Noise Statistics Estimation based on adaptive filtering

by

M. Rijkeboer & W. Sewnarain

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on July 2, 2019 at 14:00.

Project duration:	March 1, 2019 – July 5, 2019	
Thesis committee:	Prof. dr. I.E. Lager,	TU Delft, Chair
	Dr. ir. R.C. Hendriks,	TU Delft, Supervisor
	Dr. ir. A. Koutrouvelis,	TU Delft, Daily supervisor
	Dr. J. Martinez,	TU Delft, Jury member

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

At virtually every public venue, announcements for visitors are made via a public addressing system. It is important that announcements transmitted by means of a public addressing system are not only audible, but also well understood by the general public. This thesis covers one of three subsystems required to make a product that is capable of improving intelligibility of speech based on noise statistics in a room. Moreover, these statistics allow for an automatic volume control in order for listeners to experience an improved audio level. Therefore, this thesis aims at estimating the noise statistics in real time, while prior knowledge on the distorting announcement signal is available. Consequently, the concept of adaptive filtering deems suiting and is extensively studied. In order to meet the real time processing constraint, members of least mean squares algorithms are examined. Therefore, the method of steepest decent is covered, leading to the expounding of the traditional least mean squares (LMS) algorithm and the normalized LMS (NLMS) algorithm. By assessment of the algorithms regarding different step sizes and filter lengths, the NLMS algorithm is shown to be superior in terms of faster convergence speed and better stability characteristics considering similar conditions for both algorithms. Subsequently, the results show that the NLMS is able to estimate the noise in noise-to-signal ratio's higher than -15dB. Also, its low complexity allows it to be suitable for real time applications, hence meeting the requirements.

Preface

The authors, Mats Rijkeboer and Winay Sewnarain, have written this document for the Bachelor Graduation Project 'Automatic volume control of an audio amplifier' for the degree of Bachelor of Science in Electrical Engineering at Delft University of Technology. It was commissioned by Richard Hendriks and John Schmitz at the faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS).

This report documents nine weeks, which were spent developing a system that adjusts volume automatically in order for listeners to experience an improved audio level. Moreover, this system improves on speech intelligibility for listeners in a noisy environment. This report describes the principle of adaptive filtering to estimate the noise statistics in a noisy environment. Based on this information, the intelligibility of speech will be improved and the volume of the pre-amplifier will be adjusted. These two subjects will be treated in separate theses.

We would like to thank Richard Hendriks and John Schmitz for making this project available to us and for providing the necessary guidelines and guidance. Special thanks goes out to Andreas Koutrouvelis for his adequate advice during this project. We also would like to thank, Thijs Timmer, Quinten van Wingerden, Bob Luppes and Ellen Riemens for the great group dynamics and atmosphere.

M. Rijkeboer & W. Sewnarain
Delft, July 2019

Contents

1 Introduction

1.1. Intelligibility-Enhancing Automatic Volume Control objective

At many public venues, announcements for visitors are made via a public addressing system (PA). For example to inform passengers about delays at train stations or safety procedures in planes. It is important that announcements transmitted by means of a PA system are not only audible, but also well understood by the general public. In other words, announcements should be intelligible. A simplified model of a PA system can be seen in Figure 1.1. A speech signal is sent through an amplifier and played by a loudspeaker. Subsequently, the transmitted signal is altered by the involved environment. Noise present in the room will also distort the transmitted signal. Thus, the perceivable signal contains a distorted version of the initial announcement. Therefore it might occur that the message is not understandable anymore.

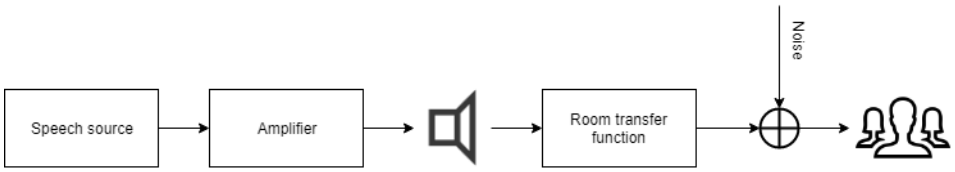

Figure 1.1: A schematic drawing of the PA system model including the amplification of an announcement and its alteration due to noise and a room transfer function

.

In order to get the message across, it is vital that a transmitted announcement is intelligible. Unfortunately, this is often not the case according to [1, 2]. When noise is present, a message will be even less understandable. One approach is to increase the volume of an announcement accordingly. Indeed, the increased volume improves audibility, but does not necessarily boost intelligibility [3]. Besides, a message should still be pleasant to listen to, so simply increasing the volume is not a viable solution. A PA system that is too loud might startle people or cause disturbance at surrounding non-target places. In extreme cases it might hurt or damage the human ear. Therefore a PA system has to keep the announcements pleasant to listen to, whilst maintaining intelligibility.

In order to tackle this problem, three sub tasks are considered. Firstly, noise statistics in the room should be estimated in order to compensate for the room transfer function. Hereafter the speech signal has to be adjusted accordingly for intelligibility enhancement. Lastly speech should be amplified to be played through a loudspeaker. This thesis will focus on one of the sub tasks, namely estimating the noise statistics. Enhancement of intelligibility and the design of the amplifier are discussed in [4] and [5].

1.2. Intelligibility-Enhancing Automatic Volume Control overview

Three subsystems will be designed to form a complete product that enhances intelligibility. Each part will solve one of the previously stated problems in section 1.1.

The noise statistics estimation subsystem will receive a recording of the speech played over the PA system in a noisy environment. It will also receive the clean speech (without noise) that is going to be played. Both signals will be used to estimate the noise.

The noise statistics estimation subsystem will then send the estimated noise to the intelligibility enhancement subgroup. This subgroup will improve the intelligibility of the speech segment that is going to be played next, based on the noise and calculate how much the signal needs to be amplified. The improved speech will be send to both subsystems and the amplification factor will be send the the amplifier and noise cancellation subsystem.

The amplifier and noise cancellation subgroup will do the pre-amplification of the signal based on the amplification factor. It will also receive a recording of the environment and use this to actively suppress low frequency noise. A schematic overview of how all parts are connected to each other can be seen in Figure 1.2.

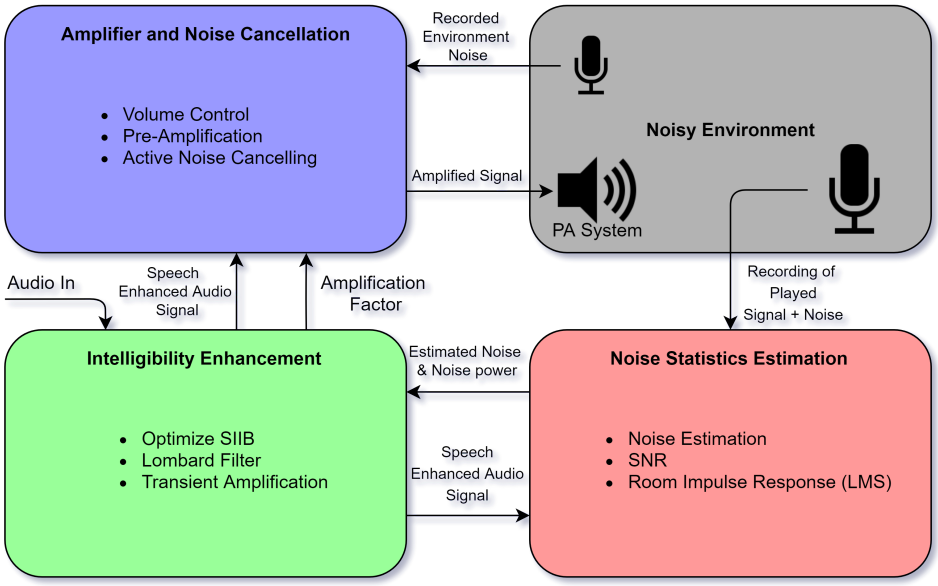

Figure 1.2: Schematic overview of the product including the different subsystems, in a variety of colors, to enhance intelligibility and adjust the volume automatically. In every box, the explanation of the subsystem is stated accordingly.

1.3. Thesis objective

Speech is played in a room through a loudspeaker. Simultaneously, a microphone will record. The received signal is the transmitted speech signal convoluted with a room impulse response (RIR) together with noise. An overview of the situation can be seen in Figure 1.3. In this diagram $s(n)$ is a speech signal played through a speaker, $h(n)$ is the RIR (Room Impulse Response) and $z(n)$ is $s(n)$ convoluted with $h(n)$. In other words, $z(n)$ is the part of the recorded speech correlated with the transmitted speech. The noise present in a room is indicated by $n(n)$ and $d(n)$ is the total recorded signal. The aim of this project is to recover $n(n)$ from $d(n)$, from which a noise power spectral density and signal-to-noise ratio will be conducted. Consequently the research question posed in this thesis will be as follows: In what ways can noise statistics be estimated from noisy speech signals provided that clean speech signals are known?

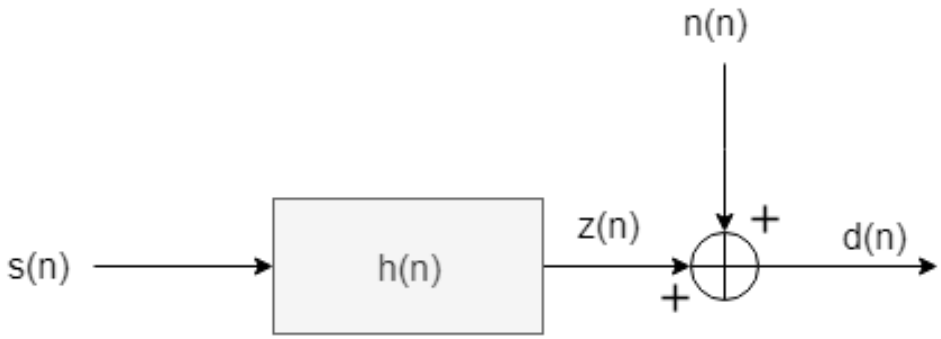

Figure 1.3: Block diagram of the situation, in which $s(n)$ is the clean speech signal and $h(n)$ represents the room transfer function. Additionally, $z(n)$ represents the altered speech signal and combined with the noise signal $n(n)$ they form the received microphone signal $d(n)$. Source:[6].

1.4. State-of-the-art Analysis

1.4.1. Spectral Subtraction

One of the possible ways to estimate the noise is the way that it is done in spectral subtraction [7]. Spectral subtraction is one of the oldest simplest algorithm to enhance speech recorded in a noisy environment. In this method, the estimated noise is subtracted from the received signal, leaving only the transmitted signal. The noise is estimated during speech pauses[8]. The noise spectrum is obtained by taking the average of the consecutive frames during speech pauses:

$$\left|\widehat{N_S}(\omega)\right|^2 = \frac{1}{M}\sum_{j=0}^{M-1}\left|N_{SP_j}(\omega)\right|^2. \tag{1.1}$$

In this equation, M is the total number of frames during speech pauses, $\left|\widehat{N_S}(\omega)\right|^2$ is the estimated noise power during a speech frame and $\left|N_{SP_j}(\omega)\right|^2$ is the noise power during a speech pause frame. The problem with this method is that it assumes that the noise is uncorrelated and stationary. However, in this case noise changes over time. Another problem is that it requires a sufficient amount of non-speech parts. Finding out when there is a pause in a noisy speech signal might be a challenge. However, in this case the clean speech signal is known beforehand, which means this can be done rather trivially by looking at the clean speech power.

1.4.2. Generalized sidelobe canceller (GSC)

The concept of GSC is based on beam forming. This in contrary to the above spectral subtraction method that has been posed for single channel estimations or single microphone setups. In [9] a so-called Griffiths-Jim beamformer or GSC approach is described for acoustic feedback and noise cancellation.

While both feedback and noise cancellation are not relevant to our design requirements, a similar approach can be used to determine the noise PSD and noise signal. In Figure 1.4 it can be seen that by determining a blocking filter B that has a zero in the direction of speech signal v(k), an uncorrelated signal to v(k), u(k) is obtained. Filter M removes signals coming form another direction than v(k). As noise might be generated from several locations (including the direction of v(k)), changes are that there is still noise present in d(k). A correlation filter f(k) will be used to remove all correlated signals with u(k) from d(k) and add them to u(k) (noise in d(k) should also belong to noise signal u(k)).

In theory SNR values and noise PSD could be calculated from the resulting signals. However this is all provided that there is no signal leakage through B, meaning that u(k) has to be uncorrelated from v(k). This means that GSC works better in a space where there is little reflections form the speech signal to prevent correlated noise. In [10] an approach is proposed to determine the Direct-to-Reverberant ratio, which can be used to specify whether a room is suitable for GSC. Another disadvantage is the fact that the direction and location of the speech has to be known for determining B and M. Also, in a different acoustical setting, B and

M will have to be adjusted. A solution to that problem might be adaptive beam forming, as it adjusts the direction path according to changes in the acoustical setting for performance enhancement[11]. Also, a setup like this requires a certain distance between microphones and loudspeaker in order to perform cancellation. Advantages are obviously noise estimation based on the location of the speech and noise signals. This in contrast with single channel setups that can only determine noise by spectral information[9].

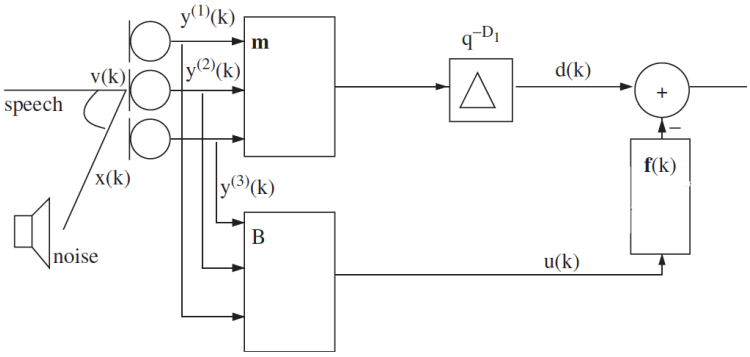

Figure 1.4: A Griffiths–Jim beamformer. Noise signal u(k) does not contain a component correlated with the speech signal v(k), and hence forms a noise reference. Source:[9].

1.4.3. Adaptive Filtering

Adaptive filtering is a technique used in many audio and image processing techniques due to its simplicity and robustness. Examples of such applications are adaptive noise cancellation (ANC), acoustic echo cancellation (AEC), equalization of communications channels or bio-medical signal enhancement [12]. A special class of adaptive filters, that are based on the minimization of the mean squared error (MSE), are particularly popular due to their robustness and ease of implementation. Additionally, computational cost is reduced [6].

By definition, adaptive filters adjust their filter coefficients based on variable parameters. This makes them able to adapt to changing circumstances, which means they can track changes in both signal statistics and involved environments. Therefore, adaptive filtering is ideal for non-stationary noise and environments with changing RIR's [13]. Furthermore, this type of filters does not require the whole sequence of a signal to be known beforehand. It bases its necessary computations on previously acquired samples and adapts when newer samples arrive. This procedure makes it well suited for real time time-varying applications.

Compared to the previously discussed spatial methods like a GNC, adaptive filtering lacks the necessity of having the location of the noise and speech sources known beforehand. This makes adaptive filtering superior in terms of location-varying systems. In other words situations in which the location of noise sources varies over time. Examples of these sources could be people walking by or arriving trains. Furthermore, only one microphone is required at minimum, whereas the spatial methods require at least an array of microphones in order to perform signal processing. Also a GNC does not use the available information on the clean speech signal, only it's location.

However, also adaptive filtering has its limitations. Adaptive filtering is an iterative method and uses a step size which controls the adaptation of the filter coefficients. The step size and the length of the filter determine the stability and convergence of the filters. Therefore, the step size influences the time it takes before the filter coefficients change into the desired values (i.e. convergence speed). Secondly, once they reached the desired values, fluctuations around the desired values are still present by means of new samples being processed. Both the convergence speed and subsequent deviation from the desired values, are quantitative ways to measure performance of the filter. Both are heavily influenced by a chosen step size. These measures will be extensively explained in Chapter 3. For real time applications it is important to have fast convergence speeds[13]. However fast convergence might come at cost of bigger deviations form the desired coefficients. Therefore one can not improve on convergence speed, without increasing subsequent fluctuations from the desired values. Several methods can be used in order to chose a step size. One can derive a step size by either trail and error [14] or by basing it on certain criteria [15]. Furthermore a variable step size can be used

[14, 16]. Additionally, length of a filter influences convergence. It should be chosen long enough, so there will be convergence. However longer filters, will slow down the convergence speed and will add extra echos due to the mismatch of the extra coefficients. Therefore research in choosing the filter length has to be conducted for adaptive filtering. [17, 18].

2

Programme of requirements

Requirements have to be met in order to make a successful design. Therefore this chapter lists the overall requirements for the Intelligibility-Enhancing Automatic Volume Control system, as seen in Figure 1.2. Furthermore, additional requirements, only relevant to this thesis, will be listed. Both sets of requirements are used in the last paragraph of this chapter in which a suitable method is chosen from the ones proposed in section 1.4.

2.1. Intelligibility-Enhancing Automatic Volume Control system

In specifying the requirements, some assumptions were made regarding the noise conditions and the existing hardware in the near-end environment. These assumptions can be found below.

Assumptions

1. The near-end noise is uncorrelated with the speech signal.
    - This is a reasonable assumption when near-end noise is defined as a signal that consists of contributions of all noise sources except the loudspeaker.
2. The input of the existing power amplifier has a standardized line level of maximally 0.447 V [19].
    - Typical value for consumer applications.
3. The gain of the existing power amplifier is equal to 25.
    - A set gain of the existing system is needed, to determine how much the system needs to amplify the signal in order to reach a certain output level at the speaker.
4. The output level of the existing power amplifier is less than 100 dBA.
    - From [20], the maximum permissible occupational noise exposure for 2 hours is 100 dBA. Assuming that PA system employees work for 8 hours per day, this means the PA system can make announcements for 25% of the time.
5. The input signal is pre-recorded and noise-free.
    - This is typical for announcements, music and audio-books.

Mandatory Requirements

The mandatory requirements need to be met in order for the overall system to be considered successful. A subdivision is made between functional and non-functional requirements, which represent requirements describing what the system has to do and requirements describing the quality of the system respectively.

1. **Functional Requirements**
   (a) The system must improve intelligibility such that the word recognition rate is at least 90 % in the presence of near-end noise.
   (b) The system must be able to suppress near-end noise in the frequency band from 0 Hz to 500 Hz.
   (c) The system must be able to process speech in the frequency band from 0 Hz to 8 kHz.
   (d) The system must be able to play audio in the frequency band from 20 Hz to 20 kHz.
   (e) The system must be able to process pre-recorded speech in advance (pre-processing).
2. **Non-Functional Requirements**
   (a) The system must operate in SNRs below 15 dB.
   (b) The system must not damage hearing.
   (c) The system must not add more than 3dBA noise to the enhanced speech signal.
   (d) The system must have a maximum pre-processing delay of 5 times the duration of the input signal with a maximum of 20 minutes.
   (e) The system must have a maximum latency of 100 ms without pre-processing.
   (f) The system must not record near-end noise when the system is not broadcasting any audio.
   (g) The system must not store recorded near-end noise longer than the system latency.

Trade-Off Requirements

The trade-off requirements are not necessary to be met, however, if the they are met, the end-users become increasingly satisfied.

1. The system should be plug-and-play.
2. The system should work on a 12 V-input.
3. The system must not add more than 1 dBA noise to the enhanced speech signal.
4. The sound coming out of the system should sound natural according to listening tests.
5. The combined price of all the individual components should not exceed 100 euro.

2.2. Noise Statistics Estimation subsystem

Not all of the requirements and assumptions made in section 2.1 are relevant when looking at the noise estimation subsystem. Moreover, there are also some additional requirements specific to this subsystem that were not mentioned previously. The assumptions and requirements regarding the noise statistics subsystem can be found below.

Assumptions

1. The near-end noise is uncorrelated from the clean speech signal
2. The near-end noise is non-stationary
3. The clean speech transmitted signal is available as an input to the subsystem.
4. The microphone and loudspeaker are a distance of 10 cm apart in height.
   - This is assumed, as if the noise estimation subsystem were to be implemented in the PA system close to the loudspeaker and not as a separate system. This way existing cabling and power of PA systems available close to the loudspeakers can be reused.
5. The room in which the PA system is situated, has dimensions [length, width, height]=[20 19 21]
   - As an example, Rotterdam Central Station has a central hall with height=30m and a passenger platform roof at height = 15m [21]. As the ratio between the two areas inside Rotterdam CS is about 1/2, a height of 21m is a good approximation of a typical space inside a station.
   - It is assumed that the loudspeaker in this theoretical situation can fully cover this area

Mandatory Requirements

The mandatory requirements need to be met in order for the noise statistics estimation subsystem to be considered successful. Again a subdivision is made between functional and non-functional requirements.

1. **Functional Requirements**
   (a) The subsystem must be able to process speech in the frequency band from 0Hz to 8kHz
   (b) The subsystem must be able to estimate the noise statistics from the incoming data samples
   (c) The algorithms must converge.
2. **Non-Functional Requirements**
   (a) The subsystem must be able to estimate the noise with incoming data samples from the microphone having SNRs below +15dB according to [4].
   (b) The subsystem must have a maximum latency of 20 ms
      - To be considered real time [22]
   (c) The subsystem must not record near-end noise when the system is not broadcasting any audio
   (d) The subsystem must not store recorded near-end noise longer than the system latency

Trade-Off Requirements

The trade-off requirements are not necessary to be met, however, if the they are met, the end-users become increasingly satisfied.

1. The subsystem should use algorithms with low computational complexity.
2. The algorithms should converge within one frame of 20 ms

2.3. Proposed Method

Based on the program of requirements, a most suitable method has to be chosen form the ones discussed in section 1.4. The first method, spectral subtraction, requires uncorrelated stationary noise. Noise in our system is uncorrelated, as mentioned in item 1. However, as mentioned in item 2, the noise is not stationary, which makes this not a viable method. The second method is the generalized sidelobe canceller. This method requires a certain distance between a microphone and loudspeaker. According to item 4, the microphone and loudspeaker are only 10 centimeters apart. This makes the GSC not a good solution. The last posed method is adaptive filtering. Adaptive filtering works with non-stationary noise and has no requirement for the distance between the microphone and loudspeaker, contrary to both the GSC and spectral substraction. Additionally, adaptive filtering has low computational cost, which satisfies item 1 and makes it easier to satisfy item 2b. Hence, this thesis poses adaptive filtering to be deemed suitable considering this program of requirements.

3

Adaptive Filters

3.1. Adaptive Noise Canceller

The aim of this project is to recover the noise signal present in a room. In Figure 1.3 $n(n)$ represents this noise signal. In order to retrieve this signal, it has to be estimated from $d(n)$. Theoretically, this could be achieved by subtracting $s(n)$ convoluted with $h(n)$ from the recorded signal $d(n)$. However, $h(n)$ is unknown. The concept of adaptive noise cancelling, which closely relates to this problem, is described in [23]. The word 'noise' in this chapter should be interpreted as the undesired signal, which is in our case the speech signal present in $d(n)$. The noise $n(n)$ however, is the desired signal. Adaptive noise cancelling tries to retrieve this desired signal.

The adaptive noise canceler (ANC) uses an adaptive filter to estimate the RIR. A diagram of the ANC is shown in Figure 3.1. In this diagram $w(n)$ are the adaptive filter coefficients. Together they form an estimation of the RIR $h(n)$. The speech signal $s(n)$ convoluted with $w(n)$ is $y(n)$. It is an estimation of the speech correlated part in $d(n)$. By subtracting $y(n)$ from $d(n)$, an error signal $e(n)$ remains. Based on this signal the filter coefficients will be adjusted to minimize the error signal. When the error is minimized, it represents an estimation of the uncorrelated noise signal $n(n)$. This will be extensively discussed in upcoming paragraphs.

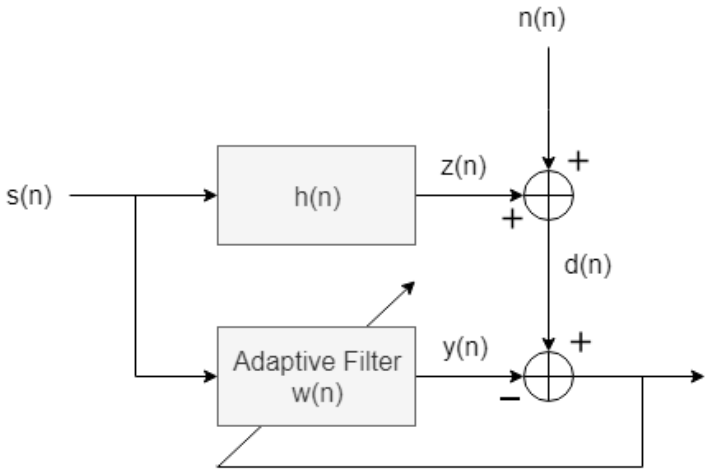

Figure 3.1: A Block diagram of Adaptive Noise Canceller. Herein, $w(n)$ are the adaptive filter coefficients. Together they form an estimation of the RIR $h(n)$. The speech signal $s(n)$ convoluted with $w(n)$ is $y(n)$, which is an estimation of the speech correlated part in $d(n)$. By subtracting $y(n)$ from $d(n)$, an error signal $e(n)$ remains. Source:[6].

3.2. Finite Impulse Response Wiener Filters

As previously introduced, certain adaptive filters modify their coefficients based on the minimization of the MSE. These filters are so-called Wiener filters. Usually, such filters are implemented using a recursive algorithm [14]. Herein filter coefficients are adjusted after each iteration to perform in real time applications. The recursive implemented Wiener filter will be a finite impulse response (FIR) filter. FIR filters have advantages over infinite impulse response (IIR) designs. First of all, FIR filters are chosen for their implementation simplicity. Moreover, they are guaranteed to be bounded input - bounded output (BIBO) stable, which improves control over stability of the filter [14, 24]. In case that the filter tabs are symmetrical around the center, the filter will have a linear phase. This means that it has a constant group delay, which is especially desirable for audio signals as every sample is delayed in the same way [24]. Finally, a FIR filter has lower quantization error sensitivity than IIR [24]. This is desirable, as a digital filter will be implemented.

By considering the adaptive filter as a FIR filter, the output $y(n)$ can be described by Equation 3.1. Here w^T is the transposed vector of filter coefficients w. The amount of filter coefficients is expressed by M. Each iteration, the filter will process M samples of $s(n)$. In Figure 3.2, this equation is visualized. Herein it is clearly visible that filter coefficients w are weights in which previous samples of $s(n)$ form the current estimated speech sample $y(n)$.

$$y(n) = \sum_{k=0}^{M} w(k)s(n-k) = w^T s(n). \tag{3.1}$$

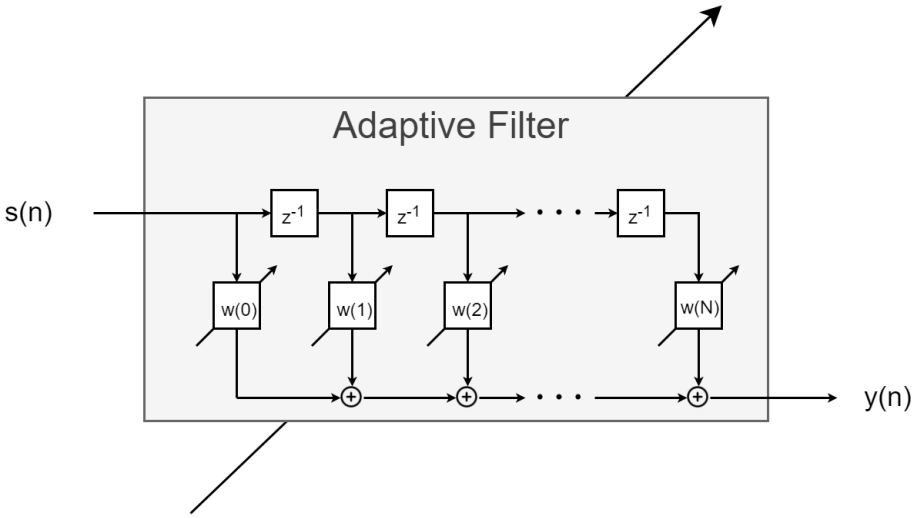

Figure 3.2: An illustration of the adaptive filter. z^{-1} denote delay elements.

The goal of a Wiener filter is to adjust $w(n)$ in such a way that $y(n)$ is as closely related to $d(n)$ as possible. It does this by finding the filter coefficients of w that minimize the MSE. Therefore, $y(n)$ has to be subtracted from $d(n)$ to estimate the error signal $e(n)$, as can be seen in Figure 3.1. The error signal can be expressed as follows:

$$e(n) = d(n) - y(n) = d(n) - w^T s(n). \tag{3.2}$$

and the subsequent mean-squared error as:

$$\xi(n) = E\left[e^2(n)\right]. \tag{3.3}$$

In this equation, $E[\cdot]$ stand for the expected value. When the MSE is minimized, the resulting error signal will be a good estimate of the noise signal. This can be proven by rewriting Equation 3.3 in the following way:

$$\begin{aligned}\xi(n) &= E\left[(d(n)-y(n))^2\right] \\ &= E\left[(z(n)+n(n)-y(n))^2\right] \\ &= E\left[n^2(n)\right]+E\left[(z(n)-y(n))^2\right]-2E\left[n(n)(z(n)-y(n))\right] \\ &= E\left[n^2(n)\right]+E\left[(z(n)-y(n))^2\right].\end{aligned} \tag{3.4}$$

By the assumption that the noise and the speech signal are uncorrelated, the cross-correlation term $2E\left[n(n)(z(n)-y(n))\right]$ from Equation 3.4 vanishes. That means that minimizing the MSE is achieved by minimizing $E\left[z(n)-y(n)^2\right]$ as $E\left[n^2(n)\right]$ is unchangeable . Thus, the MSE is minimized when $y(n)$ is equal to $z(n)$ and hence the resulting error signal is equal to the noise.

In order minimize the error, the filter coefficients need to be found that minimize the MSE. These can be found by taking the partial derivative of $\xi(n)$ with respect to $w(n)$ and setting it equal to zero [14]. The MSE can be written as:

$$\begin{aligned}\xi(n) &= E\left[(d(n)-y(n))^2\right] \\ &= E\left[d^2(n)\right]+E\left[y^2(n)\right]-2E\left[d(n)y(n)\right] \\ &= E\left[d^2(n)\right]+E\left[\left(\sum_{k=0}^{M} w(k)s(n-k)\right)^2\right]-2E\left[d(n)\sum_{k=0}^{M} w(k)s(n-k)\right].\end{aligned} \tag{3.5}$$

The derivative of Equation 3.5 can be described by Equation 3.6.

$$\begin{aligned}\frac{\partial}{\partial w(n)}\xi(n) &= \frac{\partial}{\partial w(n)}\left(E\left[d^2(n)\right]+E\left[\left(\sum_{k=0}^{M} w(k)s(n-k)\right)^2\right]-2E\left[d(n)\sum_{k=0}^{M} w(k)s(n-k)\right]\right) \\ &= 2\sum_{l=0}^{M} E\left[s(n-l)s(n-k)\right]w(l)-2E\left[s(n-k)d(n)\right] \\ &= 2\sum_{l=0}^{M} R_s(l-k)w(l)-2R_{sd}(k) \qquad k=0,\ldots,M.\end{aligned} \tag{3.6}$$

In Equation 3.6, $R_s(m)$ is the auto-correlation of the clean speech, as described in Equation 3.7. R_{sd} is the cross correlation of the speech and the received signal, as described in Equation 3.8.

$$R_s(n) = E\left[s(n)s(n+m)\right]. \tag{3.7}$$

$$R_{sd}(n) = E\left[s(n)d(n+m)\right]. \tag{3.8}$$

Setting Equation 3.6 equal to zero results in

$$\sum_{l=0}^{M} R_s(l-k)w(l) = R_{sd}(k) \qquad k=0,\ldots,M. \tag{3.9}$$

Equation 3.9 is written in matrix form in Equation 3.10[25]. The auto-correlation R_s can be written as a Toeplitz matrix since for real signals the auto-correlation is symmetric. The filter coefficients can be determined using Equation 3.11. In this equation, T^{-1} is the inverse of the auto-correlation matrix of R_s.

$$\underbrace{\begin{bmatrix} R_s(0) & R_s(1) & \ldots & R_s(N) \\ R_s(1) & R_s(0) & \ldots & R_s(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ R_s(M) & R_s(M-1) & \ldots & R_s(0) \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} w(0) \\ w(1) \\ \vdots \\ w(M) \end{bmatrix}}_{W} = \underbrace{\begin{bmatrix} R_{sd}(0) \\ R_{sd}(1) \\ \vdots \\ R_{sd}(M) \end{bmatrix}}_{v} \tag{3.10}$$

$$W = T^{-1}v. \tag{3.11}$$

This method however would require the whole signal to be available beforehand, which would prevent estimating the noise in real time. For this reason, an iterative method will be used, which does not require the whole signal at once.

3.3. The method of steepest descent

Such a procedure is the steepest decent method. In a iterative way, it tries to estimate the filter coefficients that minimize the MSE. The method starts at the first iteration by taking an initial guess of the filter coefficients. The estimate of the coefficients for the next iteration are formed by adding a correction to the current estimate, which should bring it closer to the desired state. The correction should be in the direction of the maximum descent of the quadratic error. The direction of the steepest descent points in the negative gradient direction [14]. The updated filter coefficient will be

$$w(n+1) = w(n) - \mu\nabla\xi(n). \tag{3.12}$$

Where μ is the step size, which is a positive constant that affects the rate with which the coefficients change. μ determines the rate of the convergence of the filter to the desired state. In Figure 3.3 can be seen what would happen if a small or too big of a value is chosen for μ. If μ is large, the coefficients are changed fast and the convergence will be fast, but the coefficients will not stay very stable. If μ is too large however,there will be no convergence at all. A small value for μ makes the convergence rate slow, but once it is converged, it will stay stable. This is also shown for real signals in chapter 4. According to [14], for jointly wide-sense stationary processes, d(n) and s(n), the filter coefficients will converge if Equation 3.13 is satisfied. In this equation λ_{max} is maximum eigenvalue of the auto-correlation matrix of $s(n)$.

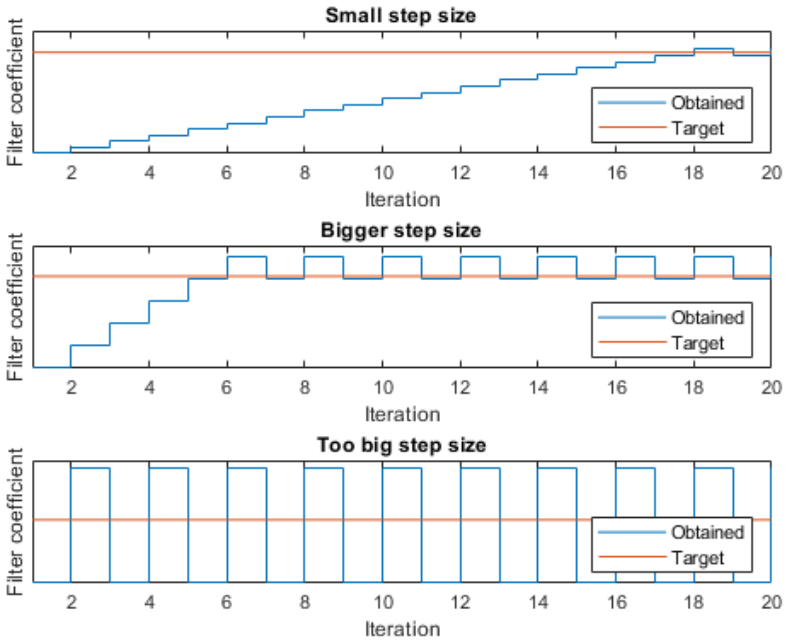

Figure 3.3: Examples of adaption with different step sizes μ.

$$0 < \mu < \frac{2}{\lambda_{max}}. \tag{3.13}$$

The gradient of the MSE is the partial derivative with respect to w. The gradient of the MSE is

$$\begin{aligned}\nabla\xi(n) &= \frac{\partial E\left[e^2(n)\right]}{\partial w} \\ &= 2E\left[e(n)\frac{\partial e(n)}{\partial w}\right] \\ &= 2E\left[e(n)\frac{\partial d(n) - w^T s(n)}{\partial w}\right] \\ &= -2E\left[e(n)s(n)\right]\end{aligned} \tag{3.14}$$

Combining Equation 3.14 and 3.12 results in Equation 3.15 for determining the new filter coefficients. The factor 2 is combined with μ since μ is a constant anyway.

$$w(n+1) = w(n) + \mu E\left[e(n)s(n)\right]. \tag{3.15}$$

Using Equation 3.15 to update the filter coefficients solves the problem of not needing the complete audio sequence. This method is still not suitable for the purpose of this thesis however, since this method requires $E\left[e(n)s(n)\right]$ to be known. This problem is solved by using the least mean squares (LMS) algorithm.

3.4. LMS Algorithm

The LMS algorithm uses a estimate of the expectation $E\left[e(n)s(n)\right]$ to update the filter coefficients[14]. The expectation is replaced with the sample mean

$$\hat{E}\left[e(n)s(n)\right] = \frac{1}{L}\sum_{l=0}^{L-1} e(n-l)s(n-l). \tag{3.16}$$

Combining Equation 3.16 and 3.15 results in

$$w(n+1) = w(n) + \mu\frac{1}{L}\sum_{l=0}^{L-1} e(n-l)s(n-l). \tag{3.17}$$

When a one-point sample mean ($L = 1$) is use, Equation 3.17 can be written as

$$w(n+1) = w(n) + \mu e(n)s(n). \tag{3.18}$$

The filter coefficients are adapted each iteration and the rate at which this is done depends on the step size μ. From Equation 3.18 can be seen why the LMS is considered to have low complexity. Updating each coefficient only takes one multiplication and one addition.

3.5. Normalized LMS

A step size has to be chosen for μ in Equation 3.18. For choosing the step size, a compromise has to be made between having fast convergence and having good stability. Using a variable step size will allow the filter to have a faster convergence while still keeping a good stability. This is because a big step size is used when the filter coefficients are far from the desired values and a small step size is used when they are close.

The maximum eigenvalue in Equation 3.13 can be replaced by the trace of the auto-correlation of $s(n)$ since

$$\lambda_{max} \leq \sum_{k=0}^{p} \lambda_k = tr(R_s). \tag{3.19}$$

where R_s is the auto-correlation of $s(n)$. If $s(n)$ is wide-sense stationary, then R_s is the Toeplitz matrix and the trace becomes

$$tr(R_s) = (p+1)r_x(0) = (p+1)E\left[|s(n)|^2\right] [14]. \tag{3.20}$$

The resulting boundary condition for the step size becomes

$$0 < \mu < \frac{2}{(p+1)E\left[|s(n)|^2\right]}. \tag{3.21}$$

$E\left[|s(n)|^2\right]$ is the power of the (current iteration of) signal and can be estimated by Equation 3.22. In this equation $s^H(n)$ is the Hermitian transpose of $s(n)$.

$$\widehat{E}\left[|s(n)|^2\right] = \frac{1}{p+1}\sum_{k=0}^{p}|s(n-k)|^2 = \frac{s^H(n)s(n)}{p+1} = \frac{\|s(n))\|^2}{p+1}. \tag{3.22}$$

Combining Equation 3.22 and 3.21 gives us the following requirement for the step size

$$0 < \mu < \frac{2}{\|s(n))\|^2}. \tag{3.23}$$

Choosing the step size

$$\mu = \frac{\beta}{\|s(n))\|^2}. \tag{3.24}$$

where β is between zero and two. Choosing the step size of Equation 3.24 for the update scheme of Equation 3.18 gives us Equation 3.25 which is also called the normalized least mean squares (NLMS) algorithm.

$$w(n+1) = w(n) + \beta\frac{s(n)}{\|s(n))\|^2}e(n). \tag{3.25}$$

One problem with Equation 3.25 is that there will be gradient noise amplification when $\|s(n))\|^2$ becomes too small. For this reason, an additional term ϵ is added in the denominator to prevent is from getting too small. The ϵ should be a small positive number. The resulting algorithm is Equation 3.26 in which β can be seen as an initial guess for the step size.

$$w(n+1) = w(n) + \beta\frac{s(n)}{\epsilon + \|s(n))\|^2}e(n). \tag{3.26}$$

3.6. Speech pause estimation

At times, distorting signal $z(n)$ might be silent. During these pauses, all signals present in the received signal $d(n)$ are considered as noise. Looking at ANC (Figure 3.1) , the LMS algorithms will adapt to this lack of distorting signal and alter the coefficients unnecessarily. When the pause ends and signal $z(n)$ is present again , the coefficients have to adapt back to the previous state before the pause. Therefore, signal pauses could be detected to prevent unnecessary adaptation of the coefficients. It is also not necessary to predict the noise signal, while the noise signal is actually present at the microphone in the form of $d(n)$. Naturally, it is desirable to take the real noise signal instead of an estimate.

When a clean speech signal is considered as distorting signal $z(n)$, speech pauses can easily be detected by setting a threshold for the signal power. Powers below this threshold can be considered as pauses. In Figure 3.4, an example of this approach is plotted, in which the speech pauses are marked by a red line.

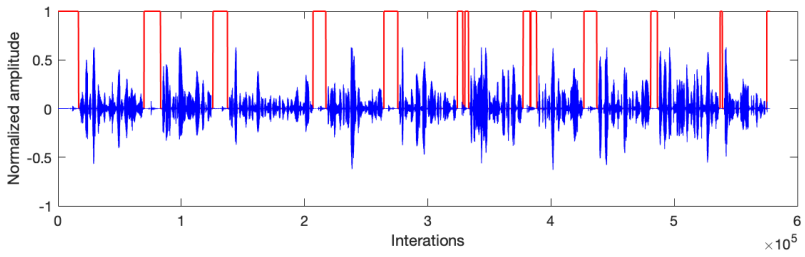

Figure 3.4: An example of speech pause estimation. The red lines represent speech pauses during a clean speech signal, given in blue.

4

Results

In this chapter the results of the performance of the LMS and NLMS algorithms in a simulated MATLAB environment are demonstrated. Firstly the selected performance measures will be discussed in order to quantify the performance. Secondly, a paragraph will be dedicated to the determination of the parameters regarding the LMS algorithms. Thirdly, the performance will be tested in a Monte Carlo simulation. Also, an example will be demonstrated of the algorithm retrieving a slowly varying noise signal from a noisy speech signal. Lastly, performance with different NSR's and computational complexity with different filter lengths are given.

4.1. Simulation set-up

The simulated environment is supposed to recreate Figure 1.3 in a MATLAB environment. The speech and noise signals were downloaded from [26]. A clean speech sample without noise and a non-stationary noise sample were used. This noise sample was selected, because it has varying noise with increasing variation over time. Therefore, it is possible to test the algorithm against different changing speeds of the noise, in other words lower and faster changes. The RIR was created using the Matlab script in subsection A.3.1. In [27] is explained how this script works. A 20x19x21 meter room with the speaker position at (19,18,19.5) and the microphone at (19,18,19.4).

4.2. Performance metrics

In order to measure performance, certain properties of the results have to be quantified using performance metrics. The noise-to-signal ratio (NSR) is the most obvious performance metric, since the goal is to reduce the speech signal as much as possible in order to obtain the estimated noise. The NSR can be calculated by Equation 4.1. A high NSR would mean that the estimated noise is close to the actual noise. This performance metric however only works for simulated signals, as it is required to know the clean noise signal.

$$NSR_{dB} = 10log_{10}\frac{|n(n)|^2}{|n(n)-e(n)|^2}. \tag{4.1}$$

Another performance measure is the convergence speed of the filter. In other words the time it takes for the filter coefficients to reach the desired values. It results in the error signal being an estimate of the noise signal. When the error signal is close to the noise signal, the algorithm has so-called converged. This way it is possible to quantify the number of iterations after which the difference between the signals is within a certain small range. Hence, the algorithm has converged.

Stability is a performance measure that is closely related to the convergence speed, since increasing the stability leads to a slower convergence speed and vice versa. The stability tells us how close to the desired values the estimated filter coefficients will stay after convergence. The stability can be expressed by looking at the Deviation (D) of the estimated squared error and the squared real noise after convergence (4.2).

$$D = \left\| n^2(n) - e^2(n) \right\|. \tag{4.2}$$

The complexity of the algorithm can be measured in terms of processing time. The algorithm was implemented in Matlab however, so this time should be considered as a rough estimate and is only meant as a indication.

Finally the last performance measure is a subjective informal one. This would be a listening test. The performance of the filters will be judged based on how much of the original speech is still heard. Also a comparison between the clean noise signal and the estimated noise signal will be done in which the estimated noise signal should resemble the actual noise.

4.3. Parameters

In order to meet the design requirements, a trade-off should be made between convergence speed and stability. Determining factors in this case are filter length and step size. It is desirable to have a minimal mean squared error as discussed in section 3.2. Thus a minimal MSE should be found within a range of filter lengths and step sizes.

As test signal, a 36-seconds clean speech signal is used. This time length is chosen considering that a typical announcement in a PA system is in the order of several tens of seconds. A non-stationary noise is chosen as distorting factor during the total clean speech signal. Also, an input NSR of -6 dB is selected. The calculation of the MSE will be based on the total noisy signal of 36-seconds. In a real-time setup, this signal will be received in segments of $20ms$ (320 samples) from the intelligibility subsystem. Therefore, 1805 frames of each 320 samples will form the full 36-seconds signal.

By reason of the real time processing constraint, filter lengths larger than 320 could be undesirable, since each segment is only 320 samples. As most of the speech signals transmitted by the total system will be in the order of several seconds, convergence within a second is desirable. Thus, very small step sizes should be avoided. Not only for the sake of convergence speed, but also for quick adaption to changing environments and non-stationary noise. With this in mind, the following subsections will determine step size bounds of the LMS and NLMS algorithms

4.3.1. NLMS

Theoretically, a maximum step size of 2, would still be able to result in the convergence of the filter coefficients. By determining the minimal MSE in a figure similar to Figure 4.1, a filter length of M = 33 and a step size of μ = 0.41 were obtained for the NLMS filter. However by listening to the results of several step sizes with M = 33, it becomes clear that step sizes above 0.1 result in quite heavily distorted estimated noise signals. Therefore step sizes are given a upper bound of 0.1. The lower bound was chosen at 0.03. This bound is chosen by looking at a 3d plot similar to Figure 4.1 in which it became clear that smaller step sizes had rapidly increasing MSE. This is due to very low convergence speed, which is apparently insufficient to adapt quickly to noise changes in the non-stationary noise signal. Therefore, step sizes lower than 0.03 were not plotted as they have significantly larger MSEs that would ruin the visibility of differences between other step sizes. The same was true for filter lengths shorter than 6. This might be because they are to few to give a correct estimate of the RIR. An upper bound of 60 was chosen for the filter length as it became clear that the MSE was only rising after that point. A filter length of M=33 and step size of μ = 0.1 results in a the smallest MSE for the NLMS, according to Figure 4.1.

4.3.2. LMS

A similar approach has been performed for LMS. The minimal MSE can be found in Figure 4.2. It was found at a filter length of M = 93 and a step size of 0.291. By listening to the results of several step sizes with different filter lengths, it was concluded that the maximum step size in which the signal was not heavily distorted was at 0.3. This upper bound is selected at M = 93, with the same reason as in subsection 4.3.1. Namely, disturbance and stability issues. Also a lower bound of 0.012 was chosen due to rapidly increasing MSE.

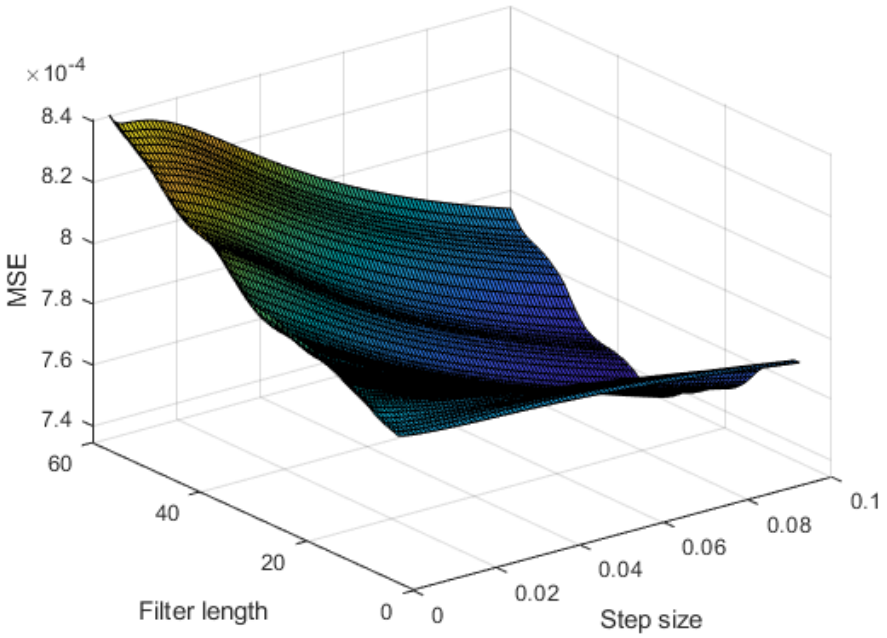

Figure 4.1: A 3D-plot of the MSE using a Normalized LMS filter. The x-axis represents step size μ, the y-axis represents filter length M and the z-axis represent the MSE. The MSE is presented when 577600 samples are processed.

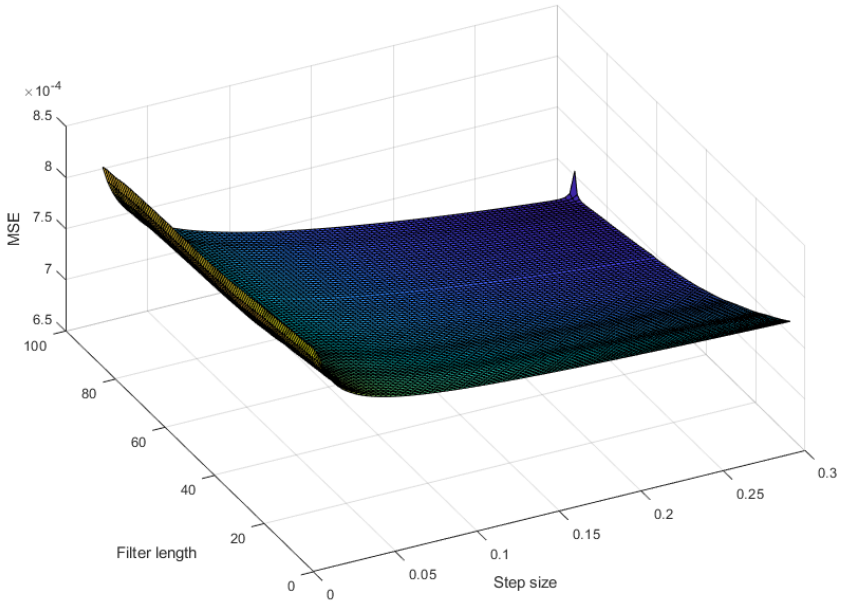

Figure 4.2: A 3D-plot of the MSE using a Traditional LMS filter. The x-axis represents step size μ, the y-axis represents filter length M and the z-axis represent the MSE. The MSE is presented when 577600 samples are processed.

4.4. Monte Carlo results

In this section, the outcome of a Monte Carlo simulation is considered. This gives insight in the average performance of the algorithms during different signal frames. Again the same non-stationary noise sample was selected to be added to a clean speech signal. The NSR is also set to -6 dB. To compare the two algorithms, filter coefficients are set to M = 33 and step size to $\mu = 0.1$, since these are the optimal variables for the NLMS. Furthermore, the initial Wiener filter coefficients will be set to zero. The total signal was divided into pieces of each 20 ms long, which is the same length as how the noise estimation subsystem will receive the intelligibility enhanced speech signal. Therefore it make sense to evaluate a similar length with a Monte Carlo (MC) simulation. In total k = 1805 different MC frames are available. Results of this simulation can be found in Figure 4.3. Using the optimal variables for the LMS result in similar differences and can be seen in the appendix in Figure B.1. In the upper plot it is seen that the normalized LMS algorithm has converged between iteration 175 and 200 iterations. In other words, it converged approximately after 13 ms. On the other hand, the traditional LMS algorithm has not converged within a single frame of 20 ms. From the lower plot one might be concluded that the traditional LMS is more stable than the normalized LMS looking at sheer fluctuations. However, in this case only the normalized LMS has converged and the traditional LMS has not. Thus, after a single 20ms frame no conclusions can be conducted regarding the stability. However when larger frames are considered, in which the LMS also converges, both algorithms will perform similarly. This is due to the fact that only a small amount of samples is considered and a Monte Carlo simulation is used that averages several different noise frames. Hence, the noise can be considered stationary. This means that once converged, the algorithms perform in a steady state. Therefore, the algorithms would display comparable performance as they both approximate the same Wiener filter and have the same step size.

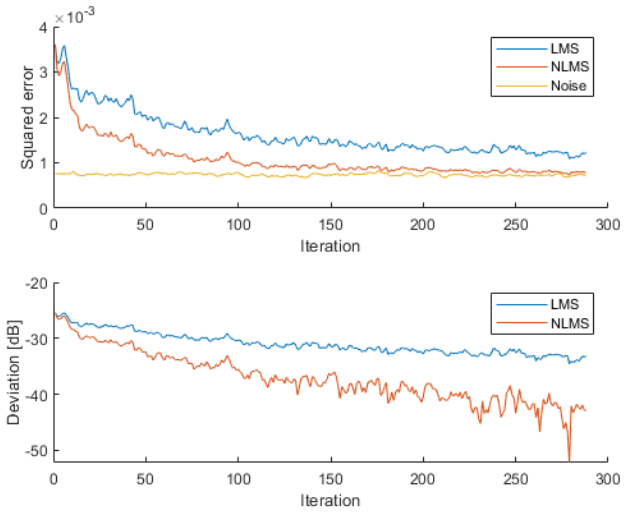

Figure 4.3: A Monte Carlo simulation of the normalized and traditional LMS filter with parameters $\mu = 0.1$ and $M = 33$. The upper graph plots the squared error against iterations of the algorithm. The lower graph plots the deviation of the error signal from the real noise signal, against iterations.

To compare the LMS and NLMS in frames of 20ms, both should converge with a filter length of M = 33. In order for the NLMS algorithm to converge, the step size should be bigger than 0.06. For the LMS algorithm, it should have a step size bigger than 0.39.

Now, a larger frame that does converge will be considered for the LMS. In Figure 4.4 a plot is shown, in which the LMS and NLMS have approximately the same convergence speed. The NLMS still has parameters $\mu = 0.1$ and $M = 33$, however the LMS has a step size $\mu = 0.72$ and $M = 33$. This shows clearly that the NLMS algorithm needs a smaller step size compared to the LMS in order to have the same convergence time.

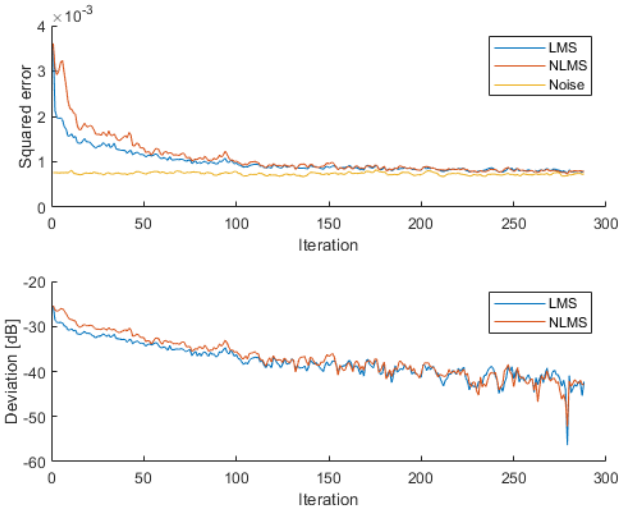

Figure 4.4: A Monte Carlo simulation of the normalized LMS with $\mu = 0.1$ and $M = 33$ and traditional LMS filter $\mu = 0.72$ and $M = 33$. The upper graph plots the squared error against iterations of the algorithm. The lower graph plots the deviation of the error signal from the real noise signal, against iterations.

4.5. Performance of the NLMS filter

This section focuses on determining performance when a 36-seconds announcement is made. Again a non-stationary noise is chosen as distorting factor. Also, a input NSR of -6 dB is selected. The total signal is again divided in frames of 20ms in order to simulate as if the frames would come in real-time from the intelligibility enhancement subsystem. Naturally, the filter coefficients will not be set to zero every frame as was done in the Monte Carlo simulation. In this case the filter coefficients and step size of the previous frame will be used as initial values of the new frame. Essentially as if frames are non-existent.

In Figure 4.5 the actual distorting noise signal is compared to the estimated noise signal during 36 seconds. Now, it can be seen that the slowly-varying noise signal is an amplitude varying noise with a changing rate that increases over time.

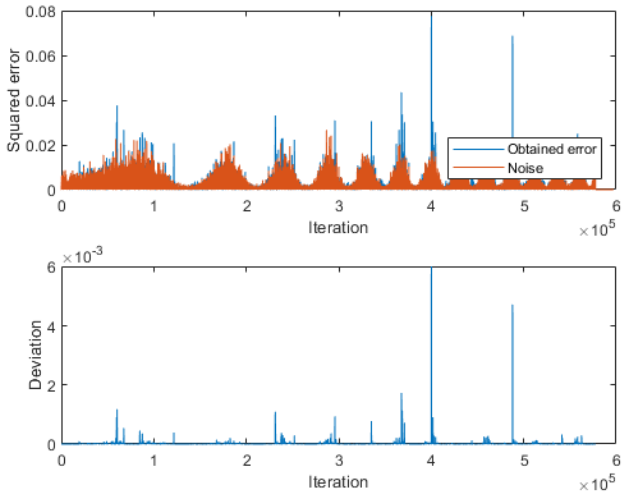

Figure 4.5: In the upper plot, the squared error against iterations of the normalized LMS filter with parameters $\mu = 0.1$ and $M = 33$.

Moreover, noise power spectral density (PSD) plots of both the estimated and real noise show that the estimated noise resembles the actual noise with the exception of some frequency components, that are estimated

to high. These are due to a parts the speech signal that are still slightly present in the estimated noise signal. In Figure 4.6 a comparison between both noise spectra can be found.

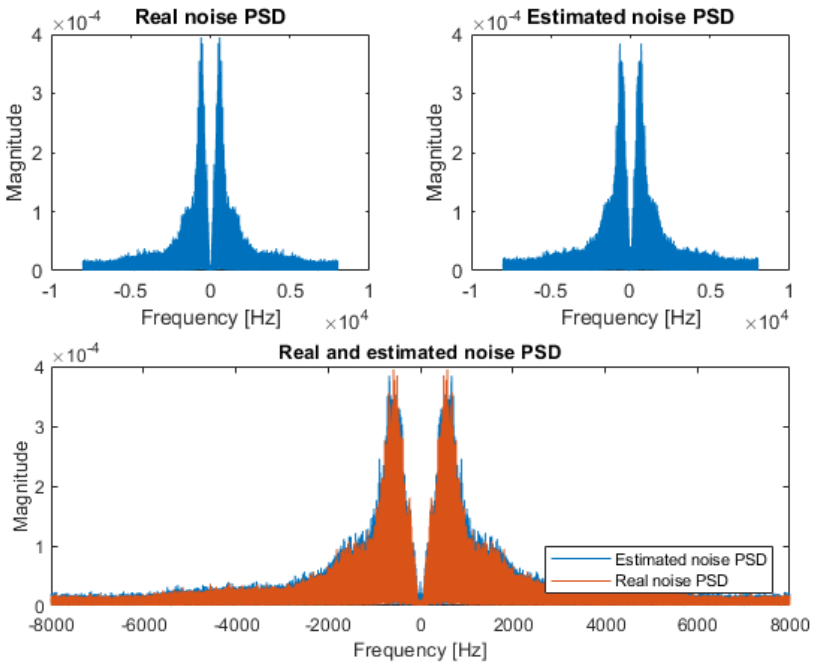

Figure 4.6: PSD of the real (left) and estimated (right) noise using the normalized LMS filter with parameters $\mu = 0.1$ and $M = 33$. Below both plots against each other.

4.6. Input and Output NSR

In previous sections a noise-to-signal ratio of -6 dB was considered. In this section the focus will be on the performance of the NLMS with different NSR. In Figure 4.7 one can see the received NSR from the microphone plotted against the output NSR which will be made available for the other subgroups. In this plot a step size of $\mu = 0.1$ and a filter length of M = 33 is used. It can be seen that the output NSR hardly changes, when the input NSR is higher than 6.2 dB.

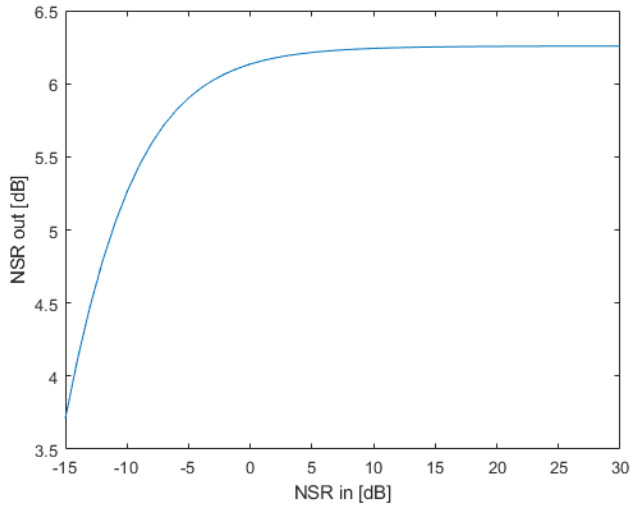

Figure 4.7: Input NSR versus output NSR using the Normalized LMS algorithm.

4.7. Complexity

The goal is to deliver the estimated noise to the other subgroup in real time. Therefore, the time between receiving the recorded signal and sending the estimated noise should be faster than 20 ms [22]. This means that processing each segment of 320 samples must not take longer than 20 ms. The time to process depends on the filter length, since longer filter lengths require less iterations. The average time it takes to process 320 samples for different filter lengths is plotted in Figure 4.8.

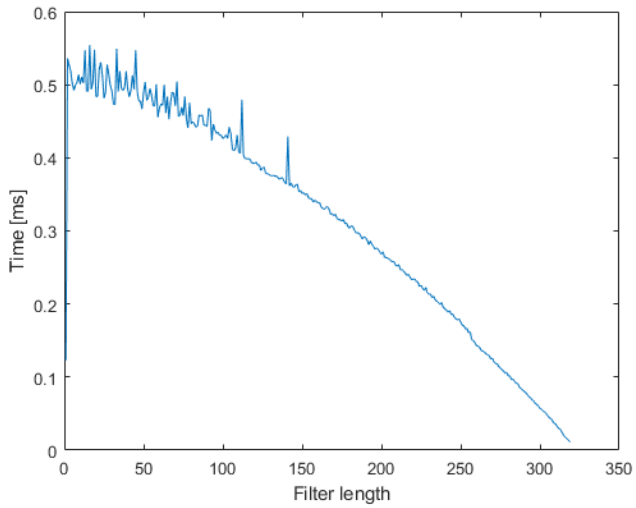

Figure 4.8: A plot of the time it takes in MATLAB to process 320 samples on the y-axis. The x-axis denotes the filter length M used with the Normalized LMS filter to process the samples.

5

Discussion

This chapter focuses on pointing out the findings of this work. First, some observations of the results are being discussed. Furthermore, the performance of the NLMS algorithm is validated against the program of requirements.

5.1. Observations Of The Results

By looking at the NLMS 3D plot of the MSE, various combinations of step sizes and filter lengths are expressed in terms of MSE. It was found that a filter length of M = 33 coefficients and a step size of 0.41 resulted in the smallest MSE. This suggests that with these parameters the error signal most accurately approximates the real noise signal. However, when a listening test was performed, it was quite noticeable that the error for step sizes larger than 0.1 included a heavily distorted version of the speech. Therefore, the 3D plot was truncated to a maximum step size of $\mu = 0.1$. According to the truncated plot, a filter length of 33 coefficients and a step size of 0.1 results in the most accurate approximation of the noise. The same procedure was followed for the traditional LMS and the optimal parameters were found as $\mu = 0.291$ and $M = 93$.

Plots of the squared error of either of the algorithms have shown that the NLMS had a faster convergence speed and better stability. Now, one concludes that the NLMS outperforms the LMS, based on simulations using both the optimal parameters for the NLMS aswell as the traditional LMS.

In Figure 4.5 was seen that the NLMS algorithm had an increased error for non-stationary parts of the noise. The algorithm performed better in the stationary parts of the noise, but still performed adequately in the non-stationary parts. By comparing the estimated noise with the real noise signal, it could be concluded that the estimated noise resembled the real noise.

The performance of the NLMS algorithm was also determined with different noise-to-signal ratio's. The NSR of the input signal affects the NSR of the output signal. In figure Figure 4.7, it could be seen that output NSR's do not increase liniairly with the input NSR's. Furthermore, for input NSR's higher than 6.2 dB, the output NSR's are not higher. A higher threshold than 6.2 dB can be obtained by lowering the step size. Anyway, this means that above a certain value estimation of the noise signal does not suppress the undesired signals, or worse, even adds undesired components. Therefore, it is undesirable to estimate the noise signal in this situation. It makes more sense to send the received microphone signal directly through instead of firstly estimating the noise.

In figure Figure 4.8, it can be seen that the time it takes to process an audio segment of 20ms takes longer when fewer filter coefficients are used. To put it into perspective, even with only one filter coefficient, it only takes less than 1 ms. This is far below the requirement of processing within 20ms. In conclusion, when choos-

ing the filter length, time complexity can be neglected.

Also, an informal listening test was performed for several combinations of parameters. An estimated noise signal was acquired by a NLMS algorithm with parameters $M = 33$ and $\mu = 0.1$. It seemed to resemble the actual input noise. According to the 3d MSE plots, parameters $M = 33$ and $\mu = 0.41$ should result in a better estimation of the noise. However, this was in contradiction with the results from the listening test. Since the MSE is an average of all iterations, the deviations of the individual iterations might still be large. But when averaged, could compensate for each other. Therefore a small MSE might be obtained, while still having large distortions in the estimated signal.

5.2. Validation

The system was tested with signals in the frequency band of 0-8 kHz, which were sampled at 16 kHz. Estimation of the noise was performed, resulting in better output NSR's than input NSR's. This means that item 1a and item 1b of the mandatory functional requirements are satisfied. Furthermore, the algorithm converges fast enough that it can produce a good real time estimate of non-stationary noise. This means that final mandatory function requirement, item 1c is met.

In Figure 4.7, it can be seen that the algorithm can improve the NSR for input signals larger than -15 dB. Therefore, item 2a of the non-functional requirements is met. The non-functional requirement of item 2a is also met, since the time to process is always below 1ms according to Figure 4.8. The algorithm does not require any recordings in the room when no speech is transmitted. The system does not need to record any data, since the incoming data samples are processed in real time. Therefore one could conclude that item 2c and item 2d are met.

The trade-off requirement item 1 is met since adaptive Wiener filters have low complexity [6, 14, 28]. As a consequence, trade-off requirement item 2 is met, which is also proven by Figure 4.8.

6 Conclusion

6.1. Outlook

Supported by the rise of computational power, development of speech intelligibility enhancement systems has seen an increase in popularity the past decades. It is implemented more and more in phones, smart devices, cars and audio systems. The problem posed in this thesis is the extraction of noise parameters in an environment, while a PA system makes announcements. These parameters are the building blocks for two adaptations in an environment: intelligibility enhancement and automatic volume control.

In order to create a simulation environment for testing, a clean speech signal and a non-stationary noise signal was selected. Furthermore a room impulse response was selected, along with locations of the loudspeaker and microphone. Now, methods to retrieve the noise signal from a transmitted noisy speech signal had to be investigated. As was encountered in the literature, this problem could be tackled form various perspectives. One assumption that let to the final selection, was the availability of the clean speech signal. Using this piece of information, could lead to advantages compared to other methods. For example, being less computationally expensive or lacking the need for pre-processing techniques on the microphone signal.

With this in mind, adaptive Wiener filtering was selected. In order to create a real time system, this method had to be solved using an iterative least squares approach. Therefore, the family of LMS algorithms was studied. Herein, the MSE error had to be minimized. To obtain an estimate of the noise signal this way, an ANC diagram was used (Figure 3.1).

Within the family of LMS algorithms, the traditional and normalized LMS algorithms were studied and compared. Therefore, performance metrics had to be determined. Hereafter, the performance regarding minimizing the MSE had to be assessed. Therefore parameters with different convergence and stability properties were considered.

Additionally, Monte Carlo simulations were done to give an inside in performance with non-stationary noise with different changing speeds. It became clear that with similar parameters the NLMS outperformed the LMS in terms of convergence speed while maintaining similar stability properties. Furthermore, performance was looked into during the estimation of noise in a simulated environment. Finally, limitations in terms of NSR and complexity were considered.

In conclusion, adaptive filtering succeeds in suppressing the undesired signal, provided that the interfering signal is available beforehand. Moreover, the NLMS outperforms the LMS in terms of convergence properties.

6.2. Recommendations and Future Work

This thesis can be seen as a starting point for further research into the subject of non-stationary noise estimation. Therefore the following recommendations are posed.

For improvement of efficiency, one would suggest to conduct further research on the subject of speech pause estimation. In subsection 1.4.3 a section was dedicated to explanation of this principle. However, no performance measures were tested on this. Hence only theoretical assumptions were made in this thesis.

In the simulated scenario, only a single non-stationary noise signal was considered to be received by the microphone. However, several noise signals from several locations could be used to simulate several noise sources. Also estimating using multiple microphones might increase the performance.

Additionally, multiple loudspeakers could be considered in a larger room, as that would improve the approximation of a real life situation in a train station. Information on the location could be incorporated to allow for beam forming and improvement on the clean speech reduction.

With both LMS algorithms, problems occurred in very low SNRs. In this case, different algorithms from the LMS family could be considered. For example, the transform domain LMS (TDLMS) algorithms, which are frequency domain realizations of the traditional LMS algorithm. These algorithms should perform more consistently and with improved suppression of the undesired signals [29] [6]. Furthermore, under these conditions, the TDLMS would outperform NLMS in terms of convergence speed.

Finally, a comparison between methods, that do and do not use the distorting signal to estimate the desired signal, could be performed. Hereby, insight might be gained into the benefits (if any) of having the distorting signal available beforehand.

A

MATLAB Code

A.1. LMS algorithms

A.1.1. Traditional LMS

```
function [y, e, W] = tlms(u, d, M, step, returnCoeffs, initCoeffs)
% This function is a wiener filter which minimizes the mean-sqaure error.
%   The filter coefficients are calculated in real time using an iterative
%   approach based on the method of steepest decent.
%
%%%% Input arguments %%%%
% u           : Input of the adaptive filter  (interferer/noise/music)
% d           : Desired signal (microphone signal)
% M           : Filter coefficients
% step        : Step size of the algorithm
% returnCoeffs: Return filter coefficients of every iteration
% initCoeffs: Initial filter coefficients (optional)
%
%%%% Output arguments %%%
% y           : Filtered signal u
% e           : minimized error (=noise)
% W           : Filter coefficients of each iteration

switch nargin
    case 4 % In case no initial filter coefficients are given
        returnCoeffs = 0; % Do not return filter coefficients of all iterations
        initCoeffs = zeros(M,1); % Make array of zeros of length M (initial filter)
    case 5
        initCoeffs = zeros(M,1); % Make array of zeros of length M (initial filter)
end

% Initialize some stuff
N = length(u)-M+1; % Number of iterations
y = zeros(N,1); % Filter output
e = zeros(N,1); % Error signal
w = initCoeffs; % Initial filter coeffs
X = zeros(M,N); % Initialize X as NxM zeros matrix
if returnCoeffs
    W = zeros(N,M); % Matrix to hold coeffs for each iteration
else
    W = NaN; % W is not used
end

for i = 1:N % Fill x with the time frames and shifted one sample each time
    X(:,i) = u(i:M+i-1);
end

Xk = flip(X); % Flip X to make the newest sample the highest
for n = 1:N % Perform the filtering
    x = Xk(:,n); % Select one frame to filter
    y(n) = dot(x,w); % Dot product of x and w
    e(n) = d(n+M-1)-y(n); % Calculate the error
    w = w + step*x'*e(n); % calculate filter
    if returnCoeffs
        W(n,:) = w; % 'save' all filters in W
    end
end

end
```

A.1.2. Normalized LMS

```
function [y, e, W] = nlms(u, d, M, step, eps, returnCoeffs, initCoeffs)
% This function is a wiener filter which minimizes the mean-sqaure error.
%   The filter coefficients are calculated in real time using an iterative
%   approach based on the Gauss-Newton update scheme.
%
%%%% Input arguments %%%%
% u          : Input of the adaptive filter  (interferer/noise/music)
% d          : Desired signal (microphone signal)
% M          : Filter coefficients
% step       : Step size of the algorithm
% eps        : Regularization term (make it small if SNR is small)
% returnCoeffs: Return filter coefficients of every iteration
% initCoeffs: Initial filter coefficients (optional)
%
%%%% Output arguments %%%
% y          : Filtered signal u
% e          : minimized error (=noise)
% W          : Filter coefficients of each iteration

switch nargin
    case 5 % In case only 5 input arguments are given
        returnCoeffs = 0; % Do not return filter coefficients of all iterations
        initCoeffs = zeros(M,1); % Make array of zeros of length M (initial filter)
    case 6 % In case no initial filter coefficients are given
        initCoeffs = zeros(M,1); % Make array of zeros of length M (initial filter)
end

% Initialize some stuff
N = length(u)-M+1; % Number of iterations
y = zeros(N,1); % Filter output
e = zeros(N,1); % Error signal
w = initCoeffs; % Initial filter coeffs
X = zeros(M,N); % Initialize X as NxM zeros matrix
if returnCoeffs
    W = zeros(N,M); % Matrix to hold coeffs for each iteration
else
    W = NaN; % W is not used
end

for i = 1:N % Fill x with the time frames and shifted one sample each time
    X(:,i) = u(i:M+i-1);
end

Xk = flip(X); % Flip X to make the newest sample the highest
for n = 1:N % Perform the filtering
    x = Xk(:,n); % Select one frame to filter
    y(n) = dot(x,w); % Dot product of x and w
    e(n) = d(n+M-1)-y(n); % Calculate the error
    normFactor = 1./(dot(x,x)+eps); % Normalize the error
    w = w + step*normFactor*x'*e(n); % calculate filter
    if returnCoeffs
        W(n,:) = w; % 'save' all filters in W
    end
end

end
```

A.2. Plots

A.2.1. Frequency Plot

```
function freqPlot(y, Fs)
% Function to make time and frequency plot
%   Detailed explanation goes here

% Time Domain signal
tint = (length(y)-1)/Fs;                % length of speech signal in seconds
t = linspace(0, tint, length(y));

% Frequancy domain
N = length(t);
Y = fftshift(fft(y));   % Fourier Transform and shift center
dF = Fs/N;                          % hertz
f = -Fs/2:dF:Fs/2-dF;               % hertz

% Plot
plot(f,abs(Y)/N)
title('Noise PSD')
ylabel('Magnitude')%[arb units]
xlabel('Frequency [Hz]')

```

```
end
```

A.2.2. Time Domain Plot

```
function timeplot(z, d, Fs)
% Function to make time plot
%   Detailed explanation goes here

% Time Domain signal
tint = (length(z)-1)/Fs;                % length of speech signal in seconds
t = linspace(0, tint, length(z));

% Plot
plot(t,d,'color','#999999')
hold on
plot(t,z,'color','#606060')
%title('Time domain spectrum')
ylabel('Amplitude')%[arb units]
xlabel('Time [s]')

end
```

A.2.3. Monte Carlo Plot

```
function [time] = monteCarloPlot(u0,n0,h,eps,samp,M,step,filter,dB)
%This function splits the signal into pieces of 'samp' samples and plots
%the squared error and MSD of the average of the estimated error. It also
%return the average time it takes matlab to filter each segment.

% Cut signal in pieces
N = floor(length(u0)/samp);
s = zeros(N,samp);
n = zeros(N,samp);
z = zeros(N,samp);
d = zeros(N,samp);
for i = 1:N
    s(i,:) = u0((i-1)*samp+1:i*samp);
    n(i,:) = n0((i-1)*samp+1:i*samp);
    temp = conv(s(i,:),h);
    z(i,:) = temp(1:samp);
    d(i,:) = z(i,:) + n(i,:); % Add noise
end

% Filter
L = samp-M+1; % Number of iterations in the filter
e = zeros(N,L);
t = zeros(N,1);
initCoeffs = zeros(1,M); % Initial filter coefficients
if isequal(filter,'nlms')
    for i = 1:N
        tic
        [~, e(i,:), ~] = nlms(s(i,:), d(i,:), M, step, eps, 0, initCoeffs); % Use normalized wiener filter
        t(i) = toc;
    end
elseif isequal(filter,'tlms')
    for i = 1:N
        tic
        [~, e(i,:), ~] = tlms(s(i,:), d(i,:), M, step, 0, initCoeffs); % Use traditional wiener filter
        t(i) = toc;
    end
end
time = mean(t); % Average time to process

% Plot
subplot(2,1,1) % plot the squared error
plot(mean(e.^2)) % MSE
hold on
plot(mean(n(:,M:end).^2)) % Mean squared real noise
xlabel('Iteration')
ylabel('Squared error')
legend({'Obtained error','Noise'},'Location','northeast')


subplot(2,1,2)
MSD = abs(mean(e.^2)-mean(n(M:end).^2)); % Mean squared deviation of noise and error
if isequal(dB,'dB')
    MSDdB = 10*log10(MSD); % Logaritmic scale
    plot(MSDdB);
    ylabel('Deviation [dB]')
else
    plot(MSD);
    ylabel('Deviation')
end
```

```
xlabel('Iteration')

end
```

A.2.4. 3D Plot

```
function [step, M] = threedMSE(s,d,eps,filter)
%This function makes  3D plot of the MSE, filter length and step size
%   A 3D plot is made of the MSE for every combination of filter length
%   and step sizes in the ranges of rangeM and stepRange
rangeM = 60:100; % The range of fiter coefficients to test
stepRange = 0.012:0.003:0.3; % The range of step sizes to test
MSE = zeros(length(rangeM),length(stepRange));
i=0;

for M = rangeM
    initCoeffs = zeros(1,M); % Initial filter coefficients
    i=i+1;
    j=0;
    for step = stepRange
        j=j+1;
        if isequal(filter,'nlms')
            [~, e, ~] = nlms(s, d, M, step, eps, 0, initCoeffs); % Use normalized wiener filter
        elseif isequal(filter,'tlms')
            [~, e, ~] = tlms(s, d, M, step, 0, initCoeffs); % Use normalized wiener filter
        end
        MSE(i,j) = mean(e.^2);
    end
end

surf(stepRange,rangeM,MSE)
xlabel('Step size')
ylabel('Filter length')
zlabel('MSE')

[A, B] = min(MSE);
[~, C] = min(A);
step = stepRange(C);
M = rangeM(B(C));
end
```

A.2.5. Noise PSD Plot

```
function [] = NoisePSDplot(e,n,Fs)
%This function makes PSD plot of real and estimated noise
%
subplot(2,2,1)
freqPlot(n, Fs) % Real noise PSD
title('Real noise PSD')
subplot(2,2,2)
freqPlot(e, Fs) % Estimated noise PSD;
title('Estimated noise PSD')
subplot(2,2,[3,4])
freqPlot(e, Fs) % Real noise PSD
hold on
freqPlot(n, Fs) % Estimated noise PSD;
title('Real and estimated noise PSD')
legend({'Estimated noise PSD','Real noise PSD'},'Location','southeast')
end
```

A.2.6. Plot Of The Squared Error

```
function [] = SEplot(e,n,dB)
%This function makes a plot of the squared error and the squared noise (the
%target of the squared error), which will tell us something about the
%convergence speed. It also plots the MSD which tells us about the
%stability.

switch nargin
    case 2 % In case only 5 input arguments are given
        dB = 0;
end

subplot(2,1,1) % plot the squared error
plot(e.^2)
hold on
plot(n.^2)
xlabel('Iteration')
ylabel('Squared error')
legend({'Obtained error','Noise'},'Location','southeast')

subplot(2,1,2)
```

```
MSD = abs(e.^2-n(1:length(e)).^2).^2; % Mean squared deviation of noise and error
if isequal(dB,'dB')
    MSDdB = 10*log10(MSD); % Logaritmic scale
    plot(MSDdB);
    ylabel('Deviation [dB]')
else
    plot(MSD);
    ylabel('Deviation')
end
xlabel('Iteration')

end
```

A.3. Other

A.3.1. Room Impulse Response

```
function [h]=rir(fs, mic, n, r, rm, src)
%RIR   Room Impulse Response.
%   [h] = RIR(FS, MIC, N, R, RM, SRC) performs a room impulse
%         response calculation by means of the mirror image method.
%
%      FS  = sample rate.
%      MIC = row vector giving the x,y,z coordinates of
%            the microphone.
%      N   = The program will account for (2*N+1)^3 virtual sources
%      R   = reflection coefficient for the walls, in general -1<R<1.
%      RM  = row vector giving the dimensions of the room.
%      SRC = row vector giving the x,y,z coordinates of
%            the sound source.
%
%   EXAMPLE:
%
%      >>fs=44100;
%      >>mic=[19 18 1.6];
%      >>n=12;
%      >>r=0.3;
%      >>rm=[20 19 21];
%      >>src=[5 2 1];
%      >>h=rir(fs, mic, n, r, rm, src);
%
%   NOTES:
%
%   1) All distances are in meters.
%   2) The output is scaled such that the largest value of the
%      absolute value of the output vector is equal to one.
%   3) To implement this filter, you will need to do a fast
%      convolution.  The program FCONV.m will do this. It can be
%      found on the Mathworks File Exchange at
%      www.mathworks.com/matlabcentral/fileexchange/.  It can also
%      be found at http://www.sgm-audio.com/research/rir/fconv.m
%   4) A paper has been written on this model.  It is available at:
%      http://www.sgm-audio.com/research/rir/rir.html
%
%
%Version 3.4.2
%Copyright  2003 Stephen G. McGovern

%Some of the following comments are references to equations the my paper.

nn=-n:1:n;                          % Index for the sequence
rms=nn+0.5-0.5*(-1).^nn;            % Part of equations 2,3,& 4
srcs=(-1).^(nn);                    % part of equations 2,3,& 4
xi=srcs*src(1)+rms*rm(1)-mic(1);    % Equation 2
yj=srcs*src(2)+rms*rm(2)-mic(2);    % Equation 3
zk=srcs*src(3)+rms*rm(3)-mic(3);    % Equation 4

[i,j,k]=meshgrid(xi,yj,zk);         % convert vectors to 3D matrices
d=sqrt(i.^2+j.^2+k.^2);             % Equation 5
time=round(fs*d/343)+1;             % Similar to Equation 6

[e,f,g]=meshgrid(nn, nn, nn);       % convert vectors to 3D matrices
c=r.^(abs(e)+abs(f)+abs(g));        % Equation 9
e=c./d;                             % Equivalent to Equation 10

h=full(sparse(time(:),1,e(:)));     % Equivalent to equation 11
h=h/max(abs(h));                    % Scale output
```

A.3.2. Main

```
%% Clear stuff
close all
```

```
clearvars
clc

%% Set parameters
% %%% Audio files: %%% (All Fs = 16kHz)
% - clean_speech
% - clean_speech_2
% - aritificial_nonstat_noise
% - babble_noise
% - Speech_shaped_noise

SNR = 6; % Choose input SNR in dB
M = 33; % Choose filter length
step = 0.1; % Choose (initial) step size
eps = 0.001; % Regularization term
returnCoeffs = 1; % Return filter coefficients of every iteration (1)(needed for plots)
initCoeffs = zeros(1,M); % Initial filter coefficients
samp = 320; % Length if segments to make Monte Carlo plots
filter = 'nlms'; % Choose which filter to use

%% Make simulated signal
[s,Fs] = audioread('AudioFiles\clean_speech.wav'); %load clean speech
[n0,~] = audioread('AudioFiles\aritificial_nonstat_noise.wav'); %load noise
%n0 = rand(length(n0),1);
h=rir(Fs,[19 18 1.6],12,0.9,[20 19 21],[19 18 1.5]); % Make an impulse response
z = conv(s,h); % Make the convolution of s and h
n0(numel(z)) = 0; % Zero pad signals to make same length
n0 = n0(1:length(z)); % Truncate noise to same length as speech
snrat = dot(z,z)/dot(n0,n0); % Calculate snr (not in dB)
SNR2 = 10^(SNR/20); % Calculate SNR (not in dB)
n = n0*sqrt(snrat)/SNR2; % Change noise power to create desired SNR
d = z + n; % Add noise and signal

%% Run the filter function
if isequal(filter,'nlms')
    [y, e, w] = nlms(s, d, M, step, eps, returnCoeffs, initCoeffs); % Use normalized wiener filter
elseif isequal(filter,'tlms')
    [y, e, w] = tlms(s, d, M, step, returnCoeffs, initCoeffs); % Use traditional wiener filter
end

%% Make figures
figure(1); [best_step, best_M] = threedMSE(s,d,eps,filter); % Find step size based on MSE (and make 3d plot :p)
figure(2); NoisePSDplot(e,n(M:end),Fs); % Make noise PSD plot of e and n
figure(3); SEplot(e,n(M:end),'ndB') % Plot squared error and MSD of the real and estimated error
figure(4); SEplot(e(9000:9300),n(9000+M-1:9300+M-1),'dB') % Zoomed in version of figure 4
figure(5); t_av = monteCarloPlot(s,n,h,eps,samp,M,step,filter,'dB'); % Plot average squared error of multiple sound
    samples
figure(6); timeplot(z,d,Fs); % Plot the recieved signal in time domain

%% Performance metrics
NSR0 = snr(n,z); % NSR before filtering
NSR1 = snr(n(M:length(s)),n(M:length(s))-e); % Noise to signal ratio in Db (can't be done for first M samples)
NSR_gain = NSR1-NSR0; % Achieved increase in NSR
samp_time = round(samp*1000/Fs);

%% Play sound
p = audioplayer(e,Fs,16);
play(p)

%% Save figures
saveas(figure(1),'Figures\3dMSE','png')
saveas(figure(2),'Figures\PSD','png')
saveas(figure(3),'Figures\SE','png')
saveas(figure(4),'Figures\SE2','png')
saveas(figure(5),'Figures\MC','png')
saveas(figure(6),'Figures\timeplot','png')

%% Display stuff
delete Figures\Performance.txt
diary Figures\Performance.txt
disp('%%%%%% Input Parameters %%%%%%')
disp(['Filter used: ', filter])
disp(['Input NSR: ', num2str(NSR0),'dB'])
disp(['Filter length: ', num2str(M)])
disp(['Step size: ', num2str(step)])
disp(['MC plot segment length: ', num2str(samp_time),' ms'])
disp('%%%%%%%%%%% Results %%%%%%%%%%')
disp(['Output NSR: ', num2str(round(NSR1)),'dB'])
disp(['NSR gain: ', num2str(round(NSR_gain)),'dB'])
disp(['Optimal filter legnth: ', num2str(best_M)])
disp(['Optimal step size: ', num2str(best_step)])
disp(['Time to filter ', num2str(samp_time),' ms: ', num2str(round(t_av*1000,2)), ' ms'])
diary off
```

B

Additional figures

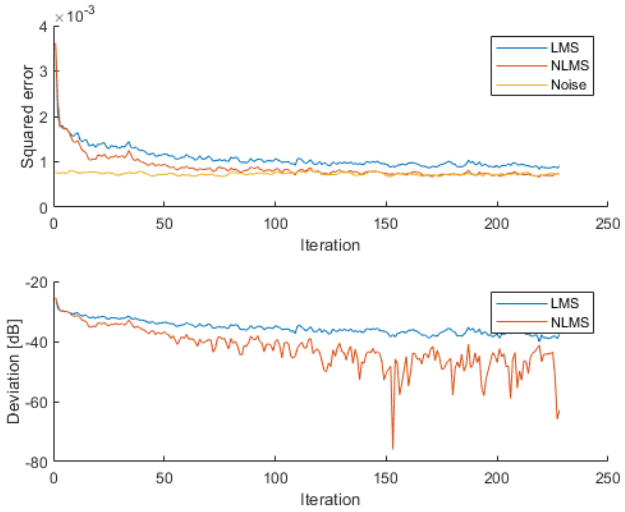

Figure B.1: Monte Carlo simulation of the normalized and traditional LMS filter with parameters $\mu = 0.291$ and $M = 93$.

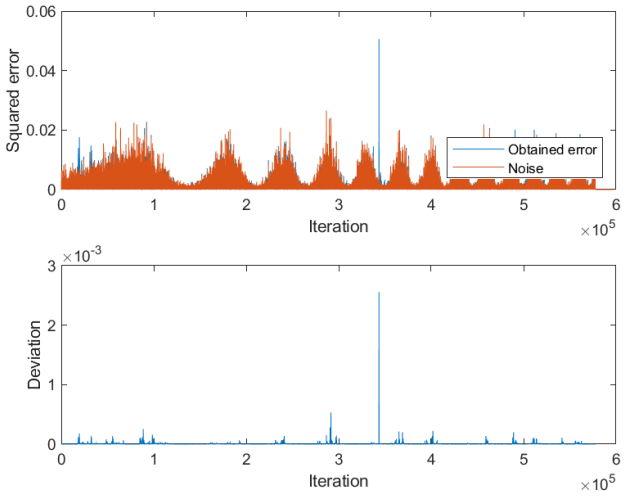

Figure B.2: Squared error of each iteration of the traditional filter with parameters $\mu = 0.291$ and $M = 93$.

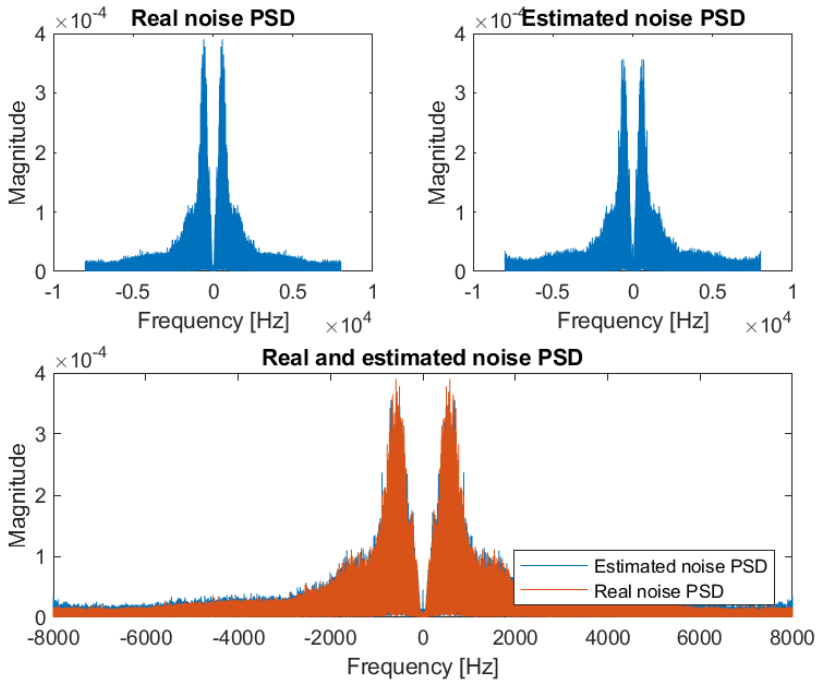

Figure B.3: PSD of the real, and estimated noise with the traditional LMS filter with parameters $\mu = 0.291$ and $M = 93$.

Bibliography

[1] P. Kootwijk, "The speech intelligibility of the public address systems at 14 dutch railway stations," *Journal of Sound and Vibration*, vol. 193, no. 1, pp. 433 – 434, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022460X96902874

[2] K. M. Slavik and A. Slavik. (2018) Speech intelligibility in pa systems. [Online]. Available: https://www.nti-audio.com/Portals/0/data/en/VDT-Speech-Intelligibility-in-PA-systems.pdf

[3] B. Sauert and P. Vary, "Near end listening enhancement: Speech intelligibility improvement in noisy environments," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 1, May 2006, pp. I–I.

[4] E. Riemens and B. Luppes, "On the enhancement of intelligibility," BSc. thesis, TU Delft, The Netherlands, 2019.

[5] T. Timmer and Q. van Wingerden, "Pre-amplifier and noise cancellation for a speech enhancement and noise dampening system," BSc. thesis, TU Delft, The Netherlands, 2019.

[6] K. Sachos, "On speech enhancement in very low snrs for smart speakers," M.S. thesis, TU Delft, The Netherlands, 2018. [Online]. Available: http://resolver.tudelft.nl/uuid:67e1ea69-6d46-4d0c-9840-a27c0b126854

[7] Navneet Upadhyay and Abhijit Karmakar, "Speech enhancement using spectral subtraction-type algorithms: A comparison and simulation study," *Procedia Computer Science*, vol. 54, pp. 574 – 584, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915013903

[8] J. S. Lim and A. V. Oppenheim, "Enhancement and bandwidth compression of noisy speech," *Proceedings of the IEEE*, vol. 67, no. 12, pp. 1586–1604, Dec 1979.

[9] Rombouts, Geert and Spriet, Ann and Moonen, Marc, "Generalized sidelobe canceller based combined acoustic feedback- and noise cancellation," *Signal Processing*, vol. 88, pp. 571–581, 03 2008.

[10] Hioka, Yusuke and Niwa, Kenta, "Psd estimation in beamspace for estimating direct-to-reverberant ratio from a reverberant speech signal," 10 2015.

[11] S. S. V. S. KOTTA and B. K. KOMMINENI, "Acoustic beamforming for hearing aids using multi microphone array by designing graphical user interface," Master's thesis, Blekinge Institute of Technology, School of Engineering, Blekinge, 2011.

[12] L. Tan and J. Jiang, "Chapter 9 - adaptive filters and applications," in *Digital Signal Processing (Third Edition)*, third edition ed., L. Tan and J. Jiang, Eds. Academic Press, 2019, pp. 421 – 474. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128150719000099

[13] S. Haykin, *Adaptive Filter Theory*. University of California at Los Angeles: Pearson Education, 2008.

[14] M. H. Hayes, *Statistical digital signal processing and modeling*. New York: John Wiley and Sons, 1996.

[15] V. Malenovsky, Z. Smekal, and I. Koula, "Optimal step-size lms algorithm using exponentially averaged gradient vector," in *EUROCON 2005*, 12 2005, pp. 1554 – 1557.

[16] Wee-Peng Ang and B. Farhang-Boroujeny, "A new class of gradient adaptive step-size lms algorithms," *IEEE Transactions on Signal Processing*, vol. 49, no. 4, pp. 805–810, April 2001.

[17] C. Schüldt, F. Lindstrom, H. Li, and I. Claesson, "Adaptive filter length selection for acoustic echo cancellation," *Signal Processing*, vol. 89, no. 6, pp. 1185 – 1194, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168409000085

[18] A. Kar and M. Swamy, "Tap-length optimization of adaptive filters used in stereophonic acoustic echo cancellation," *Signal Processing*, vol. 131, pp. 422 – 433, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168416302298

[19] E. Winer, *The Audio Expert: Everything You Need to Know About Audio.* Focal Press, 2013, ISBN 978-0-240-82100-9.

[20] "Occupational noise exposure regulation, url="https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.95", note = Accessed: 2019-06-19,."

[21] (2019) Rotterdam centraal: de feiten op een rij. [Online]. Available: https://www.rotterdam.nl/wonen-leven/rcd/RC_factsheet_NL_Definitief.LR2.pdf

[22] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973. [Online]. Available: http://doi.acm.org/10.1145/321738.321743

[23] B. Widrow, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, J. Eugene Dong, and R. C. Goodlin, "Adaptive noise cancelling: Principles and applications," *Proceedings of the IEEE*, vol. 63, no. 12, pp. 1692–1716, Dec 1975.

[24] L. Litwin, "Fir and iir digital filters," *IEEE Potentials*, vol. 19, no. 4, pp. 28–31, Oct 2000.

[25] A. V. Kisil, "A constructive method for an approximate solution to scalar wiener–hopf equations," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 469, 2013.

[26] dr.ir. Richard Heusdens and dr.ir. Richard C. Hendriks. (2019) Master course *Digital Audio and Speech Processing* on tu delft. [Online]. Available: https://cas.tudelft.nl/Education/courses/in4182/index.php

[27] S. McGovern, "The image-source reverberation model in an n-dimensional space," in *Proc. of the 14th International Conference on Digital Audio Effects*, Paris, France, Sep. 2011, pp. DAFx–11–DAFx–18.

[28] A. H. Sayed, *Fundamentals of adaptive filtering.* New Jersey: John Wiley and Sons, 2003.

[29] E. M. Lobato, O. J. Tobias, and R. Seara, "Stochastic modeling of the transform-domain ϵlms algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1840–1852, May 2008.