Decorrelation in Adaptive Feedback Cancellation for Public Address Systems

Authors
4717708 - Lucas Charles Andreas Huijbregts
4715691 - Merijn Adriaan Jongepier

TU Delft

Decorrelation in Adaptive Feedback Cancellation for Public Address Systems

by

L.C.A. Huijbregts & M.A. Jongepier

to obtain the degree of Bachelor of Science
at the Delft University of Technology,

Student numbers:	4717708 (L.C.A. Huijbregts)	
	4715691 (M.A. Jongepier)	
Project duration:	20 April 2020 – 3 July 2020	
Thesis committee:	Prof.dr. J. Schmitz,	TU Delft, proposer
	Dr.ir. R. C. Hendriks	TU Delft, supervisor
	Dr. J. Martinez Castaneda,	TU Delft, supervisor
	Dr.ir. S. Izadkhast,	TU Delft

Abstract

Public Address systems that include a setup with at least one microphone and speaker can suffer from acoustic feedback. This results in an annoying howling effect which can damage hardware and human hearing. To solve this issue, an adaptive filter that estimates the feedback path and uses this estimate to cancel the feedback can be designed. However, because the adaptive filter receives signals from both the microphone and the feedback, and because sound signals are generally correlated over time, the estimate becomes biased. To reduce this bias, the speaker signal can be decorrelated from the input. In this thesis several options to decorrelate these signals are explored, and they are evaluated based on decorrelation performance and effect on audio quality. Frequency shifting is selected as the best decorrelation method as it provides the most decorrelation while retaining audio quality. Finally it is shown that using Frequency Shifting to decorrelate the microphone and speaker signal indeed improves the estimation of the feedback path.

Preface

This thesis is written in order to obtain the Bachelor's degree in Electrical Engineering at Delft University of Technology. The subject of the thesis was conceived by Dr. John Schmitz. Dr. Schmitz found that many public address systems suffer from the problem of feedback, or howling, despite the existence of many theoretical solutions. The aim of this project is to find a practical solution to this problem. This thesis describes the design process and decisions made to create one of the three subsystems to do so. The other two subsystems are created by our colleague students Johannes Willem de Vries and Cyril Arnold Weustink, and by Cees Kos and Mathijs Bekkering.

This project could not have been completed successfully without the help of the following people. First of all, we would like to thank Dr. John Schmitz for providing us with the subject of the project. Secondly, we would like to thank Dr. Ir. Richard Hendriks and Dr. Jorge Martinez for their guidance and assistance for the duration of the project. We would also like to thank Metin Çaliş for his hands-on support and advice.

Finally we would like to thank our group members, Johannes Willem de Vries, Cyril Arnold Weustink, Cees Kos and Mathijs Bekkering for helping us through advice and motivation.

Lastly, it is important to address the Covid-19 pandemic that made the Bachelor Graduation Project quite different this year. Sadly, because of the lockdown restrictions imposed by the Dutch government and the TU Delft we are not allowed to create a physical prototype of our system during the project. This means that the end product of this project is a theoretical system implementation, only tested in digital simulation environments of MATLAB and Simulink.

Lucas Huijbregts and Merijn Jongepier
Delft, July 22, 2020

Contents

1

Introduction

1.1. Context

Public address (PA) systems are widely in use in places where an audio signal needs to be amplified to be audible to an audience. This audio signal can consist of both speech and music. Examples of these places are concert halls, open-air festivals, and theatres. When such a PA system consists of at least one microphone and one speaker, a common problem that occurs is feedback. This means that the sound created by the speaker reenters the system through the microphone. As the sound is generally amplified by the PA system, this can create an infinite loop of amplification, which manifests itself in a loud, howling tone. This howling is perceived as very annoying and can even cause damage to the human ear and audio equipment [1]. To stop this the audio equipment has to be manually adjusted or even temporarily turned off. This phenomenon also limits the maximum stable amplification of the system. Figure 1.1 shows this effect in a simulated environment of a $4mx4mx4m$ room and an audio fragment from Händel's Halleluja. The microphone and speaker are located $1.5m$ apart in the middle of the room.

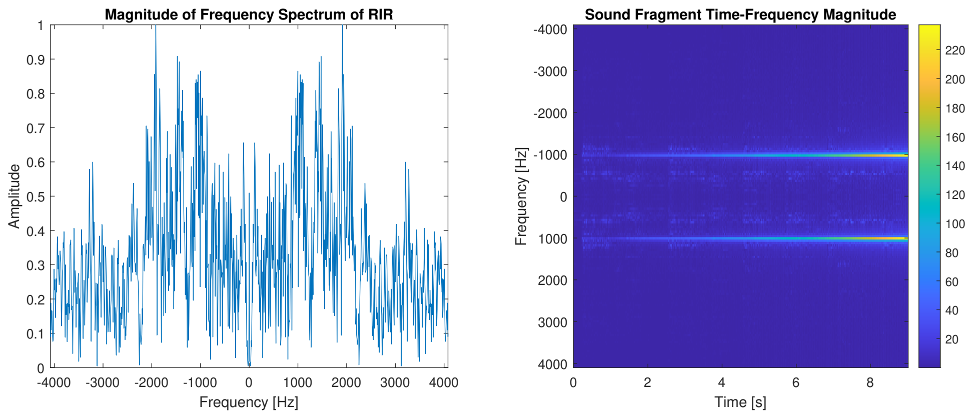

Figure 1.1: Example of feedback in an audio system. The left Figure shows the amplitude vs frequency of the RIR. It shows peaks around $1000Hz$, $1500Hz$ and $2000Hz$. These peaks can be considered resonance frequencies of the system, but are dependent on the room, objects in the room and the positions of the speaker and microphone. Right shows the audio spectrum over time of the fragment, a 9 second piece of Händel's Halleluja. It clearly shows that at around $1000Hz$, the signal becomes stronger over time. This is the howling effect. Note that because the audio fragment does not contain much energy at other frequencies, the howling effect does not occur there.

Figure 1.2 shows the phenomenon described above for a system with one microphone and one speaker [2], [3]. Signal $s(n)$ is the audio input to the microphone. Block G models the signal processing between the microphone and the speaker. The audio signal sent out by the speaker travels through the space in which the system is set up, back into the microphone. This space is referred to as the "room", and the path between the speaker and microphone is modelled by F, the Room Impulse Response (RIR).

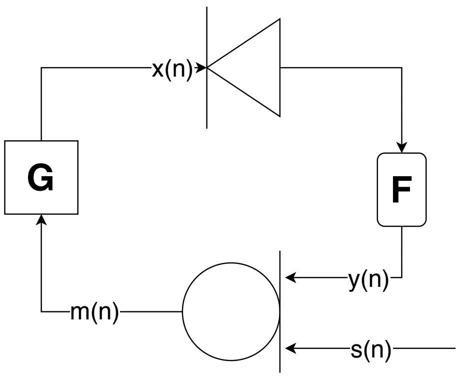

Figure 1.2: Basic PA system feedback loop.

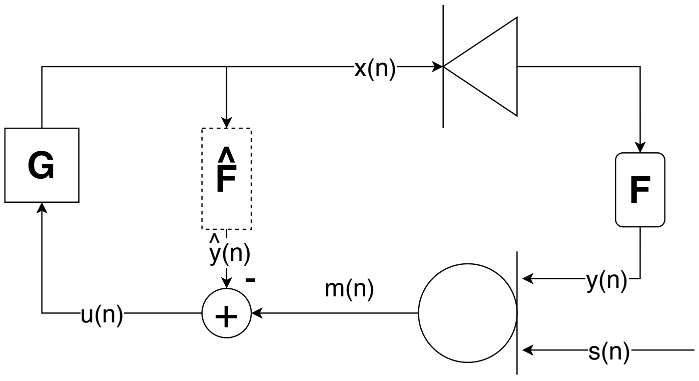

Figure 1.3: Basic Adaptive Feedback Cancellation implementation.

The microphone, G, the speaker, and F form a closed loop. In the frequency domain, the speaker input signal as a function of the audio input is given by

$$X(\omega,n) = G(\omega,n)S(\omega,n) + G(\omega,n)F(\omega,n)X(\omega,n) \tag{1.1}$$

$$\frac{X(\omega,n)}{S(\omega,n)} = \frac{G(\omega,n)}{1 - G(\omega,n)F(\omega,n)}. \tag{1.2}$$

Here, $S(\omega,n)$ and $X(\omega,n)$ are the Fourier Transforms of $s(n)$ and $x(n)$ respectively. For any system to remain stable, the Nyquist stability criterion [4] must hold. In this case, the criterion states that if there is a radial frequency $\omega = 2\pi\frac{f}{F_s}$ for which

$$\begin{cases} |G(\omega,n)F(\omega,n)| \geq 1 \\ \angle G(\omega,n)F(\omega,n) = n2\pi, \quad n \in \mathbb{Z} \end{cases} \tag{1.3}$$

then the system is unstable. In essence it states that if there is a frequency at which there is a larger than unity loop gain and G and F are in phase, then the signal reinforces itself at that frequency, causing a peak in the spectrum. In the case of PA systems this can be heard as howling.

As stated before, this effect limits the maximum stable amplification of the system. This is generally quantified by the Maximum Stable Gain (MSG). To define the MSG [3], we split $G(\omega,n)$ into its broadband component and frequency dependent component:

$$G(\omega,n) = K(n)J(\omega,n). \tag{1.4}$$

$K(n)$ represents the average gain, while $J(\omega,n)$ represents the gain that is applied differently at different frequencies. The broadband gain is then given by

$$K(n) = \frac{1}{2\pi}\int_0^{2\pi} |G(\omega,n)|d\omega \tag{1.5}$$

The MSG is then defined as

$$MSG(n)[dB] = 20log_{10}K(n). \tag{1.6}$$

such that the Nyquist criteria, see equation 1.3, only just holds, meaning a small change in phase or small increase in gain would lead to instability.

1.2. Introduction to State of the Art for Feedback Control

There are four main methods to reduce the feedback in such systems [3]. These are Phase-modulating Feedback Control, Gain Reduction Feedback Control, Spatial Filtering, and Adaptive Feedback Cancellation. In the following subsections these methods are briefly discussed.

1.2.1. Phase-modulating Feedback Control

The aim of Phase-modulating Feedback Control (PFC) is to change the phase of the signal in every loop through the system. In that way, the second part of the Nyquist criterion cannot hold in two successive iterations. PFC is achieved by placing a Phase-Modulating filter in the signal path. There are three main ways of achieving this:

- Sinusoidal Phase Modulation;
- Sinusoidal Frequency Modulation;
- Frequency Shifting.

PFC is a relatively simple and computationally light solution to the problem, but it has some significant downsides. First, the result of PFC is smoothing of the loop gain of the system. Thus, peaks that would result in feedback with a gain that is too high will be reduced, and the MSG will be determined by the average loopgain. This means there is a theoretical achievable upper bound to the MSG increase, which is around 10 dB on average for large rooms [5]. Furthermore, PFC leads to signal distortion, so to retain the signal quality one needs to use light modulation, limiting the MSG increase even more [6]. Finally, PFC is not well suited for musical signals as tones can persist for multiple cycle times in the system. Thus the perceived tone can change with every cycle when it should remain the same.

1.2.2. Gain Reduction Feedback Control

Gain reduction methods can historically be divided into three types [3]:

- Automatic Gain Control (AGC) - reduction of broadband gain $K(t)$;
- Automatic Equalization (AEQ) - reduction of gain in the necessary predetermined frequency band;
- Notch-filter-based Howling Suppression (NHS) - reduction of gain at specific howling frequency.

These three methods are given in order of historical significance and filter complexity. AGC is the simplest method, but as the entire spectrum is temporarily silenced, the MSG is not increased at all.

AEQ is already more sophisticated, as it only silences a subband of the spectrum. Therefore there is an increase in MSG, but the audio quality decreases because a part of the frequency spectrum is suddenly silenced when the system springs into action.

NHS is the most precise method. The idea is to compute the frequency spectrum using the Fast Fourier Transform (FFT), and subsequently find howling peaks using a peak-finding algorithm. If a peak is found, a notch filter is applied over that frequency, cancelling the howling. This can be done both proactively, by determining the loopgain and then inserting filters at the offending frequencies, or reactively, by detecting howling when it occurs and then inserting the filters.

There a two main drawbacks to gain reduction methods. First, in most practical applications, one cannot determine the possible howling frequencies in advance because of possible room changes, so the filter has to wait for howling to occur before it can spring into action. Due to this reactive nature, there is still a chance that audible howling will occur before it can be suppressed. Second, as gain reduction methods, similar to PFC, aim at smoothing the loop gain, there is again a theoretical limit of 10 dB to the achievable MSG increase.

1.2.3. Spatial Filtering Methods

In spatial filtering, arrays of microphones and speakers are set up in a precise manner in order to create maxima and minima of sound at the exact right places in the room by controlling the places where constructive and destructive interference occur. The aim is to have minima of the loudspeaker sound at the microphones to minimise the feedback, and to have the maxima in the audience. Obviously this method is very restrictive, as the microphones and speakers have to be set up in a very specific way and cannot be moved during use. This makes the method unsuitable for most PA applications.

1.2.4. Adaptive Feedback Cancellation

In Adaptive Feedback Cancellation (AFC) the principle is to use the microphone signal to estimate the RIR, or F, as $\hat{F}$. This is used to estimate the feedback component entering the microphone, $y(n)$. This estimate is then subtracted from the microphone signal $m(n)$, theoretically completely cancelling the feedback. This means that there is no theoretical limit to the possible MSG increase. See Figure 1.3 for the basic implementation.

1.3. Choice of Feedback Control Method

In the previous section, four methods for reducing feedback were introduced. Spatial filtering methods are too limited in their applicability, and PFC and gain reduction feedback control have a theoretical limit in the possible MSG increase of $10dB$ [3], meaning they cannot fulfill requirement 2.1.6 as found in Chapter 2. AFC, however, does have the potential for an MSG increase exceeding $10dB$. Therefore, Adaptive Feedback Cancellation was chosen to be implemented in this project.

1.4. Subsystem Division

Adaptive Feedback Cancellation, as stated before, uses the microphone signal $m(n)$ to estimate the RIR, or F, as $\hat{F}$. Using this estimate and the speaker input $x(n)$, an estimate of the feedback signal $y(n)$ is created: $\hat{y}(n)$. This estimate is subtracted from the microphone signal, which results in the estimate of the microphone input: $\hat{s}(n)$. This is the signal that is sent to G and then to the speaker.

This is AFC in its simplest form. However, in reality some additional components are needed for proper functionality [3]. First of all, there exists a nonzero correlation between the sound signals $s(n)$ and $x(n)$. Because of this and the closed loop nature of the system, the adaptive filter $\hat{F}$ estimate becomes biased and does not only cancel the feedback $y(n)$, but also part of the source $s(n)$. In order to reduce this effect, a decorrelating device is necessary somewhere in the signal path that minimises the correlation between $s(n)$ and $x(n)$. Second, a postfilter is necessary to remove any residual feedback and act as a backup for when the AFC fails.

The implementation of the AFC system has thus been split into the Adaptive Filter $\hat{F}$, the Decorrelation D and the Postfilter P, also visible in Figure 1.4. This report is dedicated to the Decorrelation Filter subsystem. Its necessity and functionality will be explained in further detail in Chapter 3. Information on the other two subsystems can be found in their respective dedicated thesis reports; see [7] for the thesis on the Adaptive Filter and see [8] for the thesis on the Postfilter.

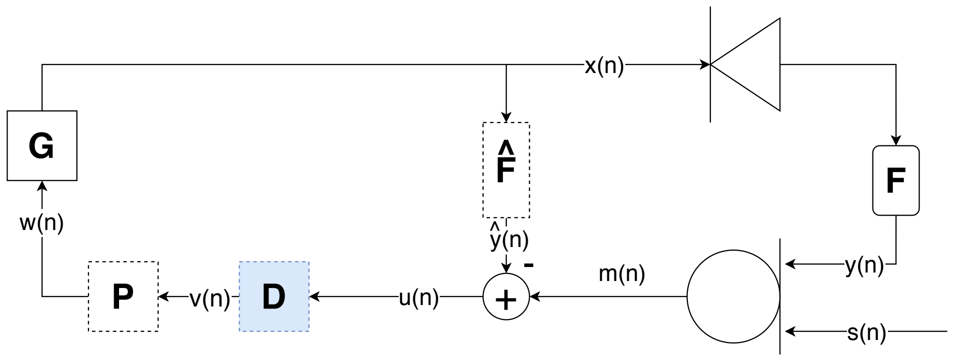

Figure 1.4: System Overview including all three subsystems. The subsystem to which this report is dedicated, the Decorrelation Filter D, is highlighted.

1.5. Thesis Overview

This thesis will first give the programme of requirements of the complete system and of the Decorrelation filter in Chapter 2. Then the problem of Decorrelation will be analysed in more detail in Chapter 3, and different solutions to this problem will be presented in Chapter 4. Each of these methods will be implemented as MATLAB functions in Chapter 5, and their performance will be evaluated based on the programme of requirements in Chapter 6. From this evaluation one method will be selected as the best decorrelation method based on our requirements. Next, in Chapter 7, this method is implemented and tested in a simulation of the full system to evaulate its performance and the results are discussed. Finally, in Chapter 8, the thesis is summarised, the results are analysed based on the Program of Requirements, and recommendations for future work are made.

2 Programme of Requirements

This chapter lists the requirements that need to be met by the system. The requirements are listed for the system as a whole, as seen in Figure 1.4 and for the Decorrelation subsystem specifically.

Two different types of requirements will be listed: Mandatory Requirements and Trade-off Requirements. Mandatory Requirements must be met, meaning they do not allow for any compromise. These requirements are indicated by the word 'must'. Trade-off Requirements are more flexible. These requirements should be met as well as possible, taking other requirements into account. These requirements are indicated by the word 'should'.

2.1. Full System Requirements

Functional Requirements

1. The system must be realisable as a module that can be inserted into the electronic part of an existing PA system.
2. The system must be able to operate on the entire frequency spectrum of human hearing. This means it should cover the frequencies between $20Hz$ and $20kHz$. Therefore, the sampling frequency of the system must be at least $40kHz$ according to the Nyquist-Shannon sampling theorem [9].
3. The system must be able to operate without human interaction. System setup is allowed to require human interaction.
4. The system must operate in real time. This means that the signal output should be produced at the same rate as the input.
5. The system should increase the MSG as much as possible, with a minimum of $10dB$ as this is the maximum MSG increase possible using PFC and NHS methods.
6. The system should output an audio signal with the highest possible quality. The quality is measured using the PEAQ algorithm, as explained further in Chapter 3. The minimum requirement limit for the ODG is -0.5, which means distortion is as good as imperceptible.
7. The system should introduce no more than $50ms$ delay [10].

Non-functional Requirements

8. The system must be suitable for indoor and outdoor use.
9. The system must be able to operate on the European mains with a nominal voltage of $230V$ and a frequency of $50Hz$.
10. The system should cost €100 or less to produce.

2.2. Subsystem Requirements

1. The subsystem should provide as much decorrelation of the input signal as possible.
2. The subsystem must keep the audio quality of the input signal as high as possible. The quality is measured using the PEAQ algorithm, as explained further in Chapter 3. The minimum requirement limit for the ODG is -0.5, which means distortion is as good as imperceptible.

3 Theoretical Problem Analysis

3.1. Mathematical Problem Description

The RIR can be approximated by a finite impulse response $\boldsymbol{f} = \begin{bmatrix} f_0 & \dots & f_{nF} \end{bmatrix}^T$ of order n_F [11]. From Figure 1.2 we can see that the feedback signal $y(n)$ at any sample time n is

$$y(n) = \begin{bmatrix} x(n) & x(n-1) & \dots & x(n-n_F) \end{bmatrix} \cdot \begin{bmatrix} f_0(n) \\ f_1(n) \\ \cdot \\ \cdot \\ \cdot \\ f_{n_F}(n) \end{bmatrix} \tag{3.1}$$

where $x(n)$ is the loudspeaker signal. This can be cast into a matrix formulation by considering multiple instances of n, resulting in the following overdetermined system of equations:

$$\begin{bmatrix} y(n) \\ y(n-1) \\ \cdot \\ \cdot \\ \cdot \\ y(1) \end{bmatrix} = \begin{bmatrix} x(n) & x(n-1) & . \ . \ . & x(n-n_F) \\ x(n-1) & x(n-2) & . \ . \ . & x(n-n_F-1) \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & \cdot \\ x(1) & x(0) & . \ . \ . & x(-n_F+1) \end{bmatrix} \cdot \begin{bmatrix} f_0(n) \\ f_1(n) \\ \cdot \\ \cdot \\ \cdot \\ f_{n_F}(n) \end{bmatrix}$$

$$\boldsymbol{y} = \boldsymbol{X} \cdot \boldsymbol{f}. \tag{3.2}$$

From this, $\hat{\boldsymbol{f}}$ may be estimated using a least squares estimate [12] as

$$\hat{\boldsymbol{f}}(n) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3.3}$$

using the pseudoinverse of $\boldsymbol{X}$ [13].

However, the feedback signal $y(n)$ can not be measured on its own. The microphone signal $m(n)$ also contains the input signal $s(n)$. Therefore the estimate becomes

$$\hat{\boldsymbol{f}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{m} \tag{3.4}$$

$$= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{y}+\boldsymbol{s}) \tag{3.5}$$

$$= \boldsymbol{f} + (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{s}. \tag{3.6}$$

We can see that if the second term is not equal to zero, the estimate is biased. Generally, the loudspeaker signal $x(n)$ and input signal $s(n)$ will be correlated, causing the bias to be nonzero [3, 11, 14]. Because of the bias, not only the feedback will be cancelled, but also part of the input signal, which will come at the cost of signal quality.

That there will be a bias when there is correlation between the signals can be seen when taking the expectation of the bias term:

$$E[bias] = E[(\boldsymbol{Z}^T\boldsymbol{s})] \tag{3.7}$$

where the expectation is taken over the rows of the matrices, and $\boldsymbol{Z}^T$ is the pseudoinverse of $\boldsymbol{X}$. Therefore, if $E[\boldsymbol{Z}^T\boldsymbol{s}] = \boldsymbol{0}$, the estimate is unbiased. However, because audio signals usually have a nonzero correlation over time, this is generally not the case. Taking a look at the covariance between $\boldsymbol{Z}^T$ and $\boldsymbol{s}$ shows that $E[bias]$ equals:

$$E[bias] = cov(\boldsymbol{Z}^T, \boldsymbol{s}) + E[\boldsymbol{Z}^T]E[\boldsymbol{s}] \tag{3.8}$$

where the covariance is $cov(\boldsymbol{Z}^T, \boldsymbol{s}) = E[\boldsymbol{Z}^T\boldsymbol{s}] - E[\boldsymbol{Z}^T]E[\boldsymbol{s}]$. Since is $\boldsymbol{s}$ an audio signal, it is zero mean and the expectation is zero. So the expectation of the bias is:

$$E[bias] = cov(\boldsymbol{Z}^T, \boldsymbol{s}). \tag{3.9}$$

Therefore, if the correlation between the loudspeaker and microphone signal is zero, the bias will be zero.

3.2. Misalignment of the Estimation

The bias of the RIR estimation $\hat{\boldsymbol{f}}$ can be quantified using the Misalignment (mis). Mathematically, this is the normalized norm of the of the difference of $\boldsymbol{f}$ and $\hat{\boldsymbol{f}}$ in decibel [15], so

$$mis = 10\log_{10}\left(\frac{||\boldsymbol{f} - \hat{\boldsymbol{f}}||^2}{||\boldsymbol{f}||^2}\right). \tag{3.10}$$

As the aim of the decorrelation is to reduce the bias, this is the eventual measure that will be used to measure the performance. However, since we will not have a working AFC system at our disposal for the greater part of the project, the system will initially be designed based on its decorrelation performance.

3.3. Decorrelation

The goal for this subsystem is to reduce the correlation of the loudspeaker $x(n)$ and input signal $s(n)$. In order to measure the performance of the subsystem, a measure of the correlation between the signals is needed. To get a single scalar as this measure, the covariance is used. This is a measure of the joint variability of two signals; if generally an increase in signal X corresponds to an increase in signal Y, the covariance is high. The formula for the covariance is

$$cov(X, Y) = E[(X - E[X])(Y - E[Y])]. \tag{3.11}$$

To compare the covariance results for different methods, the Pearson Correlation Coefficient [16] is more useful, as it is normalised with respect to the two datasets. The Pearson Correlation Coefficient is given by

$$\rho(X, Y) = \frac{cov(X, Y)}{\sqrt{Var(X)Var(Y)}}. \tag{3.12}$$

The Pearson Correlation Coefficient varies from -1 to 1. A $|\rho|$ closer to 0 means lower correlation, while an $|\rho|$ closer to 1 means high correlation. Note that in the mathematical sense, decorrelated means that the signals are completely independent, so $\rho = 0$. However, throughout this thesis we will use it to mean partially decorrelated, so $|\rho| < 1$, but not necessarily zero.

3.4. Audio Quality

Some decorrelation methods can reduce audio quality. In order to compare the influence of different decorrelating procedures on the perceived audio quality, some numerical measure is needed. In this project, the PEAQ algorithm will be used [17]. PEAQ stands for Perceptual Evaluation of Audio Quality, and it is an algorithm developed by the International Telecommunication Union's Radiocommunication Sector (ITU-R). It includes measures of nonlinear distortions, linear distortions, harmonic structure, distance to masked threshold and changes in modulation. It uses a neural network in combination with a detailed model of the human ear to generate an audio quality score.

This audio score is called the Objective Difference Grade (ODG). It corresponds directly to the ITU-R *Subjective* Difference Grade, which is a score between 1 and 5 used in audio tests with human subjects. The ODG is a score between approximately -4 and 0. Note that in reality the ODG varies from -3.98 to 0.22 [18]. A score higher than 0 means that the tested audio segment has higher perceptual quality than the segment used as reference [19]. Thus, audio with higher ODG can be interpreted as being of better quality. Table 3.1 shows the interpretation of the different scores.

Table 3.1: Table comparing ITU-R subjective Grades and their descriptions to the ODG system [17].

Description	**ITU-R Subjective Grade**	**ODG**
Imperceptible	5.0	0.0
Perceptible but not annoying	4.0	-1.0
Slightly annoying	3.0	-2.0
Annoying	2.0	-3.0
Very annoying	1.0	-4.0

4 State of the Art Analysis

Several different approaches to decorrelating signals have been proposed. These fall within the following categories [3]:

- Noise injection;
- Time-varying processing;
- Nonlinear processing;
- Forward path delay;
- Adaptive filter delay;
- Decorrelating prefilters.

In the following sections these approaches will be explained.

4.1. Noise Injection

Noise Injection (NI) is in principle the most straightforward decorrelation method. The idea is to add a noise signal somewhere along the signal path. The correlation between a random noise signal and another signal is very low. Thus, by adding noise to the speaker signal, its correlation with the input signal decreases. The simplest form of noise injection uses white noise. This means a noise signal with a constant power over the frequency spectrum is added. Of course, if the noise power is too high it will be audible to the listener. It is therefore key to find a balance between the added noise power and the signal distortion.

A way to ease this balance is to divide the signal into different frames, and determine the signal power in each frame. Based on this, different levels of noise are added to different parts of the signal. This is called Framed Noise Injection (FNI).

Another way to further decrease the impact on the audio quality is through Shaped Noise Injection (SNI). The human ear does not respond in the same way to different frequencies. We can make use of this through psychoacoustic noise shaping, where the noise power is increased in frequency bands where the noise is heard the least [20]. These bands are quantified in the Bark scale [21], which divides the human hearing range into frequency bands with similar perceptual significance. This means that the frequencies in such a band are perceived to have the same power, while the frequencies in different bands are perceived to have different power. The signal power in each of the bands can be measured, and the noise power shaped accordingly. So when the signal has most of its power in, say, the third band, that is where the most noise will be added. In this way, the noise will be 'hidden' by the signal, and much more noise can be added before it becomes audible.

4.2. Time-Varying Processing

Phase-modulating the speaker signal will have the effect that the input and output signals have a different phase. If signals are out of phase, their correlation is reduced. However, the signal itself is kept largely intact so the quality does not decrease significantly. There are three ways of phase-modulating a sound signal while retaining quality [3]:

Sinusoidal Phase Modulation

With Sinusoidal Phase Modulation (sinPM), the phase of the signal is modulated sinusoidally. A sinPM filter has frequency response:

$$H_{PM}(\omega, n) = \exp\left[j\beta \sin\left(\omega_m n\right)\right] \tag{4.1}$$

Both the modulation frequency ω_m and the modulation index β can be varied.

Sinusoidal Frequency Modulation

In sinusoidal Frequency Modulation (sinFM) the frequency is modulated sinusoidally. It is characterised by the modulation frequency ω_m or $f_m = \omega_m \frac{F_s}{2\pi}$ and modulation depth Δ_f:

$$H_{FM}(\omega, n) = \exp\left[j\frac{\Delta_f}{f_m} \sin\left(\omega_m n\right)\right]. \tag{4.2}$$

SinFM is in principle the same as sinPM where $\beta = \frac{\Delta_f}{f_m}$. Because of this, it is deemed unnecessary to implement.

Frequency Shifting

With Frequency Shifting (FS) all frequency components in the signal are shifted equally over the spectrum:

$$H_{FS}(\omega, n) = \exp\left[j\omega_m n\right]. \tag{4.3}$$

4.3. Nonlinear Processing

Another category of operations that can be done on the speaker signal to reduce the correlation is a nonlinear process. By adding a part that is not linearly dependent on the signal, the correlation is reduced. This has the advantage that, instead of adding a completely random signal like in noise injection, it adds a signal that is similar to the signal already there, reducing the impact on the audio quality [22]. The output of such a process will be

$$v(n) = u(n) + \alpha f(u(n)) \tag{4.4}$$

with f a nonlinear function, and α a tunable parameter. A simple process is Half-Wave Rectification (HW):

$$f_{HW}(u) = \frac{u + |u|}{2}. \tag{4.5}$$

This function will be zero for $u < 0$, and equal to u for $u \geq 0$. Other options found in literature are the Square-Law (SL):

$$f_{SL}(u) = \alpha u^2 \tag{4.6}$$

and the Smoothed Half-Wave Rectification (HWS):

$$f_{HWS}(u) = \frac{u}{2} + \sqrt{(u/2)^2 + c^2} \tag{4.7}$$

where c is a parameter used to smooth the discontinuous derivative at $u = 0$ [23].

4.4. Forward Path Delay

With Forward Path Delay (FPD), by adding a delay of Δ samples in the signal forward path, we make sure that the speaker signal and input signal do not line up in time, reducing the correlation between them. This technique is simple to implement and does not decrease the audio quality. The biggest drawback is that if the delay is too big, there will be an audible mismatch between when the sound enters the microphone and when it is reproduced by the speaker.

4.5. Adaptive Filter Delay

Adaptive Filter Delay (AFD) is also based on a delay in the signal path, but because it does not influence the signal itself, there should be no distortion. For AFD it is used that sound propagates with finite velocity, so there is a delay between when the sound leaves the speaker and when it enters the microphone as feedback. If this feedback delay T_f is known, we can set the first

$$d_f = T_f \cdot F_s \tag{4.8}$$

samples of the estimated impulse response $\hat{\mathrm{f}}$ to 0. These then do not have to be calculated. At the same time, a delay is added in a similar fashion to d_s in the previous section.

A large drawback of AFD is that T_f, the time it takes for the sound to travel from the speaker to the microphone, needs to be known. This is possible if the entire setup is static, but if the speaker or microphone changes position during use, this system cannot be used. As our AFC system should be able to handle all changes in the RIR (see Chapter 2), AFD will not be considered in this report.

4.6. Decorrelating Prefilters

Similar to the Adaptive filter delay method, when using a Decorrelating Prefilter (DP) there is also no distortion of the signal. This is done by placing the decorrelation device(s) only around the RIR estimate filter $\hat{F}$ and not in the signal path.

To do this, the source signal needs to be modelled as:

$$s(n) = H(\omega, n) * w(n) \tag{4.9}$$

where $H(\omega, n)$ is a time-varying model of the source signal and $w(n)$ is an excitation signal. For example, $H(\omega, n)$ models the way a trumpet reacts to being played, where the playing is modelled by $w(n)$. We then need to assume that an estimate of this model is available to us: $\hat{H}(\omega, n)$. The microphone signal $m(n)$ and speaker signal $x(n)$ are then first passed through the inverse of this estimate ($\hat{H}^{-1}(\omega, n)$) before being used to create the adaptive filter estimate $\hat{F}$. This model is then copied to the position where it is used. In this way, only the signal used to estimate the RIR is changed, not the signal in the forward path. See Figure 4.1 for a visual explanation.

The drawback of DP is that an estimate $\hat{H}(\omega, n)$ of the source signal model is needed. Modelling of this source has been extensively researched in speech applications, but not so much for audio. As our system should be able to handle all sounds that humans can hear, i.e. all sounds with frequencies between $20Hz$ and $20kHz$ (see the requirements in Chapter 2), DP will not be considered.

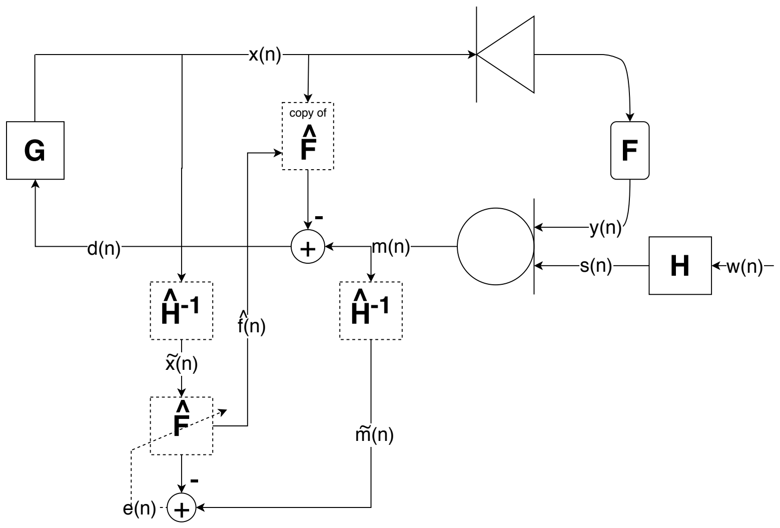

Figure 4.1: The overview of a system with a Decorrelating Prefilter that uses an inverse source model.

5
Implementation

In this chapter the implementation of each of the viable methods introduced in Chapter 4 is explained and justified. These methods are:

- Basic noise injection (NI)
- Framed noise injection (FNI)
- Shaped noise injection (SNI)
- Sinusoidal phase modulation (sinPM)
- Frequency shifting (FS)
- Half-wave rectification (HW)
- Smoothed half-wave rectification (HWS)
- Square-law (SL)
- Forward path delay (FPD)

The sections will cover the choice of the parameters of each of the methods to optimise their performance. Furthermore, the possible use of frames and with those frames the use of windowing is discussed.

Method of testing

In order to test the methods, a selection of audio clips is used. To ensure a good representation of the audio that will be used in the system, nine different audio clips of different speech and music styles are used. Three clips contain three different languages and six clips contain various music styles, each 10 seconds long. The details of the clips can be found in Appendix D.1.

All figures and data used in this chapter are based on these 9 audio clips.

Usage of Frames

Because the system needs to operate in real-time, the operations cannot be done on the entire signal all at once. Instead, they should be done on the individual samples, or when multiple samples are needed for a Fourier Transform for example, on frames. These frames then contain multiple samples, the amount of which will be determined based on the maximum delay requirement of the global system: $50ms$ as specified in Chapter 2. However, using frames introduces spectral leakage [24]. Due to the abrupt cutoff at the edges of the rectangular frames, or windows, this results in artefacts in the audio that can dramatically reduce the quality. Figure 5.1 shows an example of this.

The problem can be solved using a non-rectangular window that tapers off at the edges, reducing the abrupt cutoff and redistributing the spectral leakage. There are many possible windows. However, because we want to reconstruct the signal after the processing, the windows should be able to overlap and sum to one everywhere, so no information is lost. Therefore, a Hann window will be used [25]:

$$w(n) = \begin{cases} \sin^2(\frac{\pi n}{N}) & 0 \leq n \leq N \\ 0 & elsewhere \end{cases} \tag{5.1}$$

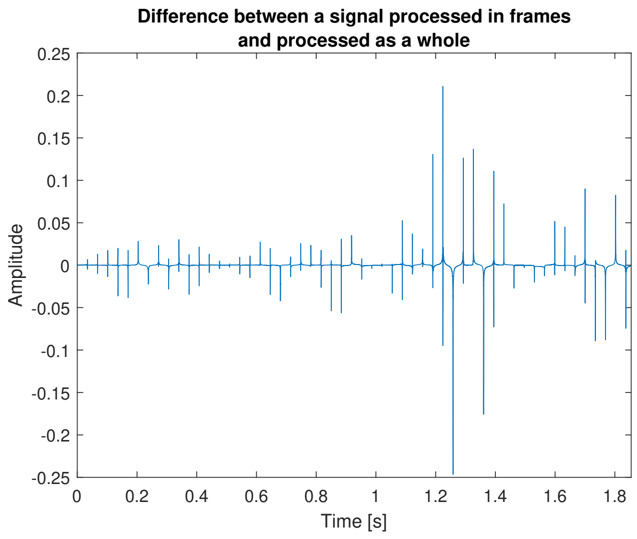

Figure 5.1: An example of artefacts introduced because of rectangular windowing. The figure shows the difference between the signal that is processed as the complete clip of ten seconds, and the signal that is processed using rectangular frames of 1500 samples. This results in spikes at every 1500 samples that dramatically decrease the audio quality. The picture does not contain the complete ten seconds in order for the spikes to be clearly visible.

The simplest way to use Hann windows is by using an overlap of 50%; the windows then add to exactly 1 over the entire signal. This means no information is lost, and the spikes seen in Figure 5.1 completely disappear.

The delay introduced by overlapping frames depends on the technology used to buffer frames. In the best case, a delay of L_{frame} is introduced when using overlapped windows. However, the subsystem needs to be able to function within Simulink. Simulink only supports using overlapped windows with a delay of $2L_{frame}$. The maximum allowed latency, as prescribed in Chapter 2, is 50 ms. With a sampling frequency of $F_s = 44.1kHz$, this gives a maximum delay of $L_{delay} = 2205$ samples. Thus, the maximum possible frame length is $L_{frame} = 1102$ samples.

The window overlap is not fixed to 50%; it can be varied if needed, as long as L_{frame} is a multiple of L_{hop}, where

$$L_{frame} = L_{overlap} + L_{hop} \tag{5.2}$$

See Appendix B.1 for more detail. The overlapping windows will then not add up to 1, but to $\frac{N_O}{2}$, where N_O is the amount of frames that overlap. So for an overlap of 50% $N_O = 2$, for 33% $N_O = 3$, etc. However, a normalisation takes care of this. Changing the window overlap does not change the delay due to framing. However, it can increase the necessary processing time, as more frames need to be processed for the same amount of data.

The advantage of more overlap between frames is that more samples are taken into account when reconstructing the signal which has the potential to increase audio quality in the end.

5.1. Noise Injection

5.1.1. Basic Noise Injection

NI is simple to implement. All that is needed is a Gaussian White Noise source that adds a single value to each signal sample passing through the decorrelation device. No framing is necessary for this procedure. The function to implement NI can be found in Appendix A.1.

The only parameter to be found is the value of the Signal-to-Noise Ratio (SNR) that gives maximum decorrelation while the signal quality does not fall below $ODG = -0.5$ as specified in the requirements in Chapter 2. Figure 5.2 shows the ODG and correlation coefficient as a function of the SNR.

Figure 5.2 shows that a lower SNR leads to a higher decorrelation. Therefore, the minimum SNR should be chosen where sound quality does not fall below the limit of $ODG = -0.5$. The minimum ODG meets the limit of -0.5 at $SNR = 62.8dB$, so this is the value chosen for the parameter.

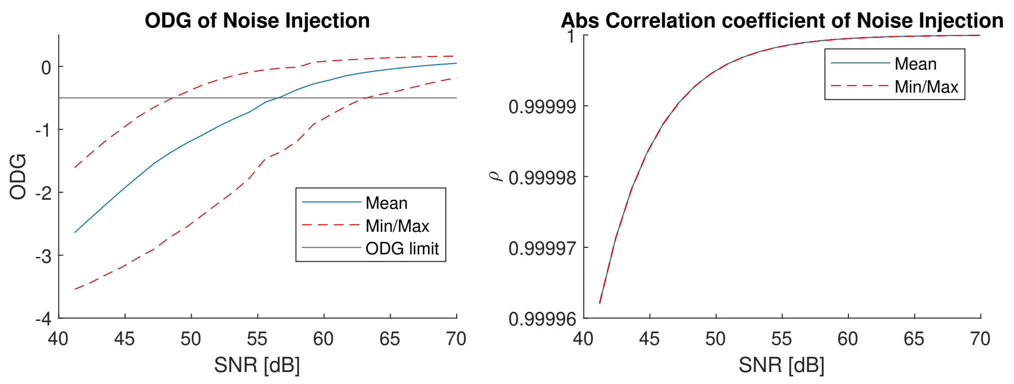

Figure 5.2: The mean, minimum, and maximum ODG and Correlation coefficient of basic NI

5.1.2. Framed Noise Injection

For FNI the signal is divided into frames of constant length. For each of these frames, a frame of noise is calculated based on a preset constant SNR. This frame of noise is then added to the audio frame. As the signal power varies with time, using a constant SNR will give a different noise power in each of the frames. The function to implement FNI can be found in Appendix A.1.

There are two parameters to optimise: the SNR and the frame length. Our experiments have shown that the frame length has a negligible influence on the performance of the technique, however a slight maximum in the ODG does appear around $L_{frame} = 400$ samples. Figure 5.3 shows the ODG and correlation coefficient as a function of the SNR at frame length 400.

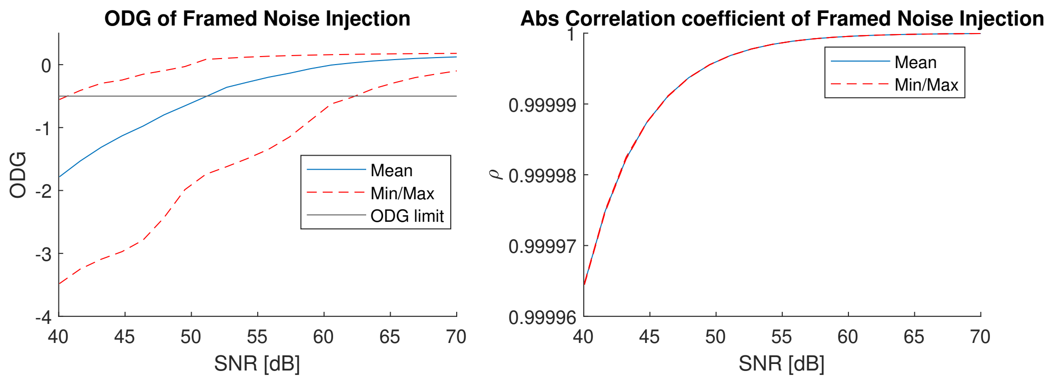

Figure 5.3: The mean, minimum, and maximum ODG and Correlation coefficient of FNI

Figure 5.3 shows that a lower SNR leads to a higher decorrelation. Therefore, the minimum SNR should be chosen where sound quality does not fall below the limit of $ODG = -0.5$. The minimum ODG meets the limit of -0.5 at $SNR = 62.12dB$, so this is the value chosen for the parameter.

As this technique uses frames, a delay of 400 samples is introduced. As shown in Section 5.4, this also provides a decorrelating effect, which will be discussed in Chapter 6.

5.1.3. Shaped Noise Injection

For SNI, to measure the signal power in different frequency bands, the FFT is used. In order to use the FFT, a fully formed signal is needed. This means the use of frames is necessary for this method. Since the FFT and IFFT are only used to find correct noise powers in the bark frequency bands and not to reconstruct the output signal, it is not necessary or beneficial to use windowing for the FFT.

The FFT is taken of each frame, after which the noise power in each band of the Bark scale is calculated. By dividing the noise power in each band by the noise power of the maximum power band, a weighting function with a maximum of 1 is created. A noise signal with some base value for the SNR is then spectrally shaped according to this weighting function. This noise is added to the signal frame. The function to implement SNI can be found in Appendix A.1.

Again, there are two parameters to optimise: the SNR and the frame length. The Bark scale is based on literature [21] and explained in Chapter 4. The frame length is again chosen as $L_{frame} = 400$, similar to FNI, as a maximum in the ODG appears around this value. Note that using frames introduces a delay in the signal which has an effect on the correlation. To fairly compare the different decorrelation methods, this effect will be discussed in Chapter 6. Figure 5.4 shows the ODG and correlation coefficient as a function of the SNR at frame length 400.

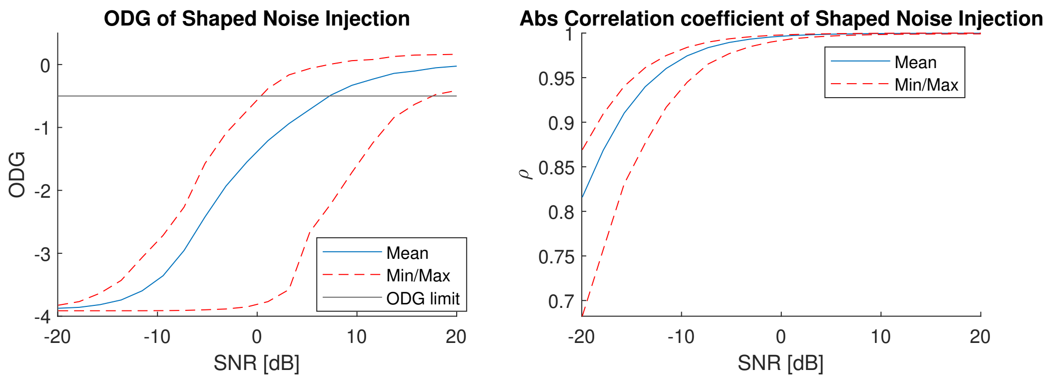

Figure 5.4: The mean, minimum, and maximum ODG and Correlation coefficient of SNI at frame length 400 samples.

Figure 5.4 shows that a lower SNR leads to a higher decorrelation. Therefore, the minimum SNR should be chosen where sound quality does not fall below the limit of $ODG = -0.5$. The minimum ODG meets the limit of -0.5 at base $SNR = 15.5dB$, so this is the value chosen for the parameter.

5.2. Time-Varying Processing

5.2.1. Sinusoidal Phase Modulation

SinPM as described in Chapter 4 can be implemented using the Hilbert Transform [3]. The output signal is then given by

$$v(n) = \mathrm{Re}\{u_a(n)H(\omega,n)\} \tag{5.3}$$

where

$$u_a(n) = u(n) + j\hat{u}(n) \tag{5.4}$$

and $\hat{u}(n)$ is the Hilbert Transform of the signal $u(n)$. This results in output

$$v(n) = u(n) \cdot \exp(j\phi) \tag{5.5}$$

$$\phi(n) = \beta \sin \omega_m n \tag{5.6}$$

In this implementation, the Hilbert Transform is approximated using the approximation of $u_a(n)$ as the inverse FFT of the one-sided spectrum of $u(n)$ [3]. To find this, the FFT of $u(n)$ is taken and the negative frequency components are all set to zero. Then, the real part of the IFFT is $u(n)$ and the imaginary part is $\hat{u}(n)$. As the FFT is needed, similar to SNI, frames are used. As the output is generated from the FFT, windowing and window overlap are implemented. The function to implement sinPM can be found in Appendix A.2.

The parameters to optimise are the constant β, the modulation frequency $\omega_m = 2\pi f_m$, the frame length L_{frame}, and the frame overlap $L_{overlap}$. Figure 5.5 shows the mean correlation coefficient for sinPM as a function of both β and f_m. Once f_m exceeds $0.02Hz$, a clear valley can be seen in the correlation at $\beta = 2.4$, where changing f_m does not have a significant effect. Therefore $\beta = 2.4$ is chosen for this parameter.

Figure 5.6 shows the ODG and correlation coefficient as a function of f_m with $\beta = 2.4$. The ODG remains relatively stable with varying f_m. The correlation coefficient shows clear maxima and minima, however the positions of these are different for the different audio clips, so they cannot be reliably located. The lowest correlation can be found at $f_m = 0.152Hz$, so this is the value chosen for this parameter.

The final parameters to determine are the frame length and frame overlap. Figure 5.7 shows the ODG and correlation coefficient as a function of the frame length for $\beta = 2.4$ and $f_m = 0.152Hz$ with

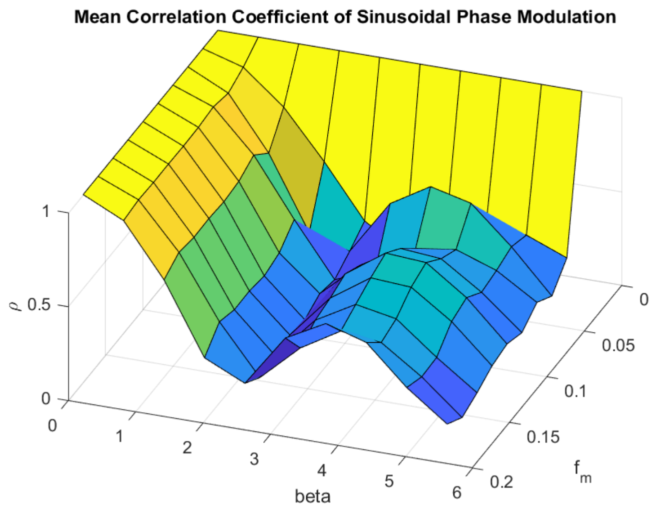

Figure 5.5: The mean Correlation coefficient of sinPM for varying β and f_m.

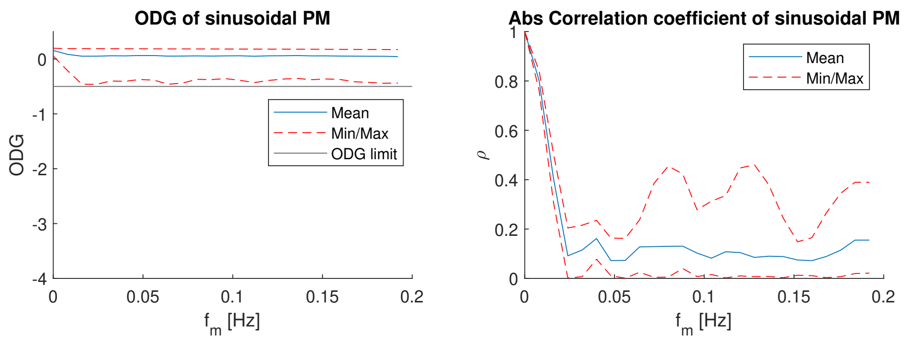

Figure 5.6: The mean, minimum, and maximum ODG and Correlation coefficient of sinPM.

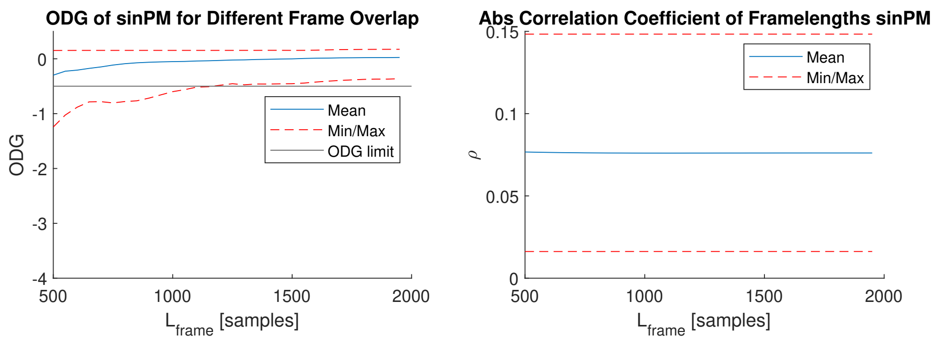

Figure 5.7: The mean, minimum, and maximum ODG and Correlation coefficient of sinPM with different frame lengths. Frames with 90% overlap are used.

an overlap of 90% as a base. The correlation coefficient does not change significantly for different frame lengths, but the ODG does. For the ODG to meet the minimum requirement of $ODG = -0.5$, a frame length of $L_{frame} = 1200$ is needed. This slightly exceeds the maximum possible frame length of $L_{frame} = 1102$ samples as explained earlier in this chapter. This means sinPM would result in more delay than the allowed $50ms$. Note that the delay introduced into the signal by using frames also has an effect on the correlation. To fairly compare the different decorrelation methods, this effect will be discussed in Chapter 6.

Figure 5.8 shows the ODG and the average computation time per sample for different amounts of samples overlap with frame length 1200 samples. Computation time was measured within MATLAB itself, as this was deemed the most accurate considering an eventual physical implementation. Both graphs also show the limits for their respective values; the minimum ODG is -0.5 and the maximum computation time per sample is the sample time $T_s = 1/F_s = 2.268 \cdot 10^{-5}s$. The figure shows that to meet the ODG requirement, an overlap of at least 1050 samples is needed. As explained earlier in this chapter and is derived in Appendix B.1, the non-overlapping amount of samples defined as L_{hop} in equation 5.2 needs to be a divisor of L_{frame} for the windowing process to function correctly. Therefore an overlap of $L_{overlap} = 1080$ samples is chosen, or 90%, resulting in $L_{hop} = 120$ which is a divisor of $L_{frame} = 1200$. The second part of the figure shows that for this value of $L_{overlap}$ the average computation time falls easily within the limit. The overlap is not increased more, as the aim for all subsystems is to minimise computational complexity in order to be able to operate in real time, see the requirements.

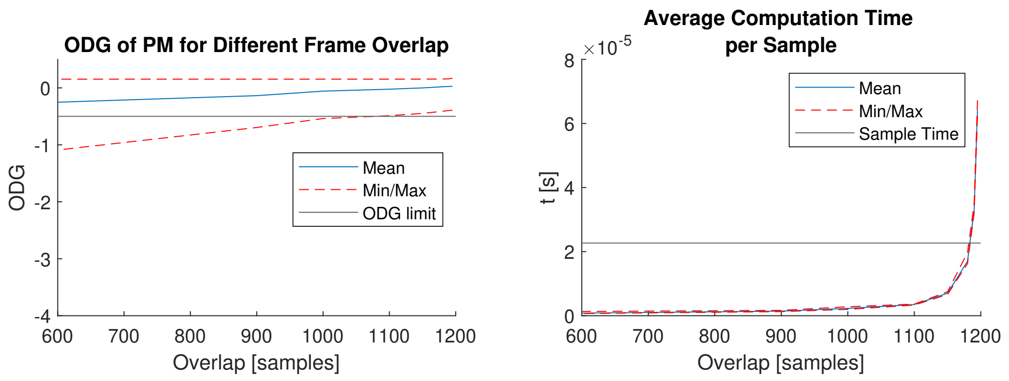

Figure 5.8: The ODG and average computation time per sample for sinPM for different amounts of overlap.

5.2.2. Frequency Shifting

FS can be implemented in a similar manner to sinPM. All that is changed is $\phi(n)$:

$$\phi(n) = \omega_m n \tag{5.7}$$

This means that again, as the FFT is needed, frames and windowing are used. The function to implement frequency shifting can be found in Appendix A.3.

The parameters that need to be optimised are the modulation frequency ω_m, the frame length L_{frame}, and the frame overlap $L_{overlap}$. The frame overlap can then be calculated from the maximum delay of 1102 samples and the frame length. Figure 5.9 shows the ODG and correlation coefficient as a function of the modulation frequency f_m, where $\omega_m = 2\pi f_m$. This figure was created without the use of framing.

Figure 5.9 shows that between $f_m = 0$ and $f_m = 2Hz$ very little change can be seen in the ODG, so any f_m between these values should give similar results in audio quality. Having f_m in this range already provides significant decorrelation. The correlation coefficient does show some peaking below $f_m = 0.5Hz$; the optimum parameter is found as $f_m = 0.6535Hz$.

Figure 5.10 shows the ODG and correlation coefficient as a function of the frame length for $f_m = 0.6535Hz$ with a base overlap of 90%. The correlation coefficient does not change significantly for different frame lengths, but the ODG does. For the ODG to meet the minimum requirement of $ODG = -0.5$, a frame length of $L_{frame} = 1000$ is needed. This falls within the maximum of $L_{frame} = 1102$ samples.

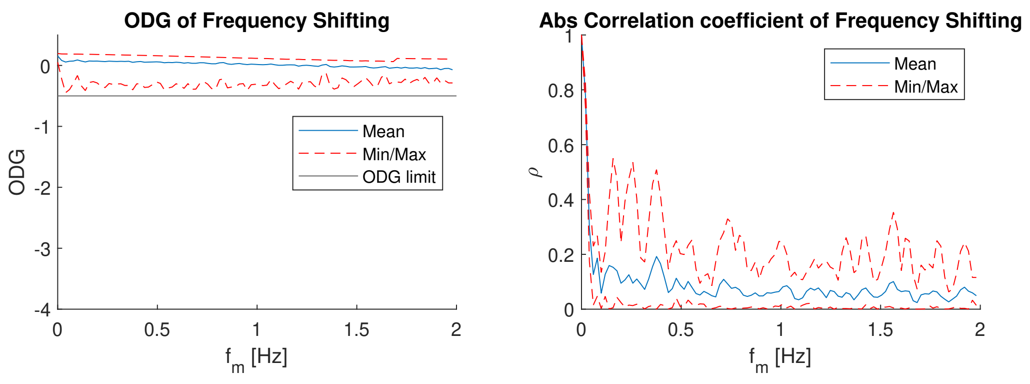

Figure 5.9: The mean, minimum, and maximum ODG and Correlation coefficient of FS.

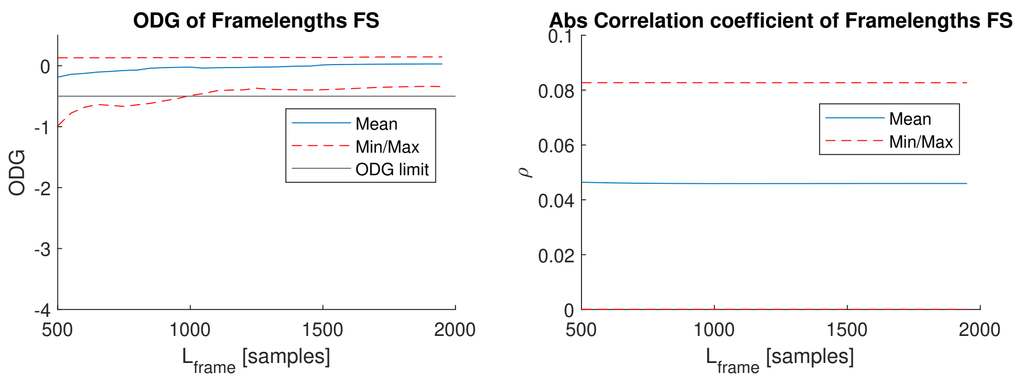

Figure 5.10: The mean, minimum, and maximum ODG and Correlation coefficient of FS with different frame lengths. Frames with 90% overlap are used.

Note that the delay introduced into the signal by using frames also has an effect on the correlation. To fairly compare the different decorrelation methods, this effect will be discussed in Chapter 6.

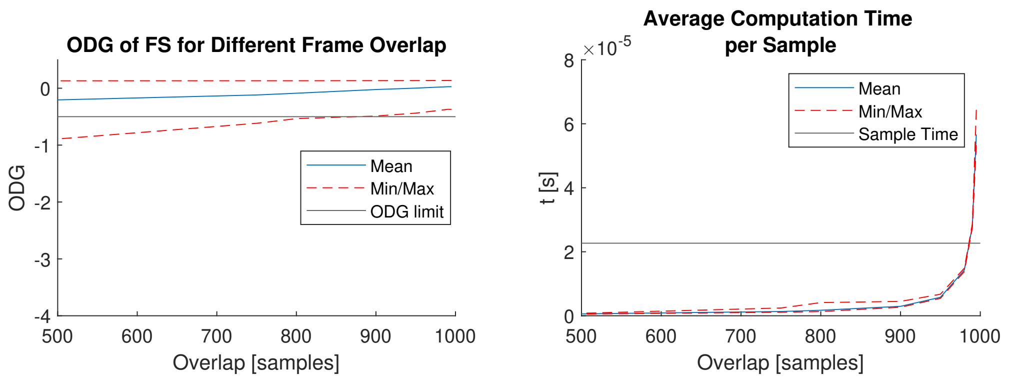

Figure 5.11: The ODG and average computation time per sample for FS for different amounts of overlap.

Figure 5.11 shows the ODG and the average computation time per sample for different amounts amounts of samples overlap with frame length 1200 samples. Computation time was measured within MATLAB itself, as this was deemed the most accurate considering an eventual physical implementation. Both graphs also show the limits for their respective values; the minimum ODG is -0.5 and the maximum computation time per sample is the sample time $T_s = 1/F_s = 2.268 \cdot 10^{-5}s$. The figure shows that to meet the ODG requirement, an overlap of at least 880 samples is needed. As explained earlier in this chapter, the non-overlapping amount of samples defined as L_{hop} in equation 5.2 needs to be a divisor of L_{frame} for the windowing process to work. Therefore an overlap of $L_{overlap} = 900$ samples, or 90% is chosen, resulting in $L_{hop} = 100$ which is a divisor of $L_{frame} = 1000$. The second part of the

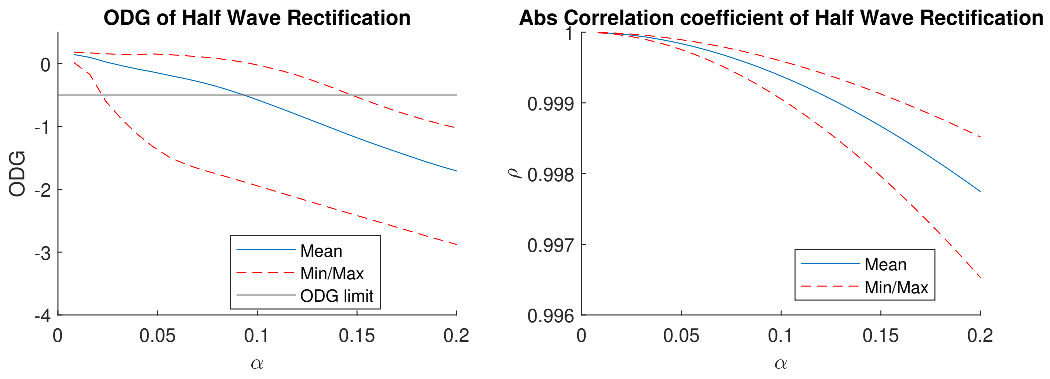

Figure 5.12: The mean, minimum, and maximum ODG and Correlation coefficient of HW.

figure shows that for this value of $L_{overlap}$ the average computation time falls easily within the limit. The overlap is not increased more, as the aim for all subsystems is to minimise computational complexity.

5.3. Nonlinear Processing

5.3.1. Half-Wave Rectification

Similar to NI, HW can be applied to individual samples, meaning there is no need to use frames. Each sample is modified as:

$$v(n) = u(n) + \frac{1}{2}\alpha(u(n) + |u(n)|) \tag{5.8}$$

The function to implement HW can be found in Appendix A.4.

The parameter that needs to be optimised is the constant α.Figure 5.12 shows the ODG and correlation coefficient as a function of α.

Figure 5.12 shows that a higher α leads to a higher decorrelation. Therefore, the maximum α should be chosen where sound quality does not fall below the limit of $ODG = -0.5$. The minimum ODG meets the limit of -0.5 (see requirement 6 in Chapter 2) at $\alpha = 0.022$, so this is the value chosen for the parameter.

5.3.2. Smoothed Half-Wave Rectification

HWS is also applied to single samples and therefore there is no need to use frames. Each sample is modified as:

$$v(n) = u(n) + \alpha\left(\frac{u(n)}{2} + \sqrt{\left(\frac{u(n)}{2}\right)^2 + c^2}\right) \tag{5.9}$$

The function to implement HWS can be found in Appendix A.5.

The parameters that need to be optimised are the constants α and c. Optimisation of two parameters is difficult to show with a 2D plot, so here it is done with two separate graphs. Note that the actual optimisation was done with simultaneous analysis of both variables, so all variations have been taken into account. These graphs simply illustrate the optimisation process. The graphs in Figures 5.13 and 5.14 show the variation in ODG and correlation coefficient when varying one parameter, while the other parameter is kept constant at its optimum value.

Figure 5.13 shows that increasing α results in increased decorrelation. Therefore the maximum α for which the ODG remains above $ODG = -0.5$ should be chosen. The ODG shows a clear peak in the minimum at $\alpha = 0.12$, so this is the value chosen for the parameter.

Figure 5.14 shows that decreasing c results in increased decorrelation. Therefore the minimum c for which the ODG remains above $ODG = -0.5$ should be chosen. The ODG shows a bump in the minimum at $c = 0.28$, so this is the value chosen for the parameter.

To summarise HWS: optimisation was done while varying both parameters simultaneously, but this resulted in data that could not easily be displayed here. The graphs in Figures 5.13 and 5.14 show an indication of the optimisation process. The values chosen for the parameters are $\alpha = 0.12$ and $c = 0.28$ as these result in minimum allowed ODG and maximum decorrelation.

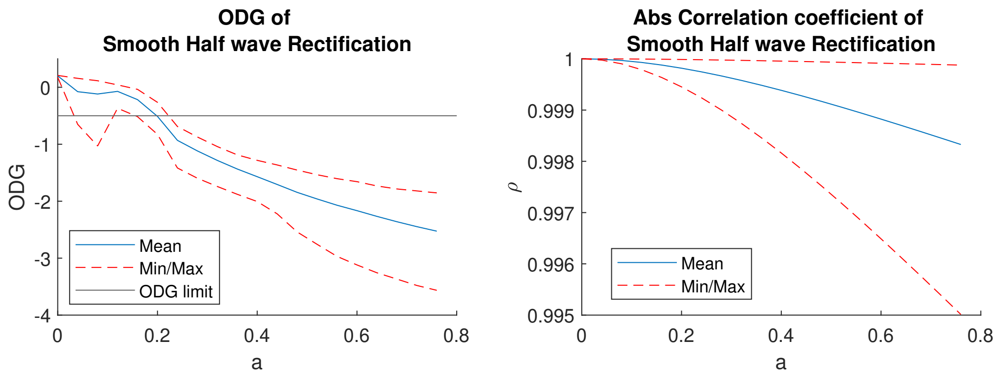

Figure 5.13: The mean, minimum, and maximum ODG and Correlation coefficient of the HWS for variation of the parameter α.

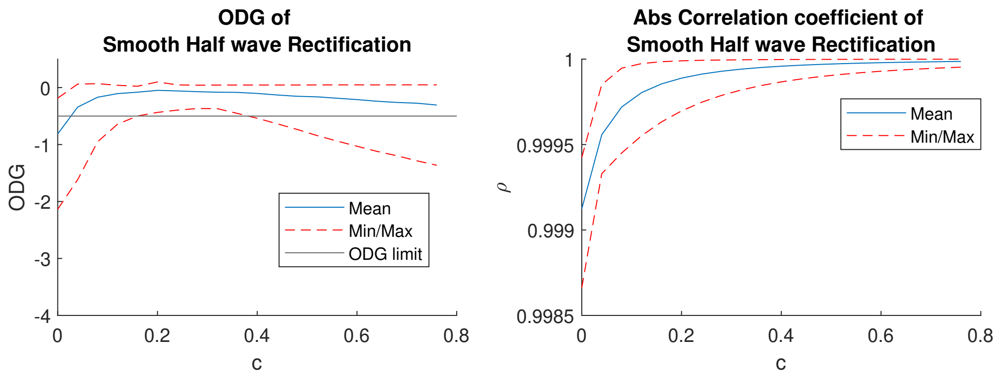

Figure 5.14: The mean, minimum, and maximum ODG and Correlation coefficient of the HWS for variation of the parameter c.

5.3.3. Square-Law

The SL is also applied to single samples, so there is no need to use frames. Each sample is modified as:

$$v(n) = u(n) + \alpha u^2(n) \tag{5.10}$$

The function to implement the SL-operation can be found in Appendix A.6.

The parameter that needs to be optimised is the constant α. Figure 5.15 shows the ODG and correlation coefficient as a function of α.

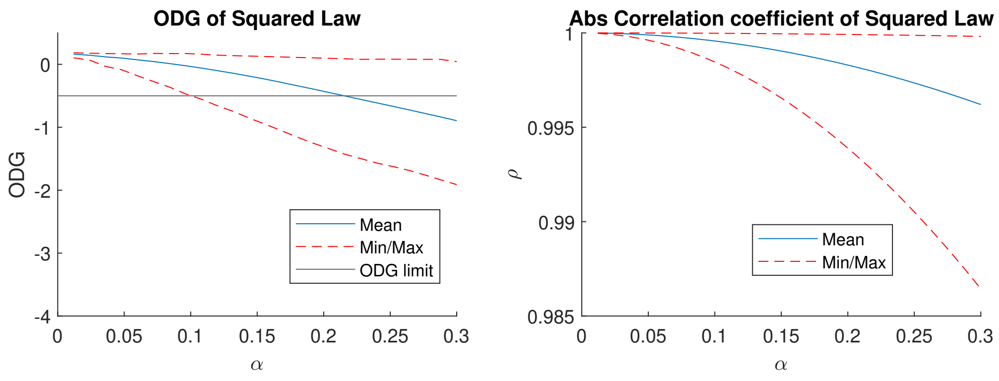

Figure 5.15: The mean, minimum, and maximum ODG and Correlation coefficient of the SL.

Figure 5.15 shows that a higher α leads to a higher decorrelation. Therefore, the maximum α should be chosen where sound quality does not fall below the limit of $ODG = -0.5$. The minimum ODG meets the limit of -0.5 at $\alpha = 0.096$, so this is the value chosen for the parameter.

5.4. Forward Path Delay

FPD is implemented by buffering the signal for a set amount of samples Δ before passing it on. As explained in earlier in this chapter in Section 5, the maximum allowed delay is $\Delta = 2205$ samples. Figure 5.16 shows the correlation coefficient as a function of the delay introduced into the signal path. The ODG is not shown, as it does not produce useful results in this case; the signal is kept the same, so its quality does not degrade. The only relevant measure of quality is the amount of delay, which is preferably as low as possible.

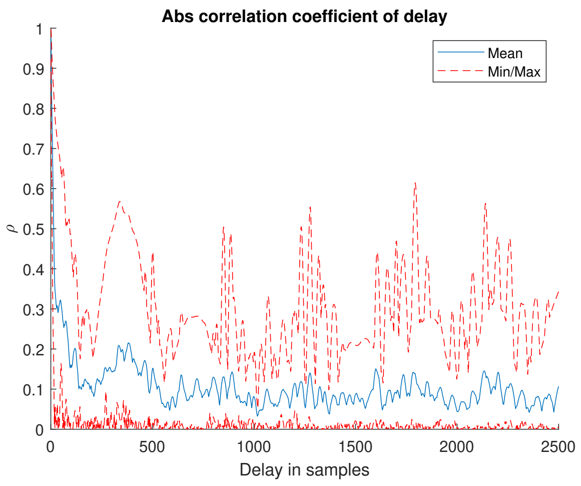

Figure 5.16: The mean, minimum, and maximum correlation coefficients of the delayed sounds with themselves (FPD).

Figure 5.16 shows multiple minima and maxima. The locations of these peaks and valleys depend on the frequencies present in each of the signals; when one signal shows a low correlation coefficient for a certain amount of delay, another signal may show very high correlation. Therefore, there is no generalised correct answer for the optimal amount of delay. Assuming the 9 audio clips used for the experiments contain most frequencies present in the sounds that the PA system should be able to handle, a delay of $\Delta = 215$ may be chosen as optimal for these types of sounds; ρ shows a dip around this value for Δ.

5.5. Overview of Parameter Design Choices

Table 5.1 shows an overview of the parameters chosen for each method.

Table 5.1: Overview of parameter choices for each of the implemented methods.

	Parameters	Value	Frame Length	Overlap
Basic Noise Injection	SNR	62.8 dB	-	-
Framed Noise Injection	SNR	62.12 dB	400	-
Shaped Noise Injection	SNR	15.5 dB	400	-
Sinusoidal Phase Modulation	β	2.4	1200	90 %
	f_m	0.152 Hz		
Frequency Shifting	f_m	0.6536 Hz	1000	90 %
Half-Wave Rectification	α	0.022	-	-
Smoothed Half-Wave Rectification	α	0.12	-	-
	c	0.28		-
Square-Law	α	0.096	-	-
Forward Path Delay	Δ	215 samples	-	-

6 Comparison of Decorrelation Methods

In this chapter the nine decorrelation methods implemented in Chapter 5 are compared when implemented using the parameters chosen in the same chapter. These parameter choices can be found in Table 5.1. All data is again based on the 9 audio clips introduced in Appendix D.1.

6.1. Results

Table 6.1: Comparison of the achievable ODG and decorrelation of the different methods given the parameters in Table 5.1.

	ODG			***ρ***		
	min	mean	max	min	mean	max
Basic Noise Injection	-0.52	-0.10	0.12	1.0	1.0	1.0
Framed Noise Injection	-0.51	0.03	0.16	1.0	1.0	1.0
Shaped Noise Injection	-0.42	0.01	0.15	0.9997	0.9999	0.9999
Sinusoidal Phase Modulation	-0.48	-0.03	0.15	0.0162	0.0760	0.1483
Frequency Shifting	-0.49	-0.02	0.13	0.0	0.0459	0.0827
Half-Wave Rectification	-0.48	0.05	0.16	1.0	1.0	1.0
Smoothed Half-Wave Rectification	-0.36	-0.03	0.08	0.9998	0.9999	1.0
Square-Law	-0.47	-0.02	0.10	0.9986	0.9996	1.0
Forward Path Delay	-	-	-	0.0253	0.0836	0.1554

Frame Delay

As mentioned in Chapter 5, four methods introduce a delay in the signal because of the use of framing. These methods are:

- Framed Noise Injection
- Shaped Noise Injection
- Sinusoidal Phase Modulation
- Frequency Shifting

As shown in Section 5.4, such a delay has an effect on the decorrelation of the signal. Table 6.1 shows the performance of the various methods without the effect of delay on the correlation. Table 6.2 shows the correlation coefficient of the four methods with and without the added delay. Framed Noise Injection and Shaped Noise Injection are delayed by 400 samples. As explained in Chapter 5, sinPM and FS use overlapping frames. As we are using Simulink for the implementation, this means that these methods introduce $2L_{frame}$ delay into the system. This is a limitation set by the Overlap-Add Method as a system block in Simulink. For this reason, $L_{frame} = 1200$ and $L_{frame} = 1000$ respectively result in delays of 2400 and 2000 samples. Note that the ODG is not considered here as the delay does not distort the signal itself.

Table 6.2: Comparing Correlation Coefficients of the decorrelation methods that use frames, with and without delay introduced by framing.

	ρ					
	Without Delay			**With Delay**		
	min	mean	max	min	mean	max
Framed Noise Injection	1.0	1.0	1.0	0.0140	0.1757	0.4995
Shaped Noise Injection	0.9997	0.9999	0.9999	0.0117	0.1766	0.5014
Sinusoidal Phase Modulation	0.0162	0.0760	0.1483	0.0	0.0577	0.2318
Frequency Shifting	0.0	0.0459	0.0827	0.0087	0.0332	0.1103

Table 6.2 shows that the added delay does not have a significant influence on the decorrelation performance of sinPM and FS. However, the added delay does significantly increase the decorrelation of the Noise Injection methods. Without delay, these methods are not able to provide any meaningful decorrelation, but the delay increases their performance with multiple orders of magnitude. However, when compared to the performance of the Forward Path Delay method as seen in Table 6.1, one can see that choosing the delay more carefully yields much better results. For example, the maximum correlation for both Noise Injection methods is around $\rho = 0.5$, while Forward Path Delay gives a maximum correlation of $\rho = 0.1554$.

In conclusion, the added delay does not change the performance of sinPM and FS significantly, but it does change the performance of both FNI and SNI. However, simply using Forward Path Delay yields better results than both the Noise Injection methods.

6.2. Conclusion

Noise Injection

All three Noise Injection methods fall short because a significant value for the SNR would result in an audible noise signal. While shaping the noise does allow for much greater added noise power, it does not significantly increase the decorrelation. When using a PA system, any audible noise will be perceived as annoying, making these three methods unsuitable for our system as set by requirement 2.1.6 in Chapter 2. Table 6.2 shows that adding the delay introduced by framing does provide more significant decorrelation for FNI and SNI, but this is purely due to the delay itself. By choosing the delay more carefully, as with the Forward Path Delay method, better results are achievable.

Nonlinear Processing Methods

Similar to the Noise Injection methods, the Nonlinear Processing methods are not able to provide significant decorrelation. This is again due to the fact that only very small changes in the signal remain inaudible; as soon as more effective nonlinear changes are made, the signal is audibly distorted.

Final Comparison

This leaves only three methods that produce significant decorrelation, which are sinPM, FS, and FPD. These three methods produce results at a similar scale; for all three methods the mean lies below $\rho = 0.1$, while all other methods do not achieve a mean below $\rho = 0.99$ without audible distortion. However, for sinPM to function well, a frame length of $L_{frame} = 1200$ samples is needed, which exceeds the maximum allowed frame length of $L_{frame} = 1102$ samples. It can therefore not be used for this project.

This means only FS and FPD remain as options. Figure 6.1 shows a comparison between these two methods, where the delayed version of FS is used to compute the correlation coefficient. The delays used are 2000 samples for Frequency Shifting and 215 samples for FPD.

The figure shows that in all aspects FS decorrelates the signal more than FPD. On top of this, as mentioned before, the performance of FPD depends heavily on the audio characteristics. The optimum delay depends mostly on the frequencies present in the signal. It is therefore not possible to choose a value for the delay that works optimally for all possible audio signals. FS does not depend on the audio clip in this way and is thus a more reliable method.

We can conclude that Frequency Shifting falls within the Programme of Requirements as specified in Chapter 2 and has the best decorrelation properties out of all the techniques researched in this

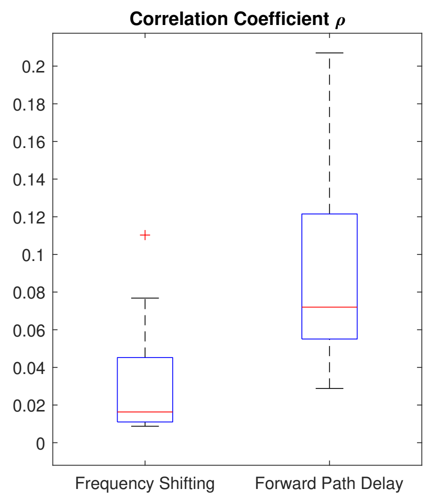

Figure 6.1: Boxplots of the correlation coefficient for Frequency Shifting (FS) and Forward Path Delay (FPD).

project. Therefore, Frequency Shifting will be implemented and used in this project as the Decorrelation Subsystem.

7 Testing and Discussion of Results

In this chapter the Decorrelation subsystem as designed in Chapters 5 and 6 is tested in a simulated version of the full Adaptive Feedback Cancellation system. The Decorrelation subsystem is implemented as a Frequency Shifting Time-Varying Filter. The subsystem performs the Frequency Shifting operation on subsequent frames of 1000 samples which are windowed using a Hann window and 90% overlap between frames. The Frequency Shifting operation itself is performed by the MATLAB function `Fs()`, which can be found in Appendix A.

The first section introduces the Testing Setup. After this, the change in misalignment (see Section 3.2) due to the Decorrelation Subsystem is tested and analysed.

7.1. Testing Setup

Figure 7.1 shows the Simulink model used to test the subsystem. The model is a simplified version of the full system created in this project, containing only the parts necessary to test the Decorrelation Subsystem. An audio signal is fed into the system in the *audio* block. This signal is fed to the Decorrelation subsystem. The signal is controlled by two switches. The *[Decorrelation: y/n]* switch controls whether the subsystem is used. The other switch is used to make sure that the first 2000 samples of each audio clip are fed through without Decorrelation. This is necessary because the Decorrelation subsystem inserts 2000 samples delay, and in this simplified version of the full system the Adaptive Filter cannot operate without a nonzero value at the *Input* port.

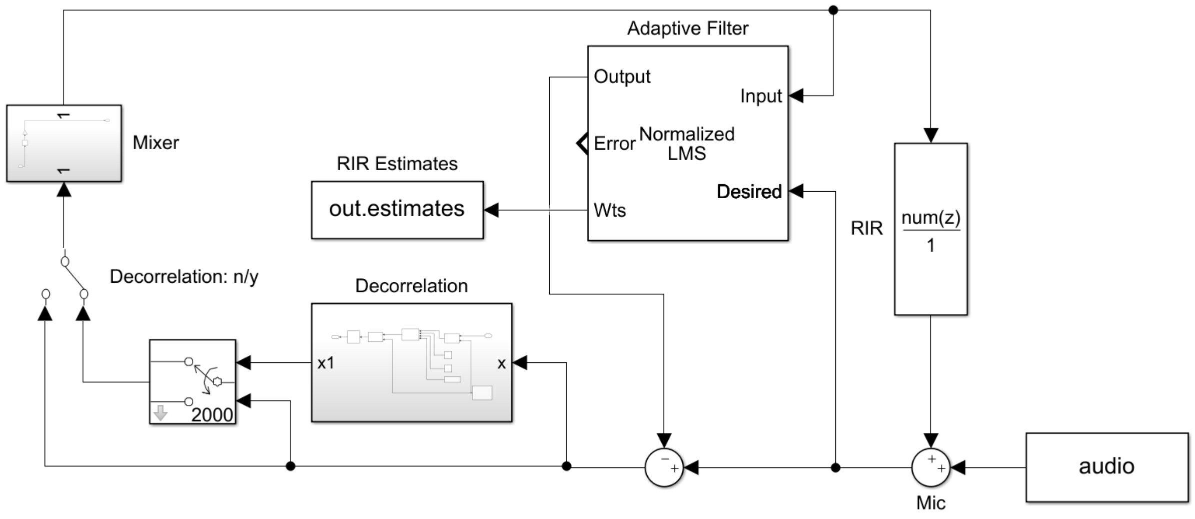

Figure 7.1: Simulink model used to test the Decorrelation Subsystem. It consists of the PA system with feedback, a simple implementation of the Adaptive Filter, and the implemented Decorrelation Subsystem.

After the Decorrelation subsystem the signal is passed through the *Mixer*, which contains a Delay and a Gain. Then the signal ends up at the *Adaptive Filter* and *RIR* blocks. The *RIR* block contains a room impulse response for testing. This impulse response can be found in Appendix D.2. The output of the *RIR* block is added to the *audio* input to create feedback.

The *Adaptive Filter* block is a Simulink library-standard Normalised Least Mean Squares (NLMS) filter, which is the most similar to the filter estimation block created by the Adaptive Filter subgroup [7]. Sadly, at the time of writing we were unable to test with the finalised Adaptive Filter Subsystem due to time constraints. Therefore the closest approximation readily available in the simulation environment of Simulink was used. The parameters of the NLMS filter were set to *Step Size* $\mu = 0.04$ and *Leakage Factor*$= 1.0$ based on rough testing, meaning this Filter Block was not fully optimised. For the optimisation of this subsystem see Thesis [7]. The output of the *Adaptive Filter* block is an estimate of the Room Impulse Response, updated every 500 samples.

7.2. Misalignment Results

Figure C.1 in Appendix C shows the misalignment over a 60 second audio clip of the 9 different sounds as introduced in Appendix D.1. The misalignment is shown for both the system with and without the Decorrelation Subsystem implemented. The misalignment without Decorrelation is quite similar between all clips. However, for the misalignment with Decorrelation a clear distinction can be made between music and speech clips, as highlighted in Figure 7.2.

For all music clips the Decorrelation misalignment has a small hitch at the start, but then it quickly converges to a low value between -20 and -10 dB. This means that the estimated RIR ($\hat{F}$) differs very little from the actual RIR (F). For all speech clips, this is not the case; every couple of seconds the misalignment peaks, indicating that $\hat{F}$ is suddenly completely different than F. These peaks are due to the pure silent periods that exist within these clips. Every couple of seconds the spoken clips are completely silent, meaning the signal has an exact amplitude of 0. Once a voice speaks, there is a mismatch between the *Desired* and *Input* ports seen in Figure 7.1, causing the estimator to fail.

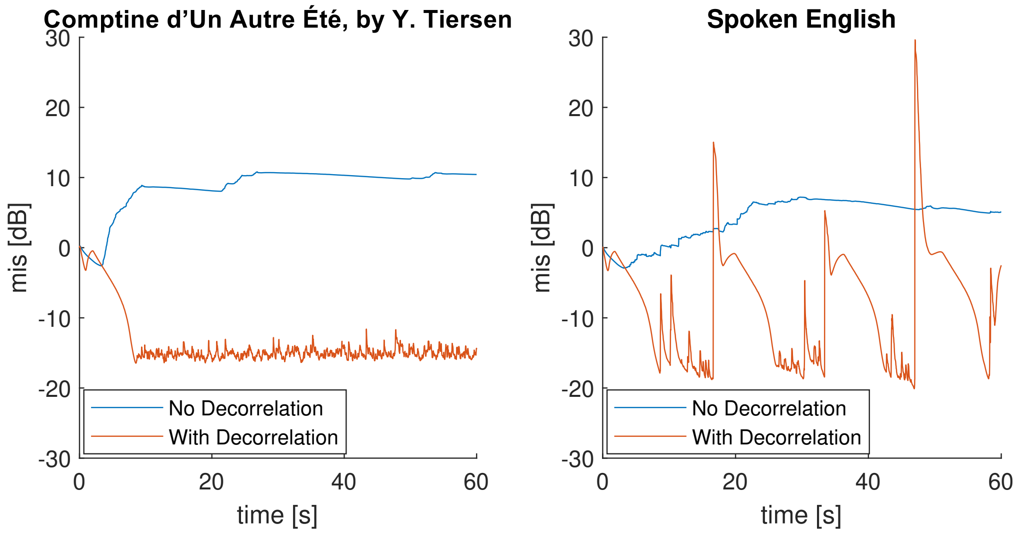

Figure 7.2: Misalignment over time for a audio clips of piano and spoken english. Both the misalgnment with and without the Decorrelation Subsystem are shown.

For all music clips the Decorrelation significantly improves the convergence of the misalignment to a low value. Without the Decorrelation subsystem, there is a clear bias in the estimator, causing $\hat{F}$ to not converge to F.

8 Conclusion and Future Work

For this project the aim was to create an Adaptive Feedback Cancellation system to prevent feedback in Public Address systems. This system should consist of an Adaptive Filter to estimate the Room Impulse Response (RIR) and cancel the feedback, a Postfilter to ensure stability and provide a back-up option, and a Decorrelation Subsystem to remove the correlation between the microphone and speaker signal in order to improve the RIR estimation. This particular thesis is dedicated to the Decorrelation Subsystem.

The Decorrelation Subsystem is a supportive system to the Adaptive Filter. As explained in Chapter 3, the Adaptive Filter cannot fully converge to its desired result, the estimation of the Room Impulse Response, because there is correlation between the input signal $s(n)$ and the speaker signal $x(n)$ as in Figure 1.4. This causes the RIR estimate to be biased. The Decorrelation Subsystem aims to remove this bias by decorrelating $s(n)$ and $x(n)$ while retaining overall signal quality.

8.1. Method Comparison

In Chapter 4 four different decorrelation methods are introduced and explained. In Chapter 5 each of these methods is implemented and optimised to assure maximum decorrelation while retaining signal quality. In Chapter 6 the different methods are compared, and the conclusion is that Frequency Shifting is the method most suitable for our needs.

8.2. Decorrelation Performance

In Chapter 7 Frequency Shifting is implemented as the Decorrelation Subsystem in a testing setup. This setup simulates the environment in which the subsystem is supposed to function. It contains the full feedback loop and an implementation of the Adaptive Filter. In the Chapter this setup is tested with the Decorrelation Subsystem deactivated and activated.

For music sound signals the Decorrelation significantly improves the RIR estimate by removing the bias. Figure C.1 shows a relatively stable misalignment for each of these signals. Some clear peaks are visible, but these do not exceed $-10dB$, meaning they indicate relatively insignificant disturbance in the RIR estimate.

For speech sound signals the Decorrelation does not improve the estimate. This is due to the fact that these signals contain pure silent periods where the signal amplitude is exactly 0. Once a sound starts again from such a silent period, the misalignment peaks enormously, meaning the RIR estimate is very far from the actual value. This peaking is due to the delay introduced by the Decorrelation Subsystem. Due to this delay, the Adaptive Filter will have a mismatch between its inputs; the *Input* is completely silent while the *Desired* port does receive a nonzero signal. A filter cannot create a nonzero output signal from an all-zero input, so the algorithm will try to find a filter that does not exist.

This means the Adaptive Filter combined with the Decorrelation Subsystem in the setup described in this report are not usable if pure silence can be present in the signal. However, the actual Adaptive Filter implemented by the dedicated subgroup [7] is able to handle such a delay. No testing has been done with the combination of the real Adaptive Filter and the Decorrelation Subsystem, but as the delay does not pose a problem for this group, the peaking due to pure silences should disappear.

8.3. Fulfillment of Requirements

Subsystem Requirements

The implementation of Frequency Shifting, given the chosen parameters, as the Decorrelation Subsystem fulfills all set requirements for the subsystem as found in Chapter 2. It provides the most decorrelation out of all tested methods, satisfying requirement 2.2.1, while the signal quality never falls below $ODG = -0.5$, satisfying requirement 2.2.2 (and requirement 2.1.7).

Full System Requirements

The full system consisting of the Decorrelation Filter, the Adaptive Filter realised in [7] and the Postfilter realised in [8] has not been tested at the time of writing. However, we can analyse if the Decorrelation Filter satisfies the functional requirements set for the system as a whole. The Decorrelation Filter can be implemented as part of an on-chip system, satisfying requirement 2.1.1. The subsystem can operate between frequencies $20Hz$ and $20kHz$ and it operates at a sampling frequency higher than $20kHz$, satisfying requirements 2.1.2 and 2.1.3. The subsystem requires no human interaction to set up or maintain functionality, and it can function in real time as the calculation time per sample is smaller than the sampling time $T_s = 1/F_s$, satisfying requirements 2.1.4 and 2.1.5. As stated before, the signal quality is not reduced by the subsystem below the limit of $ODG = -0.5$ as set by requirement 2.1.7. Finally, Frequency Shifting introduces a delay of 2000 samples, which translates to $0.045s$ delay, satisfying requirement 2.1.8.

Requirement 2.1.6, stating that the system should increase the MSG by more than $10dB$ can only be tested with the full system.

As we are unable to create a physical implementation of the Adaptive Feedback Cancellation system due to the Covid-19 pandemic, we are unable to comment on the requirements 2.1.9 - 2.1.11.

8.4. Future Work

Full System Implementation

Due to time constraints, the full system containing the Adaptive Filter and the Postfilter [7, 8] has not been implemented at the time of writing. Creating the full system is of course the first action to be taken after this thesis has been completed. Important to note, however, is that the Decorrelation subsystem has been succesfully implemented in a simulation environment that is very similar to the final full system, indicating that the full system implementation should not pose too many issues.

Subsystem Improvements

The problem that occurs when pure silence is present in the signal can be solved in multiple ways. First of all, one could simply pause the Adaptive Filter estimation process when silence is detected. No signal will be propagated anyway at this time, so the estimation algorithm does not have any data to base the estimation on. Once sound enters the Adaptive Filter block again, the estimation can continue.

Another solution is to fill up the silence with some other sound, for instance white noise or sounds in frequencies that are imperceptible to the human ear. Research would need to be done in order to determine whether enough auxiliary signal can be added to remove the problem while retaining overall signal quality.

A final solution to the problem is to use a different Adaptive Filter estimation algorithm. The algorithm used for the testing in this report is the Normalised Least Mean Squares algorithm. Research would need to be done into algorithms that are able to handle pure silence in the signal. The Adaptive Filtering subgroup has indeed been able to find such an algorithm. However, due to time constraints we were unable to test our system with this algorithm.

Physical Implementation

As mentioned before, due to the Covid-19 pandemic present during the entire duration of the project, lockdown measures rendered us unable to create a physical implementation of the system. In this subsection we briefly discuss two options for what such an implementation would look like.

The first option is the implementation of the system on a dedicated microprocessor. The code created by all three subgroups would be combined and compiled, most probably to C code, and programmed onto a microprocessor. With the addition of a protective case, power supply and connection

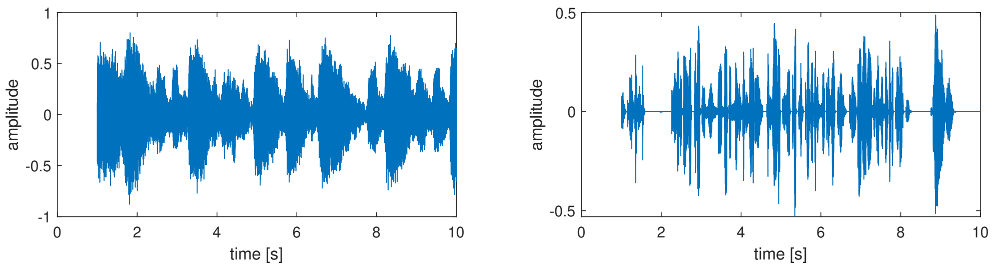

Figure 8.1: Caption

ports a simple plug-and-play system could be created than can easily be inserted into any PA system, as specified by requirement 2.1.1 in Chapter 2. By carefully selecting the components, all non-functional requirements (2.1.9-2.1.11) can then be satisfied.

The second implementation option is as a software package. This would make the system less easy to insert into a PA system, but it does allow for more flexibility. For example, one could add the software package to the existing software controlling the PA system. In this way no additional physical components would be necessary, reducing cost and increasing the lifespan of the product. Another advantage of this system is that the system as it has been created has been tested on PC's already, while the physical system on a microprocessor has not been implemented and tested.

A

Matlab code

A.1. Noise Injection

```
% Authors: Lucas Huijbregts and Merijn Jongepier
% Function: Implementation of Noise Injection variants:
%           Basic, Framed and Shaped Noise Injection

function out = noiseInjection(in, SNR, type, Fs, f_l)
in = in(:);
switch type
    % Basic Noise Injection
    case 'awgn'
        out = awgn(in,SNR, 'measured');

    % Framed Noise Injection
    case 'awgn_frames'
        N_f = ceil(length(in)/f_l);              % Amount of frames
        out = [];

        % Add White Gaussian Noise to each frame separately [using awgn()]
        for k = 1:(N_f-1)
            out = [out; awgn(in(1+f_l*(k-1):f_l*k) ,SNR, 'measured')];
        end
        out = [out; awgn(in(1+f_l*k:end) , SNR, 'measured')];

    % Shaped Noise Injection
    case 'masked'
        % Bark scale (see Bibliography)
        bark = [0 100 200 300 400 510 630 770 920 1080 1270 1480 1720 2000
            2320 2700 3150 3700 4400 5300 6400 7700 9500 12000 15500];

        % Create base noise
        n_base = awgn(in, SNR, 'measured') - in;
        N_base = fft(n_base);

        % Power and frequency
        Pin = abs(fft(in)).^2;
        Pin = Pin(1:end/2);
        f = linspace(0,Fs/2,length(Pin));

        % Bark indices
        c = (Fs/2)/length(Pin);
        bark_i = [round(bark/c) length(f)];

        % Bark scaling and spreading Bark bins over entire f−spectrum
        B = zeros(length(bark),1);
        scale = zeros(length(f),1);
        for n = 1:length(bark)
            B(n) = mean(Pin(bark_i(n)+1 : bark_i(n+1)));
            scale(bark_i(n)+1 : bark_i(n+1)) = B(n);
        end
        if(max(B) ~=0)
            mask = scale/max(B);                        % Scale so: max(B) = 1
        else
            mask = scale;
        end
        masked = [mask; flip(mask)].*N_base;     % Add negative frequencies
        out = in + real(ifft(masked));

    otherwise
        out = in;
end
end
```

A.2. Sinusoidal Phase Modulation

```
% Authors: Lucas Huijbregts and Merijn Jongepier
% Function: Implementation of Sinusoidal Phase Modulation

function out = sinPM(in, beta, f_m, Fs, T)
in = in(:);

if(mod(length(in),2)~=0)
    error('frame length is odd')                     % Only allow even length
end

% Generate Hilbert Transform of signal y
Y = fft(in);
L = length(Y);
Y((L/2)+2:end) = 0;
ya = ifft(Y);

% Generate phi, using T to assure continuity between Frames
t = linspace(0,(L−1)/Fs,L)' + T;
phi = beta*sin(2*pi*f_m*t);

% Apply modulation
out = ya.*exp(1j.*phi);

% Output one−sided fft [NEED 'symmetric' IFFT]
OUT = fft(out);
out = ifft(OUT,'symmetric');
end
```

A.3. Frequency Shifting

```
% Authors: Lucas Huijbregts and Merijn Jongepier
% Function: Implementation of Frequency Shifting
```

```

function out = FS(in, f_m, Fs, T)
in = in(:);

if(mod(length(in),2)~=0)
    error('frame length is odd')               % Only allow even length
end

% Generate Hilbert Transform of signal y
Y = fft(in);
L = length(Y);
Y((L/2)+2:end) = 0;
ya = ifft(Y);

% Generate phi, using T to assure continuity between Frames
t = linspace(0,(L-1)/Fs,L)' + T;
phi = 2*pi*f_m*t;

% Apply modulation
out = ya.*exp(1j.*phi);

% Output one-sided fft [NEED 'symmetric' IFFT]
OUT = fft(out);
out = ifft(OUT,'symmetric');
end
```

A.4. Half-Wave Rectification

```
% Authors: Lucas Huijbregts and Merijn Jongepier
% Function: Implementation of Half-Wave Rectification

function out = halfWaveRect(in, a)

out = in + 0.5*a*(in+abs(in));

end
```

A.5. Smoothed Half-Wave Rectification

```
% Authors: Lucas Huijbregts and Merijn Jongepier
% Function: Implementation of Smoothed Half-Wave Rectification

function out = halfWaveRectSmooth(in, a, c)

out = in + a*((in/2) + sqrt((in/2).^2 + c^2));

end
```

A.6. Square-Law

```
% Authors: Lucas Huijbregts and Merijn Jongepier
% Function: Implementation of Square-Law Operation

function out = squareLaw(in,a)

out = in + a*in.^2;

end
```

B

Mathematial Derivations

B.1. Overlapped Hann windows

When overlapping Hann windows, to make sure the resulting windows add up to a flat result, the length of the frames must be an integer multiple of the overlap, where both are expressed in samples. This can be seen when writing the windows as a sum:

$$W = \sum_{k=0}^{N_O-1} \sin^2 \frac{\pi(n + \frac{k}{N_O}N)}{N} = \sum_{k=0}^{N_O-1} \sin^2\left(\frac{\pi n}{N} + \frac{k}{N_O}\pi\right) \tag{B.1}$$

where N is the length of the frame in samples and N_O is the total amount of frames that overlap. So when the overlap is 50%, $N_O = 2$, when the overlap is 33%, $N_O = 3$, etc. N_O can be calculated as $N_O = \frac{1}{overlap}$ with the $overlap$ as a fraction. Using the trigonometric identity $\sin^2 x = \frac{1}{2} - \frac{1}{2}\cos 2x$, Equation B.1 can be rewritten to:

$$W = \sum_{k=0}^{N_O-1} \left(\frac{1}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{N} + \frac{k}{N_O}2\pi\right)\right) = \frac{N_O}{2} + \frac{1}{2}\sum_{k=0}^{N_O-1} \cos\left(\frac{2\pi n}{N} + \frac{k}{N_O}2\pi\right). \tag{B.2}$$

The sum can be rewritten using the sum formula for the cosine, giving:

$$S = \sum_{k=0}^{\frac{N_O}{2}-1} \left[\cos\left(\frac{2\pi n}{N}\right)\cos\left(\frac{k}{N_O}2\pi\right) - \sin\left(\frac{2\pi n}{N}\right)\sin\left(\frac{k}{N_O}2\pi\right)\right]. \tag{B.3}$$

Or:

$$S = \cos\left(\frac{2\pi n}{N}\right)\sum_{k=0}^{\frac{N_O}{2}-1} \cos\left(\frac{k}{N_O}2\pi\right) - \sin\left(\frac{2\pi n}{N}\right)\sum_{k=0}^{\frac{N_O}{2}-1} \sin\left(\frac{k}{N_O}2\pi\right). \tag{B.4}$$

Now both the first and second sum are equal to zero, which can be seen when considering the complex polynomial equation [26]:

$$z^n - 1 = 0 \tag{B.5}$$

where n is a whole, real number. The solutions of this equation are the complex roots of unity, and the sum of them equals zero:

$$1 + e^{\frac{2\pi i}{n}} + e^{\frac{4\pi i}{n}} + \dots + e^{\frac{2(n-1)\pi i}{n}} = 0. \tag{B.6}$$

Using Euler's identity, this becomes:

$$\left(1 + \cos\frac{2\pi}{n} + \cos\frac{4\pi}{n} + \dots + \cos\frac{2(n-1)\pi}{n}\right) + i\left(\sin\frac{2\pi}{n} + \sin\frac{4\pi}{n} + \dots + \sin\frac{2(n-1)\pi}{n}\right) = 0. \tag{B.7}$$

Equating the real and imaginary parts on both sides shows that both sums equal zero.

This shows that if the length of the frames is an integer multiple of the overlap, the windows sum to:

$$W = \frac{N_O}{2}. \tag{B.8}$$

C
Misalignment Improvement by Decorrelation

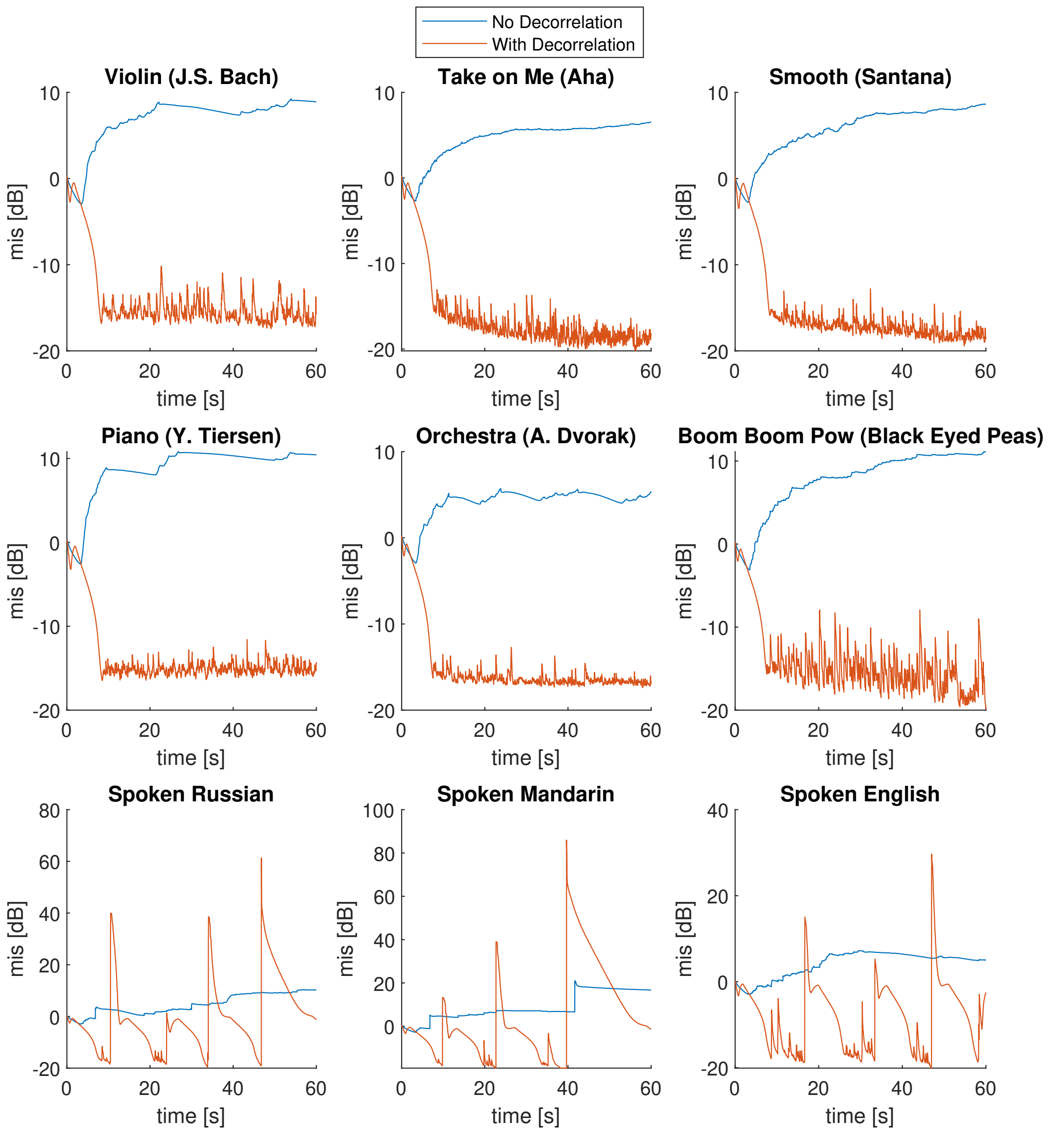

Figure C.1: Misalignment as a function of time over played audio clips. Both the misalignments with and without the Decorrelation Subsystem are shown.

D
Testing Signals

D.1. Audio Clips

For testing the decorrelation methods we selected nine different clips of audio: six different music styles and three different languages. Each of the clips is 10 seconds long. These are:

1. A clip of English speech;
2. A clip of Mandarin speech;
3. A clip of Russian speech;
4. A clip of piano music: Comptine d'Un Autre Été: L'après-midi, by Y. Tiersen;
5. A clip of solo violin music: Violin Partita No. 2 in D minor, by J.S. Bach;
6. A clip of orchestral music: Symphony No.9, "From the new World", by A. Dvorak;
7. A clip of instrumental rock music: Smooth, by Santana featuring Rob Thomas;
8. A clip of synth-pop music: Take On Me, by A-ha;
9. A clip of electronic music: Boom Boom Pow, by the Black Eyed Peas

D.2. Room Impulse Response

Figure D.1 shows the Room Impulse Response used in Chapter 7 to test the Decorrelation Subsystem in a simulation of the full system. The RIR is sourced from the Aachen Impulse Response Database [27] and was taken in their aula.

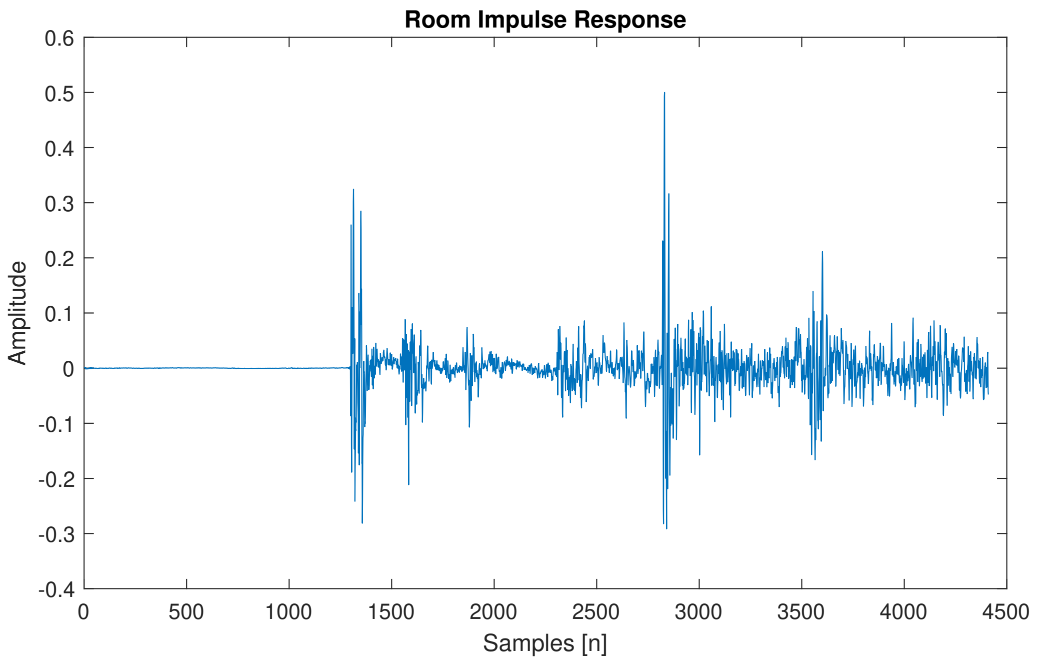

Figure D.1: Room Impulse Response as used for testing in this thesis.

Bibliography

[1] T. van Waterschoot and M. Moonen, “Adaptive feedback cancellation for audio applications,” *Signal Processing*, vol. 89, no. 11, pp. 2185 – 2201, 2009.

[2] B. C. Bispo, *Acoustic Feedback and Echo Cancellation in Speech Communication Systems*. PhD thesis, Dep. of Electr. and Comp. Eng. Faculdade de Engenharia da Universidade do Porto, Portugal, 2015.

[3] T. van Waterschoot and M. Moonen, “Fifty years of acoustic feedback control: State of the art and future challenges,” *Proceedings of the IEEE*, vol. 99, pp. 288–327, 2011.

[4] H.Nyquist, “Regeneration theory,” *Bell System Technical Journal*, vol. 11, pp. 126–147, 1932.

[5] M. R. Schroeder, “Improvement of feedback stability of public address systems by frequency shifting,” *J. Audio Eng. Soc*, vol. 10, no. 2, pp. 108–109, 1962.

[6] P.Svensson, *On reverberation enhancement in auditoria*. PhD thesis, Department Applied Acoustics, Chalmers University of Technology, Gothenburg, Sweden, 1994.

[7] C. H. Kos and M. C. Bekkering, “Adaptive filtering in automatic feedback detection and suppression for professional pa systems.” B.S. Thesis, Delft Univ. Technol., Delft, 2020.

[8] J. W. de Vries and C. Weustink, “Postfiltering in adaptive feedback cancellation for pa systems.” B.S. Thesis, Delft Univ. Technol., Delft, 2020.

[9] C. E. Shannon, “Communication in the presence of noise,” *Proc. IRE*, vol. 37, pp. 10–21, Jan. 1949.

[10] A. Mcpherson, R. Jack, and G. Moro, “Action-sound latency: Are our tools fast enough?,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 07 2016.

[11] T. van Waterschoot, G. Rombouts, and M. Moonen, “On the performance of decorrelation by prefiltering for adaptive feedback cancellation in Public Address systems,” in *Proc. 4th IEEE Benelux Signal Process. Symp. (SPS ’04)*, (Hilvarenbeek, The Netherlands), pp. 167–170, Apr. 2004.

[12] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[13] H. M. Barata J.C.A., “The moore–penrose pseudoinverse: A tutorial review of the theory.,” *Braz J Phys 42, 146–165 (2012)*, 2012.

[14] F. Strasser and H. Puder, “Correlation detection for adaptive feedback cancellation in hearing aids,” *IEEE Signal Processing Letters*, vol. 23, no. 7, pp. 979–983, 2016.

[15] K. Essafi and S. Ben Jebara, “A comparative study between different pre-whitening decorrelation based acoustic feedback cancellers,” in *2010 IEEE International Workshop on Multimedia Signal Processing*, pp. 1–6, 2010.

[16] H. L. E.M. Dekking, C. Kraaikamp and L. Meester, *A modern introduction to probability and statistics.* London, U.K.: Springer, 2010.

[17] T. Thiede, W. C. Treurniet, R. Bitto, C. Schmidmer, T. Sporer, J. G. Beerends, and C. Colomes, “Peaq - the itu standard for objective measurement of perceived audio quality,” *J. Audio Eng. Soc*, vol. 48, no. 1/2, pp. 3–29, 2000.

[18] P. Kabal, “An examination and interpretation of itu-r bs.1387: Perceptual evaluation of audio quality.” Evaluative report from the Department of Electrical & Computer Engineering at McGill University, 2002.

[19] D. Yan, R. Wang, X. Yu, and J. Zhu, “Steganography for mp3 audio by exploiting the rule of window switching,” *Computers & Security*, vol. 31, p. 704–716, 07 2012.

[20] J. D. Johnston, “Transform coding of audio signals using perceptual noise criteria,” *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 2, pp. 314–323, 1988.

[21] E. Zwicker, “Subdivision of the audible frequency range into critical bands,” *The Journal of the Acoustical Society of America*, vol. 33, no. 2, p. 248, 1961.

[22] D. R. M. Jacob Benesty and M. M. Sondhi, “A better understanding and an improved solution to the specific problems of stereophonic acoustic echo cancellation,” *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, 1998.

[23] J. L. H. Dennis R. Morgan and J. Benesty, “Investigation of several types of nonlinearities for use in stereo acoustic echo cancellation,” *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 6, 2001.

[24] A. Breitenbach, “Against spectral leakage,” *Measurement*, vol. 25, no. 2, pp. 135 – 142, 1999.

[25] R. B. Blackman and J. W. Tukey, “The measurement of power spectra from the point of view of communications engineering — part i,” *The Bell System Technical Journal*, vol. 37, no. 1, pp. 185–282, 1958.

[26] M. R. Spiegel, *Theory and Problems of Complex Variables*. New York, NY, USA: McGraw-Hill, 1974.

[27] M. Jeub, M. Schäfer, and P.Vary, “A binaural room impulse response database for the evaluation of dereverberation algorithms,” in *Proceedings of International Conference on Digital Signal Processing (DSP)*, (Santorini, Greece), pp. 1–4, IEEE, IET, EURASIP, IEEE, July 2009.