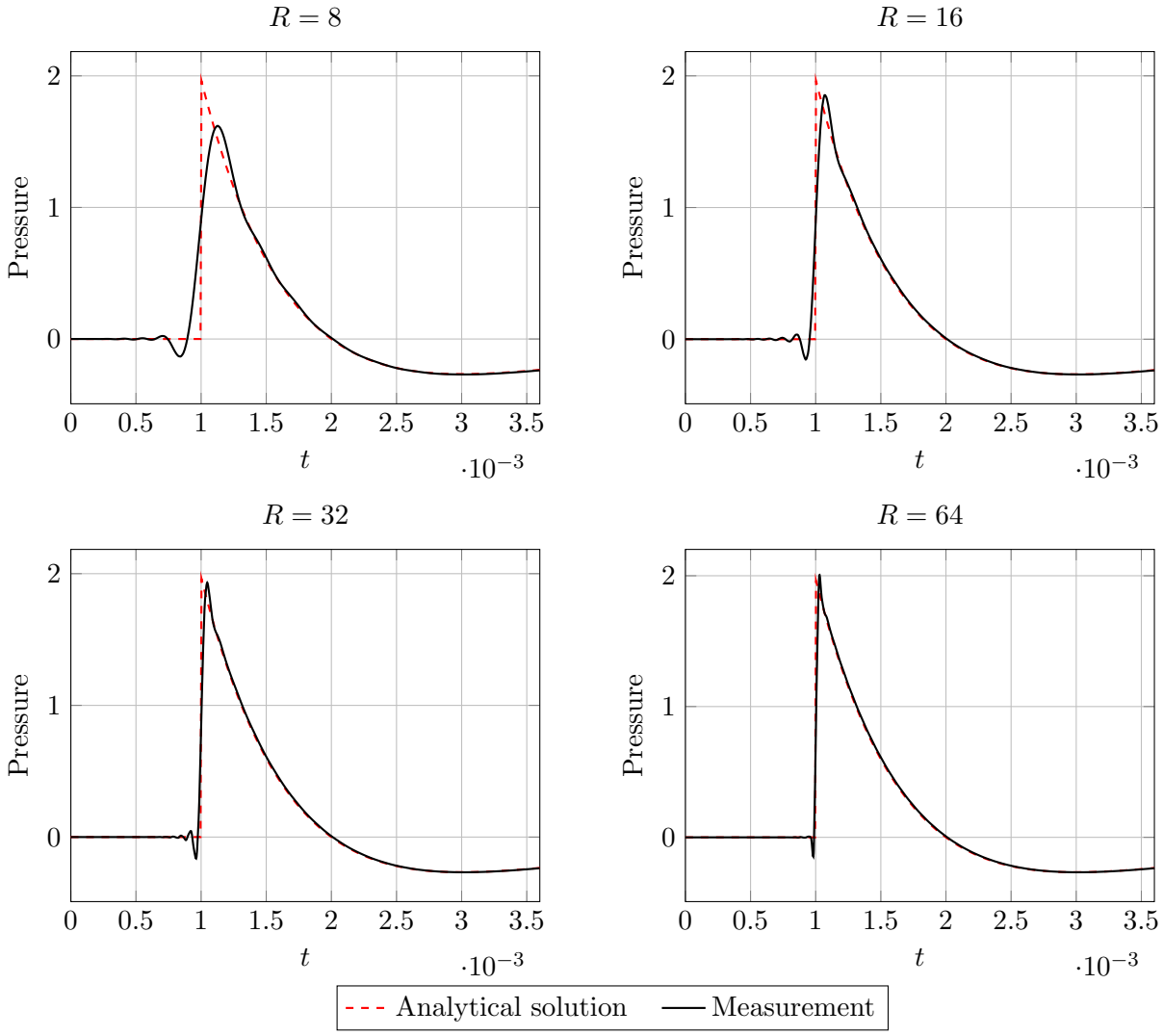
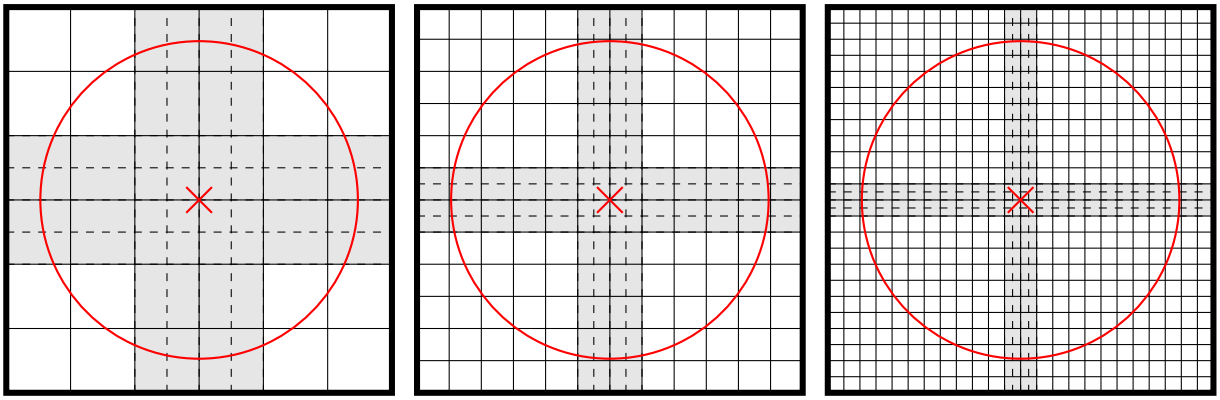


Figure 6.2: AIR measurements compared to the analytical solution for different R. The receiver point has been chosen to be located at $P_R = (1.698, 1.698, 1.698)$, such that the distance to the source is $\|P_R - P_S\|_2 = \sqrt{3 \cdot 0.198^2} \approx 0.343$ meters. Hence, the wave arrives at the receiver after $\approx 0.343/343 = 1 \cdot 10^{-3}$ seconds.



(a) $R = 2$ (b) $R = 4$ (c) $R = 8$

Figure 6.3: Illustration of the domain grid with size $\vec{S} = (3, 3, 3)$. The locally refined source point (indicated by the red cross) is located in the center of the domain at $P_S = (1.5, 1.5, 1.5)$, where $\nu = 2$ holds for each of the above shown grids. These illustrations show a two-dimensional slice of the domain at its center. The red circle displays the wave front at the end of the simulations after $t_{\text{sim}} = 0.0036$ seconds. Note, that the volume of the locally refined parts of the domain (here indicated by the gray areas) decreases, the higher R is. This is because the source point can already be isolated better with higher global resolutions.

More specifically, we can quantify to what extent higher resolutions improve the simulation accuracy. For $R = 2$ and $R = 64$, the calculated errors were about 2.693 and 0.602 respectively, which constitutes a convergence rate of

$$k = \frac{\log(2.693/0.602)}{\log(64/2)} \approx 0.432. \tag{6.17}$$

This means, that the error gets reduced by an average of

$$2^k \approx 1.349 \tag{6.18}$$

when the resolution is doubled. However, halving the cell sizes also leads to a halved simulation timestep (4.2), as well as eight times more cells. Hence, we expect about 16 times longer computing times, just to enhance the accuracy by the factor (6.18). The actual computing times for the uniform refinement shown in Figure 6.4 approve the same conclusion.

6.2.2 Local refinement at point source

In the second experiment, we assess to which degree a higher local resolution at the point source may improve the results. The experiments were designed in such a way, that the domain is first uniformly refined with a global resolution of R. Then, the slices that contain the source point P_S have been refined locally with a resolution $R_{P_S} > R$, as described in section 4.4. We introduce a new variable

$$\nu := R_{P_S}/R, \tag{6.19}$$

which expresses how many times the local resolution around the source is higher than the global resolution. To better imagine the experiments, the refined domain is examplarily illustrated in Figure 6.3 for different R.

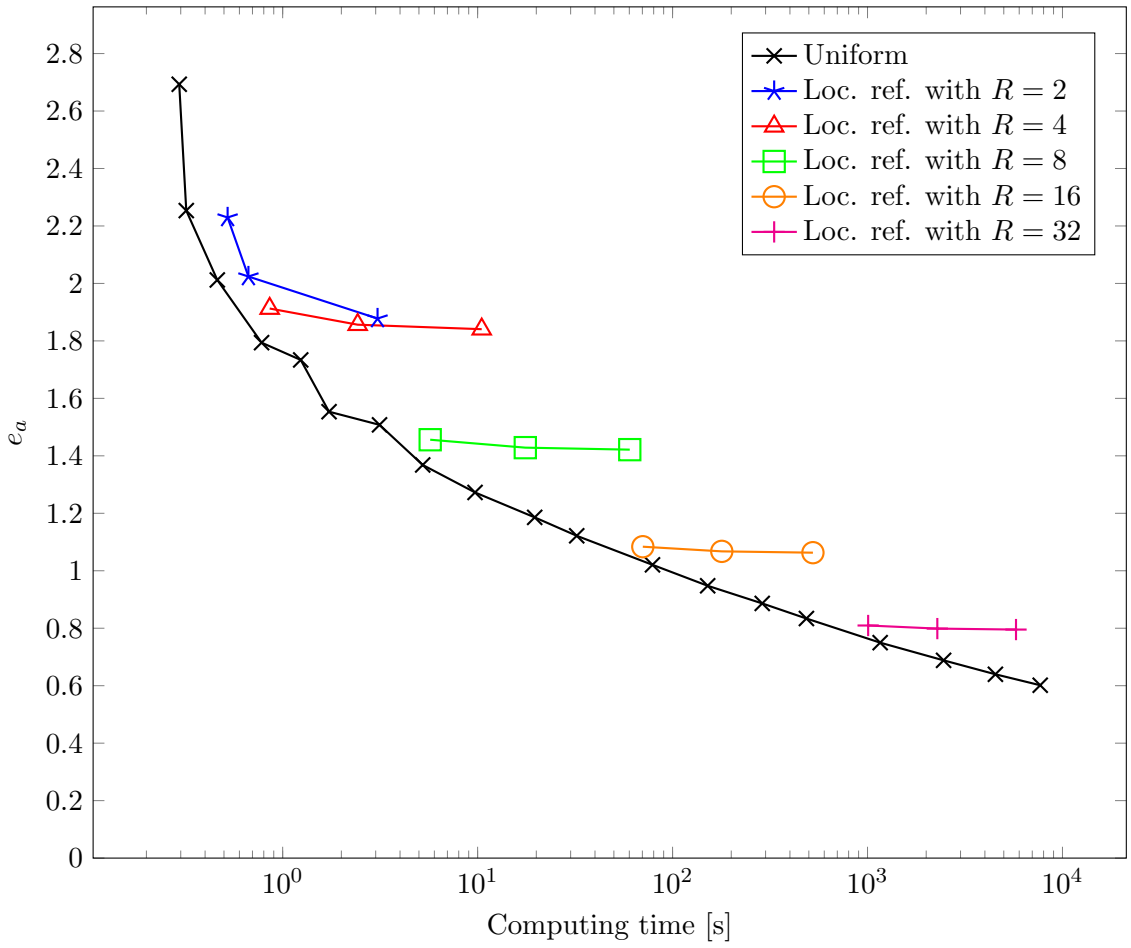


Figure 6.4: Computing times compared to the absolute pressure errors for both uniform and local refinement at the point source. Here, the plot points for all local refinement experiments indicate a change of ν to 2, 4 and 8 from the leftmost mark to the rightmost mark. As expected, by increasing the uniform resolution R, the computing time increases inversely proportional to the error. With local refinement, however, the error is virtually unchanged, while the computing time increases noticeably. Note, that the best results for local refinement can be observed with $R = 2$. But this is most likely due to the refined parts covering a large fraction of the domain and especially the spherical wave (see Figure 6.3).

The general conclusion based on the results in Figure 6.4 is, that local refinement does not contribute to a noticeably better accuracy. On the contrary, due to having significantly smaller cells, the timestep decreases correspondingly, which leads to longer computing times and thus a worse efficiency. A local timestepping (LTS) approach proposed in [9] could theoretically compensate this very well, as the fraction of cells that are locally refined is typically small, especially for higher global resolutions R (see Figure 6.3). However, besides the high computing times, the accuracy does not improve reasonably, such that even with LTS enabled, local refinement would not have outperformed the uniform method regarding the overall efficiency.

This is somewhat surprising, as we expected the error to be generated predominantely in the cell on which the source term is applied, and that the error is then disseminated across the whole domain. We therefore assumed that a locally reduced error would correspondingly improve the accuracy in the other cells, despite them having a lower resolution. Based on the results, however, this conclusion cannot be made. Thus, it is highly recommendable to always operate on a uniform grid with no further local refinement.

6.2.3 Strength T of the approximative impulse

The choice of T in (5.7) noticeably effects the absolute error, as Figure 6.5 shows. With increasing T, the error decreases continually. This may seem surprising; however, it is important to note that we are effectively using different source terms, where the analytical solutions for higher T consist of significantly smaller values (see Figure 6.6). Thus, it is more appropriate to compare the relative errors instead, which we define as

$$e_r := \sqrt{\sum_{i=1}^{10}\sum_{n=1}^{180}\gamma_i[n]^2}, \tag{6.20}$$

where

$$\gamma_i[n] := \begin{cases} \frac{\tilde{y}_i[n]-y_i[n]}{y_i[n]} & \text{if } y_i[n] \neq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{6.21}$$

Note, that the analytical solution (6.14) is 0 for all $t < r/c_0$. This is not really optimal, as incorrect signals prior to the wave incidence after r/c_0 seconds are completely neglected in (6.20). However, we can assume that those inaccuracies are similarly manifested in the measured signal after the incidence, such that the above defined relative error is still sufficiently expressive. As a consequence for all results presented in this section, we again randomly generated 10 evaluation points, but this time with the additional criterion of them having a maximal distance of $c_0 \cdot t_{\text{sim}} \approx 343 \cdot 0.0036 = 1.2348$ meters to the source. This is to ensure that there exists no evaluation point for which $y_i[n]$ is just 0 during the whole simulation, i.e. for all n.

For the relative errors, illustrated in Figure 6.5, we also observe a steady decline with increasing T, independent of the used resolutions. For $T \geq 0.003$, however, the relative errors are not changing noticeably anymore. This suggests using a value of 0.003 or above for T.

But the error is not the only quantity that should be taken into consideration. With different T, the slope of the function and its antiderivative vary as well, as can be seen in Figure 6.7. The approximative impulse $f(t)$ is designed to have its peak at $t = T$ with a value of $f(T) = (T \cdot e)^{-1}$. Hence, the smaller T is, the stronger and earlier the peak occurs. Looking at the corresponding antiderivatives, one observes that the impulse decays faster, the smaller T is.

This property is important for the choice of an adequate simulation time. Consider the fraction

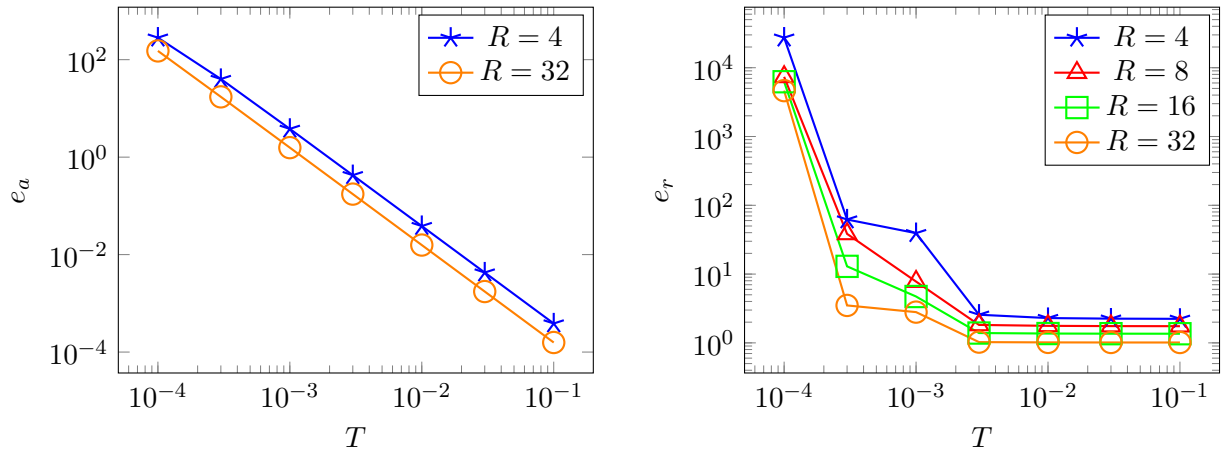


Figure 6.5: Absolute (left) and relative (right) pressure errors for different global resolutions depending on the coefficient T.

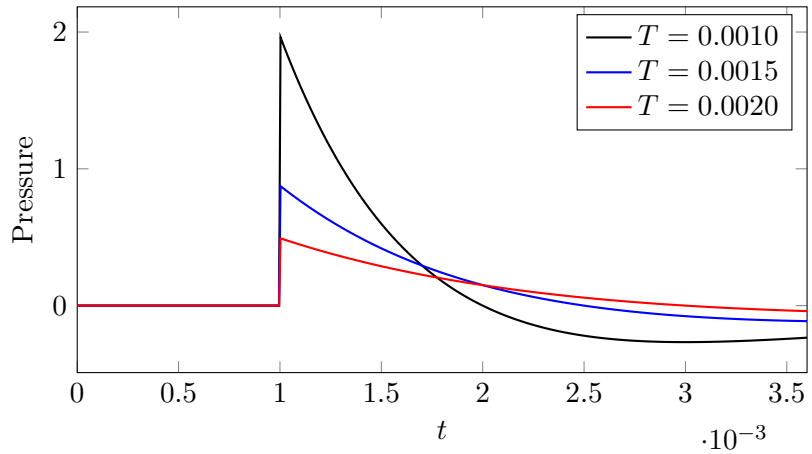


Figure 6.6: Analytical solutions for the pressure magnitude of the AIR with different T. The distance between receiver and source is ≈ 0.343 meters.

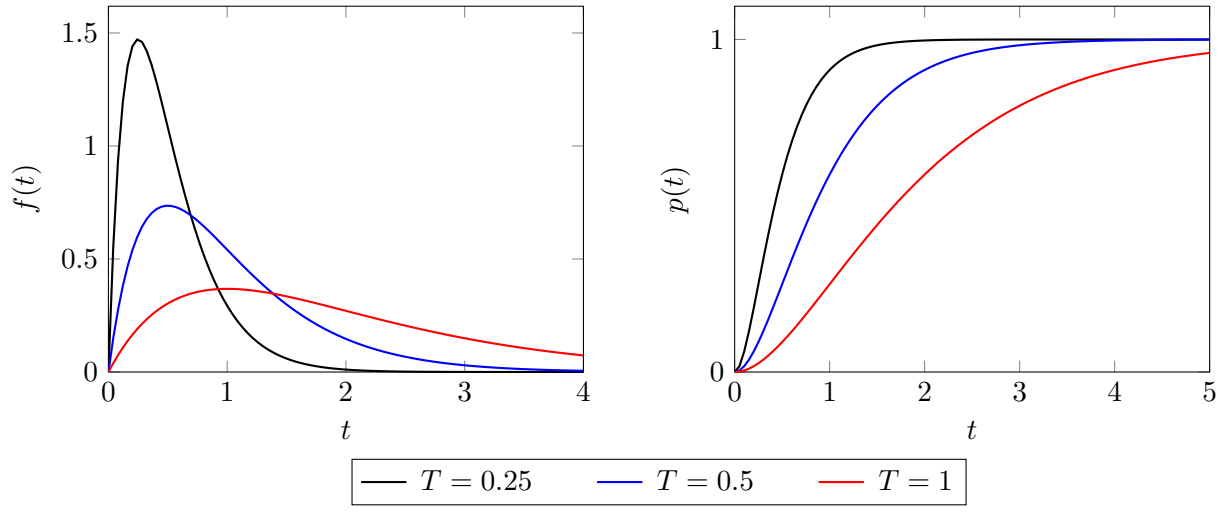


Figure 6.7: Approximative impulse (left) and its antiderivative (right) with different values for the parameter T.

$p(t)$ of the function that is simulated after t seconds, i.e.

$$p(t) := \int_0^t f(\tau)\,\mathrm{d}\tau \overset{(5.29)}{=} 1 - e^{-t/T}\left(1 + \frac{t}{T}\right). \tag{6.22}$$

For a fixed $p \in [0, 1)$, the required simulation time t increases linearily with T. That is, there exists a coefficient $\kappa(p) \in \mathbb{R}_{\geq 0}$, such that

$$p = p(t) \iff t = \kappa(p) \cdot T. \tag{6.23}$$

In the following table, some values for those linear coefficients are presented approximately:

p	0.95	0.9	0.5	0.1	0.01
$\kappa(p)$	4.744	3.890	1.678	0.532	0.149

To better understand what this means in practice, consider the experiments once with $T_1 = 0.0003$ and once with $T_2 = 0.1$. After $t_{\text{sim}} = 0.0036$ seconds, the fractions of the radiated impulse differ significantly with $p_1 = 99.992\%$ and $p_2 = 0.063\%$ respectively. According to (6.23), the simulation time for the weaker impulse with T_2 would have to be $T_2/T_1 = 333.\bar{3}$ times longer than with T_1, in order to radiate the same amount p_1.

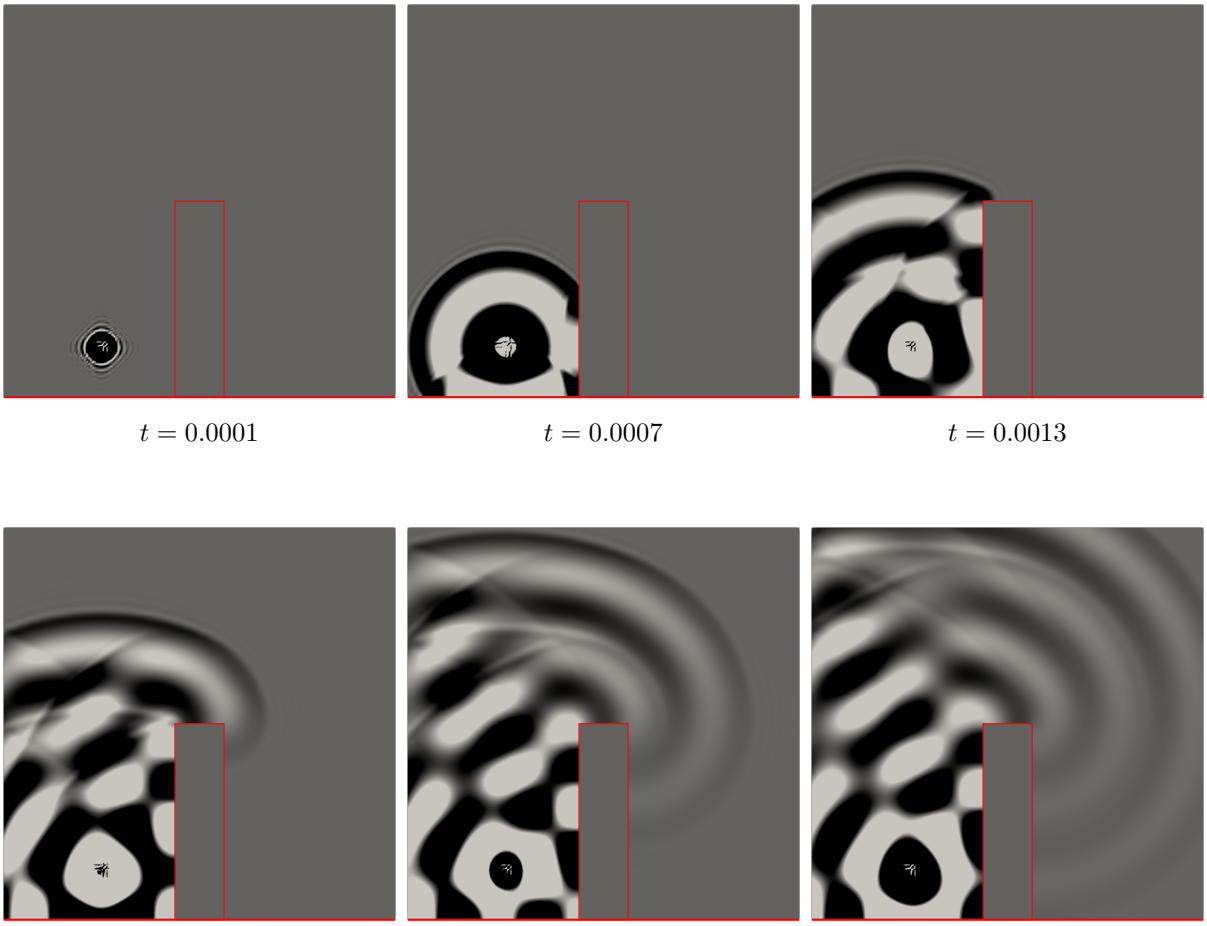
Consequently, we on one hand generally prefer a small T in order to shorten the simulation time, but on the other hand, a high T (at least greater than 0.003) should be used for a proper accuracy. In conclusion, we thus recommend to use $T \approx 0.003$ for the convolution reverberation. Note, that this suggestion is based on the results for the here presented experiment, i.e. the propagation of an undisturbed spherical wave with no boundary interactions. Hence, depending on the individual circumstances, e.g. the experiment, computational resources, or the required quality of the result, another value for T might be more appropriate.

7 Some experiments

In the following, some examplary simulations are presented, which showcase various acoustic effects, as well as some functionalities of our system. All experiments were simulated in 3D, but for better visibility, we only show a two-dimensional slice for most of them. In those cases, we omit the z-coordinates in the descriptions, and do hereby mention that the shown slices are always aligned with the z-coordinate of the source points. Furthermore, the acoustic properties are the exact same as stated in section 5.3.1 for all experiments. Green lines indicate absorbing boundaries, whereas yellow and red lines represent partially and totally reflecting boundaries. If not marked otherwise, all outer domain borders are absorbing. In all experiments, the domain has been uniformly refined with a global resolution of $R \geq 48$.

The first three experiments show diffraction effects in different environments. In Figure 7.1, the source signal gets diffracted around the corners of a wall. In Figure 7.2, a signal is emitted within a small pipe, that has small holes in it, similar to a flute. And in Figure 7.3, a similar experiment as in Figure 4.2 is presented, but in 3D.

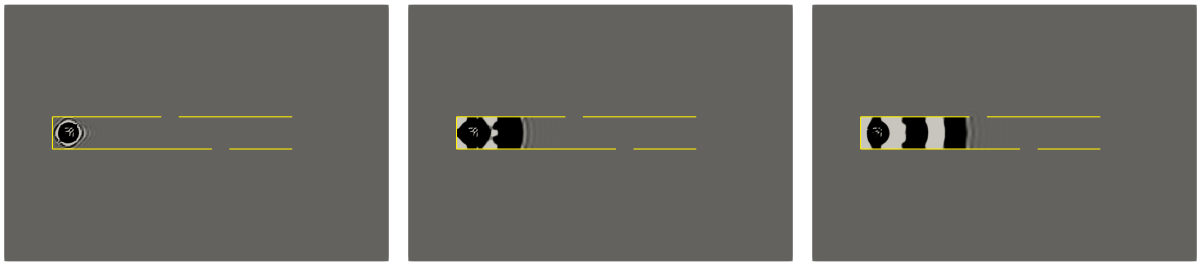
Another effect, that can be observed in Figure 7.4, is interference, which occurs when multiple waves overlap. This experiment showcases the ability to simulate multiple, individual source terms.



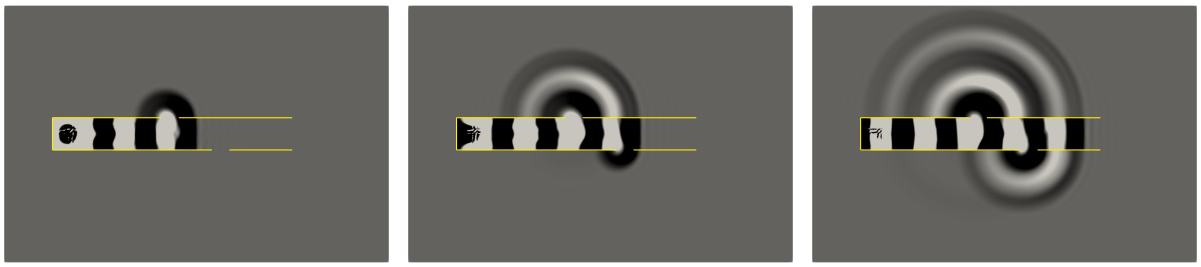
$t = 0.0001$ $t = 0.0007$ $t = 0.0013$

$t = 0.0019$ $t = 0.0025$ $t = 0.0031$

Figure 7.1: Simulation of an acoustic signal in an open environment with a totally reflecting ground and wall at the center. The domain has a size of $\vec{S} = (1, 1)$ and the wall is 0.5 meters tall and 0.125 meters wide. The point source is located left from the wall at $P_S = (0.25, 0.125)$ and emits sine waves with frequency 2000 Hz. It can be seen, that the source signal gets diffracted around the upper edges of the wall, such that it can still be received on the other side of the wall, as we are expecting it. Note, that all domain borders, except the ground, are absorbing. That is, with a geometric approach alone, as described in the introductory chapter 1, a receiver point right from the wall could not be backtraced to the source point by reflections only.



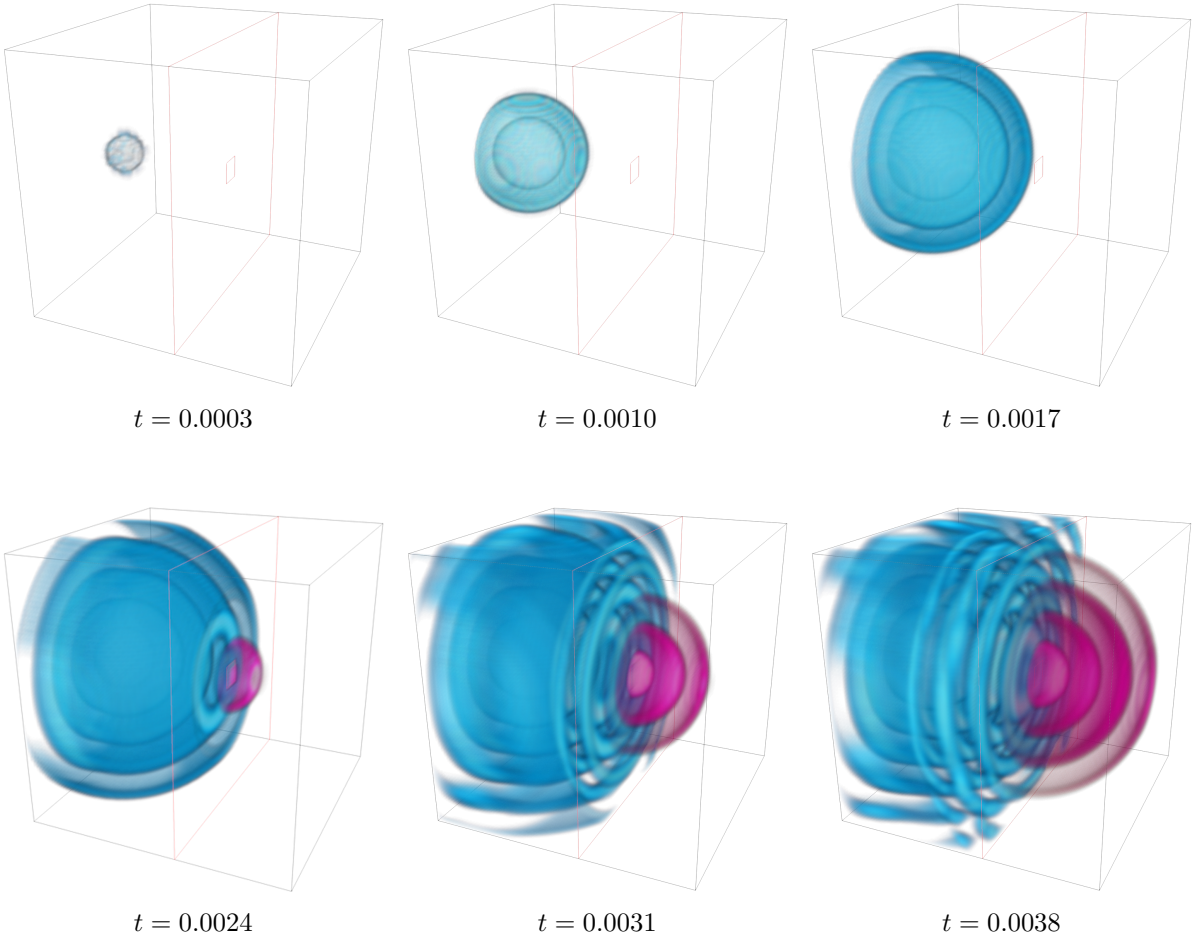
$t = 0.00001$ $t = 0.00055$ $t = 0.00100$



$t = 0.00145$ $t = 0.00190$ $t = 0.00235$

$t = 0.00280$ $t = 0.00325$ $t = 0.00370$

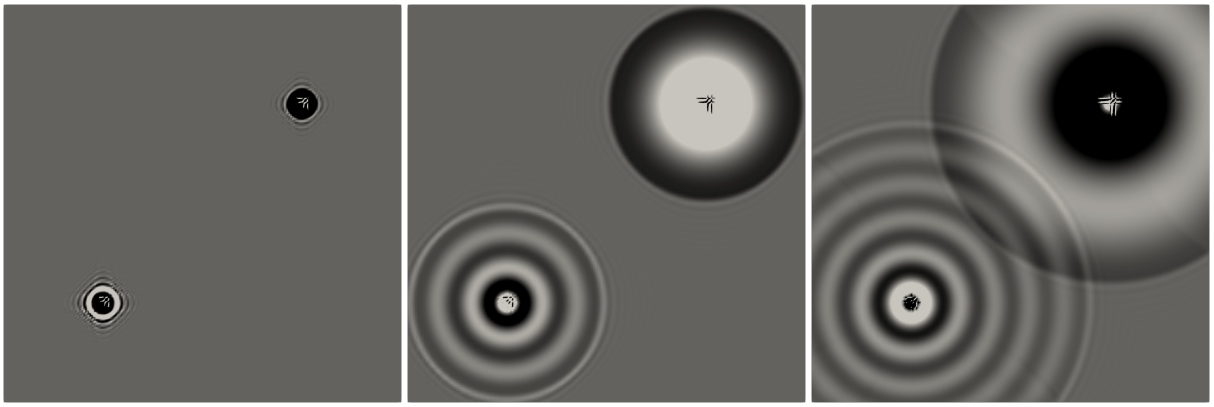
Figure 7.2: Simulation of an acoustic signal within an onesidedly opened pipe. Note, that, contrary to all previously presented experiments, the domain is not cubical, but has size $\vec{S} = (1.5, 1)$. The pipe (indicated by the yellow lines) has corner points $(0.1875, 0.4375)$ and $(1.125, 0.5625)$. Moreover, it has small holes of width 0.0625 meters, once in the top of the pipe at $x = 0.625$ (left corner), and once in the bottom at $x = 0.8125$. Partially reflecting boundary conditions have been imposed on the whole pipe with reflection factor $C_R = 0.9$.



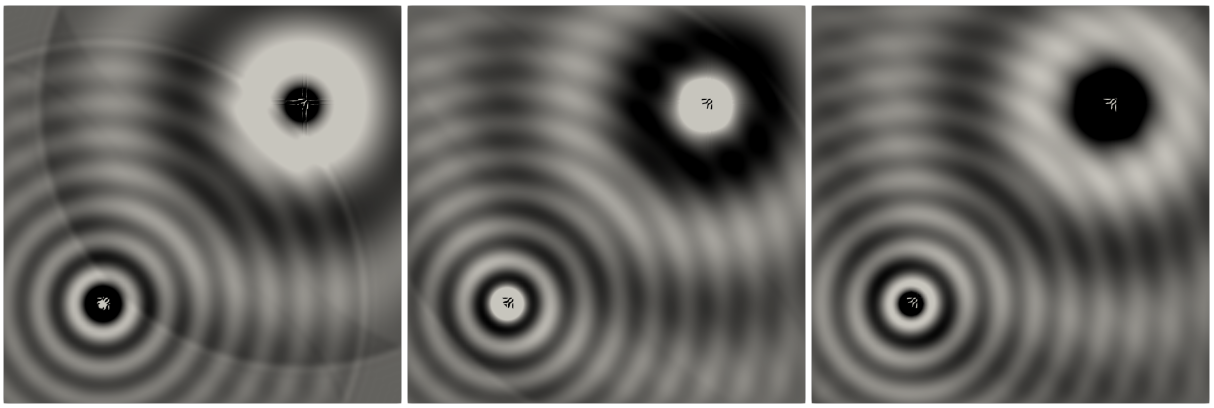
$t = 0.0003$ $t = 0.0010$ $t = 0.0017$

$t = 0.0024$ $t = 0.0031$ $t = 0.0038$

Figure 7.3: Three-dimensional simulation of a sinusoidal source signal with frequency 2000 Hz, that gets diffracted due to a small hole in a wall. The point source is located at $P_S = (0.2, 0.75, 0.75)$ in a domain of size $\vec{S} = (1.5, 1.5, 1.5)$. The totally reflecting boundary lies at $x = 0.875$ and has a square-shaped hole in it with a width of 0.125 meters (indicated by the red slice in the center of the wall). To better distinguish the waves coming from the original source (blue) and the new spherical waves that arise at the aperture (purple), we colored the areas of the domain that are segregated by the wall differently. Note further, that for reasons of visibility, the purple color map covers a more narrow range of pressure values, as the waves behind the aperture are very weak. Thus, those waves may appear stronger than they actually are. For a better comparability of the pressure magnitudes, refer to Figure 4.2, where a similar experiment is shown in 2D.



$t = 0.0001$ $t = 0.0007$ $t = 0.0013$



$t = 0.0019$ $t = 0.0025$ $t = 0.0031$

Figure 7.4: Simulation of two simultaneous point sources at $\vec{P}_{S1} = (0.25, 0.25)$ and $\vec{P}_{S2} = (0.75, 0.75)$ in a domain of size $\vec{S} = (1, 1)$. The former signal is a cosine with frequency 4000 Hz and a maximal amplitude of 0.1, whereas the latter source emits a sinusoidal signal with a frequency of 1000 Hz and a maximal amplitude of 1. In fact, the simulation system has been extended to allow for arbitrarily many individual point sources. This could be, for instance, useful for convolution reverberation, if it is known beforehand that the same source signal is always emitted at various points in the environment simultaneously, like in concert halls with multiple loudspeakers. In doing so, it can be avoided to measure the AIR for each source point separately. However, this would of course require to use the exact same signal (5.7), including the same T, for all sources.

8 Conclusion

In the course of this thesis, the existing implementation of the ADER-DG method (*LinA*) has been evolved to a functional auralization system. Especially the ability to make diffraction effects audible is a substantial advantage compared to other methods.

In order to construct virtual environments, such that the propagation of acoustic waves can be simulated reasonably realistic, we proposed the implementation of absorbing, as well as partially and totally reflecting boundary conditions. Moreover, our system supports the specification of planar boundaries (orthogonal to one dimension), which can be placed anywhere within the domain. In doing so, the domain grid is automatically adjusted in such a way, that the resulting global timestep is as optimal as possible. Overall, we were able to present a variety of experiments, where diffraction effects could be observed in differently designed environments.

Our system further enables the interpolation of the simulated signal in space-time, such that it can be measured not only at arbitrary points in the domain, but also at any desired sample frequency. As the acoustic domain can be considered as a LTI system, this allows us to retrieve an impulse response which represents the reverberation in the simulated environment.

With that, the simulation can be accelerated in two respects: On one hand, we only need to simulate the approximative impulse (5.7) instead of the whole source signal. An appropriate simulation time for that impulse depends predominantely on the reverberation time of the environment, which we can generally assume to be feasibly small, e.g. in the range of some hundred milliseconds [1, sec. 11.4]. Consequently, the computing times are kept acceptable, independent of the duration of the actual sound source. On the other hand, once the approximative impulse response is known, any simulation can effectively be reduced to a convolution operation, which can be done in real-time. This is extraordinarily more efficient than simulating each source signal individually, using the computationally expensive ADER-DG method.

Based on these considerations, our method can be considered as a relatively comprehensive and efficient auralization system, given that it also incorporates diffraction effects in a highly accurate manner. In conclusion, this suggests that our method is worthwhile to be further advanced, for instance regarding more realistic boundary conditions or more complexly shaped objects and environments, in order to become even more functional.

Bibliography

[1] G. Müller and M. Möser, *Taschenbuch der Technischen Akustik*. Springer, 2004.

[2] S. J. Orfanidis, *Introduction to signal processing*. Upper Saddle River, NJ, Prentice Hall, 1996.

[3] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics, Cambridge University Press, 2002.

[4] I. Veit, *Technische Akustik: Grundlagen der physikalischen, physiologischen und Elektroakustik*. Kamprath-Reihe Technik, Vogel, 1988.

[5] J.-P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of Computational Physics*, vol. 114, no. 2, pp. 185 – 200, 1994.

[6] M. Dumbser and M. Käser, "An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes – ii. the three-dimensional isotropic case," *Geophysical Journal International*, vol. 167, no. 1, pp. 319–336, 2006.

[7] P. Arminjon and A. St-Cyr, "New space staggered and time interleaved 2nd order finite volume methods," in *Hyperbolic Problems: Theory, Numerics, Applications* (T. Hou and E. Tadmor, eds.), pp. 295–304, Springer, Berlin Heidelberg, 2003.

[8] M. Dumbser, D. S. Balsara, E. F. Toro, and C.-D. Munz, "A unified framework for the construction of one-step finite volume and discontinuous galerkin schemes on unstructured meshes," *Journal of Computational Physics*, vol. 227, no. 18, pp. 8209–8253, 2008.

[9] M. Dumbser, M. Käser, and E. F. Toro, "An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes – v. local time stepping and p-adaptivity," *Geophysical Journal International*, vol. 171, no. 2, pp. 695–717, 2007.

[10] S. M. Day, "An informal appendix to peer reports 1a01 and 1a02." `https://steveday.sdsu.edu/BASINS/PEER_REPORT_APPENDIX.pdf`, 2011. Accessed: 07 September 2019.

[11] D. G. Manolakis and V. K. Ingle, *Applied Digital Signal Processing*. Cambridge University Press, 2012.

[12] K. Yosida, *Functional analysis*, vol. 123 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin Heidelberg New York, sixth edition, 1980.

[13] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*. Texts in applied mathematics, Springer, Berlin Heidelberg, second edition, 2007.

[14] S. S. Rienstra and A. Hirschberg, "An introduction to acoustics," Eindhoven University of Technology, 1992.