Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Mikolaj Jaworski

Sound based localization utilizing measurement signals in the audible frequency range

Master's thesis in Electronic Systems Design
Supervisor: Peter Svensson
July 2020

NTNU
Kunnskap for en bedre verden

Mikolaj Jaworski

Sound based localization utilizing measurement signals in the audible frequency range

Master's thesis in Electronic Systems Design
Supervisor: Peter Svensson
July 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Preface

The presented work is the result of the final project of the Master of Science in Electronics Systems Design at the Norwegian University of Science and Technology (NTNU). The project was carried out at the Department of Electronic Systems between January and July of 2020, and has been supervised by Peter Svensson (NTNU) and co-supervised by Audun Solvang (Spotics AS).

The initial idea of this Master's thesis was suggested by Audun Solvang at Spotics AS in Trondheim.

Acknowledgments

I would like to start by thanking my supervisor, Peter Svensson, for good discussion, comments, and helpful problem solving skills. I would also like to thank my co-supervisor Audun Solvang for providing me with the idea behind this thesis, and invaluable comments and discussions.

I would also like to thank Tim Cato Netland for his help while conducting measurements and flexibility when choosing equipment and measurement strategies.

Lastly I would like to thank my friends and family for support during this project, and throughout my studies.

-Mikolaj Jaworski

Abstract

With the rise in popularity of media like podcasts and live streaming of concerts, it is of interest to study ways to improve the listening experience and automate audio post-processing. One particularly interesting enhancement is that of fully automated audio mixing. Such an enhancement would eliminate the need of an audio engineer and possibly lead to a solution that automatically generates audio in stereo or even more complex 3D-sound formats in real-time. However, to make this possible, information about the microphone position is necessary, thus resulting in the need for a positioning system. It is reasonable to study a sound-based positioning system that considers this already existing infrastructure consisting of Commercial-Off-The-Shelf (COTS) microphones.

This thesis aimed to study a range of measurement signals in the audible frequency range. Estimation of single source-to-receiver distances at fixed positions was studied. Furthermore, estimation of the position of microphones relative to a loudspeaker-array at fixed positions in 3D, and the azimuth Angle of Departure (AoD) was conducted.

For the single source-to-receiver distance estimations, results were obtained both from simulations and measurements. The measurement signals were different sequence lengths of the Maximum Length Sequence (MLS), which is one of the Pseudo Noise (PN) sequences. In the simulations, a simple Room Impulse Response (RIR) simulator was used as the acoustic measurement channel. Different source-to-receiver distances and reverberation times were studied. A moving average filter with a different number of signal averages was studied to determine the number of averages required to detect correct source-to-receiver distance. In measurements, the number of signal averages, and signal lengths were studied.

For position and AoD estimations, results were obtained through simulations with a detailed RIR obtained from CATT-acoustics simulation software. In the simulations, a range of PN-sweeps and PN-sequences with different lengths were studied as measurement signals. Kasami and Gold sequences were used for their low cross-correlation properties, leading to continuous estimations from all loudspeakers in the loudspeaker-array. For these estimations, a range of signal averages, signal lengths, AoDs, and source-to-receiver distances were studied.

A study of moving microphones was conducted through source-to-receiver distance measurements. In the measurements, a microphone was placed on a turntable, and continuous source-to-receiver distance estimates were conducted by utilizing different measurement signals with different lengths.

In estimations of single source-to-receiver distances in fixed positions, both measurements and simulations presented that distances up to 5 m could be correctly estimated with 2047 samples long MLS without the use of a moving average filter, as long as a correct threshold of detection was utilized. In results obtained from AoD estimations, it was presented that estimates where a moving average filter averaging five periods of the measured signal resulted in an AoD estimate that, on average, deviated with $< 1^\circ$ from the correct angle. However, when the moving average filter consisted of < 4 periods/cycles, deviations of $\sim 2.5^\circ$ from the correct angle were observed.

PN-sweeps were far less prone to time variance caused by movement than MLS, but for both signal types, the implementation of moving average filter resulted in many estimates outside the window of interest (minimum and maximum source-to-receiver length). Source-to-receiver distance estimates where uncoded sweep signals and moving average filter was used, yielded good results, and all estimates where within the window of interest. The same was true for PN-sweeps coded with short sequences (≤ 255). Therefore, it can be concluded that PN-sweeps coded with short sequences should be utilized if the microphones are non-stationary.

The primary motivation for utilizing measurement signals in the audible frequency range was to use the already existing infrastructure with COTS microphones. However, because of factors like the dynamic range and self-noise, it can be concluded that a system like the one presented in this thesis would not work with all COTS microphones.

Sammendrag

Med den stadig økende populariteten knyttet til lydinnspilling i medier som podcast og livestreaming av konserter, er det av interesse å studere metoder som kan forbedre lytteopplevelsen, samt automatiseringen av postprosesseringen for disse bruksområdene.

En spesielt interessant forbedring er full automatisert lydmiksing. En slik forbedring vil potensielt eliminere behovet for en lydtekniker, og kan føre til en løsning som automatisk genererer lyd i stereo eller mer komplekse 3D-lydformater i sanntid. For å muliggjøre dette er det imidlertid nødvendig med informasjon om mikrofonplassering, hvilket resulterer i behovet for et posisjoneringssystem. Det er rimelig å studere et lydbasert posisjoneringssystem som benytter seg av denne allerede eksisterende infrastrukturen bestående av kommersielle hyllevare (COTS) mikrofoner.

Dette prosjektet tok sikte på å studere målesignaler i det hørbare frekvensområdet. Det ble utført estimeringer av: kilde-til-mottaker avstand mellom én høyttaler og én mikrofon i faste posisjoner, mikrofonplassering i forhold til et "høyttaler-array" på faste posisjoner i 3D, og azimuth avgangsvinkel (AoD).

For estimering av kilde-til-mottaker distanse ble resultater innhentet fra både simuleringer og målinger, hvor målesignalene brukt til denne estimeringen var forskjellige sekvenslengder av *Maximum Length Sequence* (MLS) som er en av *Pseudo Noise* (PN) sekvensene.

I simuleringene ble en simpel Rom Impulsrespons (RIR)-simulator benyttet som den akustiske målekanalen. Ulike distanser mellom kilde og mottaker, samt forskjellige etterklangstider ble undersøkt. Et glidende gjennomsnittsfilter med ulikt antall periodegjennomsnitt ble studert for å bestemme antall et periodegjennomsnitt som var nødvendig for å oppdage korrekt kilde-til-mottaker distanse. I målinger ble også antall periodegjennomsnitt og signallengder studert i ett rom.

For posisjons-, og AoD-estimering ble resultater innhentet fra simuleringer med en detaljert RIR som ble generert med simuleringsprogrammet CATT-acoustics. I disse simuleringene ble et utvalg av PN-sveip og PN-sekvenser med ulik lengde studert som målesignaler. Kasami-, og Gold-sekvenser ble benyttet på bakgrunn av deres lave krysskorrelasjonsegenskaper som muliggjorde kontinuerlig estimering. For disse estimeringene ble et utvalg av periodegjennomsnitt, signallengder, AoD og kilde-til-mottaker distanser studert.

En studie av tidsvarians forårsaket av bevegelige mikrofoner ble gjennomført. I disse målingene ble kontinuerlig kilde-til-mottaker estimering utført av en mikrofon plassert på et roterende bord. Disse estimatene ble utført for en rekke ulike sekvenstyper og -lengder.

Resultater fra målinger og simuleringer for én høyttaler og én mikrofon i faste posisjoner presenterte at distanser opp til 5 m kunne estimeres korrekt med 2047 punktprøver lange MLS uten bruk av et glidende gjennomsnittsfilter, så lenge en riktig deteksjonsterskel ble benyttet.

I resultatene fra AoD-estimatene ble det presentert at estimater der et glidende gjennomsnittsfilter bestående av gjennomsnittet av 5 perioder av signalet, resulterte i en estimert vinkel som

i gjennomsnitt avviket med $< 1^\circ$ fra korrekt vinkel. Når det bevegelige gjennomsnittsfilteret bestod av < 4 perioder, ble avvik på $\sim 2.5^\circ$ fra korrekt vinkel observert.

PN-sveip var langt mindre påvirket av tidsvarians enn MLS, men for begge signaltypene resulterte implementeringen av glidende gjennomsnittsfilter i en høy prosentandel av tapt deteksjon. Estimering av kilde-til-mottaker distanse der ikke-kodede sveipsignaler og glidende gjennomsnittsfilter ble brukt, gav gode resultater. Det samme gjaldt for PN-sveip kodet med korte sekvenser (≤ 255). Det kan derfor konkluderes at PN-sveip kodet med korte sekvenser burde benyttes dersom mikrofoner ikke skal være i stasjonære posisjoner.

Hovedårsaken til å bruke målesignaler i det hørbare frekvensområdet var å benytte den allerede eksisterende infrastrukturen med COTS mikrofoner. På grunn av faktorer som det dynamiske området, og selvstøyen til mikrofoner, kan det imidlertid ikke konkluderes med at et system som er presentert i denne oppgaven vil være brukenes med alle COTS mikrofoner.

Abbreviations

AoD Angle of Departure

BW Band Width

COTS Commercial Off-The-Shelf

DFT Discrete Fourier Transform

DSP Digital Signal Processing

FFT Fast Fourier Transform

FIR Finite Impulse Response

IIR Infinite Impulse Response

IR Impulse Response

KS Kasami Sequence

MLS Maximum Length Sequence

PN Pseudo Noise

PSD Power Spectral Density

RIR Room Impulse Response

RMS Root Mean Square

SNR Signal to Noise Ratio

SPL Sound Pressure Level

ToA Time of Arrival

Contents

1. Introduction

Indoor positioning and navigation is a growing industry, and the number of products for this purpose has grown significantly in the last couple of years. These products are typically used for asset tracking, indoor navigation, and many more. As opposed to outdoor positioning, where the Global Positioning System (GPS) is utilized, there exists no broadly adopted method for the indoor. This lack of a standardized method leads to the use of multiple sensors like; image (Artificial Vision), infrared, ultrasound, radiofrequency, inertial and magnetic, and also audible sound are being used for the purpose [1].

With the rise in popularity of media like podcasts and live streaming, it is of interest to study ways to improve the listening experience and to automate the process of audio post-processing. One particularly interesting enhancement is that of fully automated audio mixing. Such an enhancement would eliminate the need of an audio engineer, and possibly leading to a solution that automatically generates audio in stereo or even more complex 3D-sound formats in real-time. However, to make this possible, information about microphones' position is necessary, thus resulting in the need for a positioning system. It is reasonable to study a sound-based positioning system that considers this already existing infrastructure.

There are few commercial products that are based on this already existing technology, but a number of sound-based systems such as Active Bat [2], Cricket [3], Dolphin [4] and 3D-Locus [5] have previously been documented. These systems are based on ultrasound as the measurement signal; thus, Commercial-Off-The-Shelf (COTS) audio components cannot be used. For wide consumer adoption, interoperability with established technology and infrastructure is key.

In order to utilize COTS components as receivers, the measurement signal needs to be within the audible frequency range. Mandal et al. present such a system named Beep in his thesis [6]. The paper reports that the signal used for positioning caused annoyance and would not work in a real application, especially where sound recordings are being conducted as the measurement signal would be audible in the recordings. Therefore, it is reasonable to study a measurement signal which is either inaudible or at least not causing user annoyance.

1.1. Motivation and goal

This thesis aims to study a range of measurement signals buried in noise and study position estimation obtained by utilizing these signals. A study of estimations of source-to-receiver distance, azimuth angle, x-, y-, and z-coordinate will be presented for stationary microphone positions in the line of sight; furthermore, results of measurements of moving microphones will be presented. All discussions and assumptions regarding audibility and perceivable sound will be based on the author's subjective opinion. The results will be presented both as results

obtained from simulations and measurements. Because of unforeseen challenges caused by the Covid-19 pandemic resulting in reduced access to laboratory facilities as well as problems with equipment that was planned to be used for measurements, the result for 3D positioning will only be presented for simulations. It should be stressed that the results presented in this thesis are not fixed but serves as a tool for comparison between the cases. However, the results will be thoroughly discussed to enlighten possible improvements or errors. *The goal of this thesis is not to come up with a definitive answer, but to draw a picture of the characteristics.*

This thesis is a continuation of a project thesis written by the author in autumn 2019. The project acted as a feasibility study for this thesis and some parts presented here are directly copied from the project thesis.

1.2. Thesis structure

In chapters 2, 3, and 4, the theory regarding human hearing, the acoustic measurement chain, and the measurement signals will be presented. Chapter 5 presents the theory of how positioning will be estimated, and chapter 6 presents some theory regarding signal processing concepts utilized in this thesis. In chapter 7, the method for how the experiments were conducted will be presented. In chapters 8 and 9, the results will be presented and discussed. In chapter 10, this thesis will be concluded, followed by the appendices.

2. Human hearing

Audibility is a crucial concept of this project, and to understand this term, a short explanation of the human hearing is necessary. An in-depth explanation of human hearing and psychoacoustics can be found in [7], which forms the basis of the theory presented in this section.

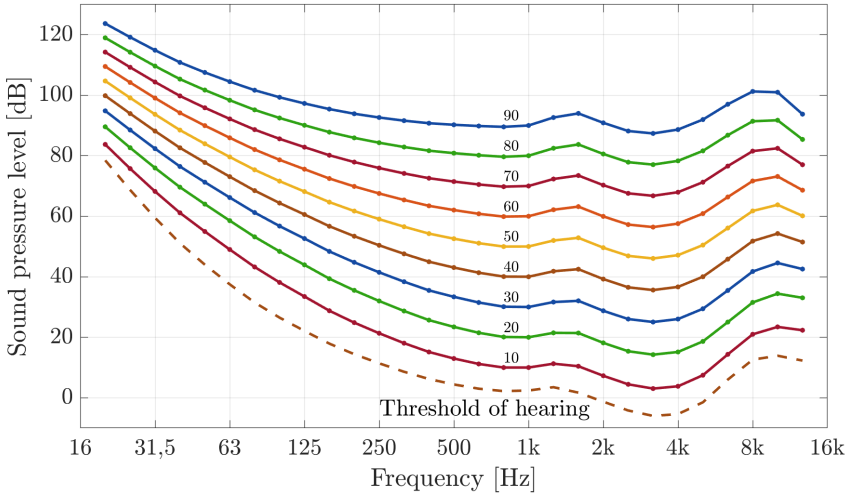


Figure 2.1.: Fletcher-Munson curves, where each curve shows frequencies which are perceived as equally loud – they result in the same loudness level [phon]. Loudness level is defined, such that the loudness level for a sinusoidal at 1 kHz is identical to the sound pressure level for the sinusoidal. The values presented in this figure were obtained from ISO 226 standard [8]

The audible frequency range of the human ear is roughly 20 Hz - 20 kHz and the frequency response of the ear appears to be non-flat. This non-flatness can be illustrated when contours of equal loudness are drawn. These contours represent all harmonic sounds that will be perceived as equally loud by an average subject for a specified Sound Pressure Level (SPL). The equal loudness contours, also known as the Fletcher-Munson curves, can be seen in figure 2.1, where the lowest curve corresponds to the threshold of audibility. This threshold is the weakest sounds a human with normal hearing can perceive. Each curve corresponds to a loudness level in *phon*[1]. The figure also shows that the human hearing performs best at frequencies between 3 kHz and 4 kHz; this is caused by the first resonance frequency of the ear canal [7].

[1]Phon is a unit of loudness level for pure tones originally introduced by scientist Heinrich Barkhausen

It is important to note that even though these contours are commonly available and standardized, the sensitivity of the human hearing can vary a lot with the listener's age, health, and prior exposition to loud sounds. The same holds for the audible frequency range; some people will perceive sounds below and above these frequencies.

Different filters for different sound levels are used to correct for the sensitivity of hearing. The most known examples of such filters are the *A-weighting*, *C-weighting*, and *Z-weighting*. The A-weighting corresponds roughly to 40 phons, C-weighted corresponds to 100 phons, and the Z-weighted corresponds to no-weighting. The different frequency weighting curves can be seen in figure 2.2. The A-weighting and C weighting filters are defined in IEC 61672-1 standard [9].

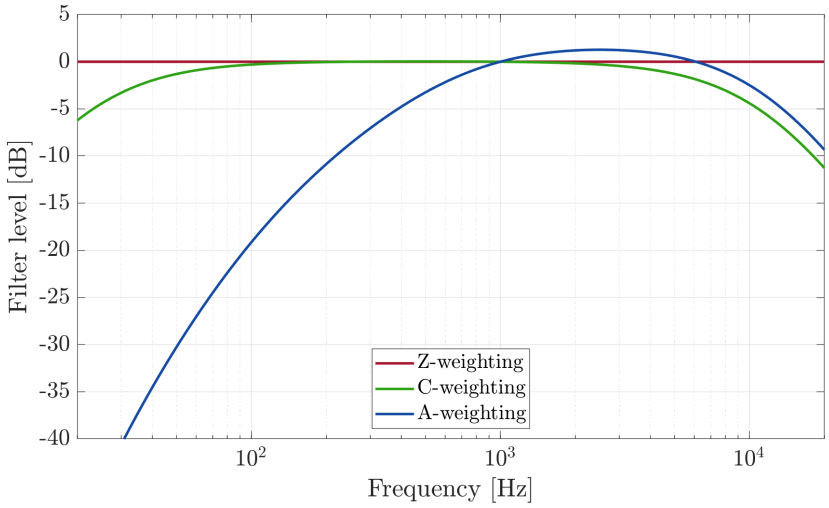


Figure 2.2.: Frequency weighting curves for the A-, C- and Z-weighting filters.

3. The acoustical measurement chain and room reverberation

To understand what can be expected from the different parts of a typical acoustical measurement chain, it's crucial to understand how all parts respond to signals that are transmitted through the system. The system response to signals is denoted by the systems Impulse Response (IR). IR has been previously explained in many signal-processing textbooks from where the theory presented here was obtained [10, 11, 12].

The IR of a Linear Time-Invariant (LTI) system is defined as the response of the system to a unit sample excitation, i.e., the response of the system when the input is a Dirac delta function, $\delta[n]$ [10]. The Dirac delta function is defined as seen in equation 3.1.

$$\delta[n] \triangleq \begin{cases} 0 & \text{if } n \neq 0 \\ 1 & \text{else} \end{cases} \tag{3.1}$$

An LTI system can be seen in figure 3.1, where $x[n]$ is the input signal, $h[n]$ is the systems IR and $y[n]$ is the output of the system. In the system presented in the figure the output $y[n]$ is given by the convolution of $x[n]$ with $h[n]$. The convolution operator $*$ is defines as seen in equation 3.2.

$$\begin{aligned} y[n] &= x[n] * h[n] \\ &= \sum_{k=0}^{N-1} h[k]x[n-k] \end{aligned} \tag{3.2}$$

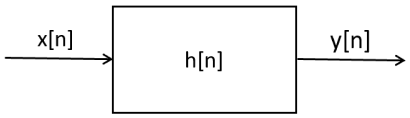


Figure 3.1.: Schematic representing of an LTI system with $x[n]$ as input, $y[n]$ as output, and $h[n]$ as the impulse response of the system.

Loudspeakers, microphones, and rooms are all dynamic systems with unique IR. These systems are often assumed to be LTI [11], which means that a sound signal transmitted and received in a room will be modified by the impulse response of:

- The loudspeaker used to generate the sound.
- The microphone used to record the sound.
- The room in which the signal is generated.
- Other parts like the microphone and loudspeaker amplifiers, analog-to-digital converters, etc. (these are often assumed to be close to ideal, and their influence is often ignored).

Mathematically, this can be expressed as seen in equation 3.3 where $x[n]$ is the input signal, $h[n]$ is the IR of the dynamic system where the index denotes the specific system, and $y[n]$ is the output signal, i.e., the measured signal.

$$y[n] = x[n] * h_{room}[n] * h_{loudspeaker}[n] * h_{microphone}[n] \tag{3.3}$$

To simplify this expression, a variable for the entire LTI system, $h_{channel}$, can be expressed as presented in equation 3.4

$$h_{channel}[n] = h_{room}[n] * h_{loudspeaker}[n] * h_{microphone}[n] \tag{3.4}$$

This yields the following expression for the measured signal $y[n]$:

$$y[n] = h_{channel}[n] * x[n] = \sum_{k=0}^{\infty} h_{channel}[k]x[n-k] \tag{3.5}$$

3.1. Typical impulse responses of the acoustical measurement chain

As previously mentioned in chapter 2, the hearing range of human spans between $20 - 20 \cdot 10^3$ Hz, which implies that equipment made for sound recording and reproduction is made to operate in that frequency range. This means that the IR of audio equipment will have bandpass characteristics, as represented in figure 3.2. These characteristics will vary between the different systems (microphones, sound cards, loudspeakers, etc.).

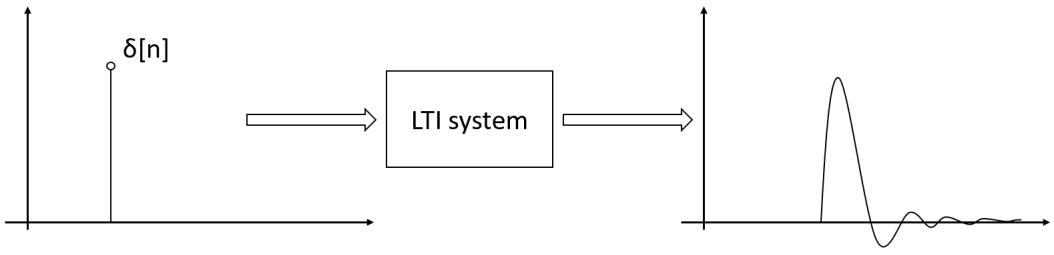


Figure 3.2.: A Dirac pulse convolved with a LTI system with bandpass characteristics.

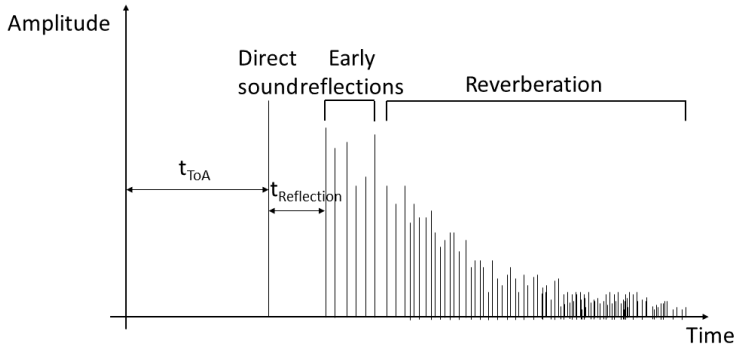


Figure 3.3.: The impulse response of a room where the input was a Dirac delta pulse at sample 0 and the IR was measured by a microphone at a distance from the impulse source. t_{ToA} is the time-of-arrival, i.e the travel time of the impulse, and $t_{Reflection}$ is the time of arrival difference between the direct sound and the first reflection.

A schematic representation of an IR of a room is presented in figure 3.3. This figure shows the Room Impulse Response (RIR) as measured by a microphone if a Dirac delta pulse was transmitted at sample 0. The time between the impulse and the direct sound is denoted by the Time of Arrival (ToA), t_{ToA}, of the signal.

The RIR can be broken into three parts: the direct sound, early reflections, and reverberation. The direct sound is the first impulse that arrives at the source. The early reflections are the first specular reflections from surfaces like walls, floors, and ceilings. The reverberation is a collection of reflected sounds that blend and overlap [13]. The direct sound in a room will generally decrease with 6 dB for each doubling of the distance from the source [14].

For small rooms, the level of reverberation is assumed to be constant everywhere in the room, according to the classic model [14]. Thus at a distance from the source, the direct sound will equal the constant reverberation level [15]. This distance is called the critical distance or room radius.

As presented in figure 3.3, the sound generated in a room will have an exponential decay. By analyzing the slope of the decay of the reverberation time, T_{60}, can be calculated. T_{60} denotes the time it takes for the SPL to decrease by 60 dB.

For an in-depth explanation of reverberation time and room impulse response, see [14, 15] from where the theory presented in this chapter was obtained.

3.2. Room impulse response modeling

To simulate an acoustical channel, a model of a typical RIR needs to be constructed. The sound pressure in a room can be expressed as shown in equation 3.6, where A is the amplitude of the signal, $w(t)$ is a sample function from a stationary white Gaussian noise with $\mathcal{N}(0,1)$[1]. The assumption of a normal distribution is a reasonable model for reverberation, as stated by Schroeder [16]. The exponential denotes the decay of the sound pressure where t is time, and τ is the exponential time constant. It's also important to note that this model is only for the reverberation part/"tail" of the RIR, and the model presented in this section doesn't contain early specular reflections.

$$p(t) = Ae^{\frac{-t}{\tau}} \cdot w(t) \tag{3.6}$$

To find an expression for τ, an expression for the expectation of sound pressured square need to be obtained as presented in equation 3.7, where the expectation of $w(t)$ is obtained from equation 3.8, where $\mathrm{E}(\cdot)$ is the expectation operator.

$$\mathrm{E}\left(p^2(t)\right) = A^2 e^{\frac{-2t}{\tau}} \cdot \mathrm{E}\left(w(t)^2\right) \tag{3.7}$$

$$\mathrm{E}\left(w(t)^2\right) = \mathrm{var}(w(t)) = 1 \tag{3.8}$$

Utilizing equation 3.7 and 3.8 the following expression for the expectation of p^2 is obtained:

$$\mathrm{E}\left(p^2(t)\right) = A^2 e^{\frac{-2t}{\tau}} \tag{3.9}$$

As previously stated, T_{60} is the time it takes for the SPL to decrease by 60 dB, and can be mathematically written as presented in equation 3.10, where $L_p(t)$ is the SPL, $L_{p,start}$ is the SPL at time $t = 0$, and the last fraction models the decay.

$$L_p(t) = L_{p,start} - \frac{60 \cdot t}{T_{60}} \tag{3.10}$$

$L_p(t)$ can be obtained by utilizing equation 3.11.

$$L_p(t) = 10\log\left(\frac{\mathrm{E}\left(p^2\right)}{\mathrm{E}\left(p_{ref}^2\right)}\right) = 10\log\left(\frac{A^2 \cdot e^{\frac{-2t}{\tau}}}{p_{ref}^2}\right) = 20\log\left(\frac{A}{p_{ref}}\right) - \frac{t}{\tau} \cdot 20\log(e^1) \tag{3.11}$$

Readers may notice that equation 3.11 has the same form as equation 3.10. This can be utilized to get an expression for τ as presented in equation 3.12.

[1] $\mathcal{N}(\mu, \sigma)$ is the Gaussian distribution with mean μ and variance σ.

$$\frac{t}{\tau}\log(e^1) = \frac{60 \cdot t}{T_{60}} \Rightarrow \frac{1}{\tau} = \frac{3}{\log(e^1)} \cdot \frac{1}{T_{60}} \approx \tau = \frac{T_{60}}{6.91} \tag{3.12}$$

In addition to the reverberation tail, a direct sound can be added to the model. The model will then contain a direct sound, p_{direct}, followed by the reverberation tail as presented in equation 3.6. The direct sound amplitude is expressed below, where r is the distance between the source and receiver.

$$A = p_{direct} = \frac{1}{r} \tag{3.13}$$

Interested readers are referred to [10, 11, 12] for more information on the topic.

4. Room impulse response measurement methods

Until the 1980s RIR measurements were conducted by utilizing impulsive sources such as balloons, high power amplifier-loudspeaker systems driven with short pulses, and blank pistols. These techniques showed to have poor performance in terms of directivity of the source, repeatability of the measurements, and proper distribution of energy over the entire frequency range [17].

The development of Digital Signal Processing (DSP) made it possible to analyze signals in the frequency domain employing the Discrete Fourier Transform (DFT) and was later improved by the Fast Fourier Transform (FFT) [18]. With this new technology, new measurement techniques based on stochastic signal theory were possible. The most popular method was the dual-channel FFT analyzer based on random excitation signals to determine the systems Transfer function (TF) [19, 20]. The problem with such signals was the stochastic nature of these excitation signals, which required averaging of multiple periods to get a reliable estimate of the RIR.

In the 1990s, new techniques utilizing periodic signals became popular, especially the Maximum Length Sequence (MLS) approach. The main reason for the popularity was the efficient deconvolution through the Hadamard transform [21, 22]. As processors became faster, the efficiency of the Hadamard transform wasn't as crucial as before, which opened the doors for new measurement techniques like the swept-sine [23].

The pursuit of the best RIR measurement signal has been the primary goal of many research papers released in the last couple of years [24]. In this thesis, both pseudo-random signals and swept-sine signals will be studied.

In the following sections of this chapter, the theory regarding three types of Pseudo-Noise (PN) sequences will be presented: The Maximum Length Sequence (MLS/m-sequence), Gold Sequence (GS), and Kasami Sequence (KS). Followed by theory regarding linearly swept sine signals, and finally, the theory behind the advantage of combining PN-sequences and swept sine signals will be presented.

4.1. Maximum length sequence

Pseudo-random or pseudo noise sequences are a group of periodic signals which seem *random-like* but are generated by mathematically precise rules, and statistically, the signals satisfy the requirements of a truly random sequence in the limiting sense [25]. In much of the litterateur on periodic sequences, the terms pseudo-random sequences, pseudo-noise sequences, and m-sequences are used synonymously [26]. However, since the '60s the term pseudo-random

sequence and pseudo-noise sequence have also been used to describe non-maximal-length sequences. The term pseudo-noise sequence and pseudo-random sequence will, in this thesis, be used for all periodic sequences with pseudo-random phase.

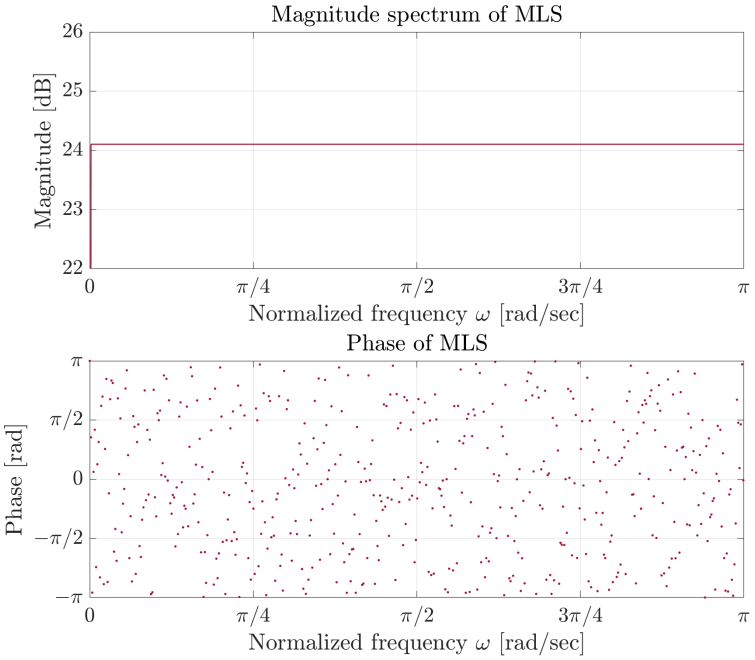


Figure 4.1.: Figure shows the magnitude and phase spectrum of a MLS.

Maximum length sequences (MLS) are a special case of a binary sequence. MLS has a perfectly flat magnitude spectrum (except at 0 Hz) and a pseudo-random phase. As previously stated, the term pseudo-random is used to emphasize that the phase is not purely random since the signal is periodic, and every repetition of the signal has the same phase, but the requirements of a truly random sequence are satisfied. The magnitude spectrum and phase spectrum are presented in figure 4.1.

To benefit from all the properties of PN-sequences, they need to be treated as periodic signals. The period of MLS is $2^m - 1$ samples, where $m \in \mathbb{Z}$. The m-term determined the period of the sequence; this is why the MLS is also known as the m-sequence [17].

4.1.1. Synthesis of MLS

Even though the synthesis of the MLS is in practice straightforward, the theory is rather complicated. An MLS is a binary sequence that satisfies a linear recurrence whose characteristic polynomial is primitive [27]. As the name implies, a binary sequence is a two-values sequence; in computer-logic, the values are typically {0,1}. A primitive polynomial is an irreducible polynomial of degree m, which is the minimum polynomial of the primitive root in GF(2^m), where GF denotes the Galois field. Interested readers are referred to [28] for more information on the

Galois field and other number theory concepts.

A binary polynomial $p(x)$ of degree m can be defined as presented in equation 4.1, where $p_0 = p_m = 1$ and the other polynomials take on values 0 and 1. It's important to note that the degree of the polynomial also denotes the sequence length given by $2^m - 1$.

$$p(x) = p_0 x^m + p_1 x^{m-1} + \ldots + p_{m-1} x + p_m \tag{4.1}$$

It is conventional to represent this polynomial as a binary vector $\vec{p}$ as presented in equation 4.2.

$$\vec{p} = [p_0, p_1, \ldots, p_m] \tag{4.2}$$

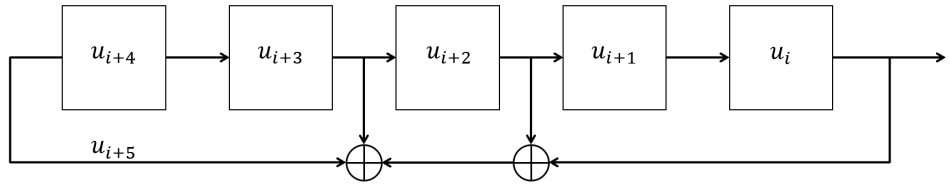
In literature the binary vector is often expressed in octal notation. For example the primitive polynomials $x^5 + x^3 + x^2 + 1$ and $x^5 + x^4 + 1$, are represented by the binary vectors $\vec{p} = [101101]_2$ and $\vec{p} = [110001]_2$, respectively and the octal notation for these polynomials is 55_8 and 61_8 respectively, where the subscript denotes the notation.

A binary sequence u generated by $p(x)$ can be obtained if equation 4.3 is fulfilled for all integers i, where $\oplus$ denotes the modulo 2 addition (XOR-operation).

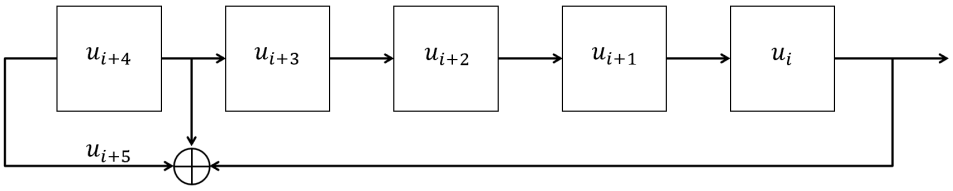
$$p_0 u_i \oplus p_1 u_{i-1} \oplus p_2 u_{i-2} \oplus \ldots \oplus p_m u_{i-m} = 0 \tag{4.3}$$

By replacing i with $i + m$ in equation 4.3, and utilizing the fact that $p_0 = 1$ the following is obtained:

$$u_{i+m} = p_m u_i \oplus p_{m-1} u_{i+1} \oplus \ldots \oplus p_1 u_{i-m-1} \tag{4.4}$$



(a)



(b)

Figure 4.2.: Figure showing a maximal-length sequence generator LFSR. (a) corresponds to polynomial $p(x) = x^5 + x^3 + x^2 + 1$ (55_8). (b) corresponds to polynomial $p(x) = x^5 + x^4 + 1$ (61_8). $\oplus$ is the XOR operation.

This yields that the sequence u can be generated by an m-stage binary Linear-Feedback Shift Register (LFSR) with feedback tap connected to the j'th element cell if $p_j = 1$ for $j \in \langle 0, m]$. LFSR's for the two primitive polynomials $p(x) = x^5 + x^3 + x^2 + 1$ and $p(x) = x^5 + x^4 + 1$ are presented in figure 4.2 (a) and (b) respectively.

Table 4.1.: List of primitive polynomials for order m. List was obtained from [21].

m	$p(x)$	m	$p(x)$
1	$x + 1$	11	$x^{11} + x^2 + 1$
2	$x^2 + x + 1$	12	$x^{12} + x^7 + x^4 + x^3 + 1$
3	$x^3 + x + 1$	13	$x^{13} + x^4 + x^3 + 1$
4	$x^4 + x + 1$	14	$x^{14} + x^{12} + x^{11} + x + 1$
5	$x^5 + x^2 + 1$	15	$x^{15} + x + 1$
6	$x^6 + x + 1$	16	$x^{16} + x^5 + x^3 + x^2 + 1$
7	$x^7 + x + 1$	17	$x^{17} + x^3 + 1$
8	$x^8 + x^6 + x^5 + x + 1$	18	$x^{18} + x^7 + 1$
9	$x^9 + x^4 + 1$	19	$x^{19} + x^6 + x^5 + x + 1$
10	$x^{10} + x^3 + 1$	20	$x^{20} + +x^3 + 1$

If the binary sequence u is a periodic sequence generated by a LFSR, $p(x)$, then for all integers i, $T^i u$ is also a sequence generated by p(x); where T^i denotes the cyclic shift operator. Such an LFSR is therefore capable of generating 2^m different sequences, but the trivial "all 0" sequence must not be included, which yields that the number of sequences generated by an LFSR is $2^m - 1$.

A sequence generated by a LFSR is an MLS, if the period of the generated sequence is $2^m - 1$, thus not all combinations yield an MLS. As previously stated LFSR produces an MLS if, and only if, $p(x)$ is a primitive polynomial [29]. A list of primitive polynomials $p(x)$ obtained from [21] is presented in table 4.1.

In most cases the binary sequence is transmitted as positive and negative pulses [21]. Conventionally the positive and negative pulses are obtained by replacing each 0 of the original binary sequence with a +1 and each 1 with a -1. This mapping can be done by utilizing a function χ as presented in equation 4.5.

$$\chi(\alpha) = (-1)^{-\alpha} \text{ for } \alpha \in \{0, 1\} \tag{4.5}$$

4.1.2. Correlation properties

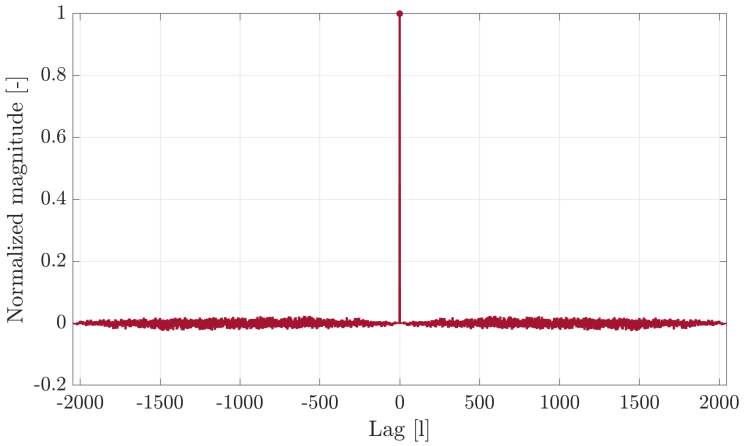


Figure 4.3.: Figure showing the auto-correlation of MLS with period $L = 2047$

One of the most important MLS properties is its close to optimal auto-correlation [29], meaning its autocorrelation approaches a dirac delta pulse. The auto-correlation r_{xx} of an MLS is mathematically expressed in equation 4.6 where L is the sequence length/period, and l is the shift/lag parameter. The derivation of this formula is out of this thesis scope, the reader is referred to [17, 29].

$$r_{xx}[l] = \begin{cases} 1 & \text{if } l = 0 \\ -\frac{1}{L} & \text{if } l \neq 0 \end{cases} \tag{4.6}$$

The auto-correlation of an MLS with period $L = 2047$ is presented in figure 4.3, as can be seen from the figure a clear peak at sample zero can be observed as well as some truncation effects surrounding the correlation peak [30].

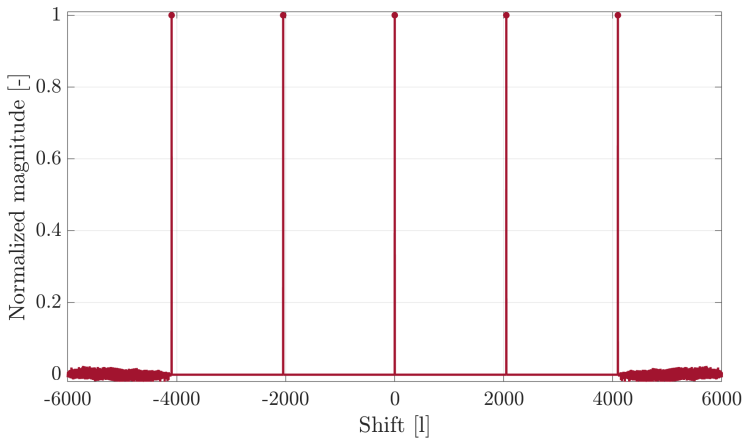


Figure 4.4.: Figure showing 5 cycles of 2047 samples long MLS cross-correlated with 1 cycle of the same MLS.

In this thesis, the purpose is to conduct continuous measurements. Continuous measurements can be done by utilizing an infinite number of cycles of MLS, where one cycle corresponds to one period of the MLS. The introduction of multiple repeated cycles is advantageous; as stated in [30], the truncation effects surrounding the correlation peaks are removed. This can be seen in figure 4.4 where 5 cycles of 2047 sample long MLS are cross-correlated with 1 cycle/period of the same MLS.

4.2. Gold sequence

Even though the auto-correlation properties of the MLS are nearly optimal, the cross-correlation between different sequences varies [21]. In this thesis, the cross-correlation properties are of utmost importance as four continuous source-to-receiver distances will be estimated in parallel; it is important to separate each of the four measurements.

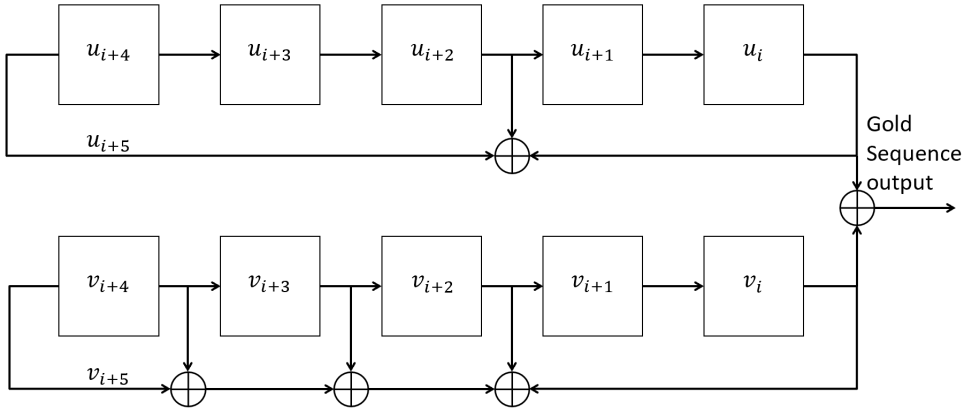


Figure 4.5.: Figure showing a GS generated by a preferred pair of m-sequences, namely: sequence u and sequence v.

In this thesis, two classes of periodic sequences with good cross-correlation properties will be studied, namely: Gold Sequences GS 's and Kasami Sequences KS 's. GS 's are a special class of MLS and were first presented by Robert Gold in his paper in 1967 [31]. These sequences are shown to yield the theoretical minimum cross-correlation that can be expected from periodic MLS [21].

GS's are obtained by modulo-2 addition of an MLS pair. Figure 4.5 is a block diagram of a GS generator where u is one MLS generated by an LFSR with polynomials= $x^5 + x^2 + 1$ and v is another MLS generated by an LFSR with polynomials= $x^5 + x^4 + x^3 + x^2 + 1$. In figure 4.5, u and v are not arbitrary sequences generated by arbitrary polynomials, they are a specific pair of sequences and they are often called a *preferred pair of MLS*. For two sequences with period $L = 2^m - 1$ to be a preferred pair the following criteria most be satisfied [21]:

- m is not divisible by 4 ($m = 2$ or m is odd).
- $v = u[q]$, where: q is odd, $q = 2^k + 1$ or $q = 2^{2k} - 2^k + 1$, and v is obtained by sampling $q'th$ symbol of u.
- $gcd(n, k)$[1]$= \begin{cases} 1 \text{ for } n \equiv 1 \ mod \ 2 \\ 2 \text{ for } n \equiv 2 \ mod \ 4 \end{cases}$

The preferred pairs of MLS has a three valued cross-correlation $r_{u,v}$ as presented in equation 4.7, where L is the sequence length and $t[n]$ is defined as presented in equation 4.8.

$$r_{uv}[l] = \left\{ -\frac{t[n]}{L}, -\frac{1}{L}, \frac{t[n]-1}{L} \right\} \tag{4.7}$$

$$t[n] = \begin{cases} 1 + 2^{\frac{n+1}{2}} & \text{for } n \text{ odd} \\ 1 + 2^{\frac{n+2}{2}} & \text{for } n \text{ even} \end{cases} \tag{4.8}$$

[1]Greatest common denominator

A list containing polynomials of several preferred pairs is presented in table 4.2[2].

Table 4.2.: List of preferred pairs of polynomials for MLS generating.

m	L	**Preferred polynomial 1**	**Preferred polynomial 2**
5	31	$x^5 + x^2 + 1$	$x^5 + x^4 + x^3 + x^2 + 1$
7	127	$x^7 + x^3 + 1$	$x^7 + x^3 + x^2 + x + 1$
9	511	$x^9 + x^4 + 1$	$x^9 + x^6 + x^4 + x^3 + 1$
11	2047	$x^{11} + x^2 + 1$	$x^{11} + x^8 + x^5 + x^2 + 1$
13	8191	$x^{13} + x^4 + x^3 + x^1 + 1$	$x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^6 + x^5 + x + 1$

One preferred pair of MLS can generate $2^m + 1$ unique GS's because the generator utilizes all phases of either u or v this is presented in equation 4.9 where T^i is the cyclic shift operator and $G[u, v]$ denotes the GS. It's also important to note that both u and v are also GS's (i.e the preferred pairs of MLS are also referred to as GS's).

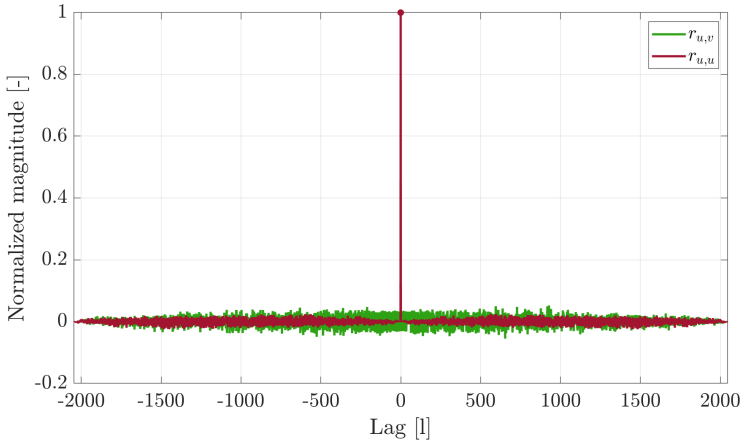


Figure 4.6.: Figure showing the auto-correlation $r_{u,u}$ of GS u, and the cross-correlation $r_{u,v}$ between GS u and v both with 2047 samples length.

$$G[u, v] = \left\{u, v, u \oplus v, u \oplus T \cdot v, u \oplus T^2 v, \ldots, u \oplus T^{L-1} \cdot v\right\} \tag{4.9}$$

The auto-correlation $r_{u,u}$ and cross-correlation $r_{u,v}$ are presented in figure 4.6 for 2047 samples sequence length. It's important to note that the advantages of utilizing multiple cycles and canceling the truncation effects of PN-sequences do not apply to GS 's generated by the modulo 2 addition.

4.3. Kasami sequence

Kasami sequences KS's are defined for even values of m, and there are two classes of KS's, namely: small set of KSs, and large set of KS's [25, 21]. In this thesis, only the small set of KS

[2]These preferred pairs were found in [32]

's are considered.

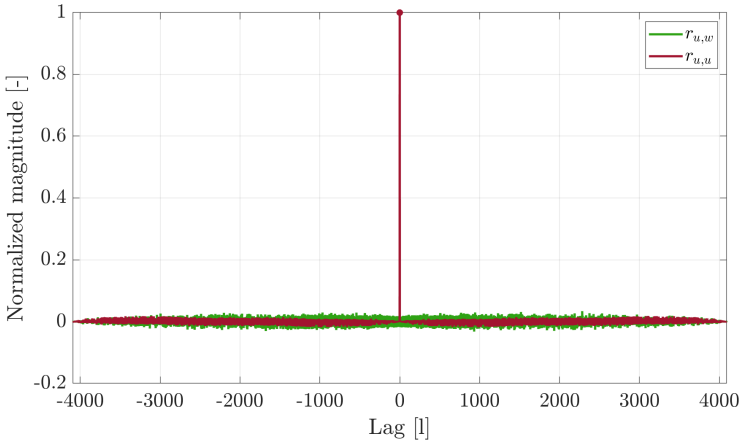


Figure 4.7.: Figure showing the auto-correlation $r_{u,u}$ of Kasami sequence u, and the cross-correlation $r_{u,w}$ between Kasami sequence u and w both with 4095 samples length.

A small set of KS's is a set of $2^{m/2}$ unique sequence each with length $L = 2^m - 1$. KS's utilize the same methodology of modulo two additions of two sequences as previously presented in section regarding GS's. However KS's are obtained by utilizing a MLS, u, with length L and decimating this sequence by $2^{m/2} + 1$. The decimation of MLS results in a new MLS v with length $L = 2^{m/2} + 1$. KS's are then obtained as presented in equation 4.10, where T is the cyclic shift operator and w is a sequence of M_{cycles} cycles of v sequences, where M_{cycles} is calculated as presented in equation 4.11.

$$K[u, w] = \left\{u, w, u \oplus w, u \oplus T \cdot w, u \oplus T^2 w, \ldots, u \oplus T^{L-1} \cdot w\right\} \tag{4.10}$$

$$M_{cycles} = \frac{2^m - 1}{2^{m/2} + 1} \tag{4.11}$$

The cross-correlation of KS's can be calculated as presented in equation 4.12, where $r_{u,w}$ denotes the cross-correlation and $s[n]$ is calculated as presented in equation 4.13 [21]. The auto-correlation of one Kasami sequence and cross-correlation between two unique KS's is presented in figure 4.7.

$$r_{u,w}[l] = \{-1, -s[n], s[n] - 2\} \tag{4.12}$$

$$s[n] = 1 + 2^{m/2} \tag{4.13}$$

4.4. Sequence length

Özdamar and Bohórquez proposed an expression for the SNR of MLS, SNR_{mls}, in [33]. The expression is presented in equation 4.14, where σ_n^2 denotes the noise variance, σ_s^2 denotes the signal variance, and L is the MLS period/length.

$$SNR_{MLS} = 2\sqrt{(1+L)\cdot\frac{\sigma_s^2}{\sigma_n^2}} \tag{4.14}$$

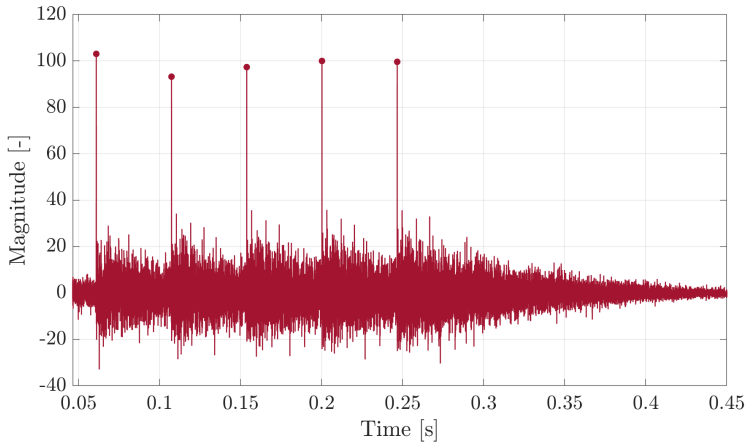


Figure 4.8.: Figure showing wrap around effects caused by late reverberation that add onto next room impulse measurement, where $T_{60} = 0.6$ second and the measurement period was $2047/44100 \approx 0.046$ seconds.

Since in this thesis continues measurements are conducted, continues RIR's will be obtained. Room reverberation will therefore play a significant role when it comes to the so-called *wrap-around effects*. When T_{60} is longer than the measurement period:

$$T_{period} = \frac{L}{F_s} \tag{4.15}$$

where F_s is the sampling frequency,and T_{period} is the measurement period, late reverberations will linger into the next RIR estimate, thus one RIR estimate will wrap-around and add to the next. This is presented in figure 4.8.

By studying equation 4.14 it can be seen that an increase in signal length/period increases SNR. This fact, combined with the wrap-around effects, points to the conclusion that longer sequences are favorable. However; as stated in many papers: time-variance causes errors to MLS based IR measurements [34, 24, 35, 36]. Long sequences yield a long measurement window, which again results in a longer window where time variance caused by factors like temperature changes, source and receiver movement, and other, can occur.

4.5. Impulse response measurements with sweeps

Müller and Massarani presented different methods for acoustical IR measurements, and they concluded that measurements utilizing sweep signals, some times also known as chirps, are more tolerant of time variance and harmonic distortion as compared to PN-sequences [24]. In this thesis, linear (LIN) sweeps will be studied as a possible measurement signal for a proposed system. The following theory about LIN sweeps is heavily based on [17, 24, 37].

4.5.1. Linear sweep

Linear sweeps are considered a "white" excitation signal within the desired frequency range. The term *linear* refers to the linear change of instantaneous frequency as a function of time, and can be mathematically shown as presented in equation 4.16 where f_{LIN} is the linearly changing instantaneous frequency that changes with time t, f_1 and f_2 are respectively the desired lower and upper frequencies, and T is the duration of the sweep.

$$f_{LIN}(t) = \frac{2\pi(f_2 - f_1)}{T}t + 2\pi f_1 \tag{4.16}$$

Equation 4.16 can further be simplified by introducing the bandwidth (BW): $BW = f_2 - f_1$, this is presented in equation 4.17

$$f_{LIN}(t) = 2\pi\frac{BW}{T}t + 2\pi f_1 \tag{4.17}$$

A *sweep factor* can be used to convey the rate of instantaneous frequency change. This factor is constant for LIN sweeps and can be calculated as presented in equation 4.18, where μ is the sweep factor. For non-linear sweeps the time varying sweep factor $\mu(t)$ can be calculated by taking the derivative of the frequency as presented in equation 4.19 where $f(t)$ is the instantaneous angular frequency.

$$|\mu| = \frac{BW}{T} \tag{4.18}$$

$$\mu(t) = \frac{df(t)}{dt} \tag{4.19}$$

Angular frequency is a derivative of phase, thus the argument of a linear sweep can be calculated by taking the antiderivative of the angular frequency as presented in equation 4.20, where $\phi_{LIN}(t)$ is the time varying phase of the LIN sweep, and ϕ_0 is the initial phase.

$$
\begin{aligned}
\phi_{LIN}(t) &= \phi_0 + \int_0^t 2\pi f_{LIN}(t)dt \\
&= \phi_0 + \int_0^t 2\pi \frac{BW}{T}t + 2\pi f_1 dt \\
&= \phi_0 + \left[2\pi \frac{BW}{T}\frac{t^2}{2} + 2\pi f_1 t\right]_0^t \\
&= \pi \frac{BW}{T}t^2 + 2\pi f_1 t + \phi_0 \\
&= |\mu|\pi t^2 + 2\pi f_1 t + \phi_0
\end{aligned}
\tag{4.20}
$$

The waveform of a LIN sweep can be obtained by inserting ϕ_{LIN} as argument of a sine function as presented mathematically in equation 4.21 where $s(t)$ is the sweep signal and $a(t)$ is the sweep amplitude.

$$
s(t) = a(t)sin\left(|\mu|\pi t^2 + 2\pi f_1 t + \phi_0\right) \tag{4.21}
$$

The obtained waveform is presented in figure 4.9 where $a(t) = 1$, $T = 1$, $\phi_0 = 0$, f_1 and f_2 are 20 Hz and 20 kHz respectively.

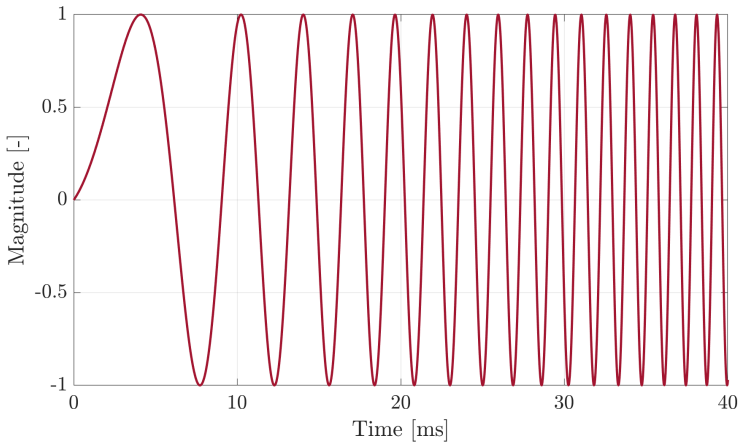


Figure 4.9.: Figure showing the waveform of a LIN sweep. The sweep duration is 1 second and the frequency sweeps from 20 Hz - 20 kHz. Only the first 40 ms are presented in the figure.

The magnitude spectrum of $s(t)$ is presented in figure 4.10. It can be seen from the figure; the magnitude is flat between 100 Hz and 20 kHz except for some ringing at the bandwidth edges. This flat spectrum can intuitively be explained by the fact that the instantaneous frequency changes linearly, this yield that each frequency component is equally excited. Even though f_1 was set to 20 Hz, the spectrum does not become flat until around 100 Hz. This is a result of too short sweep-duration T.

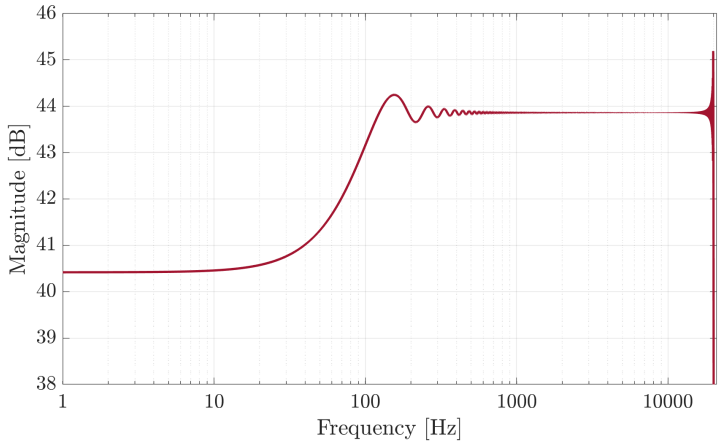


Figure 4.10.: Figure showing the magnitude spectrum of a LIN chirp, the frequencies of interest are in the range between 20 Hz and 20 kHz. The signal is not windowed. The frequencies under 100 Hz do not have enough time to evolve, and the ringing at the start and end of the frequency range is caused by the sudden start and stop of the signal in the time domain.

Synthesizing sweep signals in the time domain result in artifacts and leakage effects in the frequency domain when using DFT. This is caused by the assumption of periodic signal when doing DFT, a sudden start and stop of the signal yield discontinuities resulting in ringing and overshoot of the spectrum at the ends of the frequency range of interest seen in figure 4.10. However, these artifacts can be minimized by utilizing a window function like Hann, Hamming, Tukey, etc.

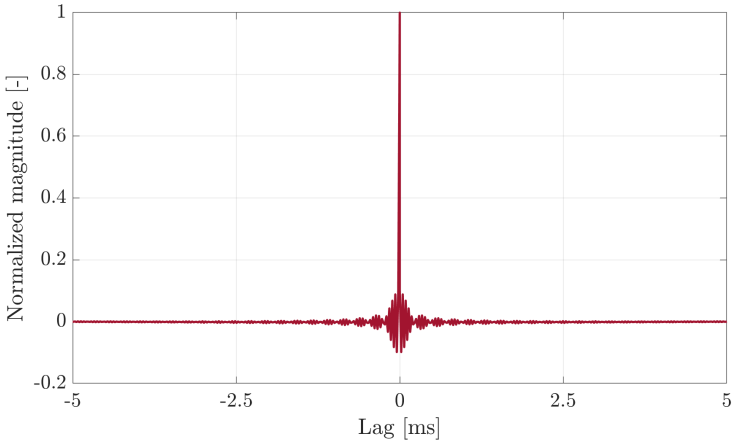


Figure 4.11.: Figure showing the auto-correlation of LIN sweep $s(t)$.

The auto-correlation of a LIN sweep $s(t)$ can mathematically expressed as presented in equation

4.22, where $r_{ss}(t)$ is the auto-correlation [37].

$$r_{ss}(t) = \sqrt{BW \cdot T}\frac{\sin\left(\pi BWt\left(1-\frac{|t|}{T}\right)\right)}{\pi BWt}\cos\left(2\pi f_0 t\right) \tag{4.22}$$

The auto-correlation $r_{ss}(t)$ is presented in figure 4.11, the "sidelobes" surrounding the auto-correlation peak can be reduced by utilizing a window function, however; this results in a reduction of maximal amplitude.

4.6. PN-sweep

As previously stated, PN-sequences have good cross-correlation properties, especially when considering Gold and KS 's, as well as excellent auto-correlation properties; however, the sequences are sensitive to time variance. Sweep signals, on the other hand, are almost immune to time-variance; thus, the advantage of combining the two methods is that by coding multiple sweep signals with different uncorrelated PN-sequence, multiple RIR estimates can be obtained simultaneously. The combination makes it possible to simultaneously conduct multiple measurements and gain higher immunity to time-variance compared to the PN-sequences [38]. The combination of these two measurement signals will, from now on, be referred to as *PN-sweep*.

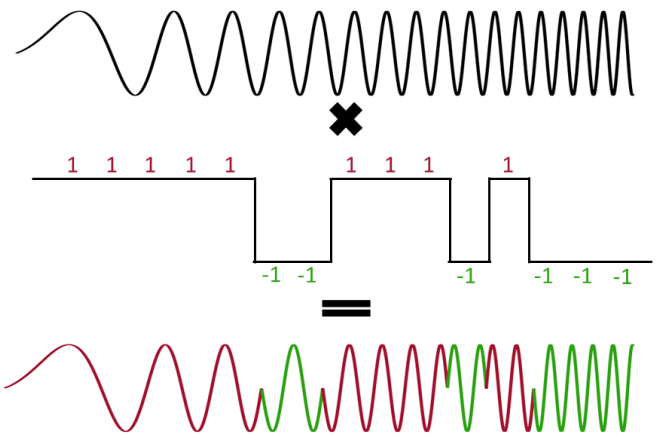


Figure 4.12.: Schematic representation of a sweep coded with a 4'th order MLS.

PN-sweeps are obtained by multiplying a sine-sweep with a PN-sequence (e.g., MLS). The PN-sequence is binary with amplitude -1 and 1, a change in PN-sequence amplitude results in a $180°$ phase-shift of the sine-sweep; this is presented in figure 4.12.

Since the PN-sweep is a combination of both a swept sine and a PN-sequence, the resulting auto-correlation and cross-correlation will also be a product of that. The auto-correlation of a

PN-sweep and cross correlation of two PN-sweeps each consisting of a 10 ms long LIN sweep with frequency sweep from 2 kHz up to 20 kHz coded with a 9'th order GS is presented in figure 4.13.

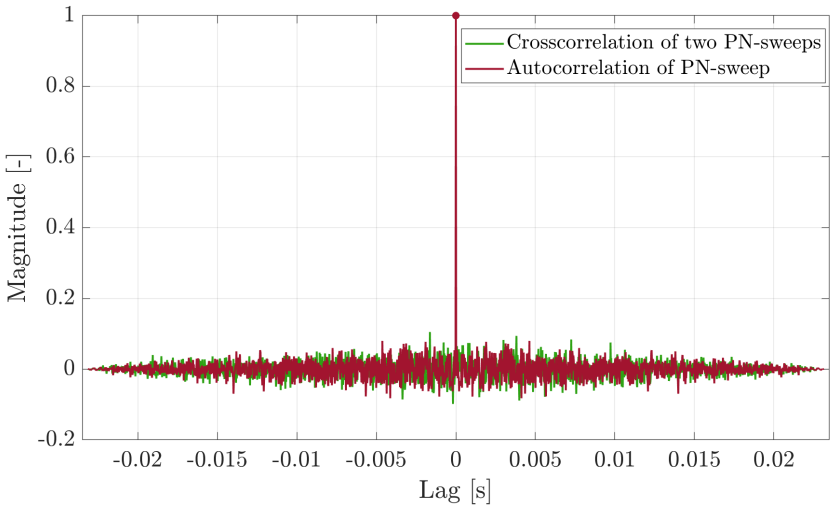


Figure 4.13.: Figure showing the auto-correlation of PN-sweep and the cross-correlation between two different PN-sweeps.

5. Audio-based positioning

In this chapter, the theory utilized in the thesis for distance and position estimation will be presented.

5.1. Single source to receiver measurements

Historically the first modern localization applications were the underwater sonar and radar [1]. In sonar, the distance from faraway objects at a certain angle is estimated by the travel time necessary for a sound wave to go from the source to the target and back. The travel time is often called the Time of Arrival (ToA).

One of the goals of this thesis is to calculate the source-to-receiver distance between one loudspeaker and one microphone; this can be done by measuring the ToA as presented in figure 5.1.

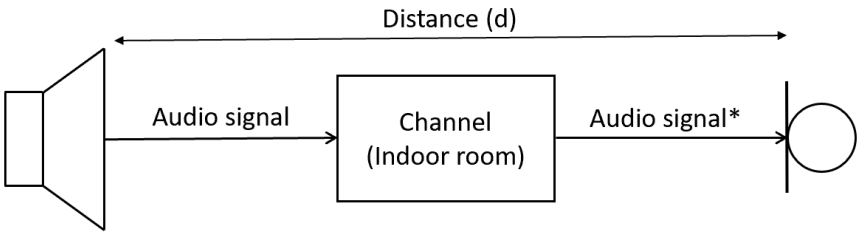


Figure 5.1.: Schematic representation of distance measurements between a loudspeaker and a microphone using audio signals. The * denotes that the audio signal has been convolved with the channel, in this case the channel is the impulse response of the room.

Assuming that sound travels at a constant speed, the distance in meters, d, as a function of ToA in second, t_{ToA}, and sound speed in the air c_{air} in m/s can be seen in equation 5.1.

$$d = t_{ToA} \cdot c_{air} \tag{5.1}$$

However in the digital domain the ToA will be calculated by an integer number of samples, n_{ToA}. Distance estimator $\hat{d}$ in this domain can be calculates as seen in equation 5.2 where F_s is the sampling frequency in Hz.

$$\hat{d} = \frac{n_{ToA}}{F_s} \cdot c \tag{5.2}$$

5.2. 3D positioning

The main goal of this thesis is to study a sound-based 3D-localization system. In this section, the localization of receivers relative to a source and the source's choice will be explained.

In this thesis, the source was assumed to be a spherical speaker array consisting of 4 speaker elements in a tetrahedron configuration. The placement of the four elements (numbered 1 to 4) on the sphere is schematically presented in figure 5.2.

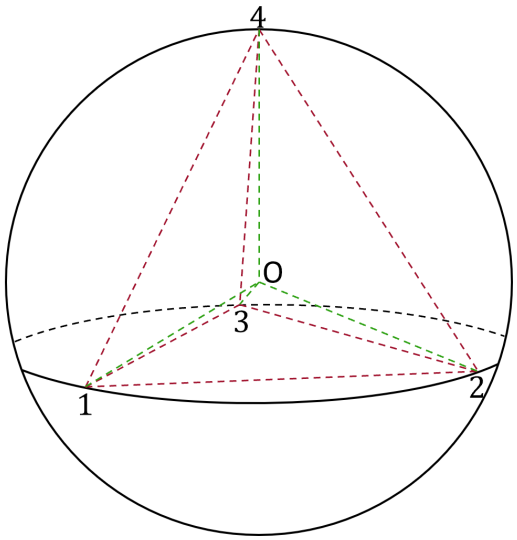


Figure 5.2.: Figure showing a schematic representation of a tetrahedron configuration of a four-element spherical speaker array. The numbers denote speaker placement, and O denotes the origin/center of the sphere.

In the late 1960s, Olson documented a comprehensive analysis of the influence of the diffraction around rigid boxes (speaker enclosure geometries) [39]. The analysis showed that the spherical shape had the flattest frequency response of all the tested geometries. This is one of the biggest advantages of utilizing a spherical enclosure as this shape will "color" the measurement signal the least, as well as this geometry allows simple implementation of the tetrahedron speaker configuration. Furthermore, the omnidirectionally of the source will only depend on the radius of the sphere. It is important to note that in this thesis, the *free-field* case will be studied, and distance estimates for diffraction around enclosures will not be studied.

5.2.1. Multiple sources to receiver distance estimation

The symmetry of the tetrahedron configuration yields a simple mathematical expression for calculating the ToA between the center of the spherical enclosure and a given microphone. This can be shown by considering a microphone m_j and a speaker element s_i, their position relative to the origin/center of the speaker enclosure can be expressed on vector form as presented in equation 5.3 where the index denotes the x-, y-, and z-coordinates.

$$\vec{s_i} = \begin{bmatrix} s_{i,x} \\ s_{i,y} \\ s_{i,z} \end{bmatrix}, \vec{m}_j = \begin{bmatrix} m_{j,x} \\ m_{j,y} \\ m_{j,z} \end{bmatrix} \tag{5.3}$$

The source-to-receiver distance between source s_i and receiver m_j can be expressed as a vector $\vec{x}_{i,j}$, this vector can be calculates as presented in equation 5.4.

$$\vec{x}_{i,j} = \vec{s_i} - \vec{m}_j \tag{5.4}$$

The estimated ToA between speaker element i and microphone j can be calculated by utilizing equation 5.5 where $\hat{T}_{i,j}$ is the ToA estimate and c_{air} is the sound speed in air in m/s.

$$\hat{T}_{i,j} = \frac{|\vec{x}_{i,j}|}{c_{air}} \tag{5.5}$$

Now considering the 2D situation presented in figure 5.3 $|\vec{x}_{i,j}|$ can be calculated by considering the two right-triangles. The first right-triangle consists of $\vec{s_i}$, $\vec{g}$, and $\vec{h}$, and the second consists of $\vec{m}_i$, $\vec{x}_{i,j}$, and $\vec{h}$ where $\vec{g}$ is the scalar projection of $\vec{s_i}$ on $\vec{m}_j$, and $\vec{h}$ denotes the shortest path between vector m_j and the source.

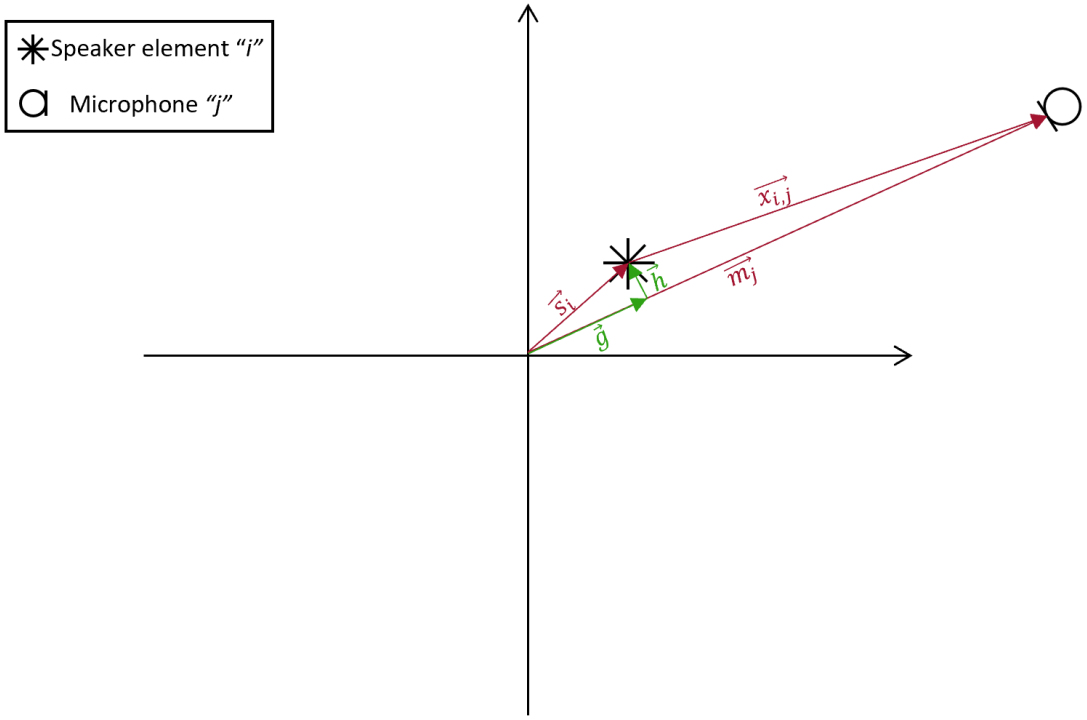


Figure 5.3.: Figure showing schematic representation of a 2D localization situation.

Using the Pythagorean theorem $|\vec{x}_{i,j}|$ can be calculated as presented in equation 5.6 and the scalar projection can be calculated by the means of the dot product as presented in equation 5.7.

$$|\vec{x}_{i,j}|^2 = (|\vec{m}_j| - |\vec{g}|)^2 + |\vec{h}|^2 \tag{5.6}$$

$$|\vec{g}| = \vec{s_i} \cdot \frac{\vec{m}_j}{|\vec{m}_j|} \tag{5.7}$$

In this thesis the approximation presented in equation 5.8 will be true in all cases. By utilizing this approximation and combining equation 5.6 and 5.7, the source-to-receiver distance can be calculated as presented in equation 5.9.

$$\vec{m}_j >> \vec{s}_i \rightarrow \vec{h} \approx 0 \tag{5.8}$$

$$|\vec{x}_{i,j}|^2 \approx (|\vec{m}_j| - \vec{s_i} \cdot \frac{\vec{m}_j}{|\vec{m}_j|})^2 \Rightarrow |\vec{x}_{i,j}| \approx (|\vec{m}_j| - \vec{s_i} \cdot \frac{\vec{m}_j}{|\vec{m}_j|}) \tag{5.9}$$

Now lets study the case of the spherical speaker array consisting of 4 elements in tetrahedron configuration. Assuming plane waves ($\vec{m}_j >> \vec{s}_i$) and symmetry of the tetrahedron speaker

array the mean of the four ToA's will denote the ToA from the spheres origin. The estimated mean ToA from the four speaker elements to one microphone can be obtained by calculated the mean of both sides of equation 5.5 as presented in 5.10 where i denotes the speaker element.

$$\frac{1}{4}\sum_{i=1}^{4}\hat{T}_{i,j} = \frac{1}{4c}\sum_{i=1}^{4}|\vec{x}_{i,j}| \tag{5.10}$$

By utilizing the approximation in 5.9 in equation 5.10 the following is obtained:

$$\frac{1}{4}\sum_{i=1}^{4}\hat{T}_{i,j} \approx \frac{1}{4c}\sum_{i=1}^{4}\left(|\vec{m}_j| - \vec{s}_i \cdot \frac{\vec{m}_j}{|\vec{m}_j|}\right) \tag{5.11}$$

By further simplification the following expression is obtained:

$$\frac{1}{4}\sum_{i=1}^{4}\hat{T}_{i,j} \approx \frac{1}{4c}\left(\sum_{i=1}^{4}|\vec{m}_j| - \sum_{i=1}^{4}\vec{s}_i \cdot \frac{\vec{m}_j}{|\vec{m}_j|}\right) \Rightarrow \frac{1}{c}\vec{|}m_j| - \frac{\vec{m}_j}{4\cdot|\vec{m}_j|}\sum_{i=1}^{4}\vec{s}_i \tag{5.12}$$

Because of symmetry the sum of the speaker element vectors can be calculated to be:

$$\sum_{i=1}^{4}\vec{s}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.13}$$

By utilizing this and solving for $|\vec{m}_j|$ an estimate of the source-to-receiver distance can be calculated as:

$$|\hat{\vec{m}}_j| = \frac{c}{4}\sum_{i=1}^{4}\hat{T}_{i.j} \tag{5.14}$$

5.2.2. Direction of arrival estimation

As the estimate of the source-to-receiver distance has been calculated, it's time to calculate an estimated Angle of Departure (AoD) of the microphone relative to the origin of the speaker array. In this thesis, only the AoD in the x,y-plane is of interest. The AoD θ relative to the center of spherical speaker array is presented in figure 5.4.

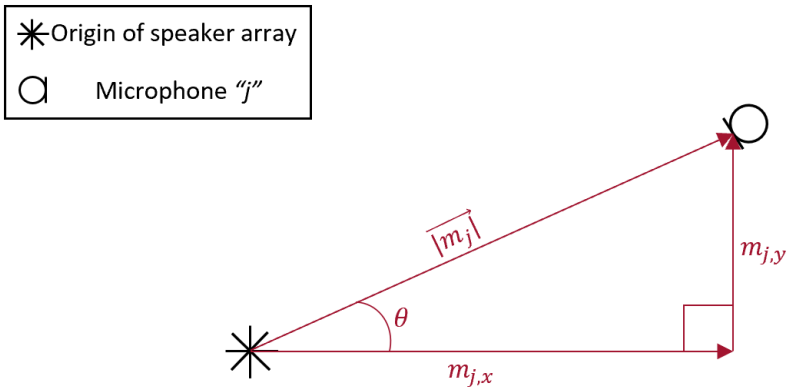


Figure 5.4.: Figure showing schematic representation of the azimuth angle in respect to the origin of the speaker array.

To obtain an estimate of the azimuth $\hat{\theta}$, simple trigonometrical operations can be utilized as presented in equation 5.15, where $|\hat{\vec{m}}_j|$ is the estimated source-to-receiver distance presented in equation 5.14 and $\hat{m}_{j,y}$ is the y-coordinate estimate.

$$\hat{\theta} = \arcsin\left(\frac{\hat{m}_{j,y}}{|\hat{\vec{m}}_j|}\right) \tag{5.15}$$

In this operation the angle θ is wrapped in range [−90°,90°], to find the estimated unwrapped angle $\hat{\theta}_{unwrapped}$ (angle in range [0°,360°]) the following condition is applied:

$$\hat{\theta}_{unwrapped} = \begin{cases} 180° + \hat{\theta} & \text{for } \hat{m}_{j,x} < 0 \\ \hat{\theta} & \text{for } \hat{m}_{j,x} \geq 0 \end{cases} \tag{5.16}$$

To solve equation 5.15 an expression for the microphone coordinates needs to be found. This can be done by defining a vector containing 4 estimates of ToA (one for each speaker element). The ToA vector is defined as:

$$\hat{\vec{T}}_j = \begin{bmatrix} \hat{T}_{1,j} \\ \hat{T}_{2,j} \\ \hat{T}_{3,j} \\ \hat{T}_{4,j} \end{bmatrix} \tag{5.17}$$

A matrix denoting the position of each speaker element in respect to the origin of the spherical enclosure can be expressed as presented in equation 5.18 where $\widetilde{S}$ denotes the position matrix [1]. The rows of the matrix on the right hand side of the equation denote the speaker element and the columns denote x, y and z coordinates respectively. $|\vec{s}_i|$ is the distance to the speaker

[1]It is important to note that this matrix assumes that all speaker elements are in free-field.

element from origin.

$$\widetilde{S} = |\vec{s}_i| \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \tag{5.18}$$

$\hat{\vec{m}}_{i,j}$ can be calculated by adapting equation 5.5 and 5.9 to matrix form as presented in equation 5.19, here it is important to note that the inverse matrix $\widetilde{S}^{-1}$ cannot be calculated directly but it can be calculated as a Moore-Penrose pseudoinverse matrix.

$$\hat{\vec{m}}_j = |\vec{m}_j| \cdot (\widetilde{S}^{-1} \cdot (|\vec{m}_j| - c \cdot \hat{\vec{T}}_j)) \tag{5.19}$$

5.3. Retrieving the time of flight from room impulse response estimates

Since the system's sampling frequency is finite, the direct sound might be recorded at non-integer time-steps of the RIR. This yields that the spacial resolution of the system is bounded as expressed below.

$$d_{min} = \frac{1}{F_s} \cdot c_{air} \tag{5.20}$$

$$c_{air} \approx 331.6 + (0.606 \cdot T_{temp}) \tag{5.21}$$

where: d_{min} is the smallest measurable distance, F_s is the sampling frequency and c_{air} is the speed of sound where c_{air} can be calculated as presented in equation 5.21 where T_{temp} is the air temperature. This means that all microphone positions within d_{min} will result in a direct sound at the same sample. Since the sampling frequency is fixed, the sampling frequency can be increased in post processing, this technique is called upsampling. In this thesis upsampling will be done by means of 1D linear interpolation as presented in figure 5.5.

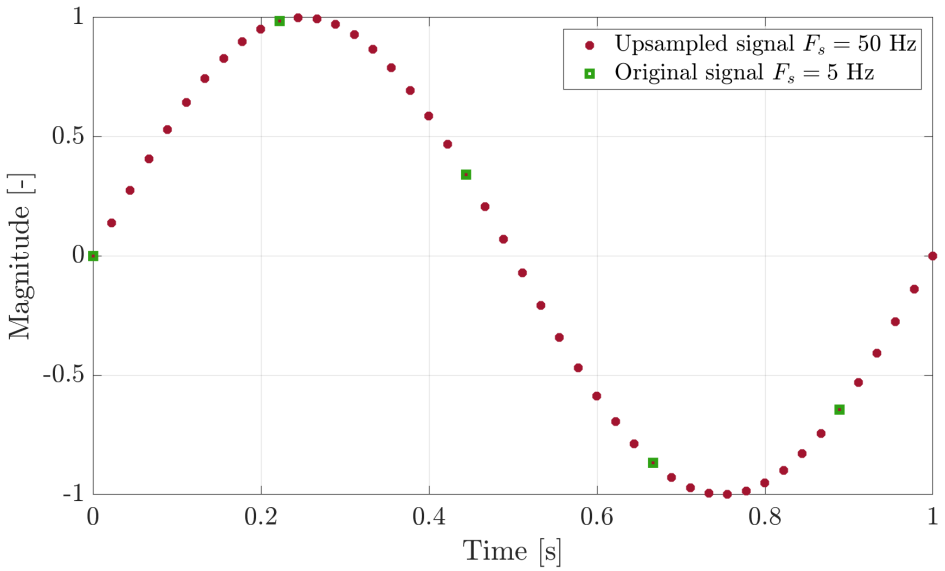


Figure 5.5.: Figure showing a 1 Hz sine wave sampled with a sampling frequency of $F_s = 5$ Hz and the same signal upsampled by factor 10 resulting in a sampling frequency of $F_s = 50$ Hz.

The interpolation presented in the figure would result in a minimum distance that is ten times smaller than the original sampled signal.

6. Signal processing

6.1. Signal-to-noise ratio

The signal-to-noise ratio (SNR) is defined as the ratio between, power of a signal P_{signal}, and power of noise P_{noise}. This ratio in dB is represented mathematically in equation 6.1 where A_{signal} and A_{noise} are respectively, the root mean square (RMS) values of signal and noise.

$$SNR = 10\log(\frac{P_{signal}}{P_{noise}}) = 20\log(\frac{A_{signal}}{A_{noise}}) \tag{6.1}$$

For simulation purposes it is desirable to set the SNR. This can be done by calculating a scaling factor as presented in equation 6.2 where $SNR_{scaling}$ is the scaling factor, SNR is the desired SNR, and A_{noise} is the RMS value of the noise signal.

$$SNR_{scaling} = 10^{\frac{SNR}{20}} \cdot A_{noise} \tag{6.2}$$

When the scaling factor is obtained the signal needs to be scaled as presented in equation 6.3, where $s_{original}$ is the signal which is being scaled, and A_{signal} is the RMS value of signal $s_{original}$.

$$s_{scaled} = s_{original} \cdot \frac{SNR_{scaling}}{A_{signal}} \tag{6.3}$$

To represent the audible SNR with a single number, an A-weighted SNR, SNR_A, was used in previous studies like the one presented in [40]. SNR_A is obtained by first filtering the signal and noise with an A-weighting filter and using these filtered values to calculate the RMS of both signal and noise. These RMS values are then utilized in equation 6.1 however resulting in SNR_A.

6.2. Signal detection

Some form of noise distorts all practically and simulated measurements. Correlation can be used to detect the known measurement signal embedded in noise. The objective of correlation is to measure the degree to which two signals are similar. In [10] two types of correlations are defined, namely the cross-correlation and auto-correlation as presented in equation 6.4 and 6.5 respectively. The cross-correlation is a measure of the degree of similarity between two different signals, e.g., two signals: $x[n]$ and $y[n]$. The expression for the cross-correlation between these two signals is the sequence $r_{xy}[l]$. The auto-correlation is a measure of the degree of similarity between the same signal, e.g., $x[n]$, and is expressed as the sequence $r_{xx}[l]$. In both cases, the index l is the shift/lag parameter.

$$r_{yx}[l] = \sum_{n=-\infty}^{\infty} y[n]x[n-l] = y[l] * x[-l] \tag{6.4}$$

$$r_{xx}[l] = \sum_{n=-\infty}^{\infty} x[n]x[n-l] = x[l] * x[-l] \tag{6.5}$$

By utilizing the definition of the received signal $y[n]$ from equation 3.5 and the cross-correlation definition from equation 6.4 the following expression for the cross-correlation between the transmitted and received signal can be obtained:

$$r_{yx}[l] = y[l] * x[-l] = h_{channel}[l] * (x[l] * x[-l]) = h_{channel} * r_{xx}[l] \tag{6.6}$$

This means that the cross-correlation between $x[n]$ and $y[n]$ is determined by the auto-correlation of the input signal convolved with the IR of the channel. If r_{xx} is a Dirac delta pulse $y[n]$ would be equal to $h_{channel}$.

6.3. Matched filtering

A practical implementation of cross-correlation is the matched filter. Matched filtering is a process for detecting a known piece of a signal embedded in noise, as explained in [41]. The main task of a matched filter is to maximize the SNR. Figure 6.1 shows a schematic representation of a known signal $s[n]$, embedded in noise $n[n]$, which is passed through a matched filter.

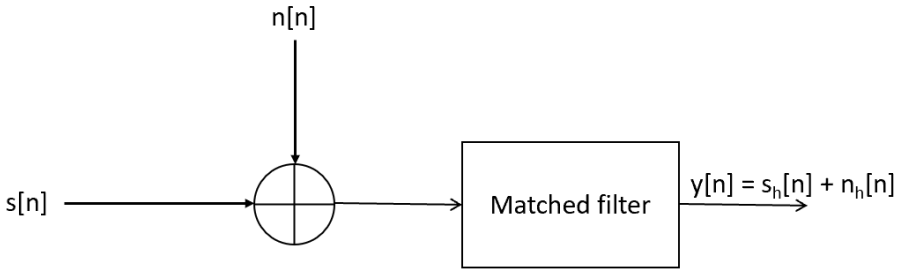


Figure 6.1.: Schematic representation of a signal, $s[n]$, embedded in noise, $n[n]$, passed through a matched filter. Index h denotes that the signal was convolved with the filter.

If the input signal, $s[n]$, is a known signal with good auto-correlation properties (close to a Dirac delta pulse), and $n[n]$ is white noise, then matched filter theory states the maximum SNR at the output will occur when the filter has an impulse response that is the time-reverse of the known signal $s[n]$. This means that the impulse response of the matched filter $h_{matched}[n]$ is defined as:

$$h_{matched}[n] = s[L-n] \tag{6.7}$$

where L is the length of the known signal.

It's important to note that $s[n]$ can be a continuous periodic signal, and $h_{matched}[n]$ could be one period of such signal. In this case, maximum SNR would be obtained when the phase of $s[n]$ would be the same as $h_{matched}[n]$.

As defined in equation 6.5, the convolution of a signal with the time-reversed of the same signal results in the auto-correlation sequence.

6.4. Pulse compression

Inaudible measurement sound can be achieved by utilizing sound signals that are either "hidden"/masked by the acoustical background noise in the room where the measurements are conducted ($SNR < 0$), or by using sound with levels below the threshold of audibility. The former alternative can be achieved by utilizing pulse compression.

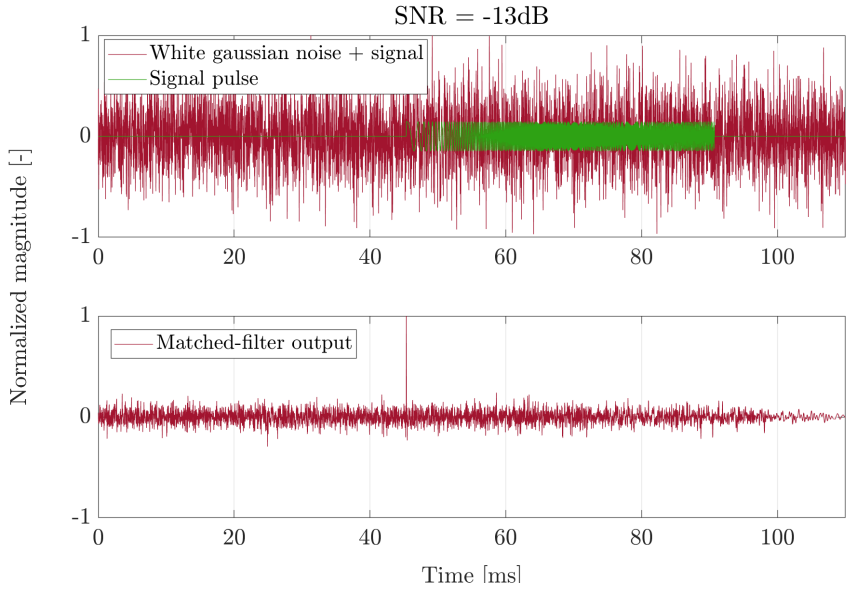


Figure 6.2.: Example showing how matched-filtering brings out a signal buried in noise.

The signals need to be designed so that the width of the signal passed through a matched filter is smaller than the width obtained by a standard sinusoidal pulse, hence the name of this technique: pulse compression. As stated in [42].

The pulse compression aspect of the matched filter results in a good probability of detection for signals well below the noise floor, and this can be seen in figure 6.2 where the signal buried in noise was a LIN sweep, and the SNR was -13 dB [43, ch.11]. From the figure, a peak at the output of the matched filter can be observed; this peak corresponds to the start time of the LIN sweep.

6.5. Signal averaging

Signal averaging is a method used in signal processing to increase SNR. Signal averaging is used when periodic signals in random uncorrelated noise are used. By aligning corresponding samples of a periodic signal and summing these time-aligned samples, the amplitude will increase with the number of aligned periods. However, if the noise is random and has zero mean, the RMS will increase with the root of the number of sums [44].

This can be mathematically presented by considering a sampled signal $s[n]$ which consists of a periodic signal $x[n]$, and random uncorrelated noise $n[n]$ as presented in equation 6.8.

$$s[n] = x[n] + n[n] \tag{6.8}$$

If N additions of the signals are done the summed signal can be expressed as:

$$\sum_{n=1}^{N} s[n] = \sum_{n=1}^{N} x[n] + \sum_{n=1}^{N} n[n] \tag{6.9}$$

The periodic signal component for sample point i is the same at each repetition if the signal is stable and the summations are aligned perfectly. This yields:

$$\sum_{n=1}^{N} x[n] = N \cdot x[n] \tag{6.10}$$

Considering the noise component the summation after many repetitions will result in the RMS of the noise. This is presented in equation 6.11.

$$\sum_{n=1}^{N} s[n] = \sqrt{N\sigma_n^2} = \sqrt{N} \cdot \sigma_n \tag{6.11}$$

By comparing the result of equation 6.10 and 6.11 it can be seen that the amplitude of the known signal will increase as a function of N, and the noise signal will increase with the square root of N. This fact yields the following expression for the SNR obtained after averaging N periods of the signal:

$$SNR_{AVG} = \frac{N \cdot x[n]}{\sqrt{N} \cdot \sigma_n} = \sqrt{N} \cdot SNR \tag{6.12}$$

where SNR_{AVG} is the SNR obtained from averaging and SNR is without averaging. It can be seen that the SNR will increase with the square root of the number of sums.

6.6. Power spectral density

In information theory, additive white Gaussian noise (AWGN) is often used as a model to mimic the behavior of many random processes that are found in nature [45]. However, AWGN has a flat frequency, which is rarely a good representation of acoustical background noise [46].

It's impossible to make a model that describes the background noise in all rooms as they vary in size, acoustical properties, and have different sources of noise. Therefore, it is essential to estimate the frequency content of each specific room's background noise and use it as a model. This can be done by assuming that the background noise is a stationary random process. However, this is just an assumption as transient noise like footsteps, speech, and knocking sound will occur in most rooms. A good way to estimate the noise content of a room is to estimate its power spectral density.

Power spectral density (PSD) is the measure of the density of power in a stationary random process, $X(t)$ per unit of frequency [10]. Using the Wiener-Khinchin theorem the PSD for a wide sense stationary random process can be calculated as:

$$S_{xx}(f) = \int_{-\infty}^{\infty} r_{xx}(\tau) e^{-j2\pi f\tau} d\tau \tag{6.13}$$

where Sxx is the power density, and $r_{xx}(\tau)$ is the auto-correlation process of signal $X(t)$ given by:

$$r_{xx}(\tau) = \mathrm{E}\left[X(t)X(t-\tau)\right] \tag{6.14}$$

where $\mathrm{E}(\cdot)$ denotes the expectation operator. Equation 6.13 is only valid for wide-sense stationary process as its auto-correlation function is only a function of the time lag τ and not the absolute time t. Thus in most practical cases the PSD needs to be estimated. A popular approach for PSD estimation is the periodogram. Using the periodogram approach the PSD can be calculated as presented in equation 6.15, where $\hat{S}_{xx}(f)$ is the estimated PSD, N is the window size, and $X(f)$ is the frequency content of $X(t)$ obtained by utilizing the FFT of the signal with the window size N. This equation is derived in chapter 14 in [10], and will therefore not be derived in this report.

$$\hat{S}_{xx}(f) = \frac{1}{N}|X(f)|^2 \tag{6.15}$$

A popular averaging method used for PSD is the Bartlett method, which involves three steps. First, the N-point sequence is subdivided into K non-overlapping segments, each having length M. The K segments are then averaged to obtain the Bartlett power spectrum estimate. Interested readers are referred to [10, sec. 14.2.1] for full derivation.

6.6.1. Auto-regressive process

Figure 6.3 is an idealized frequency plot of an MLS buried in background noise. As can be seen from the figure, the SNR will vary between frequency banks. This is unfavorable as a lower SNR yields a lower probability of detection. By filtering the measurements signal to "mimic" the background noise, a more even SNR for all frequency banks would be achieved, thus maximizing the probability of detection.

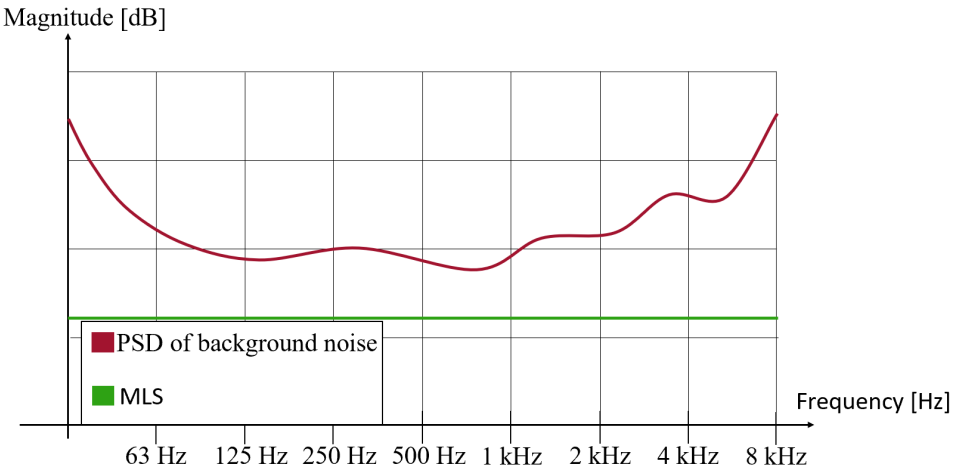


Figure 6.3.: Schematic representation of MLS buried in acoustical background noise.

An auto-regressive process (AR-process) is a way of modeling weak stationary processes as a output signal $x[n]$, from a causal LTI system $h_{AR}[n]$, which is subject to white noise $w[n]$. I.e:

$$x[n] = h_{AR}[n] * w[n] \tag{6.16}$$

In an AR-processes, $h_{AR}[n]$ is an all-pole filter that can be described as:

$$H_{AR}(z) = \frac{1}{1 + \sum_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)} \tag{6.17}$$

where $H_{AR}(z)$ is the Z-transorm of $h_{AR}[n]$, a is the filter coefficient, p denotes the filter order, and z is the Z-operator. To find the filter coefficients for a given AR order (a given number of filter coefficients) the Yule-Walker equation can be used [10, sec. 12.2.2]. The Yule-Walker equation is presented in 6.18 where γ_{xx} is the auto-correlation sequence of x and a are the filter coefficients.

$$\begin{pmatrix} \gamma_{xx}(0) & \gamma_{xx}(-1) & \gamma_{xx}(-2) & \cdots & \gamma_{xx}(-p) \\ \gamma_{xx}(1) & \gamma_{xx}(0) & \gamma_{xx}(-1) & \cdots & \gamma_{xx}(-p+1) \\ \vdots & \vdots & \vdots & & \vdots \\ \gamma_{xx}(p) & \gamma_{xx}(p-1) & \gamma_{xx}(p-2) & \cdots & \gamma_{xx}(0) \end{pmatrix} \cdot \begin{pmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} \gamma_{xx}(1) \\ \gamma_{xx}(2) \\ \vdots \\ \gamma_{xx}(p) \end{pmatrix} \tag{6.18}$$

To efficiently solve this equation, the Levinson-Durbin Algorithm can be utilized [47].

7. Experiments

7.1. Detecting the direct sound arrival time using a simple room simulator

In a first experiment, a simulator, simulating a single source-to-receiver measurement situation consisting of one loudspeaker and one microphone, was programmed in MATLAB. In this simulator, the source and receiver, as well as the entire measurement chain, were assumed to be ideal. A block diagram of the simulator is presented in figure 7.1. In the following sections, each of the blocks of the simulator will be explained.

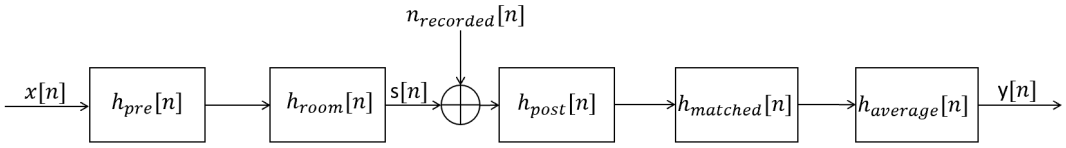


Figure 7.1.: Block diagram of source-to-receiver measurement simulator.

The signal $x[n]$ was a signal consisting of N-number of cycles of MLS with L sequence length. The MLS was generated by utilizing the builtin *mls*-function in MATLAB[1]. The signal $x[n]$ was then convolved with filter $h_{pre}[n]$ by utilizing the *conv*-function in MATLAB.

Filter $h_{pre}[n]$ was constructed by convolving two filters, as presented in figure 7.2, namely: $h_{shaping}$ and h_{HP}.

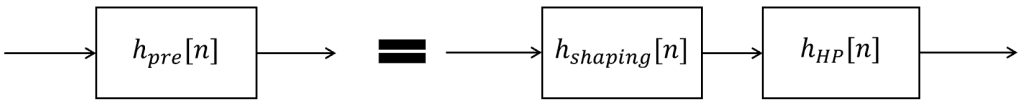


Figure 7.2.: Block diagram showing the construction of filter $h_{pre}[n]$

The filter $h_{shaping}$ was a 2'nd order filter obtained by utilizing the *aryule*-function in MATLAB and was a filter that shaped the measurement signal to mimic the frequency spectrum of the background noise. This function generates filter coefficients of the p'th order AR-process by utilizing the Levinson-Durbin algorithm as previously presented in section 6.6.1. The AR-model

[1]URL to MLS-function: `https://se.mathworks.com/help/audio/ref/mls.html`

was of the stationary part of the recorded background noise, $n_{recorded}$. The stationary part was found by listening to the recording through uncalibrated headphones. Stationary is defined as *parts of the recording where no transient noise, like sound of footsteps, and knocking, occurred.* Filter h_{HP} was a 1'st order high-pass Butterworth filter with a cut-off frequency at 500 Hz [2]. The reason for the high-pass filter is to firstly filter the signal to match the frequency response of the loudspeaker, and secondly to filter the DC-offset caused by $h_{shaping}[n]$. The Z-transform of $h_{shaping}[n]$ and $h_{HP}[n]$ had the form as presented in equation 7.1 and 7.2 respectively, where a_i and b_i coefficients were obtained by the respective MATLAB-functions, and $H(z)$ denotes the Z-transform of the respective filter.

$$H_{shaping}(z) = \frac{1}{1 + a_{1,shaping} \cdot z^{-1} + a_{2,shaping} \cdot z^{-2}} \tag{7.1}$$

$$H_{HP}(z) = \frac{b_{0,HP} + b_{1,HP} \cdot z^{-1}}{1 + a_{1,HP} \cdot z^{-1}} \tag{7.2}$$

This yields that the Z-transformed $h_{pre}[n]$, $H_{pre}(z)$, was obtained by solving the following equation:

$$\begin{aligned} H_{pre}(z) &= H_{shaping}(z) \cdot H_{HP}(z) \\ &= \frac{b_{0,HP} + b_{1,HP} z^{-1}}{1 + (a_{1,shaping} + a_{1,HP}) z^{-1} + (a_{1,shaping} a_{1,HP} + a_{2,shaping}) z^{-2} + a_{1,HP} a_{2,shaping} z^{-3}} \end{aligned} \tag{7.3}$$

Filter $h_{pre}[n]$ was then obtained by utilizing *impz*-function[3] in MATLAB by inserting the a_i and b_i values of $H_{pre}(z)$. The signal obtained from convolving $x[n]$ with $h_{pre}[n]$ was then convolved with $h_{room}[n]$ which was a simulated RIR resulting in signal $s[n]$. The RIR was simulated by utilizing a MATLAB function by professor Peter Svensson at Norwegian University of Science and Technology (NTNU). This function is based on the theory presented in section 3.2 and can be seen in appendix A.

Noise, $n_{recorded}[n]$, was then added to $s[n]$. $n_{recorded}[n]$ was a recorded background noise. Two unique background noise recordings were utilized as noise in the simulator. The background noises were recorded in:

- A quiet library where people read but no speaking occurred during recording
- An empty conference room with ventilator noise

The background noises were recorded in mono with a *Zoom H4N pro digital recorder*, at a 44.1 kHz sampling frequency. To define imperceivable measurement signal at a source-to-receiver distance of 1 m, a $h_{room}[n]$ with 1 m source-to-receiver was generated. A *.wav* file containing $s[n] + n_{recorded}$ was then created. This audio file was listened to, and $s[n]$ was scaled until it was imperceptible in casual listening. Scaling was done as previously described in section

[2] This filter was constructed by utilizing *butter*-function in MATLAB

[3] Note: the impz-function approximates an Infinite Impulse Response (IIR) to be a Finite Impulse Response (FIR) by utilizing the MATLAB filter-function `https://se.mathworks.com/help/signal/ref/impz.html`.

6.3, resulting in an imperceivable SNR. When imperceivable SNR was obtained, five scaling factors of the measurement signal $x[n]$ were found empirically. The five scaling factors were a range from a scaling factor which yielded an imperceivable $s[n]$, to 3 dB lower SNR than the imperceivable $s[n]$.

The sum $s[n] + n_{recorded}[n]$ was then convolved with filter $h_{post}[n]$ as presented in figure 7.3, where $h_{HP}[n]$ is the same high-pass Butterworth filter as previously, and $h'_{shaping}[n]$ is the inverse of the previously presented shaping filter. Also here convolution was done by utilizing the filter-function. The Z-transform of this filter had the following form:

$$H'_{shaping}(z) = 1 + a_{1,shaping} \cdot z^{-1} + a_{2,shaping} \cdot z^{-2} \tag{7.4}$$

where the coefficients are the same as previously presented in equation 7.1.

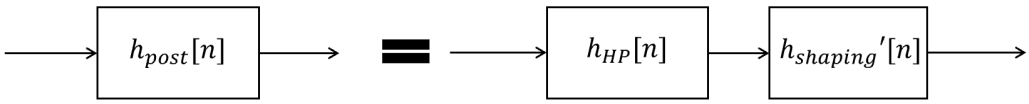


Figure 7.3.: Block diagram showing the construction of filter $h_{post}[n]$

The Z-transform of filter $h_{post}[n]$ was obtained by solving:

$$H_{post}(z) = H'_{shaping}(z) \cdot H_{HP}(z) \tag{7.5}$$

The signal on the output of $h_{post}[n]$ was then convolved with a matched filter, $h_{matched}[n]$, by utilizing the filter-function. This filter was obtained by taking the time inverse of one cycle of the MLS, $x[n]$, as previously described in section 6.3. As a result, the output of the matched-filter was N number of RIR estimates $\hat{h}_{room}[n]$ (i.e., the same number of estimates as the number of MLS cycles). Each estimate had the same number of samples as the MLS length L.

The output of the matched-filter was finally convolved with a moving average filter, $h_{averaging}[n]$, by utilizing the *conv*-function. The filter was constructed as presented in equation 7.6, where M is the number of averages, and L corresponds to the period of the known signal (i.e., the same as the length of one cycle of $x[n]$).

$$h_{averaging}[n] = \frac{1}{M} \sum_{m=0}^{M-1} \delta[n - mL] = \frac{1}{M} \cdot (\delta[n] + \delta[n - L] + \cdots + \delta[n - (M-1) \cdot L]) \tag{7.6}$$

This averaging filter was expressed in the Z-domain by doing a Z-tranform of equation 7.6. This is presented in equation 7.7, where $H_{averaging}(z)$ is the filter in Z-domain.

$$H_{averaging}(z) = \frac{1}{M} \cdot \sum_{m=0}^{M-1} z^{-Lm} \tag{7.7}$$

This equation was solved by utilizing the sum of geometric series:

$$H_{averaging}(z) = \frac{1}{M} \cdot \sum_{m=0}^{M-1} z^{-Lm} = \frac{1}{M} \cdot \sum_{m=0}^{M-1} \left(\frac{1}{z^{-L}} \right)^m = \frac{1}{M} \cdot \frac{1 - z^{-LM}}{1 - z^{-L}} \tag{7.8}$$

The filter $h_{average}[n]$ was then obtained by utilizing the impz-function in MATLAB. The output of the filter was $y[n]$.

7.1.1. Monte Carlo simulations

Monte Carlo simulations were conducted to verify the results. The purpose of the Monte Carlo simulations was first to study the number of signal averages (number of periods of the received signal that are averaged by the moving average filter) that were required to detect correct ToA for several source-to-receiver distances and reverberation times and secondly study a threshold of detection for the same parameters.

The simulations were conducted by utilizing 500 different parts of the recorded background noises, and utilizing the simulator previously presented in figure 7.1. The part of the background noise recordings used as $n_{recorded}[n]$ was changed for each of the 500 iterations of the Monte Carlo simulation, resulting in 500 different measurement situations. The lowest number of signal averages in each iteration that yielded the correct ToA was saved. A correct ToA was the exact sample at which the direct sound occurred in $h_{room}[n]$. The ToA was estimated as presented in equation 7.9, where $\hat{n}_{ToA}$ is the ToA estimate expressed in samples, and $y[n]$ is the output of the simulator presented in figure 7.1. This estimate was based on the theory presented in section 5.1.

$$\hat{n}_{TOF} = \arg\max_n |y[n]| \tag{7.9}$$

The code for the Monte Carlo simulation used to find number of averages can be seen in Appendix B.

Monte Carlo simulations with 500 iterations were conducted to determine a threshold of detection. For each Monte Carlo simulation, an attempt was made to estimate the ToA. For an estimate to be calculated, an impulse response peak needed to exceed a threshold, so the ToA was:

$$\hat{n}_{TOF} = \begin{cases} \arg\max_n |y[n]| & \text{if } \frac{\max|y[n]|}{y^*_{\text{RMS}}} >= Threshold \\ \text{not calculated} & \text{if } \frac{\max|y[n]|}{y^*_{\text{RMS}}} < Threshold \end{cases} \tag{7.10}$$

where y^*_{RMS} is the RMS-value calculated *without the 4 samples around the peak*. For each iteration, there were three possible outcomes:

1. n_{TOF} was estimated, and correct (identical to direct sound of $h_{room}[n]$)

2. n_{TOF} was estimated, but incorrect.

3. n_{TOF} could not be estimated.

The counts of these three types of results lead to (1) the probability of correct detection, p_{CD}, (2) the probability of incorrect detection, or false alarm, p_{FA}, and (3) the probability of a missed detection, p_{MD}. These probabilities were computed for a number of threshold values. The code for this simulation can be seen in Appendix C.

7.2. Detecting the direct sound arrival time - measurements

A practical test was conducted in an empty conference room with noise from a ventilator to verify the results obtained from simulations. The room was located at the acoustics laboratory at NTNU. A block diagram showing the principle of the measurements is presented in figure 7.4. The two filters $h_{pre}[n]$, $h_{post}[n]$, and $h_{average}[n]$ are the same as previously defined.

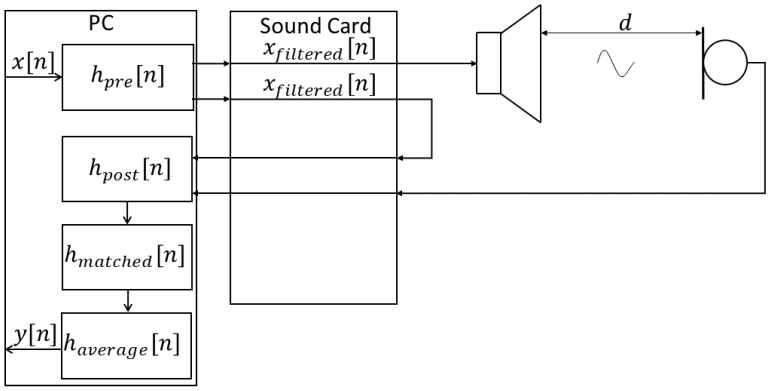


Figure 7.4.: Block diagram showing the the measurement chain used for the field measurements, where d is the distance between the source/loudspeaker and receiver/microphone.

As can be seen from figure 7.4, one microphone and one loudspeaker placed at a distance d from each other were utilized to conduct the measurements. The loudspeaker and microphone were connected to a sound-card and a trigger signal, which was connected directly from the output of the sound-card to an input. The sound-card was connected to a computer running Audacity[4]. Audacity did the recording and playback of the measurement signal.

The measurements were split into three stages:

- Stage 1: Pre-processing

[4]A free open source audio software URL: `https://www.audacityteam.org/`

- Stage 2: Conducting the measurements
- Stage 3: Post-processing

These three stages will be explained in the following sections, and the equipment utilized to conduct the measurements is listed in table 7.1.

Table 7.1.: Equipment used to conduct the measurements

Equipment	**Manufacturer**	**Type**
Microphones	BSWA	216
Microphone amplifier	BSWA	4000
Sound Card	Roland	Studio-capture UA-1610
Loudspeaker	Adam Audio	A5X
Software	Audacity	Version 2.3.3
Software	Mathworks	MATLAB Version R2018b
Sound calibrator	Norsonic	Nor1256

7.2.1. Stage 1: Pre-processing

In the first phase, background noise in the room was recorded using the sound card and the microphone. When a recording was obtained for the room, the filtering of the measurements signal was done. The measurement signal contained multiple cycles of an MLS.

The measurement signals were filtered by $h_{pre}[n]$ in the same manner as for the simulator. When filtering was completed, an audio-file of $x_{filtered}[n]$ was generated by utilizing the MATLAB's audiowrite function. The audio-file was a 16-bit .wav format file.

The code for pre-processing can be seen in Appendix D.

7.2.2. Stage 2: Conducting the measurements

Figure 7.5.: Pictures of the measurement setup in the conference room showing microphone and loudspeaker placement.

Measurements were conducted for 5 m source-to-receiver distance. A laser distance meter was used to measure the distances. The source-to-receiver distance was measured from the microphone directly between the mid-range woofer and the tweeter of the loudspeaker. The

microphone was placed on a stand directly in front of the loudspeaker in line of sight. The loudspeaker was placed on a stand, see figure 7.5.

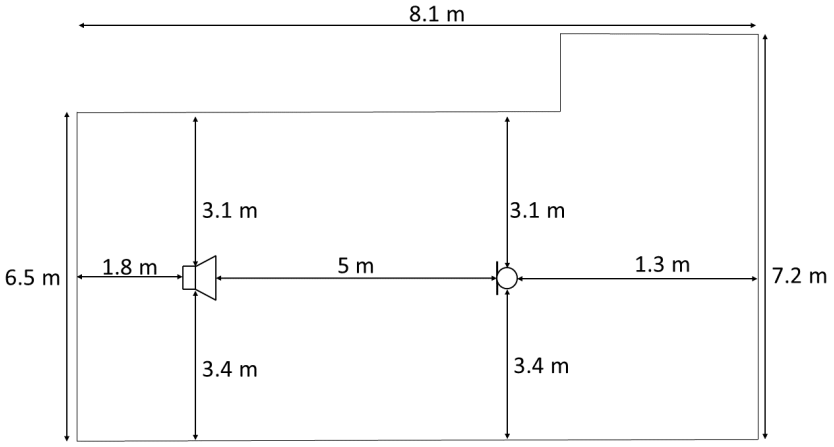


Figure 7.6.: Figure showing the three measurement positions. Note figure is not to scale.

The distance from the floor to between the mid-range woofer and the tweeter was 1.2 meters. The microphone was extended to match the height of the loudspeaker. The microphone and loudspeaker placements during the measurement can be seen in figure 7.6.

At first, the generated audio-file of $x_{filtered}[n]$ was transmitted with a loud volume to the loudspeaker, and a recording of the microphone signal was done simultaneously. This recording of the loud measurement signal was done to determine the SPL at the receiver for a given amplification of the builtin sound-card amplifier. Recordings of the calibrator were also conducted to determine the digital value corresponding to 94 dB SPL. When the loud recording was obtained, a new recording/measurement was conducted. This time, the sound card's amplification was reduced until the measurement signal was not perceivable by the author of this report at 1 m from the source, i.e., the measurement signal was masked by the background noise in the room.

The measurements were conducted for a range of SNRs.

7.2.3. Stage 3: Post-processing

When all the measurements were completed, the audio-files obtained were imported to MATLAB. The audio files were then filtered by $h_{post}[n]$, $h_{matched}[n]$, and $h_{average}[n]$. The matched filter coefficients were the time inverse of the MLS used as $x[n]$. The trigger signal $x_{trigger}[n]$ was utilized to separate $y[n]$ into single measurement situations where each peak of the trigger denoted $t_{ToA} = 0$.

The ToA estimate expressed in samples was obtained by up-sampling $y[n]$, by a factor of 16, by utilizing the *interp*-function in MATLAB. This resulted in a new variable $y_{upSampled}[n]$. The ToA estimate was then calculated as presented in equation 7.11 where $\hat{t}_{ToA}$ is the ToA estimate, F_S is the sampling frequency, and $upFactor$ is the up-sampling factor.

$$\hat{t}_{ToA} = \frac{1}{F_s \cdot upFactor} \arg\max_n |y_{upSampled}[n]| \tag{7.11}$$

The distance was then calculated as follows:

$$\hat{d} = \hat{t}_{ToA} \cdot c \tag{7.12}$$

where c is the sound speed in air (343 m/s).

Also in these measurements a range of threshold was studied as previously presented in equation 7.10, however, here utilizing equation 7.11 instead of the sampled ToA estimate $\hat{n}_{ToA}[n]$.

The code used for post-processing can be seen in appendix E.

7.3. Estimating the direction of arrival using detailed room simulator

A 3D localization system was simulated in MATLAB. Figure 7.7 presents the block diagram of the simulator.

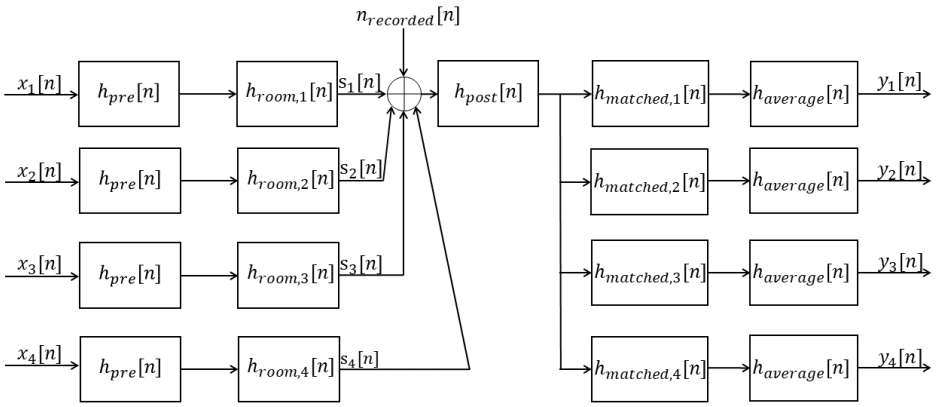


Figure 7.7.: Block diagram of the 3D simulator.

The measurement signals in this simulator were either pure PN-sequences (Gold or Kasami) or PN-sweeps. Gold sequences were utilized for odd orders sequence lengths, and Kasami sequences were utilized for even orders sequence lengths. The sweep signals were generated according to the theory presented in 4.5.1. Each of the 4 PN-sweeps was coded with a unique Kasami or Gold sequence. This resulted in four measurement signals denoted $x_1[n]$ to $x_4[n]$ in the block-diagram. Each measurement signal contained multiple cycles of the respective sequence/sweep.

The PN-sweep signals were time-delayed with respect to each other. This was done by appending zeros in-front of signals $x_2[n]$ to $x_4[n]$. The number of zeros appended corresponded to $T/4$, $T/2$,

and $3T/4$ for $x_2[n]$, $x_3[n]$, and $x_4[n]$ respectively, where T denotes the period of the PN-sweep[5]. This was done to make the sound generated by the four signals played simultaneously sound more like "noise". The sweeps used in the PN-sweeps were generated by utilizing equation 4.21 in section 4.5.1, where the values of f_1 were either 2 kHz or 5 kHz, and f_2 was 20 kHz.

Gold and Kasami sequences were obtained by utilizing the MATLAB functions in Appendix F. These functions were developed based on the theory presented in chapter 4.

Each measurement signal was then individually filtered by $h_{pre}[n]$, which was the same as presented previously, except two cut-off frequencies of the high-pass filter were studied here, namely; 2 kHz and 5 kHz. This was done to make the simulations more realistic, and match the frequency response of a real loudspeaker array where small loudspeaker elements are utilized.

In this thesis, a simulated loudspeaker array was utilized. The array consisted of four-point sources in free-field (no loudspeaker box) in a tetrahedron configuration, as previously presented in figure 5.2. The elements were spaced 200 mm apart. Each loudspeaker element to receiver distance resulted in a unique RIR, which are denoted $h_{room,1}[n]$ to $h_{room,4}[n]$ in the block-diagram. Thus for each receiver position, four unique RIR's were utilized. The receivers were ideal point sources.

The RIRs were simulated in the room acoustics simulation software *CATT-acoustics*[6]. The CATT-acoustics settings are attached in appendix G. The absorption of the walls, ceiling, and the floor was set arbitrarily until a T_{60} of about 0.6 seconds was obtained. Four different sources were utilized at different positions in the simulated room to test different arrival angles. One of the sources was at 1 m source-to-receiver distance and was used to determine imperceivable measurements signal in the same manner as for the simplified RIR.

All signals on the outputs of the RIR's, named $s_1[n]$ to $s_4[n]$ where then added together. The recorded background noise $n_{recorded}[n]$ was also added to the signals. The measurement signals $x[n]$ were scaled until $s[n]$ was deemed imperceivable by the author. When an imperceivable measurement signal was obtained the power of $x[n]$ was further scaled down by 3 dB.

All the added signals and noise were then filtered by $h_{post}[n]$, which was the same as previously defined, but also here the cut-off frequency of the high-pass filter was adjusted. Finally, four signals, $y_1[n]$ to $y_4[n]$, containing 4 RIR estimates were obtained by filtering the whole signal with four individual matched filters (one for each measurement signal), and averaging the outputs of the filters with a moving average filter as previously defined.

Finally signals $y_1[n]$ to $y_4[n]$ were utilized to estimate the receiver positions in respect to the loudspeaker array center. The ToA of each loudspeaker element to microphone position were estimated by up-sampling each $y[n]$ and utilizing equation 7.11[7]. The ToA estimates were then utilized in equations 5.14 and 5.19 to obtain an estimated microphone position. the MATLAB function pinv was used to solve equation 5.19. The angle was finally calculated by utilizing equation 5.15.

[5] This was done only to the first cycle of each signal, resulting in all cycles having the same time-delay in respect to each other for the entire measurement signal.

[6] URL to CATT-acoustics website: `https://www.catt.se/`

[7] $y[n]$ consisted of multiple cycles and was therefore split into many single estimates by utilizing a trigger to determine time = 0 seconds of each estimate.

Monte Carlo simulations with 500 iterations were conducted. In these simulations, a more straightforward threshold of detection was used. If the source-to-receiver distance in any coordinate (x, y, or z) was longer than 15 m, the estimate was discarded, and a missed detection was saved.

7.4. Detecting the direct sound arrival time for a moving receiver - measurements

A study of moving microphones was conducted to determine which measurement signal would be best suited in a final system. A range of different lengths of PN-sequences, sine-sweeps, and PN-sweeps was studied by utilizing the system presented in figure 7.8.

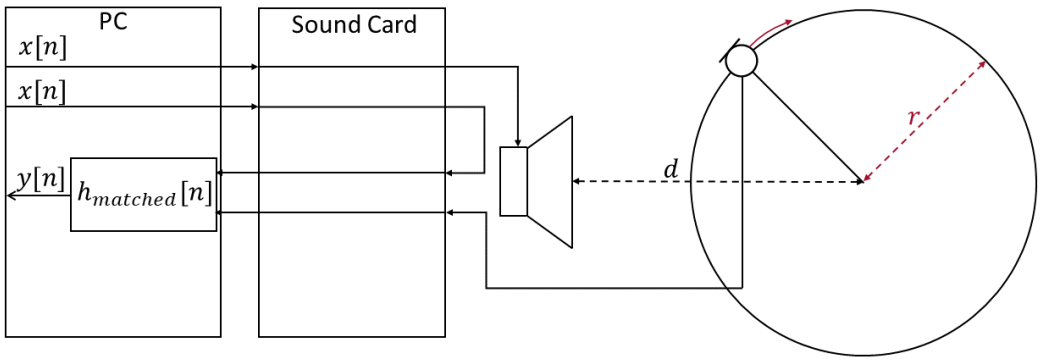


Figure 7.8.: Figure is a schematic representation of the measurement setup utilized to study moving microphones.

A turntable was utilized to rotate a microphone placed on a stand and a loudspeaker placed at distance d from the table center. The microphone was extended at different lengths from the table center, and the rotation speed of the microphone was calculated as presented in equation 7.13, where $v_{microphone}$ is the rotation speed of the microphone, $t_{rotation}$ is the time it takes for the table to rotate 360°, and r is the distance at which the microphone was extended from the table center.

$$v_{microphone} = \frac{2\pi \cdot r}{t_{rotation}} \tag{7.13}$$

The measurements were conducted by rotating the microphone at a constant rotation speed at

different extensions and playing a continuous stream of measurement signals of different types and lengths through the loudspeaker. The measurement signals were constructed in MATLAB by utilizing the audiowrite-function, and the signals were constructed in the same manner as previously, however, no pre-filtering was done. The measured signal from the microphone and a measured trigger signal, which was obtained by simply connecting one output of the sound card directly to an input, were passed through a matched filter, which was constructed of the time inverse of one cycle of $x[n]$. The distance was then estimated as previously presented in equation 7.12.

The measurements were conducted in an empty conference room with noise from a ventilator. The room was located at the acoustics laboratory at NTNU and material used for these measurements are presented in table 7.2

Table 7.2.: Equipment used to conduct the measurements

Equipment	**Manufacturer**	**Type**
Microphones	BSWA	216
Microphone amplifier	BSWA	4000
Sound Card	Roland	Studio-capture UA-1610
Loudspeaker	Adam Audio	A5X
Software	Audacity	Version 2.3.3
Software	Mathworks	MATLAB Version R2018b
Turntable	Norsonic	NOR265

8. Results

In this chapter, the results obtained from the experiments will be presented. Firstly the results of the signal shaping filter will be presented, followed by results obtained from the simple room simulator, and the field measurement conducted to estimate the arrival time. Then the results obtained from the detailed room simulator utilized to estimate the direction of arrival and receiver coordinates, and finally, the results obtained from measurements of the moving receiver.

8.1. Measurement signal shaping

As described in the experiments chapter, the measurement signal was shaped to mimic the frequency spectrum of the background noise in the given rooms.

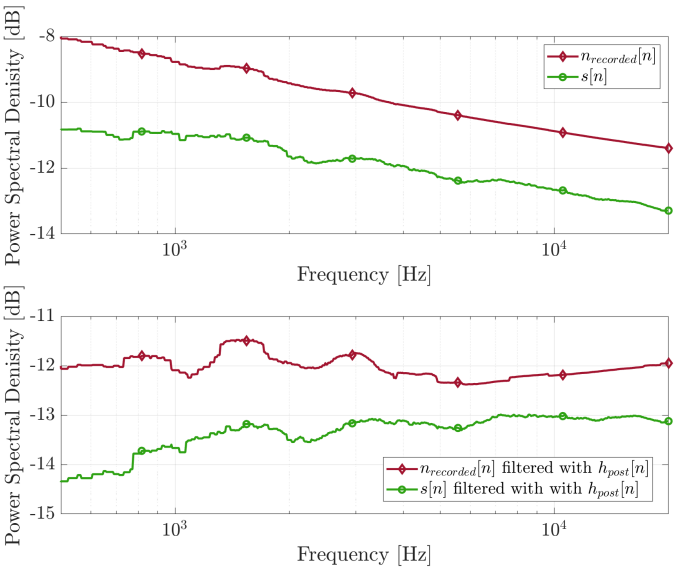
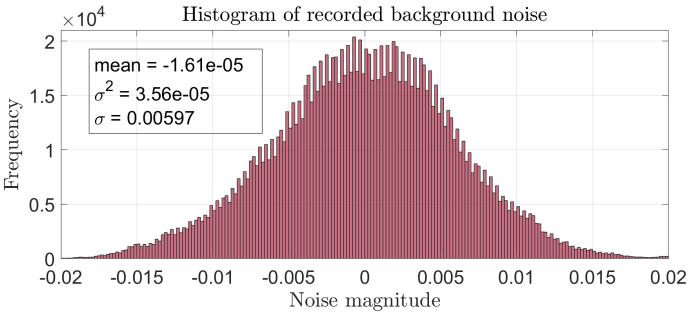
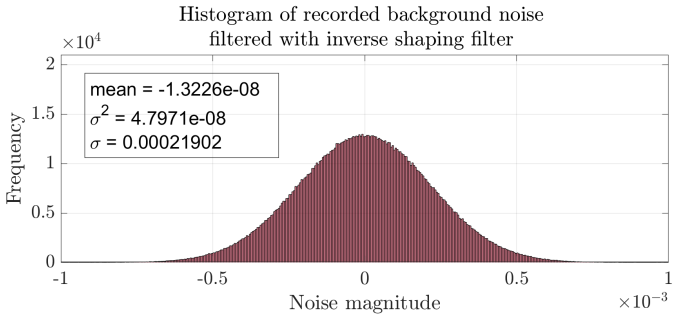


Figure 8.1.: Figure presents results of the shaping filter $h_{shaping}[n]$.

The effect of the shaping filter is presented in figure 8.1. The top graph presents the shaped signal $s[n]$ (the same as previously presented in figure 7.1) and the measured background noise $n_{recorded}[n]$, and the bottom graph presents signal $s[n]$ filtered by $h_{post}[n]$ and background noise filtered with $h_{post}[n]$. As can be seen from the top graph, the shaped MLS follows the background noise's spectral shape with little deviation. It can be seen that the magnitude of the recorded



(a)



(b)

Figure 8.2.: Figure showing two histograms where histogram (a) shows the distribution of recorded background noise, and histogram (b) shows the distribution of recorded background noise filtered with the inverse shaping filter $h'_{shaping}[n]$. Note that the scaling of x-axis differ between the two figures

background noise is reduced with ~ 3 dB per frequency decade. This reduction corresponds to "pink"-noise characteristics, which is often used to generalize acoustical background noise. From the bottom graph, it can be seen that background noise filtered with the inverse shaping filter results in a "whitened" noise signal, which oscillates around -12 dB. The histograms of the background noise and filtered background noise are presented in figure 8.2 (a) and (b), respectively. By studying the histograms, it can be seen that by filtering the background noise with the inverse shaping filter, the noise becomes white and Gaussian with ~ 0 mean.

8.2. Direct sound arrival time - simple room simulator

The mean number of signal averages required to detect correct ToA (the correct ToA was the correct direct sound sample) obtained from Monte Carlo simulations are presented in figure 8.3, where results for two different background noise recording and two different signal lengths are presented. As can be seen from the figure, when the source-to-receiver distance increases, so does the number of averages, $n_{avg}[-]$, required to detect correct ToA. It can also be observed that 4095 samples long MLS requires roughly half of the necessary averages compared to 2047 samples long MLS. The results are for an A-weighted SNR of -12 dB at 1 m source-to-receiver distance. The signal was imperceivable for the author at -9 dB A-weighted SNR.

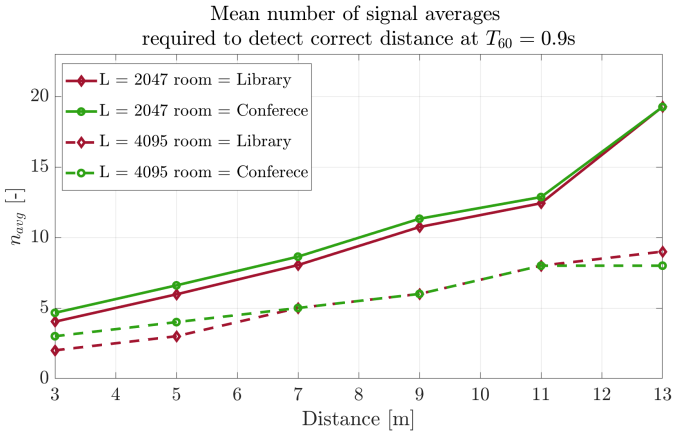


Figure 8.3.: Figure presents the mean number of averages necessary to detect correct source-to-receiver distance for two different $n_{recorded}[n]$ and L.

The results presented in figure 8.3 only show the mean number of signal averages required to detect correct source-to-receiver distance without any form of threshold of detection (all estimates are over threshold). It was therefore decided to study a specific case of the results presented in figure 8.3 to analyze how a threshold of detection would influence the required number of averages to detect the direct sound correctly. The threshold of detection for 5 m source-to-receiver with 0.9 s reverberation time, with the background noise recorded in the conference room as the noise signal is presented in figure 8.4 and 8.5.

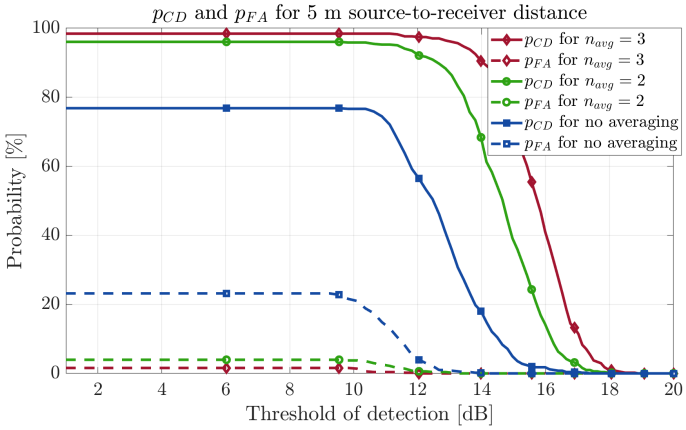


Figure 8.4.: Figure showing the probability of correct detection p_{CD} and probability of false alarm p_{FA} for a range of thresholds of detection for a 2047 samples long measurement signal and a 5 m source-to-receiver distance with conference room background noise.

The probability of correct detection p_{CD} and the probability of false alarm p_{FA} are presented in figure 8.4, and as can be seen from the figure the probability of correct detection increases drastically, and the probability of false alarm decreases when the number of signal averages

increases. Figure 8.5 presents the probability of missed detection p_{MD} for the same situation as in figure 8.4. As observed in figure 8.5, p_{MD} increases at a lower threshold for a lower number of signal averages. By studying the two figures, it can be seen that utilization of a moving average filter with three averages and a threshold of detection at 12 dB, p_{FA} approaches 0% and p_{CD} is around 97%. It is important to note that without averaging and a threshold of detection at 13 dB, p_{CD} is around 40% and p_{FA} is around 0%, which yields that in 1 second of measurement around 8 correct distance estimates are obtained.

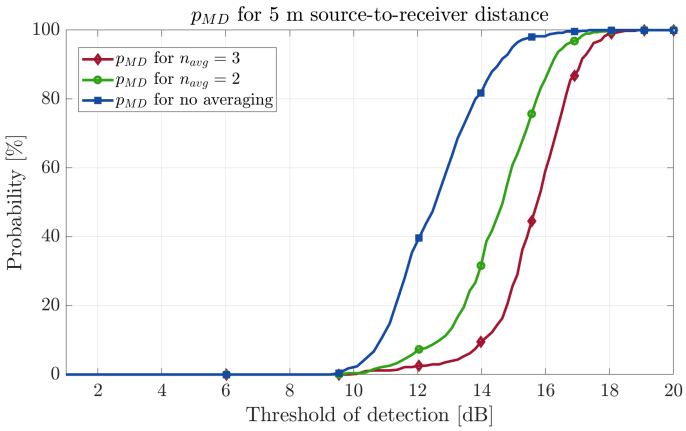


Figure 8.5.: Figure showing the probability of missed detection p_MD for a range of thresholds of detection for a 2047 samples long measurement signal and a 5 m source-to-receiver distance with conference room background noise.

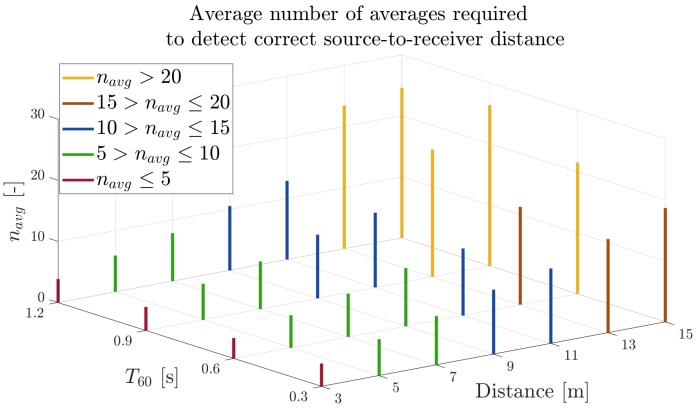


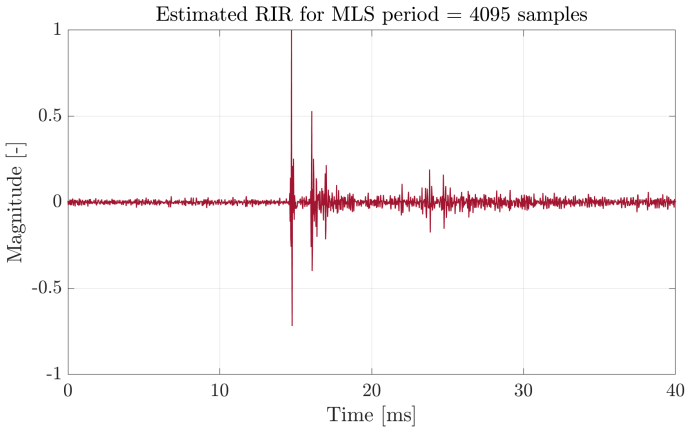
Figure 8.6.: Figure showing mean number of averages, $n_{avg}[-]$ that is required to detect correct direct sound pulse.

It is also important to note that the number of averages required to detect correct source-to-receiver distance will depend on the reverberation of the room in which the measurements are conducted. The effect of reverberation is presented in figure 8.6, where the conference room background noise was utilized as a noise signal, and a 2047 samples long MLS was used as

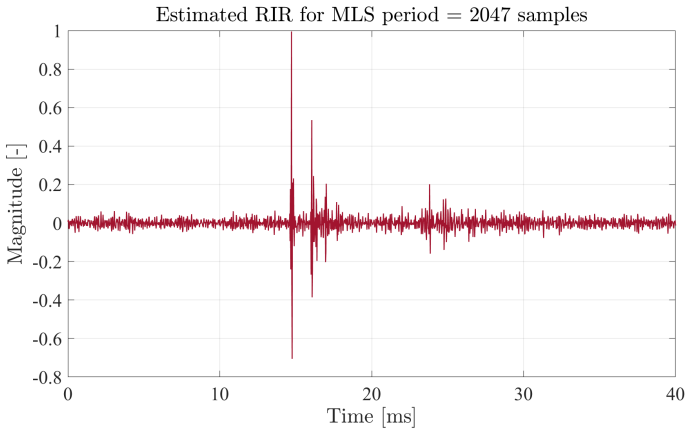
the measurement signal. As can be observed from the figure, the number of averages required increases both as a product of source-to-receiver distance and reverberation time.

8.3. Direct sound arrival time - measurements

The estimated RIR of the room where the measurements were conducted is presented in 8.7. These estimates were obtained using multiple MLS cycles and transmitting the signal with 17 dB A-weighted SNR. Figure (a) and (b) were obtained by utilizing 4095 samples and 2047 samples long MLS, respectively. The wrap-around effect causes the noise-like signal before the direct sound pulse in both (a) and (b). It can be observed that the noise-like signal has a smaller amplitude for the 4095 samples long estimate as compared to 2047 samples, which is logical as the wrap-around effects will be smaller for the longer sequence. The estimate where 2047 samples were utilized will be used as a reference in the results presented in this section, and the estimated source-to-receiver distance of the reference RIR is ~ 5.05 m (ToA ~ 14.7166 ms).



(a)



(b)

Figure 8.7.: Estimated RIR with 4095 samples long MLS (a), and 2047 samples long MLS (b). Figure (b) is used as reference in the following figures.

The mean ToA obtained from measurements for a range of A-weighted Sound Pressure Levels (SPL_A) is presented in figure 8.8. For each SPL_A, the presented results are a mean of ~ 4000 ToA estimates. As can be observed from the figure, the most significant deviation from reference occurs at 6 dB SPL_A, and for all levels under 8 dB, largest deviations are observed for the non-averaged signal. At 6 dB SPL_A, the non- averaged signal deviates with ~ 0.34 ms, which yields an error of ~ 0.12 m for 343.3 m/s sound speed. It can also be observed that the deviation decreases with the number of averaged cycles. The SPL_A was based on measurements of the measurement signal several dB above the noise floor and was calculated only ones during the measurements.

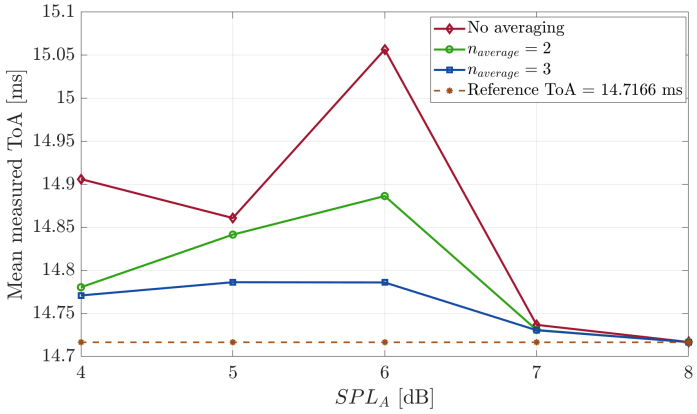


Figure 8.8.: Figure showing mean of 4000 ToA measurements for a range of SPLs.

The mean measured ToA for 5 dB SPL_A for a range of detection thresholds is presented in figure 8.9. As can be observed from the figure the deviation approaches ~ 0 at 14 dB, 15 dB, and 16 dB threshold of detection for 3, 2, and none cycles averaged respectively. Outliers in the measurement data-set cause the increase in deviation at ~ 13 dB threshold of detection for the non-averaged signal.

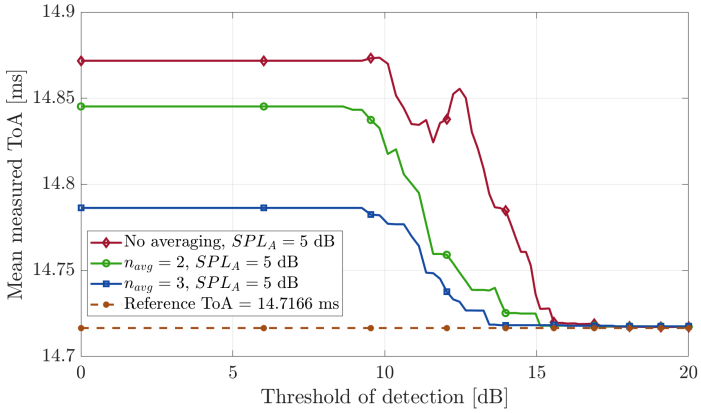


Figure 8.9.: Figure shows mean of 4000 ToA measurements for a range of thresholds of detection.

Because of factors like temperature fluctuations, inaccuracies in the distance measurements, and other uncertainties, a measure of how good the estimated ToA is, can be the standard deviation as presented in figure 8.10. As can be observed from the figure, the signal where 3 cycles are averaged has a maximum standard deviation of $\sim$ 1.1 ms, which corresponds to $\sim$ 0.378 m. Also here the curves approach ~ 0 for 14 dB, 15 dB and 16 dB thresholds of detection. When the threshold of detection increases, the standard deviation decreases, resulting in the mean estimated ToA approaching reference.

The probability of missed detection p_{MD} is presented in figure 8.11. By comparing figure 8.9, 8.10, and 8.11 it can be observed that by choosing a threshold of detection at 16 dB, the standard deviation, and deviation from reference approach zero for all three cases. For the non-averaged measurements 16 dB threshold of detection results in $p_{MD} =\sim 50\%$, which results in $\sim$ 10 estimated distances per second.

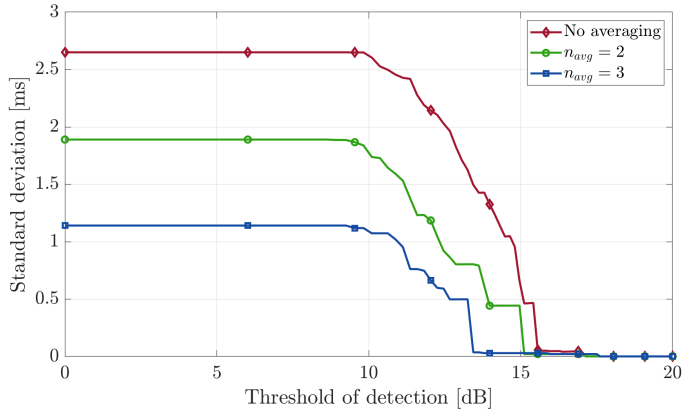


Figure 8.10.: Figure shows standard deviation of 4000 ToA measurements for a range of thresholds of detection.

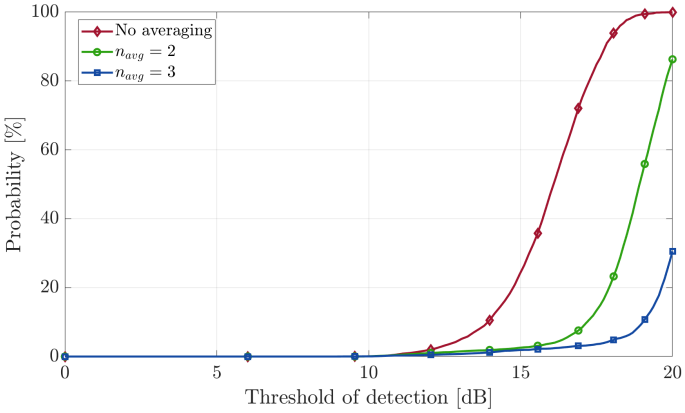


Figure 8.11.: Figure shows probability of missed detection of 4000 ToA measurements for a range of thresholds.

8.4. Direction of arrival and position estimates using detailed room simulator

The simulated room, microphone positions, and loudspeaker element positions from CATT-acoustics are presented in figure 8.12. The microphones numbered 1 to 4 are colored blue in the figure, and the loudspeaker elements are red. Microphones 01, 02, 03, and 04 will be referred to as Mic 1, Mic 2, Mic 3, and Mic 4.

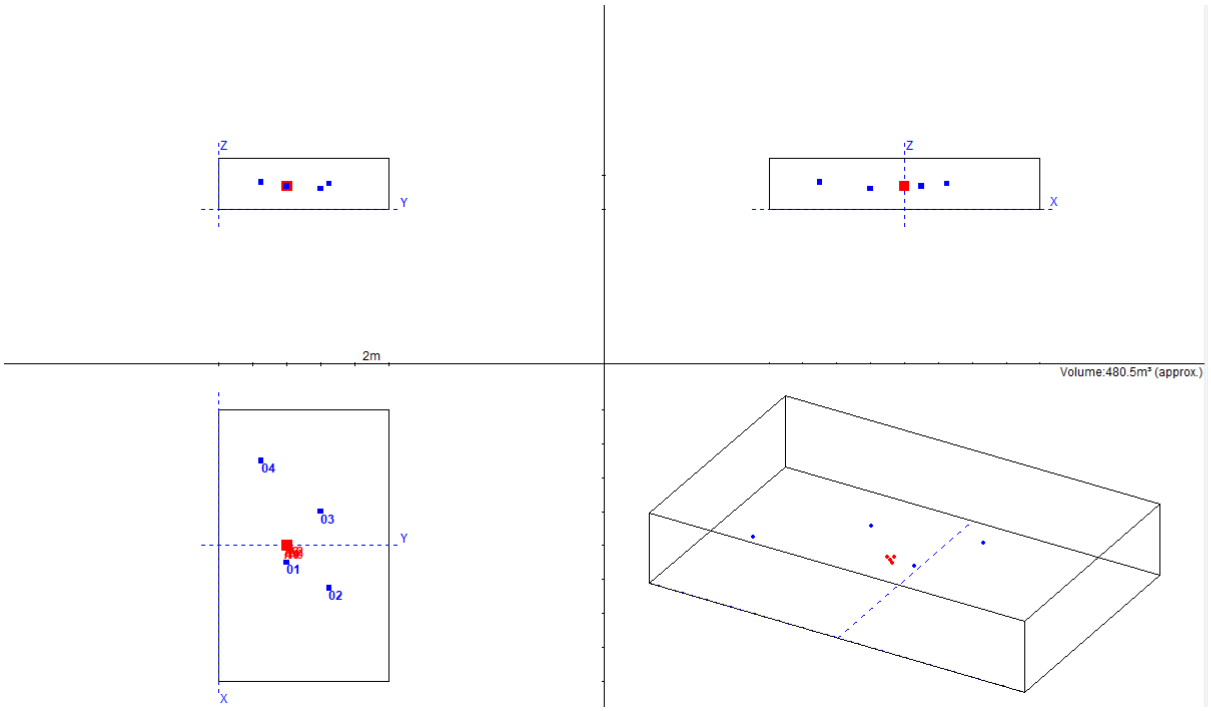
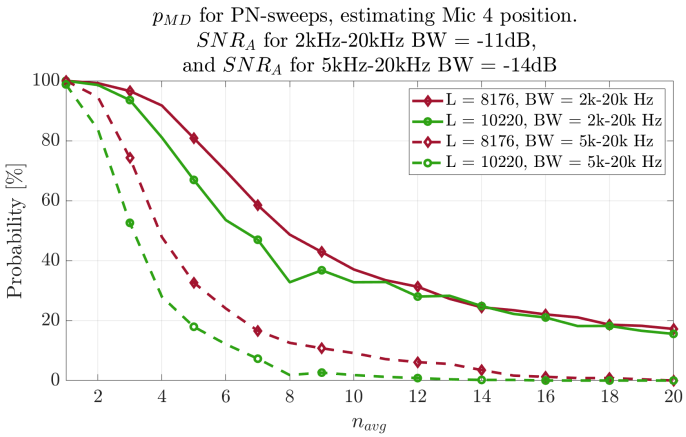
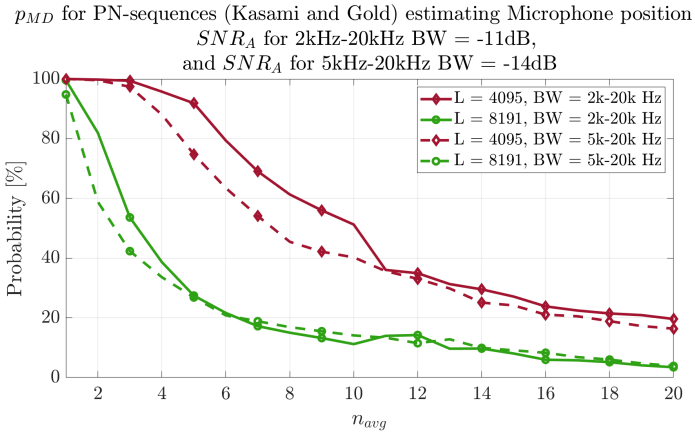


Figure 8.12.: Simulated room, loudspeaker positions and microphone positions. Microphones are blue, and loudspeaker are red. Figure obtained from CATT-acoustics

The result will be presented for two PN-sweep lengths; namely, 8176 samples and 10220 samples. Each of the PN-sweeps was coded with a unique 511 samples long Gold sequence. Two PN-sequence lengths will be presented, namely; 4095 samples long Kasami sequences, and 8191 samples long Gold sequences. Other sequence lengths were studied, but shorter sequences had a very high probability of missed detection and are therefore not presented here. The same two $n_{recording}$ noise signals as for the simple room simulator were studied, but also here the results obtained were similar between the "Conference" and "Library" background noise; thus only the conference background noise will be utilized in the results presented here. In this simulator, only one kind of threshold was used, which was a maximum source-to-receiver distance threshold. If a source-to-receiver distance was greater than 15 m, the estimate was rejected, and a missed detection was counted.



(a)



(b)

Figure 8.13.: Figure showing p_{MD} for two cases, (a) presets p_{MD} for PN-sweeps with sequence lengths L = 8176 sample, and L = 10220, and (b) presents PN-sequences (Kasami and Gold) with length, L = 4095, L = 8191. In both (a) and (b) two band-widths are presented, namely; 2kHz-20kHz and 5kHz-20kHz.

The probability of missed detection, p_{MD}, for PN-sweeps with periods L = 8176, and L = 10220 (for the sweeps L denotes the number of samples in a single sweep period), and for PN-sequences (Kasami and Gold) with sequence length, L = 4095, and L = 8191, are presented in figure 8.13 (a) and (b) respectively. In the two figures two different bandwidths (BW) can be observer namely; 2 kHz - 20 kHz, and 5 kHz - 20 kHz. For PN-sweeps, the BW was set as previously presented in the experiments chapter, and the BW of the PN-sequences was obtained by changing the cut-off frequency of the high-pass filter. The signals with BW = 2 kHz$-$20 kHz were deemed imperceivable by the author at $SNR_A = -8$ dB and for BW = 5 kHz$-$20 kHz the signal was imperceivable at $SNR_A = -11$ dB. It is important to note that the SNRs were calculated for the entire audible frequency range of the noise.

As can be observed from the figures by reducing the bandwidth of the PN-sweep, p_{MD} was reduced significantly, and for both 8176 samples long, and 10220 samples long sweeps the

probability of missed detection approached 0 at $n_{avg} = 15$. For the PN-sequences, the reduction of bandwidth resulted in little reduction of p_{MD}. The probability of missed detection for the 8191 samples long Gold sequence with bandwidth 2 kHz - 20 kHz has the lowest values of all the signals presented for this bandwidth.

The lowest p_{MD} was obtained for PN-sweeps with bandwidth $5k$ Hz- 20 kHz; it was decided to study the two PN-sweeps further. Figure 8.14 presents the mean estimated AoD. The mean is of all Monte Carlo iteration estimates. As can be observed from the figure for both signal lengths, the estimate deviate by a couple of degrees from the correct AoD when $n_{avg} = 2$, and the 8176 and 10220 samples long sequences cross the correct AoD at $n_{avg} = 4$ and $n_{avg} = 3$ respectively. As can be seen, the 10220 samples long PN-sweep deviates more than the 8176 samples long PN-sweep from the correct AoD for $n_{avg} \leq 4$, but it is also important to note that both signals deviate by a maximum of 1° from the correct AoD for $n_{avg} \geq 4$.

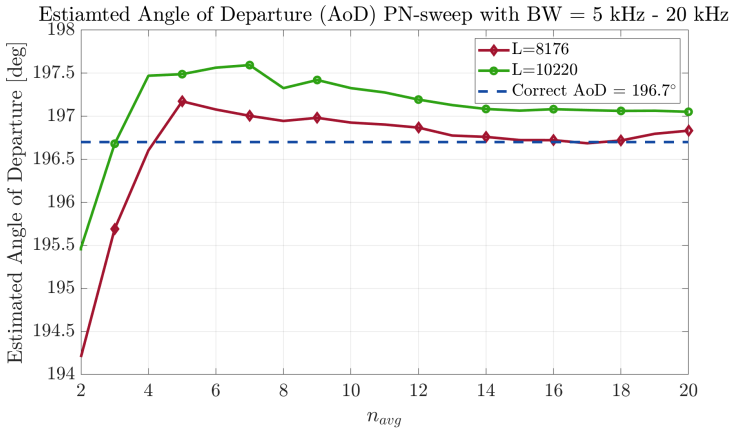


Figure 8.14.: Mean estimated Angle of Departure (AoD) for MIC 4, by utilizing 8176 and 10220 samples long PN-sweeps with a frequency bandwidth = 5 kHz - 20 kHz, all sweeps utilized were coded with an unique Gold sequence.

The standard deviation is presented as a measure to determine how well the system operates for a range of averages of the moving average filter. The standard deviation of the two signals for a range of averages can be seen in figure 8.15. As can be seen from the figure, the deviation has a large decrease for $n_{avg} > 2$. The lowest standard deviation for both signal lengths is at $n_{avg} = 20$ and the standard deviation is $\sim 1°$.

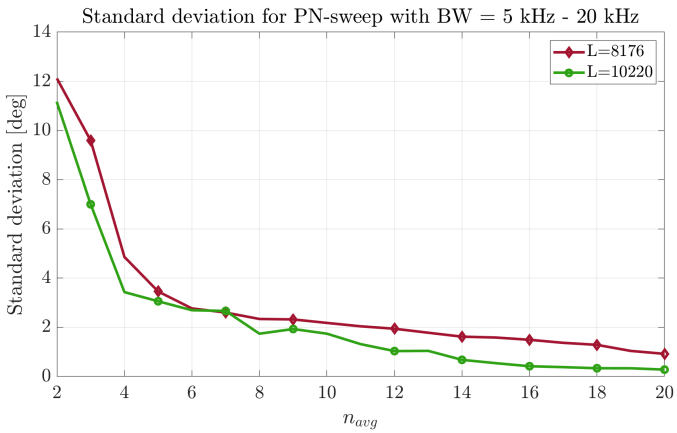


Figure 8.15.: Standard deviation for MIC 4 AoD estimate, for 8176 and 10220 samples long PN-sweeps with a frequency bandwidth = 5 kHz - 20 kHz, all sweeps utilized were coded with an unique Gold sequence.

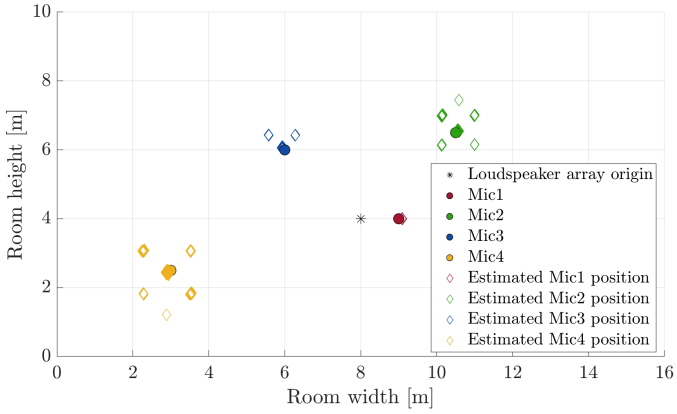


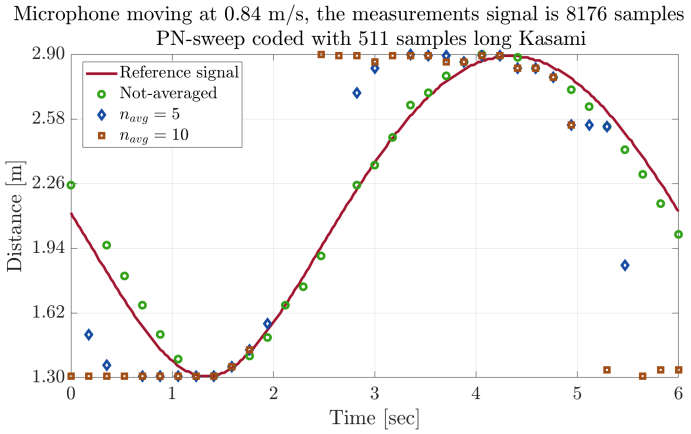
Figure 8.16.: Microphone position estimate by utilizing 500 Monte Carlo simulation iterations, and multiple cycles of an 8176 samples PN-sweep, and a moving average filter where five periods of the sweep are averaged.

The position estimate obtained from Monte Carlo simulations in 2D is presented in figure 8.16, where five signal averaging has been used for the PN-sweep with $L = 8176$. As can be seen from the figure, a little deviation from the correct position can be seen at Mic 1, and the deviations increase as a function of source-to-receiver distance; thus, the largest deviations can be observed for estimates of Mic 4. As can be seen from the figure, a deviation from the correct position of ~ 1.5 m is estimated for Mic 4 (yellow diamonds). However, it is important to note that most estimates are within a couple of cm deviation from the correct position.

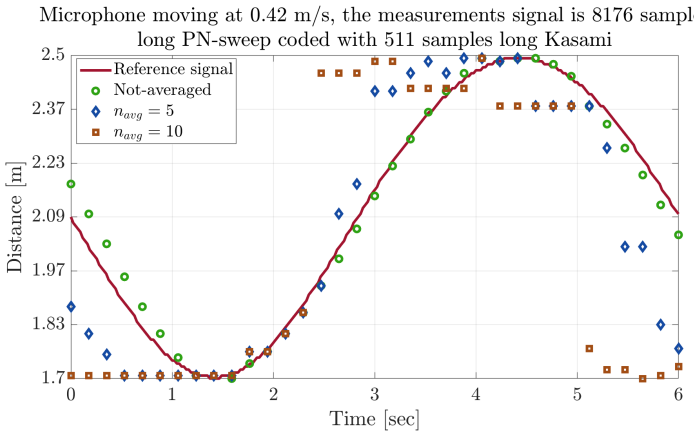
8.5. Estimates of direct sound arrival time with moving microphones

Measurements of moving microphones were conducted to study how time-variance influences the distance estimates. Many signal lengths and types were studied, but only the PN-sweep and LIN-sweep will be presented here.

The estimated source-to-receiver distance for a microphone moving at 0.84 m/s and 0.42 m/s are presented in figure 8.17 (a) and (b) respectively. The measurements were conducted by transmitting many cycles of the measurement signals to the loudspeaker, thus conducting continuous measurements. The figure presents results for 8176 samples long PN-sweep coded with 511 samples long MLS. The reference signal was 1000 samples long, not coded sweep. The received signal was averaged in post-processing by utilizing a moving average filter.



(a)



(b)

Figure 8.17.: Figure showing estimated source-to-receiver distance for moving microphone. The microphone moves with a speed of 0.84 m/s (a) and 0.42 m/s (b). The measurement signal was 8176 samples long PN-sweep coded with 511 samples long Gold code. The estimates were averaged in post-processing by a moving average filter.

As can be seen from the figure, the non-averaged PN-sweep gives a good estimate at both microphone rotation speeds with little deviation from reference. For 0.42 m/s, the measurement signal where five periods were averaged also follows the reference signal curve, however, with a lag. As can be observed from graphs (a) and (b), the number of estimates presented for $n_{avg} = 5$ and $n_{avg} = 10$ is lower than the number of estimates of the non-averaged case. The reason for this is that estimates outside the window of interest (estimates larger or smaller than the limits of the figures) were not included.

Figure 8.18 shows the influence of averaging. As can be seen, the unaveraged estimate looks like a sine wave for the entire 30 seconds measurement; the change at 15 seconds is caused by a change of direction of the turntable. The not-averaged case estimates are also very evenly spaced as can be observed the increase in the number of averages of the moving average filter results in a distorted estimate, and uneven spacing between estimates. It is also important to note that all three graphs show similar estimates when the change in source-to-receiver distance is small (peaks of the sine-wave).

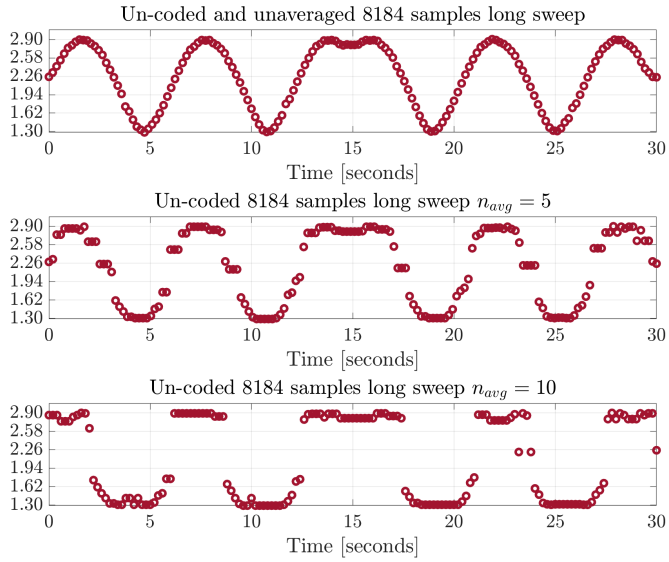


Figure 8.18.: Figure showing distance estimate of microphone moving at 0.84 m/s and the measurement signal was an un-coded 8176 samples long sweep. The graph from top to bottom represented not-averaged, $n_{avg} = 5$, and $n_{avg} = 10$ estimates.

PN-sweeps were generally more affected by time-variance as compared to un-coded sweeps. MLS with the same sequence length as the PN-sweeps were significantly affected by time-variance, especially when averaged. PN-sweeps coded with shorter PN-sequences resulted in better robustness to time variance than longer sweeps coded with longer sequences.

9. Discussion

In this chapter, a discussion of the results presented in the previous chapter will be presented. First, each of the experiments will be discussed separately, followed by a discussion of the entire concept, future work, and possible implementation of such a system.

9.1. Measurement signal shaping

As could be observed in figure 8.1, the recorded background noise had almost perfectly pink characteristics. The shaped signal $s[n]$, on the other hand, followed the 3 dB magnitude reduction per decade but oscillated a lot more than the recorded background noise. The reason for this might be that the simulated RIR with which the measurement signal (MLS) was convolved to obtain $s[n]$ might not have entirely flat magnitude since a finite normally distributed signal is utilized to generate the reverberation tail.

The imperfect white characteristics of the recorded noise filtered with the inverse shaping filter might be caused by the low order of the AR-model that was utilized to obtain the filter coefficients. It might be assumed that the filtered background noise would become "whiter" if a higher-order AR-model would be utilized for the inverse shaping filter.

In all measurements where the shaping filter was utilized, the filter coefficients were obtained only once during the simulations/measurements; thus, if changes in the background noise were present during the measurements/simulations, the signal would not be shaped to match these changes. However, on the other hand, if the coefficient were to be obtained to often, the signal shaping could result in a signal that is shaped by transients, or noise sources that are not always part of the background noise, which again could result in amplification at some frequencies which could potentially make the measurement signal more audible.

By utilizing a second-order AR-model to obtain the filter coefficients, the model would follow the shape of the frequency spectrum but would not "fit" the spectrum to match details like ventilator noise or other noise sources that have a specific frequency range. It can be discussed that a higher-order AR-model could be used to determine if any tonal noise sources are present, and then utilizing these sources to mask the measurement signal. However, this approach could result in an audible measurement signal if the noise sources used for masking would suddenly disappear, and is therefore not recommended.

Furthermore, it might be discussed if an AR-model is the right approach since the frequency spectrum of the noise has almost perfectly pink characteristics. For the swept signal, a logarithmic sweep could be used without any shaping filter at all.

9.2. Direct sound arrival time - simple room simulator

As could be previously observed in figure 8.3 for distances up to ~ 11 m, the 4095 MLS length needs roughly half the number of signal averages as compared to 2047 samples long MLS. This means that the number of samples necessary to detect correct source-to-receiver distance is roughly the same for 2047 samples long MLS and 4095 samples long MLS, because if, e.g., $n_{avg} = 4$ for 2047 and $n_{avg} = 2$ for 4095 then 8188 samples are necessary to detect correct distance for 2047 samples long MLS, and 8190 samples are necessary for 4095 samples long MLS.

When the source-to-receiver distance was greater than 11 m, the shorter MLS needed roughly three times the number of signal averages compared to the longer MLS. This might be caused by the wrap-around effects, as the shorter MLS will result in a more significant part of the reverberation tail being wrapped-around.

The maximum distance that is possible to measure will depend on the MLS length. For 2047 samples long MLS, the maximum source-to-receiver distance that could be measured would be ~ 15.9m. For distances longer than that, the direct sound would also wrap-around to the next estimate. In this thesis, 15 m was used as an absolute maximum distance of the system. In most cases, 15 m would be sufficient for an indoor system used for sound recording.

From simulations, it can be seen that for distances up to ~ 5 m, signals, where the moving average filter was not utilized, can still yield correct source-to-receiver distance estimates. If the right threshold is utilized, the probability of false alarm is also negligible. This means that in applications where a couple of estimates per second is sufficient, the moving average filter could be omitted.

In the results, one single case was studied in detail, namely; 5 m source-to-receiver distance with $T_{60} = 0.9$ seconds, these parameters were studied as 5 m was considered to be a representative distance for most use-cases, and the reverberation represents a "worst case" scenario.

It can be concluded that the system will be influenced by both the source-to-receiver distance as well as reverberation. In a product, a maximum distance for the system is more straightforward to define than a maximum reverberation, as most people have an intuitive understanding of distance rather than reverberation.

The simulations were conducted with a simple RIR simulation, which made it possible to define a correct and a wrong estimate. It is important to note that if wrong distances were estimated, the estimates were utterly random, e.g., if 5 m was the correct distance, a wrong distance estimate could be any distance between 0 m and 15 m.

9.3. Direct sound arrival time - measurements

As seen in section 8.3, the reference distance obtained from sound measurements was calculated to be 5.05 m, but the distance measured with a laser distance meter was 5.008 m. Multiple factors can cause this deviation of ~ 4 cm, but the two factors that are most likely to cause this error are; the measurement conducted with the laser meter was to the geometrical center of the

loudspeaker and not the acoustical center, which can yield an error, and the reference distance is calculated by assuming 20°C air temperature. The room was most likely colder than that, which results in a slower sound speed, which again can result in the distance error (estimated distance is longer than actual distance).

As could be observed in figure 8.8, the SPL that yielded the most significant deviation from reference was at 6 dB A-weighted SPL. This indicates that even though the SPL generated by the loudspeaker was constant, the SPL of background noise was varying. The A-weighted SPL of the background noise was measured to be 28 dB, but this measurement was conducted only ones during the tests. This fact can indicate that a constant measurement of the background noise level should have been conducted to continuously adapt the volume of measurement signal to obtain a constant SNR.

One possible error in figure 8.8 might be the fact that the SPL was calculated to be 8 dB A-weighted SPL, by studying the datasheet of the microphone utilized to conduct the measurements it can be seen that the self-noise of the microphone is at ~ 25 dB A-weighted SPL. This means that the measurement signal is well under the self-noise of the microphone. The SPL was calculated by transmitting a loud measurement signal through the loudspeaker and calculating the A-weighted SPL of this signal. The volume of the measurement signal was then adjusted in the sound card software until the measurement signal was not perceivable at 1 m distance. The reduction in dB was then used to determine the SPL of the imperceivable measurement signal. Here it was assumed that the reduction given by the sound card software was correct, which might not be the case. However, since the microphone's self-noise is only expressed as a single A-weighted value, it is hard to determine if the measurement signal was under the self-noise for all frequencies. This should be further studied.

A further interesting fact is that he SPL of the acoustic background noise presented in thi thesis might also be lower than presented in the results, since the self-noise of the microphone might be higher than the actual acoustic noise.

It can also be discussed that the mean of all ~ 4000 estimates might not be the most representative measure. By studying figure 8.9 it can be seen that a threshold of detection at 13 dB resulted in a more significant deviation from reference than at 12 dB. This was caused by the fact that some correct estimates were under the threshold, and a wrong distance estimate (~ 7 m) was over the threshold. This resulted in the wrong estimate affecting the average. It can be discussed that the median of the data would be a correct representation of the data because the distribution of the estimated distances was not Gaussian.

9.4. Direction of arrival and position estimates using detailed room simulator

The choice of the array size used in the simulations was made by testing many different sizes. When a smaller distance between the loudspeaker elements was utilized, the AoD estimates deviated by a couple of degrees from correct AoD. However, the reason for this deviation is unknown, but it is assumed that the resolution of CATT acoustics causes this deviation. When the array size was smaller, the ToA estimates from the four loudspeakers deviated only by

a couple of samples, while for larger array size, the ToA difference between the speakers was several samples in difference, which results in each sample being less important for the estimate.

The array was four omnidirectional loudspeaker elements in free-field, which is not a realistic assumption as diffraction caused by the loudspeaker enclosures and loudspeaker elements will be present in real measurements. Diffraction will cause different deviations in source-to-receiver distances for different frequencies. Also, the radiation pattern of a real loudspeaker array would not be omnidirectional, as most loudspeakers become more directional for higher frequencies.

The autocorrelation peak of the PN-sweep signals is dependant on the sweep length in samples; therefore, in theory, utilizing longer PN-sweeps would yield a better probability of detection. However, when the sweep signals were longer than $\sim$ 11000 samples, the measurement signal became very tonal, resulting in a more perceivable signal than the shorter sweeps. This is why longer PN-sweeps were omitted in this thesis. For PN-sequences (Gold, Kasami, and MLS), the audibility was similar for all samples length since these sequences have noise-like characteristics, but the shorter sequences were utilized in this thesis to compare the results with the two types of measurement signals, and since longer sequences were assumed to be more prone to time variance.

As seen in figure 8.14, the estimates where longer sweeps were utilized deviated more from the correct AoD than the estimates obtained with shorter sweeps. By studying both signals in estimations where no noise was present showed that both signal lengths performed in the same way, this indicated that the "better" estimate obtained by the shorter sweep is random. It is however important to note that AoD estimates would still be within $\sim 1°$.

The frequency range that was studied in the results was 5 kHz - 20 kHz. This frequency range resulted in a significantly better probability of detection for PN-sweeps than when 2 kHz - 20 kHz was used because this range was less audible for the author. However, as previously stated, the loudspeaker's directionality will be frequency-dependent and will be more directional for higher frequencies. Furthermore, shorter wavelengths will also be more prone to obstacles.

As can be observed from figure 8.16 all position estimates deviate by a factor from the correct position. This deviation is most likely caused by the Matrix-equation used to calculate the x, y, and z coordinates. By only utilizing the x, y, and z coordinates to estimate the AoD and utilize the mean ToA to estimate the source-to-receiver distance, a combination of this to estimates could be used to determine the position of the microphone, which could potentially be a better approach since the source of the error caused by the matrix equation will only be a part of the angle estimate and not the distance estimate.

For 8176 samples, long PN-sweep with moving average filter consisting of $n_{avg} < 4$ a large standard deviation and a deviation of$\sim 2.5°$ from correct AoD was observed. This can indicate that to estimate a correct AoD at 5 m source-to-receiver distance, a moving average filter consisting of $n_{avg} > 4$ should be utilized. Furthermore the number of missed detection was $< 50\%$ for $n_{avg} < 4$.

The estimator presented was dependent on four correct ToA estimates to estimate position. If one or more of the four estimates was wrong (a couple of samples), a long source-to-receiver distance was estimated; therefore, a threshold of the maximum distance of 15 m was implemented.

However, since this was the only threshold, the estimations' confidence is only based on the number of averages of the moving average filter. Nevertheless, by studying the 2D estimate presented in figure 8.16, it can be observed that the maximum deviation was just a couple of cm, this might be sufficient in some applications.

In this thesis, only the azimuth angle was studied, but in theory, the same results can be expected for the elevation angle, as for this estimation, the x- and z-coordinate estimates would be utilized the estimation of these coordinates showed good results.

9.5. Estimates of direct sound arrival time with moving microphone

Multiple measurement signals were tested with the turntable. The results showed that the PN-sweeps were far less prone to time variance than MLS, but in both cases, the implementation of moving average filter resulted in a high percentage of missed detection. Missed detection was when estimates that were not in the region of interest (1.3m - 2.9m for 0.84 m/s speed, and 1.7m - 2.5m for 0.42 m/s speed) were calculated. By reducing the microphone's rotation speed, the probability of missed detection for $n_{avg} = 5$ was also reduced. Therefore, it can be concluded that if the moving object's speed is not too high, the position can be estimated even with $n_{avg} = 5$, but with some error.

PN-sweep coded with 255 long MLS were also studied as a part of this thesis, only for the moving microphone measurements. The results for PN-sweeps coded with 255 samples long MLS for the moving microphone were very similar to the results obtained for uncoded sweeps; this means that they followed the movement similarly as presented in figure 8.18. The length of the code, which was utilized to code the sweeps, will be the decisive factor if it were to be used with not stationary microphones. Shorter codes result in better movement estimation, but shorter codes also result in higher crosscorrelation.

Distance estimation for moving microphones was only done with measurement signals with SPL well above the background noise level. It should be, therefore, further studied how movement combined with low SNR affects the estimations.

9.6. Audibility

The author determined the audibility in this thesis; however, in all estimations, except for the moving receivers, the SNR was 3 dB below the authors' perceivability. Therefore, it can be concluded that the measurement signal would be imperceivable for most people, as the author has normal hearing and is 25 years old. When listening to recordings and in field-measurements, the author knew how the measurement signal sounded and knew what to listen for, it might be assumed that for people that do not have previous knowledge of how the measurement signal sound would probably not notice it as easy as the author did, thus resulting in a higher SNR that could be utilized in a final system.

Audibility was in this thesis only expressed in terms of A-weighted SNR, which is a single number measure. The in-audible SNR should be further studied in a range of frequencies (octave bands

or 1/3 octave bands) and a broader range of background noises.

9.7. Comparison and real-world application

When studying the one source and one receiver case, it was presented that for distances up to 5 m, the use of a moving average filter was unnecessary as long as a threshold of detection was correctly implemented. In the positioning case, however, none such threshold of detection was implemented since the estimator dependent on 4 ToA estimates; thus if any of the four estimates were under threshold no estimation was obtained, and a maximum source-to-receiver distance threshold was therefore considered to be the best solution in this case. This means that the number of averages is the decisive factor in the AoD estimates when considering the estimate's confidence.

By considering the effects of moving average filters on moving microphones, it can be concluded that a final system should not implement such filters, or at least as few periods as possible should be averaged, especially when PN-sequences are used. A combined solution might be considered in a final product where four time-delayed PN-sweep signals are sent through one loudspeaker are used for source-to-receiver estimation and using a different technology like Bluetooth for AoD estimates. In such a solution, the signals would be less affected by movement as long as the sweeps are coded with short PN-sequences (≤ 255 samples), and a threshold of detection could be implemented. By combining different technologies, the solution would be more robust as different technologies have different noise signals which are not necessarily correlated with each other.

The reason for utilizing measurement signals in the audible frequency range was to make the system less dependent on the equipment used. However, when studying the results obtained from measurements, the A-weighted SPL of the imperceivable measurement signal was ~ 8 dB, and speech signals can be assumed to have an A-weighted SPL of up to ~ 96 dB at 0.5 m between the speaking person and microphone. This means that a microphone with a dynamic range of > 88 dB would have to be implemented, and if this system was used with an instrument, an even more extensive dynamic range would be necessary. Furthermore, the noise floor of a typical condenser microphone is in the range of $20 - 30$ dB A-weighted SPL. This means that in some rooms, the background noise in the room would be under the microphone's self-noise. However, it can be discussed if the microphone's self-noise could be used as a measure of impermeability. By using the noise floor of the microphone, the signal would be masked by the microphone's self-noise in the recordings, but would possibly be perceivable by the users of the system since the background noise might be under the self-noise. The annoyance in this approach is unknown and should be further studied. This approach would also assume that the self-noise of all microphones utilized would be the same, which again might not be the case. Therefore, it can be concluded that the users of a system like the one presented in this thesis could not simply choose any COTS microphone and assume that the system would work because of the deviations in noise floor and dynamic range. Therefore, it can be discussed if an approach utilizing ultrasound would be more comfortable and cheaper to implement; however, this is only speculations, and an assessment should be made.

In this thesis, moving microphones were only studied with constant speed and signals well above

the noise floor. Therefore, the result might not be representative for signals under the noise floor and should be further studied. In all cases presented here, the microphones were in line of sight, in reality; however, people wearing lavalier microphones may cover the microphones. The result of covering and "natural" movement should be further studied. Furthermore, in all measurements and simulations, the only source of the noise was background noise and microphone self-noise; in reality, noise generated by the system user would be present, i.e., noise from speech, instruments, etc.

All results presented in this thesis were post-processed, and the processing time was not taken into account. In a real-world application, this processing would have to be done in real-time and large parts of the code would have to be optimized to obtain real-time processing.

A higher level estimator like the Kalman-filter could also be utilized in a final system to make the system more robust against errors.

10. Conclusion

In estimations of single source-to-receiver distances in fixed positions, both measurements and simulations presented that distances up to 5 m could be correctly estimated with 2047 samples long MLS without the use of a moving average filter, as long as a correct threshold of detection was utilized. In results obtained from AoD estimations, it was presented that estimates where a moving average filter averaging five periods of the measured signal resulted in an AoD estimate that, on average, deviated with $< 1^\circ$ from the correct angle. However, when the moving average filter consisted of < 4 periods/cycles, deviations of $\sim 2.5^\circ$ from the correct angle were observed.

PN-sweeps were far less prone to time variance caused by movement than MLS, but for both signal types, the implementation of moving average filter resulted in many estimates outside the window of interest (minimum and maximum source-to-receiver length). Source-to-receiver distance estimates where uncoded sweep signals and moving average filter was used, yielded good results, and all estimates where within the window of interest. The same was true for PN-sweeps coded with short sequences (≤ 255). Therefore, it can be concluded that PN-sweeps coded with short sequences should be utilized if the microphones are non-stationary.

The primary motivation for utilizing measurement signals in the audible frequency range was to use the already existing infrastructure with COTS microphones. However, because of factors like the dynamic range and self-noise, it can be concluded that a system like the one presented in this thesis would not work with all COTS microphones.

Bibliography

[1] João Neves Moutinho, Rui Esteves Araújo, and Diamantino Freitas. *Indoor localization with audible sound—Towards practical implementation.* Pervasive and Mobile Computing 29, 2016.

[2] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. *Implementing a sentient computing system.* IEEE, 2001.

[3] Nissanka B Priyantha, Anit Chakraborty, and Hari Balakrishnan. *The cricket location-support system.* ACM, 2000.

[4] Yasuhiro Fukuju, Masateru Minami, Hiroyuki Morikawa, and Tomonori Aoyama. *DOLPHIN: An Autonomous Indoor Positioning System in Ubiquitous Computing Environment.* WSTFES, 2003.

[5] José Carlos Prieto, Antonio R Jiménez, Jorge I Guevara, Joao L Ealo, Fernando A Seco, Javier O Roa, and Francisco X Ramos. *Subcentimeter-accuracy localization through broadband acoustic transducers.* IEEE, 2007.

[6] Atri Mandal, Cristina V Lopes, Tony Givargis, Amir Haghighat, Raja Jurdak, and Pierre Baldi. *Beep: 3D indoor positioning using audible sound.* IEEE, 2005.

[7] Hugo Fastl and Eberhard Zwicker. *Psychoacoustics: Facts and models (book).* Springer, 2007.

[8] *ISO 226:2003. Acoustics — Normal equal-loudness-level contours.* International Organization for Standardization, 2003.

[9] *IEC 61672-1:2013. Electroacoustics - Sound level meters - Part 1: Specifications.* International Electrotechnical Commission, 2013.

[10] John G Proakis. *Digital signal processing (book).* Pearson Prentice Hall, 4th ed. 2007.

[11] Adil Alpkocak and Kemal Sis. *Computing Impulse Response of Room Acoustics Using the Ray-Tracing Method in Time Domain.* Archives of Acoustics, 2010.

[12] William G Gardner. *Efficient convolution without input/output delay.* Audio Engineering Society Convention 97, 1994.

[13] Steven M Schimmel, Martin F Muller, and Norbert Dillier. *A fast and accurate "shoebox" room acoustics simulator.* IEEE, 2009.

[14] Steven L Garrett. *Understanding Acoustics (book).* Springer International Publishing, 2017.

[15] Heinrich Kuttruff. *Room acoustics.* CRC Press, 4th ed. 2000.

[16] Manfred R Schroeder. *Modulation transfer functions: Definition and measurement.* S. Hirzel Verlag, Acta Acustica united with Acustica, 1981.

[17] Antoni Torras Rosell. *Methods of measuring impulse responses in architectural acoustics.* Master's thesis, Tech. Univ. of Denmark, Kongens Lyngby, Denmark, 2009.

[18] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. 1965.

[19] Julius S Bendat and Allan G Piersol. *Random data: analysis and measurement procedures (book).* John Wiley & Sons, 2011.

[20] Henrik Herlufsen. *Dual channel FFT analysis (Part II).* Brüel & Kjær Technical Review, 1984.

[21] Jessie F MacWilliams and Neil JA Sloane. *Pseudo-random sequences and arrays.* IEEE, 1976.

[22] Martin Cohn and Abraham Lempel. *On fast M-sequence transforms (Corresp.).* IEEE, 1977.

[23] Angelo Farina. *Simultaneous measurement of impulse response and distortion with a swept-sine technique.* Audio Engineering Society Convention 108, 2000.

[24] Swen Müller and Paulo Massarani. *Transfer-function measurement with sweeps.* Journal of Audio Engineering Society, 2001.

[25] Abhijit Mitra. *On pseudo-random and orthogonal binary spreading sequences.* International Journal of Information and Communication Engineering, 2008.

[26] Dilip V Sarwate and Michael B Pursley. *Crosscorrelation properties of pseudorandom and related sequences.* IEEE, 1980.

[27] Robert J McEliece. *Finite fields for computer scientists and engineers (book).* Springer Science & Business Media, 2012.

[28] Manfred Schroeder. *Number Theory in Science and Communication: With Applications in Cryptography, Physics, Digital Information, Computing and Self-Similarity (book).* Springer Berlin Heidelberg, 5th ed. 2009.

[29] Henrik Schulze. *Theory and applications of OFDM and CDMA wideband wireless communications (book).* Wiley, 2005.

[30] Shinnosuke Hirata and Hiroyuki Hachiya. *Truncation-noise characteristics of finite-length M-sequence.* Acoustical Society of Japan, 2015.

[31] Robert Gold. *Optimal binary sequences for spread spectrum multiplexing (Corresp.).* IEEE Transactions on Information Theory, 1967.

[32] Kemal M Kiymik, İnan Güler, Orkun Hasekioglu, and Mustafa Karaman. *Ultrasound imaging based on multiple beamforming with coded excitation.* Elsevier, Journal of Signal processing, 1997.

[33] Jorge Bohórquez and Özcan Özdamar. *Signal to noise ratio analysis of maximum length sequence deconvolution of overlapping evoked potentials.* The Journal of the Acoustical Society of America, 2006.

[34] Peter Svensson and Johan L. Nielsen. *Errors in MLS Measurements Caused by Time Variance in Acoustic Systems.* Journal of Audio Engineering Society, 1999.

[35] Jiyuan Liu and Finn Jacobsen. *An MLS coherence function and its performance in measurements on time-varying systems.* 6th International Congress on Sound and Vibration, 1999.

[36] Michael Vorländer and Malte Kob. *Practical aspects of MLS measurements in building acoustics.* Elsevier, Applied Acoustics, 1997.

[37] Andreas Springer, Mario Huemer, Leonhard Reindl, Clemens CW Ruppel, Alfred Pohl, Franz Seifert, Wolfgang Gugler, and Robert Weigel. *A robust ultra-broad-band wireless communication system using SAW chirped delay lines.* IEEE, 1998.

[38] Max Kowatsch and Johann T Lafferl. *A spread-spectrum concept combining chirp modulation and pseudonoise coding.* IEEE, 1983.

[39] Harry F Olson. *Direct radiator loudspeaker enclosures.* Journal of the Audio Engineering Society, 1969.

[40] Timothy Van Renterghem, Annelies Bockstael, Valentine De Weirt, and Dick Botteldooren. *Annoyance, detection and recognition of wind turbine noise.* Elsevier, Science of the Total Environment, 2013.

[41] John C Bancroft. *Introduction to matched filters.* CREWES Research, 2002.

[42] Nadav Levanon and Eli Mozeson. *Radar signals.* John Wiley & Sons, 2004.

[43] Leif Bjørnø. *Applied Underwater Acoustics (book).* Elsevier, 2017.

[44] Umer Hassan and Sabieh Anwar. *Reducing noise by repetition: Introduction to signal averaging.* European journal of Physics, 2010.

[45] Kevin McClaning and Tom Vito. *Radio Receiver Design.(The Book End).* Horizon House Publications, Inc. Microwave Journal, 2002.

[46] Reto Pieren. *Auralization of Environmental Acoustical Sceneries: Synthesis of Road Traffic, Railway and Wind Turbine Noise.* PhD thesis, Delft University of Technology, 2018.

[47] Peter J. Brockwell and Rainer Dahlhaus. *Generalized Levinson–Durbin and burg algorithms.* Elsevier, Journal of Econometrics, 2004.

A. MATLAB function for room IR generating

```
function impres = creexpimpres(fs,Nsamp,T60,V,dist);
%  impres = creexpimpres(fs,Nsamp,T60,V,dist);
%
% Create an impulse response with an ideally
% exponential decay. Relates to the direct sound
% at 1 m distance.
% If the optional parameter 'dist' is given, a direct sound is added
% as a perfect pulse and the exponential part starts after this.

c = 343;
impres = randn(Nsamp,1);
t = [0:Nsamp-1]/fs;
tau = T60/6.91;
expwin = exp(-t/tau)';
clear t
impres = impres.*expwin;
clear expwin
scale = sum(impres.^2)/(100*pi*T60/V);
impres = impres/sqrt(scale);
if nargin == 5,
   ncut = floor(dist/c*fs);
   impres(1:ncut) = zeros(ncut,1);
   impres(ncut) = 1/dist;
end
```

B. MATLAB code for Monte Carlo average simulator

```
% Code written by Mikolaj Jaworski for master thesis 2020.
% This code is a Monte Carlo simulation for different source-to-receiver
% distances for three different background noise measurements.

clear all
clc
close all

%% Constants %%
room.speed = 343; % defines the speed of sound in the room, c.
room.reverberation = 1.2; % defines sound reverberation in the room, T60.
room.volume = 300; % defines the room volume
room.distance = [3:2:15]; % source-to-receiver distance
room.samples = 2^14; % defines how many samples the IR of the room contains
%% imports %%
[noise.background,sampling.rate] = audioread('Conference.wav'); % imports background noise
sampling.period = 1/sampling.rate; % sampling period
noise.background = noise.background(sampling.rate*1:end); % signal start at 1sec
%% Change theese variables %%
name = 'conference'; % room in which background noise recording was conducted
L = [2047,4095]; % vector containing different M-sequence lengths
L = L(2); % m-sequence length choise for Monte Carlo
N = 20; % Number of averages
signal_length = 2*N*L+2000+room.samples-1; % legnth of noise signal

monte_carlo_size = 500; % number of Monte Carlo iterations

x.single = mls(L); % One cycle of M-sequence

% Switch for choosing signal scaling which yield unperceivable
```

```
    signal at 1m
% the first argument of the vector is unperceivable for the author.
    For
% each element of the vector a damping of -1dB is done. E.g for the
    hallway
% recording the SNR.scaling elements result in the following A-
    weighted
% SNR's: [-9dB,-10dB,-11dB,-12dB] at 1m source-to-receiver distance.
%
switch name
    case 'conference'
        stationary_start = 3;
        stationary_stop = 6;
        SNR.scaling = [0.00005685,0.00005063,0.00004515,0.00004025];
    case 'library'
        stationary_start = 5;
        stationary_stop = 8;
        SNR.scaling = [0.0000635,0.00005658,0.000050475,0.00004495];
end
%% Filter design %%
h.room = zeros(length(room.distance),room.samples); % Matrix
    containing differnt RIR

for i = 1:length(room.distance)
    % function generating different RIR's by Peter Svensson
    h.room(i,:) = creexpimpres(sampling.rate, room.samples,
                    ...
                        room.reverberation,room.volume, room.
                            distance(i));
end

Butteworth_order = 1; % High-pass filter order
cut_off_frequency = 500; % High-pass filter cut-off frequency
AR_order = 2; % AR-model order

% Defines stationart noise (none transients in noise signal)
stationary_noise.start = stationary_start*sampling.rate;
stationary_noise.stop = stationary_stop*sampling.rate;
noise.backgroundStationary = noise.background(stationary_noise.start
    : ...
                                                1:stationary_noise.
                                                    stop);

b.averaging = zeros(1,L*N+1);
b.averaging(1) = 1;
```

```
67 b.averaging(end) = -1;
68
69 a.averaging = zeros(1,L+1);
70 a.averaging(1) = 1;
71 a.averaging(end) = -1;
72
73 h.averaging = impz(b.averaging,a.averaging);
74
75 h.matched = x.single(end:-1:1);
76
77 a.aryule = aryule(noise.backgroundStationary,AR_order); % AR
       coefficients
78 h.aryule = impz(1,a.aryule);
79 h.aryuleInverse = impz(a.aryule,1);
80
81 [b.butter,a.butter]=butter(1,2*cut_off_frequency/sampling.rate,'high
       '); % HP coefficients
82 h.HP = impz(b.butter,a.butter);
83
84 h.pre = conv(h.HP,h.aryule); % defines h_pre filter
85 h.post = conv(h.aryuleInverse,h.HP); % defines h_post filter
86 %% Measurement signal generator%%
87 x.multiple = x.single; % Makes a vector containing multiple cycles
       of MLS
88
89 for i = 1:N-1
90     x.multiple = [x.multiple;x.single];
91 end
92
93 % Zero padding to minimize truncation effects.
94 zero_pading = zeros(N*L,1);
95 x.multiple = [zero_pading; x.multiple; zeros(2000,1)];
96
97 % filters MLS
98 x.filtered = conv(h.pre,x.multiple);
99
100 x.filtered = x.filtered.*SNR.scaling(4); % chooses SNR 3dB below
       authors perceivability
101
102 s.clean = zeros(length(room.distance),signal_length);
103 for i = 1:length(room.distance)
104     s.clean(i,:) = conv(x.filtered,h.room(i,:));
105 end
106
107 % Defines all variables used for Monte Carlo simulations
```

```
108 max_of_bank = zeros(monte_carlo_size,N,length(room.distance));
109 value_of_bank = zeros(monte_carlo_size,N,length(room.distance));
110 correct_value = zeros(monte_carlo_size,length(room.distance));
111 correct_bank = zeros(monte_carlo_size,length(room.distance));
112
113 % Makes a matrix containing Monte Carlo number of different noise
      signals.
114 noise.monteCarlo = zeros(signal_length,monte_carlo_size);
115 for i = 1:monte_carlo_size
116     noise.monteCarlo(:,i) = noise.background(i:1:i+signal_length-1);
117 end
118
119 %% Post-processing clean signal %%
120 y.clean = filter(h.post,1,y.clean);
121 y.clean = filter(h.matched,1,y.clean);
122
123 y.averagedClean = conv(h.averaging,y.clean);
124
125 averaging_start = N*L+round((room.distance(i)/room.speed)*sampling.
      rate)-1;
126 averaging_stop = 2*N*L+round((room.distance(i)/room.speed)*sampling.
      rate)-1;
127
128 y.averagedClean = y.averagedClean(averaging_start:1:averaging_stop);
129
130 %% Noise signal whitening filter %%
131
132 noise.whitenedPlaceholder = zeros(signal_length+length(zero_pading),
      monte_carlo_size);
133 noise.whitened = zeros(signal_length,monte_carlo_size);
134 noise.averaged = zeros(averaging_stop-averaging_start+1,
      monte_carlo_size);
135
136 %% Post-processing noisy signal %%
137 for j = 1:monte_carlo_size
138     for i = 1:length(room.distance)
139         % filters s[n]+ one of the recorded noise signals with
              h_post[n]
140         y.noisy = filter(h.post,1,s.clean(i,:)'                    ...
141                                                  + noise.monteCarlo
                                                       (:,j));
142
143         y.noisy = filter(h.matched,1,y.noisy);
144         y.averaged = conv(h.averaging,y.noisy);
145         % Scaling done to place correct direct sound at first sample
```

```
            .
146         averaging_start = N*L+floor((room.distance(i)/room.speed)*
                sampling.rate);
147         averaging_stop = 2*N*L+floor((room.distance(i)/room.speed)*
                sampling.rate);
148
149         y.averaged = y.averaged(averaging_start:1:averaging_stop);
150
151         % for-loop for finding max value of each bank
152         for ii = 1:N
153             [max_value,max_sample] = max(y.averaged((ii-1)*L+1:ii*L
                    +1).^2);
154             max_of_bank(j,ii,i) = max_sample;
155             value_of_bank(j,ii,i) = max_value;
156         end
157
158         % for-loop for finding averaging bank that yield correct
                value
159         for ii = 1:N
160             [max_value,max_sample] = max(y.averaged((ii-1)*L+1:ii*L
                    +1).^2);
161             if max_sample == L
162                 correct_bank(j,i) = ii;
163                 correct_value(j,i) = max_value;
164                 break
165             end
166         end
167     end
168 end
```

C. MATLAB code for Monte Carlo threshold simulator

```
% Code written by Mikolaj Jaworski for master thesis 2020.
% This code is a Monte Carlo simulation for different source-to-receiver
% distances for three different background noise measurements. The purpose
% of this simulator is to find the number of errors at different thresholds
% of detection.

clear all
close all
clc
%% imports %%
[noise.background,sampling.rate] = audioread('Library.wav'); % imports background noise
sampling.period = 1/sampling.rate; % sampling period
noise.background = noise.background(sampling.rate*1:end); % signal start at 1sec
name = 'library'; % room in which background noise recording was conducted

% Switch for choosing signal scaling which yield unperceivable signal at 1m
switch name
    case 'conference'
        stationary_start = 3;
        stationary_stop = 6;
        SNR.scaling = [0.00005685,0.00005063,0.00004515,0.00004025];
    case 'library'
        stationary_start = 5;
        stationary_stop = 8;
        SNR.scaling = [0.0000635,0.00005658,0.000050475,0.00004495];
end

% Defines stationart noise (none transients in noise signal)
stationary_noise.start = stationary_start*sampling.rate;
```

```
30 stationary_noise.stop = stationary_stop*sampling.rate;
31 noise.backgroundStationary = noise.background(stationary_noise.start
       : ...
32                                                 1:stationary_noise.
                                                       stop);
33
34 load data_3D_plot.mat % Loads data optained from Monte Carlo to find
       number of averages
35
36 T60 = reverberation;
37 r = distance;
38 nresults = correct_bank;
39
40 i = 6; % Changed to find threshold at different source-to-receiver
       distances
41 %% Constants %%
42 room.speed = 343; % defines the speed of sound in the room, c.
43 room.reverberation = 0.9; % defines sound reverberation in the room,
       T60.
44 room.volume = 300; % defines the room volume
45 room.distance = distance(i); % source-to-receiver distance
46 room.samples = 2^14; % defines how many samples the IR of the room
       contains
47
48 Number_of_averages = nresults(3,i); % Number of averages found by
       means of Monte Carlo
49 L = 2047; % MLS length
50
51 monte_carlo_size = 400; % number of Monte Carlo iterations
52
53 x.single = mls(L); % One cycle of MLS
54
55 threshold = [1:0.1:10]; % Threshold for monte carlo
56
57 %% Variables %%
58 % function generating different RIR's by Peter Svensson
59 h.room = creexpimpres(sampling.rate, room.samples, room.
       reverberation,  ...
60                                                 room.volume, room.
                                                       distance);
61 %% Filter design %%
62 h.placeholder = zeros(L,1);
63 h.placeholder(1) = 1;
64
65
```

```
66 b.averaging = zeros(1,L*N+1);
67 b.averaging(1) = 1;
68 b.averaging(end) = -1;
69
70 a.averaging = zeros(1,L+1);
71 a.averaging(1) = 1;
72 a.averaging(end) = -1;
73
74 h.averaging = impz(b.averaging,a.averaging);
75
76 h.matched = x.single(end:-1:1);
77
78 Butteworth_order = 1; % High-pass filter order
79 cut_off_frequency = 500; % High-pass filter cut-off frequency
80 AR_order = 2; % AR-model order
81
82 a.aryule = aryule(noise.backgroundStationary,AR_order); % AR
       coefficients
83 [b.butter,a.butter]=butter(1,2*cut_off_frequency/sampling.rate,'high
       '); % HP coefficients
84
85 h.aryule = impz(1,a.aryule); % Defines AR-model filter
86 h.aryuleInverse =  impz(a.aryule,1); % Defines inverter AR-model
       filter
87 h.HP = impz(b.butter,a.butter); % Defines high-pass filter
88
89 h.pre = conv(h.HP,h.aryule); % defines h_pre filter
90 h.post = conv(h.aryuleInverse,h.HP); % defines h_post filter
91 %% Signal generator
92 x.multiple = x.single; % Makes a venctor containing multiple cycles
       of MLS
93
94 for i = 1:(monte_carlo_size+2*Number_of_averages+3*2-1)
95     x.multiple = [x.multiple;x.single];
96 end
97
98 % Zero padding to minimize truncation effects.
99 number_of_pads = 1000;
100 x.multiple = [zeros(number_of_pads,1);x.multiple;zeros(
       number_of_pads,1)];
101
102 %% Signal filtering %%
103 x.filtered = conv(h.pre,x.multiple); % MLS filtering
104 x.scaled = x.filtered .* SNR.scaling(1); % Scaling to make signal
       inperceivable at 1m
```

```
105
106 s.clean = conv(x.scaled,h.room); % Convolution of inperceivable
    signal with RIR
107 noise_length = length(s.clean); % Finds the length of noise = signal
108
109 noise.background = noise.background(1:noise_length); % Makes a
    vector containing recorde background noise
110
111 s.noisy = noise.background+s.clean; % Clean signal added noise.
112
113 % Post-processing of clean signal
114 y.clean = conv(h.post,s.clean); % Filtered with inverse AR and HP
    filters
115 y.clean = conv(y.clean,h.matched); % Matched filtering
116 y.clean = conv(y.clean,h.averaging); % Signal averaging
117 % Some samples of the signal are discarded because of instability
    caused
118 % by filtering, averaging and other factors
119 y.clean = y.clean(number_of_pads+(3+Number_of_averages)*L:end-(
    number_of_pads+(3+Number_of_averages)*L));
120
121 % Post-processing of trigger signal
122 y.trigger = conv(h.post,x.scaled); % Filtered with inverse AR and HP
     filters
123 y.trigger = conv(y.trigger,h.matched); % Matched filtering
124 y.trigger = conv(y.trigger,h.averaging); % Signal averaging
125 % Some samples of the signal are discarded because of instability
    caused
126 % by filtering, averaging and other factors
127 y.trigger = y.trigger(number_of_pads+(3+Number_of_averages)*L:end-(
    number_of_pads+(3+Number_of_averages)*L));
128
129 % Post-processing of signal added with noise
130 y.noisy = conv(h.post,s.noisy); % Filtered with inverse AR and HP
    filters
131 y.noisy = conv(y.noisy,h.matched); % Matched filtering
132 y.noisy = conv(y.noisy,h.averaging); % Signal averaging
133 % Some samples of the signal are discarded because of instability
    caused
134 % by filtering, averaging and other factors
135 y.noisy = y.noisy(number_of_pads+(3+Number_of_averages)*L:end-(
    number_of_pads+(3+Number_of_averages)*L));
136
137 [row, column] = find(y.trigger > 0.9*max(y.trigger)); % Finds
    trigger samples
```

```
138
139 measurement_situations = zeros(length(row),L); % matrix containing
      all monte carlo iterations
140
141 for i = 1:length(row)
142     measurement_situations(i,:) = y.noisy(row(i):row(i)+L-1);
143 end
144
145 monte_carlo_size = length(row); % Defines number of Monte Carlo size
146
147 % Defines correct sample and value used to caluclate number of
      errors
148 [correct_value, correct_index] = max(y.clean(row(1):row(1)+L-1));
149 false_alarm = zeros(length(threshold),1); % Makes a vector to
      contain number of false alarm
150 missed_detection = zeros(length(threshold),1); % Vector containing
      failed detections
151 correct_detection = zeros(length(threshold),1); % Vector containing
      correct detections
152
153
154 %% Monte Carlo simulation %%
155 for i = 1:monte_carlo_size
156     for ii = 1:length(threshold)
157         y.rms = measurement_situations(i,:);
158         [value, index] = max(measurement_situations(i,:));
159         if index<3
160             y.rms(1:index+4) = [];
161         else
162             y.rms(index-2:index+2) = [];
163         end
164         y.rms = rms(y.rms);
165         if value/y.rms < threshold(ii)
166             missed_detection(ii,1) = missed_detection(ii,1) +1 ;
167         else
168             if index == correct_index
169                 correct_detection(ii,1) = correct_detection(ii,1)
                      +1;
170             else
171                 false_alarm(ii,1) = false_alarm(ii,1) + 1;
172             end
173         end
174     end
175 end
176 %% File saving %%
```

```
177 fileName = ['Threshold_T60_09_Distance_',num2str(room.distance),'m_L
    ', ...
178         num2str(L),'Number_of_averages',num2str(Number_of_averages),'
                .mat'];
179
180 save(fileName,'false_alarm','missed_detection','correct_detection',
        ...
181                                         'monte_carlo_size','
                                            Number_of_averages')
```

D. MATLAB code for signal preprocessing

```
clc
clear all

close all
%% Variables %%
% Background noise recording with 44100 kHz sampling rate
[noise.background,sampling.rate] = audioread('Opptak_Bakgrunn.wav');
noise.background = noise.background(:,1);

% Stationary part of background noise
stationary_noise.start = 3*sampling.rate;
stationary_noise.stop = 6*sampling.rate;

noise.backgrounStationary = noise.background(stationary_noise.start:
    ...
                                                1:stationary_noise.
                                                   stop);
L = [2047,4095]; % The two signal lengths utilized
L = L(1); % Defines which signal length to generate

AR_order = 2; % Order of AR-model

fcutofffirHP = 500; % High-pass filter cut-off frequency
%% Generate M-sequence %%
x.single = mls(L);
x.single = x.single.*0.1; % Scaling done because of clipping when
    converted to .wav

x.multiple = x.single;

numberOfSamples = 200 * sampling.rate; % Calculates number of
    samples corresponding to 200 seconds
numberOfMsequences = round(numberOfSamples/L); % Finds number of
    sequences

for i = 1:numberOfMsequences-1
    x.multiple = [x.multiple;x.single];
```

```
33 end
34
35 x.multiple = [zeros(2000,1);x.multiple;zeros(2000,1)]; % zeropadding
36
37 %% Filter design %%
38 a.aryule = aryule(noise.backgroundStationary,AR_order); % AR
       coefficients
39 [b.butter,a.butter]=butter(1,2*cut_off_frequency/sampling.rate,'high
       '); % HP coefficients
40
41 b.pre = b.butter;
42 a.pre = [1,a.butter(2)+a.aryule(2),a.butter(2)*a.aryule(2)+a.aryule
       (3), ...
43                                                    a.butter(2)*a.
                                                          aryule(3)];
44 h.pre = impz(b.pre,a.pre);
45 %% Pre-processing of signal used in measurements %%
46
47 x.filtered = filter(b.pre,a.pre,x.multiple); % HP filtering
48
49 %% Generate wave-files %%
50 % Generates .wav file for use in measurements.
51 audiowrite(strcat(num2str(L),'x_msequence.wav'),x.filtered,sampling.
       rate)
```

E. MATLAB code used for signal postprocessing

```
% Mikolaj Jaworski 2020
% Master thesis
% Code for post processing of measured RIR with different SNR's

clc
clear all

close all
%% Variables %%

% Recorded background noise used for AR-model is imported
[noise.background,sampling.rate] = audioread('Opptak_Bakgrunn.wav');
noise.background = noise.background(:,1);

% Stationary part of background noise is defined
stationary_noise.start = 3*sampling.rate;
stationary_noise.stop = 6*sampling.rate;

noise.backgrounStationary = noise.background(stationary_noise.start:...
    1:stationary_noise.stop);

SNR = -4; % SNR based on writers perceivability
N=3; % Number of averages
% Importing measurements from trigger and microphone
[x.measured,sampling.rate] = audioread('2047x_msequence_-4.wav');

x.trigger = x.measured(:,2); % Trigger is one of two stereo channels
x.measured = x.measured(:,1); % Microphone signal is one of two stereo channels

x.measured = x.measured.*-1; % Flips phase because of wrong phase at speaker

x.trigger = x.trigger(1:sampling.rate*60); % Uses 1 minute of recording for simulations
```

```
33 x.measured = x.measured(1:sampling.rate*60);
34
35 upSampling = 16; % up sampling factor
36 sampling.rateUp = sampling.rate*upSampling; % up sampled sampling
       rate
37 samples_included = 5; % Number of samples used to estimate time of
       arrival
38
39 L = [2047,4095]; % MLS lengths
40 L = L(1); % Defines which period of MLS is used
41
42 threshold = [1:0.1:10]; % Threshold for monte carlo
43
44
45 AR_order = 2; % AR-model order
46
47 fcutofffirHP = 500; % cut-off frequency high-pass filter
48
49 x.single = mls(L); % one cycle of MLS used in matched filter
50 %% Filter design %%
51 a.aryule = aryule(noise.background,AR_order); % AR-model filter
52
53 [b.butter,a.butter]=butter(1,2*125/sampling.rate,'high'); % HP
       filter
54
55 b.matched = x.single(end:-1:1); % Matched filter
56
57 b.averaging = zeros(1,L*N+1);
58 b.averaging(1) = 1;
59 b.averaging(end) = -1;
60
61 a.averaging = zeros(1,L+1);
62 a.averaging(1) = 1;
63 a.averaging(end) = -1;
64
65 h.averaging = impz(b.averaging,a.averaging);% Averaging filter
66 h.matched = b.matched; % Matched filter
67 h.HP = impz(b.butter,a.butter); % High-pass filter
68 h.yuleInverse = impz(a.aryule,1); % inverse AR-model
69
70 h.post = conv(h.yuleInverse,h.HP); % Filter post microphone
71 %% signal filtering %%
72 y.trigger = conv(x.trigger,h.post);
73 y.trigger = conv(y.trigger,h.averaging);
74 y.trigger = conv(y.trigger,h.matched);
```

```
75
76 y.measured = conv(x.measured,h.post);
77 y.measured = conv(y.measured,h.averaging);
78 y.estimated = conv(y.measured,h.matched);
79
80 %% Measurement situation defined %%
81 % Some triggers are discarded because of in-stationary caused by
82 % filtering, averaging and other factors
83 [trigger] = find(y.trigger>0.9*max(y.trigger)); % Finds stationary
       triggers
84
85 %% Up-sampling %%
86 monteCarloSize = length(trigger); % Monte-Carlo size same as number
       of measurement situations
87 estimatedTimeOfArrival = zeros(monteCarloSize,length(threshold)); %
       estimated TOA
88 % Number of missed detections cause by signals under threshold
89 missed_detection = zeros(length(threshold),1);
90
91 % Monte Carlo simulations
92 for i = 1:monteCarloSize
93     for ii  = 1:length(threshold)
94
95         y.monteCarlo = y.estimated(trigger(i):trigger(i)+L-1);
96         time.normalSampling = 1/sampling.rate*[0:length(y.monteCarlo
               )-1];
97
98         [value,index] = max(y.monteCarlo);
99         y.rms = y.monteCarlo;
100
101        % Excludes max values from rms calculations
102        if index < samples_included
103            y.rms(1:index+2) = [];
104        elseif index > L - 3
105            y.rms(index-2:end) = [];
106        else
107            y.rms(index-2:index+2) = [];
108        end
109
110        y.rms = rms(y.rms); % caluclates rms of noise seround peak
111
112        % If peak is under threshold missed detection is added and
               none
113        % TOA estimate is obtained for this iteration
114        if value/y.rms < threshold(ii)
```

```
115                 missed_detection(ii,1) = missed_detection(ii,1) +1 ;
116             else
117                 % If statement finds outliers
118                 if index > L - (samples_included+1)
119                     estimatedTimeOfArrival(i,ii)=0;
120                 elseif index < samples_included
121                     estimatedTimeOfArrival(i,ii)=0;
122                 else
123                     % Makes a small window surrounding the maximum peak
124                     y.forUpSampling = y.monteCarlo(index-
                            samples_included:       ...
125                                                     index+
                                                        samples_included
                                                        -1);
126                     % Makes the same window for time
127                     time.forUpSampling = time.normalSampling(index -
                                ...
128                                         samples_included:index+
                                               samples_included-1);
129                     % Up-sampling
130                     y.upSampled = interp(y.forUpSampling,upSampling);
131                     % Makes a new time vector with upSampling times
                            better resolution
132                     temp = length(time.forUpSampling)*upSampling;
133                     time.upSampled = 1/sampling.rateUp.*(0:temp-1);
134                     % Find max of up-sampled peak
135                     [~,index] = max(y.upSampled);
136                     % Estimates TOA based on maxima of up-sampled peak
                            added
137                     % since peak is rarely at 0 seconds the time at
                            which the
138                     % window is generated is added. The 10000
                            multiplication
139                     % factor is there because of MATLAB's rounding of
                            floating
140                     % point arithmetic
141                     estimatedTimeOfArrival(i,ii) = time.upSampled(index)
                            *10000 + time.forUpSampling(1)*10000;
142                 end
143             end
144         end
145 end
146
147 %% File saving %%
148 fileName = ['Estimator','_L_',num2str(L),'_Number_of_averages_'
```

```
                ...
149                                         ,num2str(N),'_SNR_',num2str(SNR),'.mat'
                                              ];
150 save(fileName,'SNR','missed_detection','estimatedTimeOfArrival',
                ...
151                                                                                 ,
                                                                                    monteCarloSize
                                                                                    ','N')
```

F. MATLAB functions for PN-sequence generation

```
function [sequence,L]=genMLS(poly)
% Function for generating maximum length sequences (MLS)
% Function input:
%                 poly = a vector containing primitive polynomial of order m
% Function output:
%                 sequence = The generated MLS
%                 L = Sequence length
% Two example showing input vector structure:
% The primitive polynomial: x^5 + x^2 + 1 should be defined as : [5 2 0]
% The primitive polynomial: x^7 + x + 1 should be defined as : [7 1 0]
% - Created by Mikolaj Jaworski, 2020.
L=2^poly(1)-1;
% Converts polynomial vectors to binary representation.
% examples of conversion:
%                         [5 2 0] = [1 0 1 0 0 1]
%                         [7 1 0] = [1 1 0 0 0 0 0 1]
temp=zeros(1,max(poly)+1);
for i=1:length(poly)
    temp(length(temp)-poly(i))=1;
end
poly=temp;

% Defines the initial state of the Linear Feedback Shift Register (LFSR)
x=[1 zeros(1,length(poly)-2)];
y=zeros(1,L);
for i=1:L
    y(i)=x(end); % y is the variable containing the sequence;
    xi=mod(sum(x.*poly(2:end)),2); % XOR operation
    x(2:end)=x(1:end-1);
    x(1)=xi;
end
sequence = y;
```

```
function [goldSequence] = genGold(numberOfSequences,L)
% Function that generates Gold sequences with sequence period = 2047
% samples. This function takes in desired number of uncorrelated
    sequences
% (minimum 2) and outputs a matrix where each row corresponds to a
% sequence. This function was written by Mikolaj Jaworski in 2020.

% Minimum number of sequences is 2
if numberOfSequences < 2
    fprintf('Number of sequences most be greated than 2\n');
    return
elseif numberOfSequences > 2049
    fprintf('Number of sequences most be greated smaller than 2049 \
       n');
    return
elseif (L ~= 2047) && (L ~= 8191) && (L ~= 511)
    fprintf('The preferred sequence length is not allowed in this
       function \n');
    return
else
    if L == 2047
    % preferred pairs of polynomials
        poly1 = [11 8 5 2 0];
        poly2 = [11 2 0];
    elseif L == 8191
        % preferred pairs of polynomials
        poly1 = [13 4 3 1 0];
        poly2 = [13 12 10 9 7 6 5 1 0];
    elseif L == 511
        poly1 = [9 4 0];
        poly2 = [9 6 4 3 0];
    end
    % sequence period based on polynomial
    n = poly1(1);

    % uses genMLS function to generate two MLS's based on the
       preferred
    % pairs
    [u,~]=genMLS(poly1);
    [v,L]=genMLS(poly2);
    % Makes a matrix for Gold sequence storage
    goldSequence = zeros(numberOfSequences,L);
    % Stores the two preferred pairs
    goldSequence(1,:) = u;
    goldSequence(2,:) = v ;
```

```
42
43      % Checks how many sequences are to be generated
44      if L < 3
45      % If only more that two sequences are desired, these sequences
            are
46      % generated by mod_2 addition of dequence u and the cyclic shift
            of
47      % sequence v
48      else
49          for i = 3:numberOfSequences
50              k = i-3;
51              x = mod(u+circshift(v,k),2);
52              goldSequence(i,:) = x;
53          end
54      end
55
56      [row,col] = size(goldSequence);
57      % For-loop that changes all 1's with -1's and all 0's with 1's.
58      for ii = 1:row
59          for i = 1:col
60              if goldSequence(ii,i) == 1
61                  goldSequence(ii,i) = -1;
62              elseif goldSequence(ii,i) == 0
63                  goldSequence(ii,i) = 1;
64              end
65          end
66      end
67  end
68  end
```

```
function kasamiSequence = genKasami(numberOfSequences,L)
% Function that generates Kasami sequences with sequence period =
    4095
% samples. This function takes in desired number of uncorrelated
    sequences
% (minimum 2) and outputs a matrix where each row corresponds to a
% sequence. This function was written by Mikolaj Jaworski in 2020.

if L == 4095
    poly = [12 7 4 3 0]; % Polynoimial for m-sequence generating
elseif L == 1023
    poly = [10 3 0];
end
n = poly(1); % Order of polynomial
decimator = (2^(n/2)+1); % Order of decimation

u= genMLS(poly); % Generates one MLS
v = downsample(u,decimator); % Decimates the generated MLS

kasamiSequence = zeros(numberOfSequences,L); % Makes a matric for
    sequence storage

v = v';
u = u';

v_multiple = v;

for i = 1:decimator-1
    v_multiple = [v_multiple;v];
end

kasamiSequence(1,:) = u;
kasamiSequence(2,:) = v_multiple;

if numberOfSequences < 3


else
    for i = 3:numberOfSequences
        k = i-3;
        w = circshift(v,k);
        w_multiple = [w];
        for ii = 1:decimator-1
            w_multiple = [w_multiple;w];
        end
```

```
43          kasamiSequence(i,:) = mod(u+w_multiple,2);
44      end
45
46  end
47
48  [row,col] = size(kasamiSequence);
49
50  % For-loop that changes all 1's with -1's and all 0's with 1's.
51  for ii = 1:row
52      for i = 1:col
53          if kasamiSequence(ii,i) == 1
54              kasamiSequence(ii,i) = -1;
55          elseif kasamiSequence(ii,i) == 0
56              kasamiSequence(ii,i) = 1;
57          end
58      end
59  end
60  end
```

G. Catt acoustics settings

G.1. General settings

Project : no project info Creator : CATT-Acoustic v9.1e (build 1.01) / TUCT v2.0e:1.02 (prerelease)

Date/Time : 2020-07-02 13:13:38

General

no of planes : 6

surface area : 476.00 m^2

mean abs. : 35.5 35.5 35.5 35.5 35.5 35.5 35.5 35.5

mean scatt. : 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

no of src. : 1 (incl. array elements : 1)

no of rec. : 4

bkg noise : 45.0 38.0 32.0 28.0 25.0 23.0 21.0 19.0 dB (given average) 35.8 dBA (NCB:28)

res. noise : 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 dB (for noisemap) 7.2 dBA (NCB:0)

head dir. : source

room considered open (pending a full Predict SxR calculation).

Air

rel. humidity : 50

temperature : 20.0 °C

sound speed : 343.3 m/s

density : 1.200 kg/m^3

cha. impedance : 412.0 kg/m^2s

absorption : 1.125E-04 3.208E-04 6.492E-04 1.131E-03 2.498E-03 7.692E-03 2.734E-02 9.249E-02 1/m (125-16k Hz)

no audience maps defined.

SOURCE-id (POS m) DIRECTIVITY (AIMPOS m) ROTATION MAPRAYFACTOR Directivity-file date/time:

A0 (0.100 4.100 1.450) OMNI.SD0 (0.100 5.100 1.450) 0.0˙ 1.000

Lp1m_a : 96.0 96.0 96.0 96.0 96.0 96.0 96.0 96.0 dB (125-16k Hz)

Incoherent source

A1 (0.100 3.900 1.250) OMNI.SD0 (0.100 4.900 1.250) 0.0˙ 1.000

Incoherent source

Lp1m_a : 96.0 96.0 96.0 96.0 96.0 96.0 96.0 96.0 dB (125-16k Hz)

Incoherent source

A2 (-0.100 4.100 1.250) OMNI.SD0 (-0.100 5.100 1.250) 0.0˙ 1.000

Lp1m_a : 96.0 96.0 96.0 96.0 96.0 96.0 96.0 96.0 dB (125-16k Hz)

Incoherent source

A3 (-0.100 3.900 1.450) OMNI.SD0 (-0.100 4.900 1.450) 0.0˙ 1.000

Lp1m_a : 96.0 96.0 96.0 96.0 96.0 96.0 96.0 96.0 dB (125-16k Hz)

Incoherent source

RECEIVERS (pos m) (aimpos m) [aimvector m] individual_noise:

01 (1.000 4.000 1.350) (-0.100 3.900 1.450) [-0.992 -0.090 0.090]

02 (2.500 6.500 1.500) (-0.100 3.900 1.450) [-0.707 -0.707 -0.014]

03 (-2.000 6.000 1.200) (-0.100 3.900 1.450) [0.668 -0.739 0.088]

04 (-5.000 2.500 1.600) (-0.100 3.900 1.450) [0.961 0.275 -0.029]

G.2. Room geometry

;MASTER.GEO

;PROJECT=Project

;Constant declaration

LOCAL h = 3 ;room height in m

LOCAL w = 16 ;room width

LOCAL d = 10 ;room depth

ABS walls ¡53 53 53 53 53 53¿ L ¡10 10 10 10 10 10¿ 255 255 0 ;

ABS floor ¡01 01 01 01 01 01¿ L ¡10 10 10 10 10 10¿ 0 255 0 ;

CORNERS

;floor corners

1 -w/2 0 0

2 -w/2 d 0

3 w/2 d 0

4 w/2 0 0

;ceiling corner

11 -w/2 0 h

12 -w/2 d h

13 w/2 d h

14 w/2 0 h

PLANES

[1 floor / 4 3 2 1 / floor]

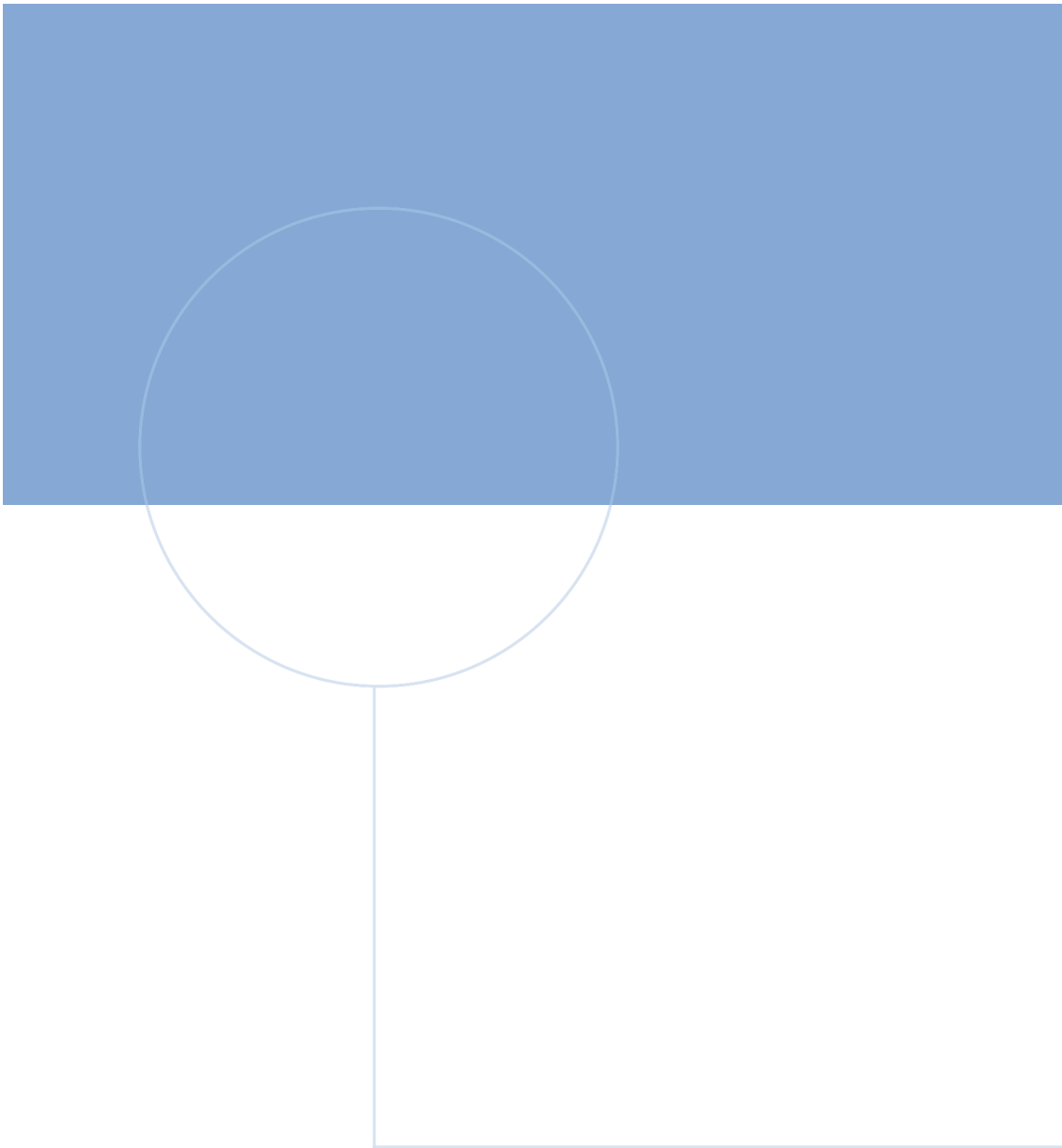
[2 ceiling / 11 12 13 14 / walls]

[3 first wall / 1 11 14 4 / walls]

[4 second wall / 3 13 12 2 / walls]

[5 third wall / 2 12 11 1 / walls]

[6 fourth wall / 4 14 13 3 / walls]



NTNU

Kunnskap for en bedre verden