FDTD Simulation of a Sound Field

Mini Project

Group 18gr872

Aalborg University
Electronic Engineering and IT

This report is compiled in LaTeX. Additionally is Mathworks MATLAB, Adobe Illustrator, Inkscape, Lucidcharts.com, Altium Designer, and Xfig used to draw figures and charts.

Electronic Engineering and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
FDTD Simulation of a Sound Field

Theme:
Signal processing

Project Period:
MSc, 8th Semester 2018

Project Group:
Group 18gr872

Participants:
Jonas Buchholdt
Christoph Kirsch

Supervisor:

Number of Pages: 34

Date of Completion:
26th may 2017

Abstract:

The paper deals with the creation of different sound effects for an electric guitar on the a Digital Signal Processor. Some of these effects are the reverb, the flanger and the equalizer. The report includes a thorough explanation of each of the effects followed by the used design approach. Simulations on MATLAB were done to verify the design. All the effects have been coded in assembly for the DSP implementation. The Assembly code works with the TMS320C5515 DSP from Texas Instruments. In order to make the DSP usable on a variety of electric guitars, a preamplifier was built. All details relating to the design and the implementation of this component are included in the paper as well.

Preface

This Mini Project is composed by group 18gr872 during the 8th semester of Electronic Engineering and IT at Aalborg University. The general purpose of the report is the development and implementation of an Finite-Difference Time-Domain (FDTD) simulation for sound fields in Python and documenting aspects of scientific computing encountered while doing so.

For citations, the report employs the Harvard method. If citations are not present by figures or tables, these have been made by the authors of the report. Units are indicated according to the SI standard.

Aalborg University, May 7, 2018

Jonas Buchholdt
<jbuchh13@student.aau.dk>

Christoph Kirsch
<ckirsc17@student.aau.dk>

Contents

Glossary

FDTD Finite-Difference Time-Domain. v, vi, 5, 6, 7, 9, 11, 12, 13, 14, 15, 16, 17, 21, 24, 25, 26, 29, 31

NSC Numerical Scientific Computing. vi, 17, 23

RMS Root Mean Square. 17, 21, 25, 26

Part I

Problem Description

Chapter 1

Numerical Simulation

1.1 Finite-Difference Time-Domain (FDTD) for sound-field simulation

According to the project requirements for the mini-project, the students are supposed to choose a computational aspect related to their semester project. This, in case of the mini-project at hand, is FDTD-simulation.
When in acoustics the behaviour of sound sources in a room is to be investigated, there are basically for different approaches. The simplest approach is considering only the amount of energy, that is brought into the room and how it is absorbed. The most prominent example is Sabine's formula that can be used to estimate the reverberation time RT_{60}. This very simplistic method only has limited uses. More elaborate methods are based on raytracing, but because of physical limitations they can only represent the higher part of the human hearing frequency range accurately. The very low end of the frequency range can be covered by modal models, that are based on the room geometry and which are not feasible to use towards the mid frequencies. FDTD-simulation follows the fourth approach, which is based on simulating wave behaviour and is most suited to the mid frequency range. [Botteldoore, 1995]
Because the subject of the main project in the current semester is *low-mid frequency acoustical beamforming*, an FDTD-simulation can serve as helpful tool in order to estimate the behaviour of a speaker array under different acoustical conditions. As with most simulations, the goal is, to be able to evaluate the performance of different configurations of loudspeakers and signal processing parameters relatively quickly and accurately. This can aid in the overall development process and keeps the number of real world measurements to a minimum, saving time and resources.

1.2 The FDTD

The goal of this section is to outline the basics of numerical sound field simulation by using the FDTD method. The principles this kind of numerical simulation will be described, so that the method can be adapted to investigate the behaviour of one or more loudspeakers in a sound field. The approach of FDTD is to solve the wave equation by a finite-difference approximation for both time and space derivatives. This makes it possible to easily simulate the sound pressure and particle velocity of a speaker at any time step. For using FDTD with a specific loudspeaker, all simulations have to be done in a relatively narrow frequency band, in order for the simulation to give a good approximation to the real world behaviour. An FDTD cannot cover the whole human hearing range accurately. The scope of the semester project is a frequency range from 60 Hz to 300 Hz. An important property of FDTD-simulations is, that the calculations are performed in time domain, which means, that the pressure and the particle velocity at any specified time step can be analyzed directly by solving two coupled equations.[C. Kleinhenrick and Karhe, 2009]. This section will end out with a FDTD model of a 3 dimensional space.

1.2.1 FDTD wave equation

In FDTD-simulation, there typically are two equations, which need to be solved. When simulating a sound field, the first formula is the Euler Equation 1.1, which describes the relation between the gradient of the pressure p and the derivative of the particle velocity $\vec{v}$ with respect to time.

$$\frac{\partial \vec{v}}{\partial t} = -\frac{1}{\rho} \vec{\nabla} p \tag{1.1}$$

Where:

ρ is the density of the medium	[kg/m^3]
∂t is an infinitesimal time step	[s]
p is the pressure	[Pa]
$\vec{v}$ is the particle velocity	[m/s]

Equation 1.1 is only valid with small variation in pressure. The second Equation 1.2 is the linear continuity equation. The equation describes the relation between the derivative of the pressure p with respect of time and the velocity gradient $\nabla \vec{v}$. They are related through the density of the medium and the speed of sound.

$$\frac{\partial p}{\partial t} = -\rho c^2 \vec{\nabla} \vec{v} \tag{1.2}$$

Where:

ρ is the density of the medium	[kg/m^3]
∂t is an infinitesimal time step	[s]
p is the pressure	[Pa]
c is the speed of sound	[m/s]
$\vec{v}$ is the particle velocity	[m/s]

By use of the derivation, both equations are approximated linearly at every point in a three dimensional cartesian grid. This is done with discrete time steps.

1.2.2 FDTD using Cartesian grid

Using a Cartesian grid for FDTD approximation is a well known technique (see [Botteldoore, 1995]) and will also be employed in this project. The Cartesian grid is set up using the sound pressure Equation 1.2 and the particle velocity Equation 1.2 as the unknown quantities, which have to be solved for in every point in space. A small grid is visualized in Figure 1.4

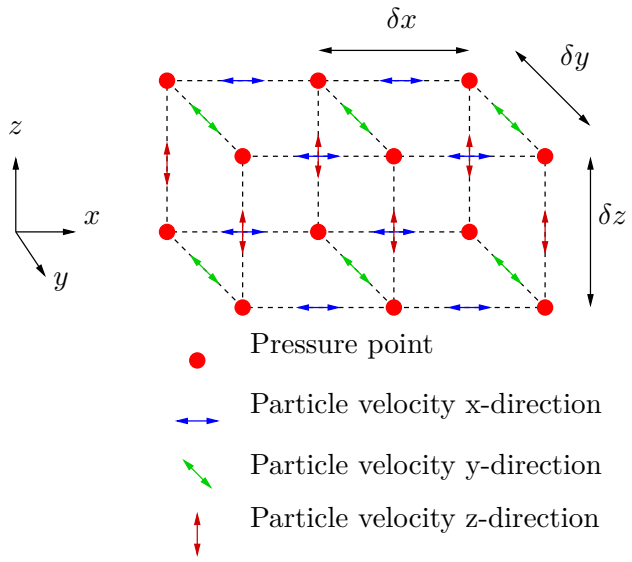


Figure 1.1: A 3 dimensional example of a Cartesian grid.

The grid points are built of positions that are described as $(i\,\delta x, j\,\delta y, k\,\delta z)$ at a time $t = [l]\delta t$. The time step is visualized in Figure 1.2.

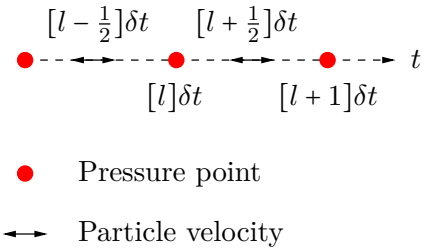


Figure 1.2: Transient definition points of sound p pressure and particle velocity $\vec{v}$

$\delta x, \delta y, \delta z$ are the spatial discretization steps as shown in Figure 1.4 and δt is the time spatial discretization step as shown in Figure 1.2. i, j, k are the discrete indices for the points in the grid and l is the discrete time index. For every axis, the corresponding particle velocity has to be determined at a position in Equation 1.3 at an intermediate time $t = [l \pm \frac{1}{2}]$.

$$\vec{v} = \begin{bmatrix} v_x[(i \pm \frac{1}{2})\,\delta x, j\,\delta y, k\,\delta z] \\ v_y[i\,\delta x, (j \pm \frac{1}{2})\,\delta y, k\,\delta z] \\ v_z[i\,\delta x, j\,\delta y, (k \pm \frac{1}{2})\,\delta z] \end{bmatrix} \tag{1.3}$$

The pressure is determined at position $p^{[l+1]}_{(\mathrm{i,j,k})}$. It can be chosen to start with either pressure or velocity arbitrarily. The time step δt can be regarded as a scaling factor for time, because Python only works with integer indices. This means δt is implemented in the formulas and not in the iteration step. The time $\pm\frac{1}{2}$ is also changed to a integer with adding $\frac{1}{2}$. This scalar is only relevant in the implementation and is disregarded in the rest of this section. The same applies for the step sizes δx, δy and δz.

Solving for $(v_x)^{[l+\frac{1}{2}]}_{(\mathrm{i+\frac{1}{2},j,k})}$ leads to following three Equations 1.4:

$$(v_x)^{[l+\frac{1}{2}]}_{(\mathrm{i+\frac{1}{2},j,k})} = (v_x)^{[l-\frac{1}{2}]}_{(\mathrm{i+\frac{1}{2},j,k})} - \frac{\delta t}{\rho_0 \delta x}\left(p^{[l]}_{(\mathrm{i+1,j,k})} - p^{[l]}_{(\mathrm{i,j,k})}\right) \tag{1.4a}$$

$$(v_y)^{[l+\frac{1}{2}]}_{(\mathrm{i,j+\frac{1}{2},k})} = (v_y)^{[l-\frac{1}{2}]}_{(\mathrm{i,j+\frac{1}{2},k})} - \frac{\delta t}{\rho_0 \delta y}\left(p^{[l]}_{(\mathrm{i,j+1,k})} - p^{[l]}_{(\mathrm{i,j,k})}\right) \tag{1.4b}$$

$$(v_z)^{[l+\frac{1}{2}]}_{(\mathrm{i,j,k+\frac{1}{2}})} = (v_z)^{[l-\frac{1}{2}]}_{(\mathrm{i,j,k+\frac{1}{2}})} - \frac{\delta t}{\rho_0 \delta z}\left(p^{[l]}_{(\mathrm{i,j,k+1})} - p^{[l]}_{(\mathrm{i,j,k})}\right) \tag{1.4c}$$

Solving for $p^{[l+1]}_{(\mathrm{i,j,k})}$ leads to Equation 1.5.

$$p^{[l+1]}_{(\mathrm{i,j,k})} = p^{[l]}_{(\mathrm{i,j,k})} - \rho_0 c^2 \delta t \left(\frac{(v_x)^{[l+\frac{1}{2}]}_{(\mathrm{i}+\frac{1}{2},\mathrm{j,k})} - (v_x)^{[l+\frac{1}{2}]}_{(\mathrm{i}-\frac{1}{2},\mathrm{j,k})}}{\delta x} + \frac{(v_y)^{[l+\frac{1}{2}]}_{(\mathrm{i,j}+\frac{1}{2},\mathrm{k})} - (v_y)^{[l+\frac{1}{2}]}_{(\mathrm{i,j}-\frac{1}{2},\mathrm{k})}}{\delta y} + \frac{(v_z)^{[l+\frac{1}{2}]}_{(\mathrm{i,j,k}+\frac{1}{2})} - (v_z)^{[l+\frac{1}{2}]}_{(\mathrm{i,j,k}-\frac{1}{2})}}{\delta z} \right) \quad (1.5)$$

1.2.3 FDTD grid boundary conditions

The meaning of boundary conditions in the given context is the behaviour of the simulation near and at the boundary surfaces (walls), that occur at the ends of the grid. The sound waves react differently on the walls opposed to propagation in free field conditions. Walls will act like either a reflecting surface, an absorbing surface or both. This boundary behaviour from the wall is described as an frequency dependent impedance. It is necessary to analyse and implement this in the simulation because the sound field will show a different behaviour compared to sound field without any boundaries. Within this project, the frequency dependent boundary conditions will only be a good approximation and not accurately represent free field conditions. An accurate frequency dependent model would require heavy calculations with convolution at each boundary point and at each time step [Botteldoore, 1995]. This kind of calculation has a high time consumption and therefore an approximation will be used.
The approximation in this project will be based on the impedance approach (see [Jeong and Lam, 2000]) as described above. The impedance approach can be used when walls is present in simulation and does not contain a perfect matched layer. Therefore the impedance approach can not be used for free field simulation unless the simulation is stopped just before the wave hits the boundary. Because of the way the pressure spreads along the grid, the stopped simulation will lead to some areas, typically every corner, to which the wave has not yet spread. This results in a circular shape on the simulation grid if the sound sources are placed in the middle. In this project a large room is used and the simulation will be stopped just before the boundary to simulate free field condition. Afterwards the data are cropped such that only simulating data within the area to which the sound had already spread are used.

The impedance approach is usable at low frequency, meaning in a frequency range, in which the sound sources behave approximately omnidirectional [Jeong and Lam, 2000]. Two kinds of absorbing boundaries are common in real life and therefore also in simulation. These boundaries are as follows:

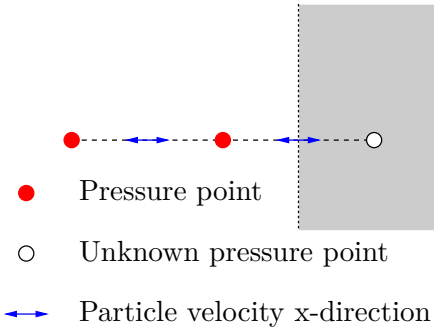


Figure 1.3: The figure visualized a boundary plan through particle velocity plan x in 1 dimension.

For solving the problem visualized in Figure 1.3, an asymmetric finite-difference approximation for the space derivative is used [Botteldoore, 1995]. **??** shows the asymmetric finite-difference approximation.

$$\frac{\partial p}{\partial x}|^{[l]}_{(i_0+\frac{1}{2},j,k)} = \frac{2}{\delta x}\left(p^{[l]}_{(i_0+\frac{1}{2},j,k)} - p^{[l]}_{(i_0,j,k)}\right) \tag{1.6}$$

The advantage of Equation 1.6 is that it only requires knowledge of one nearest pressure point, but it is only valid within δx. Using the same procedure as in Equation 1.4 just with plugging in Equation 1.6 instead, the particle velocity at the boundary is approximated as Equation 1.7

$$(v_x)^{[l+\frac{1}{2}]}_{(i_0+\frac{1}{2},j,k)} = (v_x)^{[l-\frac{1}{2}]}_{(i_0+\frac{1}{2},j,k)} - \frac{2\delta t}{\rho_0 \delta x}\left(p^{[l]}_{(i_0+\frac{1}{2},j,k)} - p^{[l]}_{(i_0,j,k)}\right) \tag{1.7}$$

The only unknown in Equation 1.7 is $p^{[l]}_{(i_0+\frac{1}{2},j,k)}$, but it can be found by using **??**, where v_n is changed with $(v_x)^{[l]}_{(i_0+\frac{1}{2},j,k)}$ and becomes Equation 1.8.

$$\begin{aligned}(v_x)^{[l+\frac{1}{2}]}_{(i_0+\frac{1}{2},j,k)} = (v_x)^{[l-\frac{1}{2}]}_{(i_0+\frac{1}{2},j,k)} - \frac{2\delta t}{\rho_0 \delta x}\Bigg(& Z_0 (v_x)^{[l]}_{(i_0+\frac{1}{2},j,k)} \\ &+ Z_1 \frac{\partial (v_x)^{[l]}_{(i_0+\frac{1}{2},j,k)}}{\partial t} + Z_{-1}\int_{-\infty}^{t} (v_x)^{[l]}_{(i_0+\frac{1}{2},j,k)}(\tau)d\tau - p^{[l]}_{(i_0,j,k)}\Bigg)\end{aligned} \tag{1.8}$$

The integral in the Equation 1.8 is replaced with a sum from minus infinity to l in Equation 1.9.

$$(v_x)^{[l+\frac{1}{2}]}_{(i_0+\frac{1}{2},j,k)} = (v_x)^{[l-\frac{1}{2}]}_{(i_0+\frac{1}{2},j,k)} - \frac{2\delta t}{\rho_0 \delta x}\Bigg(Z_0 (v_x)^{[l]}_{(i_0+\frac{1}{2},j,k)} + Z_1 \frac{(v_x)^{[l+\frac{1}{2}]}_{(i+\frac{1}{2},j,k)} - (v_x)^{[l-\frac{1}{2}]}_{(i+\frac{1}{2},j,k)}}{\delta t} + Z_{-1}\delta t \sum_{m=-\infty}^{l} \left((v_x)^{[m+\frac{1}{2}]}_{(i+\frac{1}{2},j,k)} \right) - p^{[l]}_{(i,j,k)} \Bigg) \quad (1.9)$$

The last unknown variable is the particle velocity v_x at time $t = [l]$. To find a solution for v_x at time $t = [l]$, a linear interpolation between v_x at time $t = [l \pm \frac{1}{2}]$ is used [Botteldoore, 1995]. The resulting particle velocity will be expressed as Equation 1.10

$$(v_x)^{[l+\frac{1}{2}]}_{(i_0+\frac{1}{2},j,k)} = \alpha (v_x)^{[l-\frac{1}{2}]}_{(i_0+\frac{1}{2},j,k)} + \beta \frac{2\delta t}{\rho_0 \delta x}\Bigg(Z_0 (v_x)^{[l]}_{(i_0+\frac{1}{2},j,k)} - Z_{-1}\delta t \sum_{m=-\infty}^{l} \left((v_x)^{[m+\frac{1}{2}]}_{(i+\frac{1}{2},j,k)} \right) - p^{[l]}_{(i,j,k)} \Bigg) \quad (1.10)$$

Where:

$$\alpha = \frac{1 - \frac{Z_0}{Z_{\mathrm{FDTD}}} \frac{2Z_1}{Z_{\mathrm{FDTD}}} \delta t}{1 + \frac{Z_0}{Z_{\mathrm{FDTD}}} \frac{2Z_1}{Z_{\mathrm{FDTD}}} \delta t} \quad [1]$$

$$\beta = \frac{1}{1 - \frac{Z_0}{Z_{\mathrm{FDTD}}} \frac{2Z_1}{Z_{\mathrm{FDTD}}} \delta t} \quad [1]$$

$$Z_{\mathrm{FDTD}} = \frac{\rho_0 \delta x}{\delta t} \quad [\mathrm{Nsm}^{-3}]$$

1.2.4 FDTD grid cell size

The choice of grid cell size for FDTD is a critical parameter, which must fulfill some problem specific constraint [Kunz and Luebbers, 1993]. The grid cell size has to be small enough to contain data for all specified simulated frequencies, which means that the grid cell size has to be smaller than the smallest wavelength λ. As the frequency rises the wave length is decreasing. This means the grid cell size constrain is determined by the highest frequency of interest in the FDTD simulation. Opposed to that, the grid cell size should be relatively large, in order to keep the computational power down, that is required to run the simulation. The grid cell size therefore has to be chosen intelligently, for which [Kunz and Luebbers, 1993] displays a solution. After the grid cell size is chosen the Courant stability condition determines the maximum time step. The maximum time step size, which will be calculated based on the grid cell size, will be the used. A smaller time step size does not improve the

accuracy in general.

The boundary for the smallest grid cell size is the Nyquist rate, which states that the wavelength shall at least be twice as big as the grid cell size δ. Since δx, δy and δz have the same size only δ will be used to represent the grid step size. The Nyquist rate is the lower boundary, but since the simulation is an approximation and is not exact and the smallest wavelength is not precise, δ has to be more than two samples per wavelength. To find a optimal grid size the grid dispersion error which relates to the wave propagation speed through the grid will be taken intro account. The error occurs because the wave propagates with different slightly speeds through the grid. This error is depending on the relative direction of the wave. The grid dispersion error is propertional to the grid cell size, which means that the error is decreased with smaller δ[Kunz and Luebbers, 1993]. Often if $\delta \leq \frac{1}{10}\lambda_{\min}$ the formerly mentioned constraint is met. The solution therefore a good compromise between computation resource and approximation error. The grid cell size in this project is therefore set according to Equation 1.11.

$$\delta x = \delta y = \delta z \leq \frac{1}{10}\frac{c}{f_{\max}} \tag{1.11}$$

Where:

δ is the grid cell size	[1]
x, y and z is the direction	[1]
c is the speed of sound	[m/s]
$f_{\max}$ is the maximum frequency in the simulation	[Hz]

1.2.5 FDTD time step size and stability

The time step size for FDTD follows from the Courant condition [Kunz and Luebbers, 1993]. The aim of the project is not to analyse the condition of Courant. This section will therefore only give a short overview on the most important aspects of the condition and on how to use the condition to calculate the time step size. Considering a plane wave, the Courant condition states that, in one time step, any point on the wave must not pass through more than one cell. During one time step the wave can propagate only from one cell to its nearest neighbors [Kunz and Luebbers, 1993]. To determine the time step size the time step δt can therefore be determined by the speed of sound and the grid cell size as in Equation 1.12.

$$\delta t \leq \frac{1}{\sqrt{\frac{1}{(\delta x)^2} + \frac{1}{(\delta x)^2} + \frac{1}{(\delta x)^2}} \cdot c} \tag{1.12}$$

Where:

δ is the step size	[1]
t is the time indicator	[s]
c is the speed of sound	[m/s]

Making the time step size smaller than stated by Equation 1.12 will not improve the result, in fact the equation calculates the time step size where the grid dispersion error is minimal [Kunz and Luebbers, 1993]. Unless the dispersion error is minimized, the time step might even be smaller because of the stability condition. A stable simulation is only guaranteed under certain conditions. Because Equation 1.10 is applied to different conditions e.g. in corner or flat walls, a stable simulation is not possible in general, only under certain conditions which depend on time and grid cell size. It has been shown, that the simulation is stable if Z_0 and Z_1 are positive for all simulation regions and if Equation 1.13 is satisfied [Botteldoore, 1995].

$$\delta t \leq \sqrt{\frac{2}{3}} \left(\frac{1}{\sqrt{\frac{1}{(\delta x)^2} + \frac{1}{(\delta x)^2} + \frac{1}{(\delta x)^2}} \cdot c} \right) \tag{1.13}$$

If Z_{-1} is nonzero the time step shall furthermore satisfy Equation 1.14

$$c\delta t \leq \delta x \left(\frac{1 + \frac{2Z_1}{\rho_0 \delta x}}{1 + \frac{2Z_{-1}\delta x}{\rho_0 c^2}} \right)^{\frac{1}{2}} \tag{1.14}$$

1.2.6 FDTD sound source

The so called acoustical center of the loudspeaker used in the main project is about 17 cm in the front of the speaker cabinet. It also has to be noted that the FDTD sound source for the simulation is at the acoustical center and not at the position of the loudspeaker itself. Therefore the FDTD sound source is modelled as a transparent source. The speaker cabinet may have a different effect when building a speaker array in the real world, but not at the position of the sound source in FDTD. Therefore the speaker cabinet will not be incorporated into the simulation. The following section will briefly explain the three most common way of implementing sources as described in [Saarelma, 2013].

There are two simple methods to implement a FDTD sound source and one more advanced way to implement a FDTD sound source. The simple ways of implementing at sound source are the hard- and the soft sources. The problem with implementing a hard source is, that the hard source overwrites the update step in the source point and therefore effectively scatters any incident field. This might correspond to a real scenario if the speaker cabinet was at the acoustical center and very reflective, but it

is not and therefore this kind of source is not suitable in the given context. Secondly, the soft source is set up in a way, that the pressure from the source is added to the pressure source point, which means, that this source does not scatter. The problem with this method is that the actual excitation does not match the time function of the source. To make a source that acts like a hard source but does not scatter, the transparent source is used according to [John B. Schneider and Broschat, 1997]. The explaining the implementation of transparent source is quite lengthy and has only little benefit for understanding the computational issues that are the subject of this mini-project. It will therefore be skipped.

1.3 FDTD implementation

The goal of this section is to show the parameters for setting up a FDTD-simulation with one omnidirectional sound source placed in the middle of the room.

1.3.1 FDTD step size

In this section, the step size for both grid- and the time step will be determined. Because the time step size is depending on the grid step size, the grid step size has to be determined first. Equation 1.11 states, that all three dimentions will have the same step size in this project. The calculation is featured in Equation 1.15.

$$\delta d = \delta x = \delta y = \delta z \leq \frac{1}{10}\frac{c}{f_{\max}} \tag{1.15a}$$

$$\delta d = \frac{1}{10}\frac{343\,\mathrm{m/s}}{300\,\mathrm{Hz}} \tag{1.15b}$$

$$\delta d = 0.11\,\mathrm{m} \tag{1.15c}$$

Where:

δd is the maximum grid cell size	[m]
c is the speed of sound at a temperature of $20\,^\circ$°C	[m/s]
$f_{\max}$ is the maximum frequency in the simulation	[Hz]

Since the maximum grid cell size is 11 cm, the grid cell size is in this project will be less or equal to 10 cm, to be sure that a simulation does not suffer from problems due to grid cell size.

In a complicated optimization process in the main project it has been determined, that for the intended function as a loudspeaker array, the optimal distance between the acoustical centers of the two side loudspeakers is 40 cm and the distance to the center of the front speaker is 40 cm as well. When arranging the sources in a triangular shape, the greatest common divisor for putting them into a grid is 20 cm.

However this is bigger then than the formerly stated 11 cm, which makes it unsuitable for simulation. Instead, it has been decided to set the grid step size at 5 cm in all simulations, to ensure flexibility during development. Next the time step has to be determined. The condition for the time step is stated in Equation 1.13 and dictates the following Equation 1.16.

$$\delta t \leq \sqrt{\frac{2}{3}} \left(\frac{1}{\sqrt{\frac{1}{(\delta x)^2} + \frac{1}{(\delta x)^2} + \frac{1}{(\delta x)^2}} \cdot c} \right) \tag{1.16a}$$

$$\delta t \leq \sqrt{\frac{2}{3}} \left(\frac{1}{\sqrt{\frac{1}{(0.05\,\mathrm{m})^2} + \frac{1}{(0.05\,\mathrm{m})^2} + \frac{1}{(0.05\,\mathrm{m})^2}} \cdot 343\,\mathrm{m/s} \right) \tag{1.16b}$$

$$\delta t \leq 687\,\mu\mathrm{s} \tag{1.16c}$$

This corresponds to a sampling frequency of 14 552 Hz. To be sure that a simulation does not suffer from a limiting time step size, the sampling frequency will be rounded up to 14 560 Hz.

1.3.2 FDTD implementation

The aim of this section is to convert the grid to arrays. Since the grid is in three dimensions, where the time is a fourth dimension, the whole array system for pressure and all of the particle velocity grids will build on four dimensional arrays. The following Figure 1.4 shows a simple grid in a corner with entry notation of h as the height of the room, w as the width of the room and l as the length of the room.

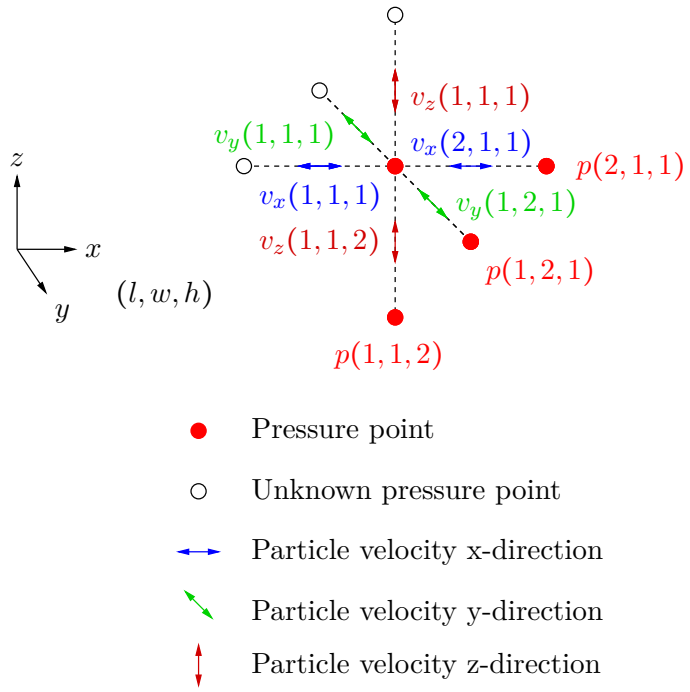


Figure 1.4: The figure visualized a boundary plan through particle velocity plan x in 1 dimension

The simulation is implemented in Python. The x direction is implemented as matrix row direction and y direction is implemented as matrix column direction. The height h is implemented as the third dimension and the time is implemented as the fourth dimension. The particle velocity will be calculated as the first step and the pressure will be calculated as the second step. The time dimension is only as small as 2 pages, because it is only necessary to save at $[l-1]$ and at $[l]$. A page is one dimension in a matrix system, where the number before ´´page" describes the time in this context and not a two dimension page. There can be many dimensions for pages, where the implementation of the FDTD has two page dimensions. All time steps further away than $[l-1]$ are not used in the calculation and will be deleted for keeping the storage requirements as low as possible.

1.3.3 FDTD boundary

The particle velocity at the boundary in all direction has to be calculated as in Equation 1.10 and this is done as a step between the particle velocity matrices and the pressure matrices. This means, that for the particle velocity in x direction the first and last matrix row are calculated by the boundary formula and in y direction the first column and the last matrix column is calculated by the boundary formula.

1.3.4 FDTD plot

The speakers will always be feeded with a gain of one, which correspond to a pressure of ±1 Pa and zero phase. The number of simulation run steps is the smallest room size divided by the grid size, such that the waves do not reflect from the wall.

A wave will expand in a circular form and since the simulation stops just before the it hits the first wall, the simulation result will be a circular shape. Some of the area is not simulated and is therefore without a valid pressure. Because of this, the simulation is made so large that the polar plot with a radius of 10 m can still be calculated, where the used simulation data is only a squared area inside the circular shape and the rest data is discarded.

The polar plot from the analytical model is based on the Root Mean Square (RMS) pressure, therefore the RMS pressure of the FDTD has to be calculated. The RMS of the FDTD simulation is calculated as following in Equation 1.17.

$$P_{\text{rms}}(i,j,k) = \sqrt{\frac{\left(p_{(\text{i,j,k})}^{[l=\delta t]}\right))^2 + \left(p_{(\text{i,j,k})}^{[l=2\delta t]}\right)^2 + \ldots + \left(p_{(\text{i,j,k})}^{[l=n\delta t]}\right)^2}{n}} \tag{1.17}$$

1.4 Numerical Scientific Computing (NSC)-aspects to be investigated

Because the reason for doing the mini-project at hand is learning about NSC, some aspects of the latter have to be approached. One rather obvious point is the performance of the algorithm, where relative changes in computation time between different implementations can be measured and discussed. When talking about performance, one thing, that comes to mind is parallelization. It is advantageous, when a solution to a computational problem can be computed in parallel. However, for FDTD the fact, that the algorithm relies on information that is computed in previous iterations of a loop, makes parallelization rather difficult. Operations like the multiplication of large matrices can still be implemented as parallel computation, but it is not possible to run iterations of the calculation in the pressure grid in parallel. However, for particle velocity matrices, which are directional, it might be possible to parallelize computations and also for a three dimensional case, further parallelization might be possible.

Part II

Software Development

1.5 Considerations of the design

Most of the considerations of the algorithm is already explained in section 1.2, where the over all design is explained. The following list provide the name of the function , input and output.

- The pressure formula is named _p()
- The particle velocity in x direction is named _vx()
- The particle velocity in y direction is named _vy()
- The particle velocity in z direction is named _vz()
- The particle velocity at the left boundary in x direction is named _vxlb()
- The particle velocity at the right boundary in x direction is named _vxrb()
- The particle velocity at the top boundary in y direction is named _vytb()
- The particle velocity at the bottom boundary in y direction is named _vybb()
- The input is a transparent source which is calculated for every iteration and added to the center point of the grid.
- The output is the RMS pressure of every point in grid.

1.6 Data type

Since the algorithm estimate the pressure and particle velocity in space with respect to time, both pressure and particle velocity is stored in a 4 dimension matrix where the last dimension is the time. Both pressure and particle velocity is a shared variable which value is one or less, as long as the transparent source have an amplitude of one. All array is stored as a shared float, because the array data excite 50 mb, and all formula is made as an function. To avoid to much data transfer between parts of the code, data is shared, where the pointer name is the direction name.

1.7 Optimization

Every calculation is depending on each other and have to be calculated for each time step in the right order in serial, with the exception of the individual particle velocity for each time step. This means that the particle velocity x, y, z and the boundary can be calculated in parallel. Another optimization possibility is to go from double precision to single precision. A test function will be made to see if de calculation error is critical or not with using single precision.

1.8 Test Plan

The test plan is to compare the pressure drop of the simulation with the analytical model. By physical knowing, the pressure will attenuate by approx. 6 dB for every

double of distance in fare field and approx. 3 dB in near field. Since the FDTD is an approximation of the far field scenario, an analytical model will be set up as a doctest, where the test compare the analytical model and the simulation. To keep the computation time down, the chosen distance is one and two meter. The test will be run in both double precision and single precision, to see if the single precision is as precise as double precision.

1.9 Profiling and benchmarking

Most for the code have to run in series but the particle velocity can be run i both parallel and serial, and therefore the profiling and benchmarking will be focused on this part. To do the benchmarking of the particle velocity, the time before the particle velocity starts will be recorded, and the time after the particle velocity is finish will be recorded. This two time will be subtracted to fine the spend calculation time of the calculation. All three implementation will be tested to find the best benchmark of the code. The following **??** shows the benchmark time for all three implementation with different room size. The resend to do more than one room size is to determine if the processing is lacking from memory or processing power.

Chapter 2

Adhesion to the Software Development Plan

2.1 Recap: The Waterfall Model

In the software development plan, that had to be submitted as an excercise, following the NSC lecture on the 28^{th} of March 2018, it was stated, that a Waterfall-based development method was to be followed during the course of this mini-project. According to [Larsen et al., 2020, p. 141] and based on [Royce, 1987] a modified Waterfall approach can be described with the following points:

- Functionality
- Algorithm
- Documentation
- Coding
- Quality Metrics
- Evaluation
- Refactoring
- Optimization

Because the requirements for Mini-project are clearly specified, a plan-driven development model like the modified Waterfall scheme is suited to the software project. Another aspect, that helps to ensure a smooth development, is that, due to the group size, only two developers are working on the code. This enables a form of communication and up-to-date-keeping, that would be much more complicated when a larger number of participants was involved.

2.2 Development Log

In this section, brief statements to how the points from section 2.1 were addressed in the course of the mini-project will be made.

2.2.1 Functionality

Because the FDTD-simulation is part of another project, which deals with low-mid frequency acoustical beamforming, many of the functional requirements were dictated by the usage in the main project. The usage in the main project demands, that the simulation routine sets up a FDTD-grid with a given room size and different boundary conditions. Furthermore, one ore several transparent sound sources have to be placed in the grid at given positions and the sound emitted by those sources has to be individually adjustable in amplitude and phase. It must be possible to export plots of the pressure field at a given point in time in order to illustrate the behaviour of arrangement of sound sources in the simulation. In the main project, also polar plots are extracted from the FDTD-data, but this is not part of the implementation made in the mini-project.

2.2.2 Algorithm

An algorithm is needed that insures that the formerly stated functional requirements are achieved. While the theoretical basics are described in section 1.2, some more detailed explanations on a more technical level are given in **??**. These go hand in hand with section 2.2.4. While the theory of FDTD provides well established fundamentals, of what mathematical construct should be implemented, there is still a number of choices, regarding what the implementation should look like considering the intended application.

2.2.3 Documentation

Specific requirements on the documentation were given in the mini-project task description. For documentation purposes the document at hand has been written and the code itself has been augmented with comments, based on the proposals given in [van Rossum et al., 2013]. When looking at the modified Waterfall model, the documentation step is a point, that has had a more iterative nature than other points. This is mainly due to the fact, that, within the development process, some difficulties in the implementation have led to more complex coherencies then initially anticipated. Keeping the documentation up to date means coming back to it basically after every change that is being made to the software.

2.2.4 Coding

In the coding phase, everything that had been planned previously has been put into code. However, there was an iterative component when the evaluation (section 2.2.6)

revealed some faulty behaviour of the code. Coding has been executed by two different programmers who kept track of the versions via the means of *git*. All code was written in Python 3.6.5. Several implementations of the code have been made, in order to investigate their behaviour in terms of efficiency (see also section 2.2.8).

2.2.5 Quality Metrics

When quality metrics are to be considered, there are three main issues that come into mind. The functional requirements (section 2.2.1) have to be fulfilled, for reference cases the results should match those known to those cases that are obtained from trusted sources and finally the computation time should be as low as possible.
In order to design a routine that checks the feasibility of simulation results by comparing it to results, that are known to be correct, a suitable example of such a reference case has to be found. Independent of the boundary setup, the conditions in a finite size FDTD-grid correspond to acoustical free field conditianons as long as the waves emitted by sources have not reached the boundary. This means, a correct behaviour of the simulation can be confirmed by investigating one of these "early" samples. Based on [Kinsler et al., 2009, p. 172], the sound pressure difference at two points with distances r_1 and r_2 from a pulsating sphere can be described as follows:

$$\Delta L_{\mathrm{p,a}}(r_1, r_2) = 20 \log_{10} \frac{|j\rho_0 c V_0 (\frac{a}{r_1}) ka(e)^{j(\omega t - k r_1)}|}{|j\rho_0 c V_0 (\frac{a}{r_2}) ka(e)^{j(\omega t - k r_2)}|} \,\mathrm{dB} \tag{2.1}$$

Where:

$\Delta L_{\mathrm{p,a}}$ is the sound pressure level difference.	[dB]
r_1, r_2 are the distances, for which the pressures are being calculated, $r > a$.	[m]
t is the time, for which the pressure is being calculated.	[m]
ρ_0 is the specific density of air.	[kg/m^3]
c is the speed of sound.	[m/s]
V_0 ist the velocity at the surface of the spherical source.	[m/s]
a is the radius of the spherical source.	[m]
ω is the angular frequency.	[s^{-1}]
k is the wavenumber.	[m^{-1}]

Equation 2.1 can be simplified to a great extent when adapting into the context of the FDTD-simulation:

$$\Delta L_{\mathrm{p,a}}(r_1, r_2) = 20 \log_{10} \frac{r_2}{r_1} \,\mathrm{dB} \tag{2.2}$$

When a single sound source is positioned in the FDTD-grid, the RMS-pressure at two points with known distances to the source can be recorded. If level difference at these points approximately matches the value according to Equation 2.2, that can be interpreted as a sign for the simulation to work correctly.
In order to assess computation time, different implementations are run on the same

machine with the same parameters and the time is measured. In this context, the "bottleneck" of the computation can be determined and discussed: MORE TEXT HERE?

2.2.6 Evaluation

In order to get an rough estimate over which order of magnitude of precision can be achieved in FDTD-simulation, a comparison test was set up. A single transparent sound source was set up amidst a three-dimensional cubic room with an edge length D. The grid size was set to $\delta d = \delta x = \delta y = \delta z = 0.05\,\mathrm{m}$. The source radiated sound at a frequency of $f = 300\,\mathrm{Hz}$. The RMS pressure was logged at distances of $r_1 = 1\,\mathrm{m}$ and $r_2 = 2\,\mathrm{m}$ from the source. The difference between the simulated pressures $\Delta L_{\mathrm{p,s}} = L_{\mathrm{r2}} - L_{\mathrm{r1}}$ was compared to the analytical pressure difference according to Equation 2.2. For the distances as stated above $\Delta L_{\mathrm{p,a}} = 6.021\,\mathrm{dB}$. The error will denoted as $\epsilon_s = \Delta L_{\mathrm{p,a}} - \Delta L_{\mathrm{p,s}}$. This means, that negative values of ϵ_s indicate, that the simulated pressure drop is bigger then it is expected according to the analytical model. Simulations were carried out with single- and double precision floats, in order to see if this impedes the error ϵ_s.

Table 2.1: Error evaluation

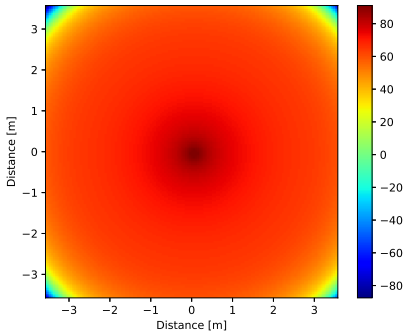
	ϵ_s [dB]	
D [m]	Single Precision	Double Precision
10	- 0.686	- 0.686
20	- 0.750	- 0.750
30	- 0.040	- 0.040

Table 2.1 indicates, that there is no advantage in using double precision variables over single precision variables. In an acoustical context, the results are virtually indistinguishable. Because single precision calculations offer increased performance, they are preferable. The simulation grid size appears to have an influence on the error ϵ_s. Grid size and error do not appear to be reciprocal, but it can be noted, that ϵ_s is significantly smaller for a edge length of $D = 30\,\mathrm{m}$ then at the other lengths investigated. Supposedly this is also depending on the grid resolution δd.

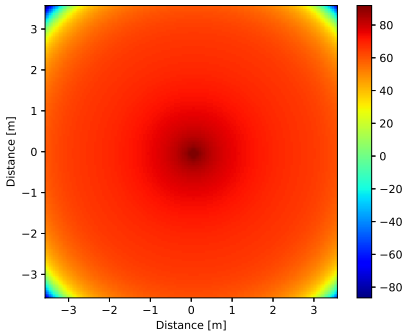
Table 2.2: Velocity step calculation times in a 3D-grid, calculated on a computer with an I7 920 processor and 12 GB RAM, (*) extrapolation from the time measured in a 2D-grid

	Calculation time [s]					
	Single Precision			Double Precision		
(D) [m]	naive*	single	multi	naive*	single	multi
10	200.00	0.15	0.19	0.16	0.37	0.27
20	2340.00	2.19	1.10	27,37	29.93	20.02
30	48150.00	53.38	61.03	81870.00	251.8	487.26

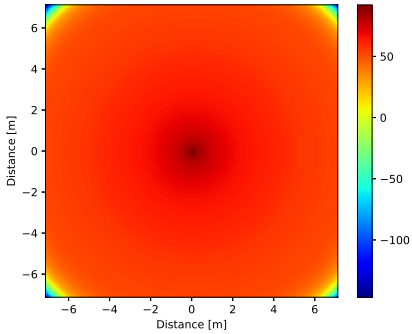
Table 2.2 shows the times it took to calculate a single velocity step in a grid with the given edge lenght D. In every scenario, except the 10 m Double Precision grid, the naive implementation proves to be the slowest by far. The implementations that utilize array operations do not show an unambiguous behaviour. For some cases the multicore implementation has performance benefits. However, this seems to only be the case for grids within a certain sweet spot size, where the additional thread management begins to pay off and the computation is not yet slowed down by limitations related to the shared memory.



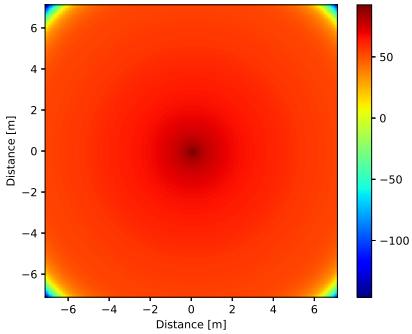
(a) The figure shows the RMS pressure for a room of 10 m and with single precision calculation



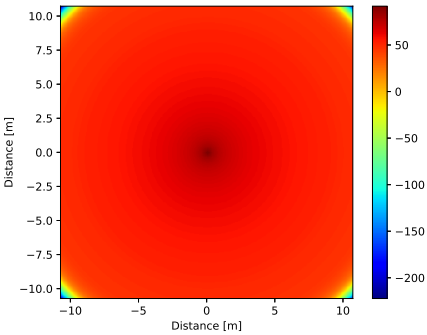
(b) The figure shows the RMS pressure for a room of 10 m and with double precision calculation



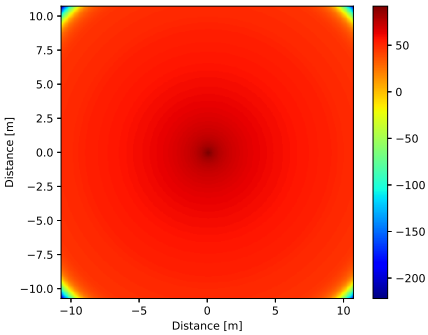
(c) The figure shows the RMS pressure for a room of 20 m and with single precision calculation



(d) The figure shows the RMS pressure for a room of 20 m and with double precision calculation



(e) The figure shows the RMS pressure for a room of 30 m and with single precision calculation



(f) The figure shows the RMS pressure for a room of 30 m and with double precision calculation

Figure 2.1: The figure shows the resulting RMS pressure for...

2.2.7 Refactoring

Refactoring has partly been an element of the mini project, as there were different implementations (naive, vector-operation, parallel) to be made. In the context of a larger software project, refactoring could be done on a larger scale with different elements of the code. However, the whole concept of refactoring is in contrast to the classical waterfall approach. In the modified waterfall approach that is employed in this mini-project, a refactoring step is included, perhaps to retrofit some of the advantages of agile programming.

2.2.8 Optimization

One of the tasks listed in the problem description for the mini-project involves implementing the algorithm in a way, that parallel processing is used. This can be done for FDTD-simulation. However, because of the nature of the computational tasks, it takes a great deal of effort to make an implementation using parallel computing on the CPU, that comes even remotely close to a well designed implementation without the explicit use of parallel computing. However, it might be possible to implement the simulation on a GPU and gain performance by doing so.

Chapter 3

Conclusion

Within in the context of the mini-project at hand, a sound field simulation has been implemented using a FDTD-based method. Investigations regarding precision and performance have been conducted. It has been established, that in an acoustical context, there is no point in using double precision variables instead of single precision.

The chosen development plan (modified waterfall model) turned out to be a suitable choice for the given project. This might have been due to the clear target definition and the small number of participants, which made it easy to keep track of the development. For more complex software projects a more agile approach might be better suited.

Bibliography

Botteldoore, D., 1995. Finite-difference time-domain simulation of low-frequency room acoustic problems. *Tidskrift.* [online], p. 7. Available at: <https://asa.scitation.org/doi/pdf/10.1121/1.413817>. [Accessed 09.03.2018].

C. Kleinhenrick, T. W., A. Niepenberg and Karhe, D., 2009. A Three-Dimensional Acoustic Simulation for the Development and Evaluation of Active Noise Control System using the FDTD Method. *Tidskrift.* [Online], (NAG/DAGA), pp. 584–587. Available at: <http://pub.dega-akustik.de/NAG_DAGA_2009/data/articles/000505.pdf>. [Accessed 09.03.2018].

Jeong, H. and Lam, Y. W., 2000. FDTD modelling of frequency dependent boundary conditions for room acoustics. *Tidskrift.* [online], p. 7. Available at: <https://www.researchgate.net/publication/265818356_FDTD_modelling_of_frequency_dependent_boundary_conditions_for_room_acoustics>. [Accessed 16.03.2018].

John B. Schneider, C. L. W. and Broschat, S. L., 1997. Implementation of transparent sources embedded in acoustic finite-difference time-domain grids. *Tidskrift.* [online], p. 7. [Accessed 03.04.2018].

Kinsler, L. E., Frey, A. R., Coppens, A. B., and Sanders, J. V. *FUNDAMENTALS OF ACOUSTICS, 4TH ED* -. John Wiley & Sons, Inc, New York, 4 edition, 2009. ISBN 0-471-84789-5.

Kunz, K. S. and Luebbers, R. J. *The Finite Difference Time Domain Method for Electromagnetics* -. CRC Press, Boca Raton, Fla, 1993. ISBN 978-0-849-38657-2.

Larsen, T., Arildsen, T., and Jensen, T. L. *Behavioral Simulation and Computing for Signal Processing Systems.* Wiley, Hoboken, NJ, preliminary edition, 2020. ISBN 978-0-470-71134-7.

Royce, W. W. Managing the Development of Large Software Systems: Concepts and Techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pp. 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press. ISBN 0-89791-216-0. Available at: <http://dl.acm.org/citation.cfm?id=41765.41801>.

Saarelma, J., 2013. Finite-difference time-domain solver for room acoustics using graphics process- ing units. Speciale, Aalto University, School of Electrical Engineering. Available at: <https://pdfs.semanticscholar.org/3de4/fad7e947af077745e73717e4edf3e8d66b3f.pdf>.

van Rossum, G., Warsaw, B., and Coghlan, N., 2013. *PEP 8 – Style Guide for Python Code*. Available at: <https://www.python.org/dev/peps/pep-0008/>.