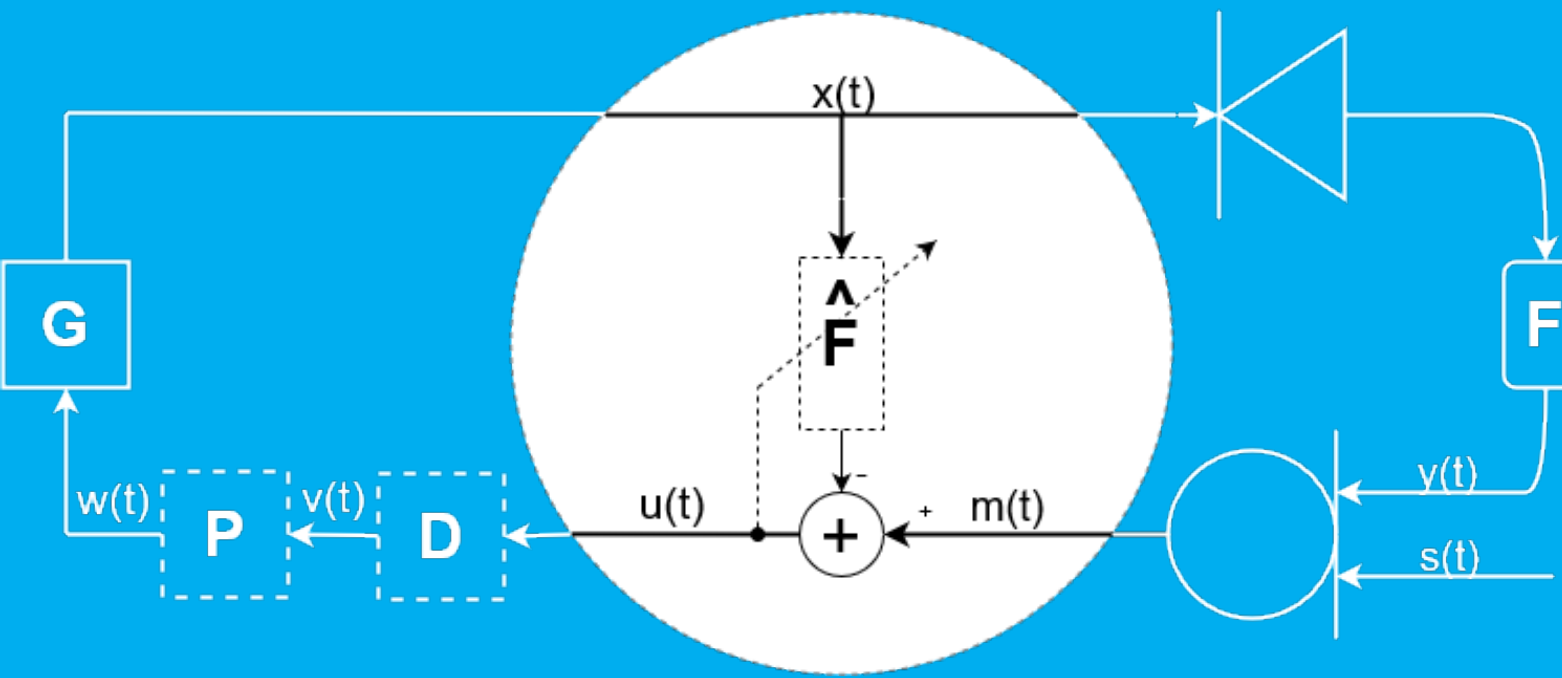
Adaptive filtering in adaptive feed-back cancellation for PA systems

Authors
Cees Kos
Mathijs Bekkering



TU Delft

Adaptive filtering in adaptive feedback cancellation for PA systems

by Cees Kos and Mathijs Bekkering

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

Student numbers:	4697030 (C.H. Kos)	
	4443683 (M.C. Bekkering)	
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

TU Delft

Abstract

This report is written as part of the Bachelor Graduation Project in the third year of the Electrical Engineering Bachelor programme at the Delft University of Technology. This report is made by a subgroup that is part of a larger project dedicated to finding a solution to frequency feedback in electroacoustic systems, also known as the Larsen effect.

This problem has a long history of literature and many solutions to this problem have been presented. For this project, the focus lies on the application of Adaptive Feedback Cancellation (AFC). This technique uses estimations of the Room Impulse Response (RIR) to minimise the feedback component. This report focuses on estimating such a RIR in the case of a PA system. Several methods will be discussed and compared by implementing and testing them in a simulation environment built in Matlab. The different methods are subjected to a set of performance measurements that provide the necessary information to see for each method and situation if and how the results measure up against a set of predetermined system requirements. From the researched algorithms, one is chosen as the one to be implemented in the final design of the group. At last, some considerations and recommendations are given for implementing one of the algorithms into software and hardware components.

Preface

This thesis was made as part of a bachelor graduation project, which is the final course of the Bachelor Electrical Engineering Programme at the TU Delft University of Technology. The goal was to realise an implementation to stop the howling effect known in public address systems.

Firstly, we want to thank dr. John Schmitz for providing the idea for the project. Secondly, we thank dr.ir. Richard Hendriks and dr. Jorge Martinez for being our supervisors for this project and being able to guide us through the course of the project, as well as Metin Çaliş, who acted as our daily supervisor and helped us with some of the problems and questions that came up during the project.

The members of our group were all interested in working with electroacoustic systems, which made this project perfect for us. When reading the document we hope to give insight to the problem, the aspects of a solution and how we as a team confronted the task resulting in our implementation of the adaptive filter design that fits into the complete system.

Cees Kos and Mathijs Bekkering
Delft, 19-06-2020

Contents

1

Introduction

Acoustic systems are used on a large scale. Such systems make use of both recording devices and playback devices. Systems like that can be found in different forms, from mobile phones to public address (PA) systems used at festivals and concerts. These systems are of a different scale, but represent the same. That makes that they also have a problem in common, one that holds for any system that uses interconnected microphones and loudspeakers. A problem called frequency feedback. This problem describes the situation in which sound from the loudspeaker is fed back to the microphone and played again. When certain components of the signal are amplified with a gain larger than unity, a loud and irritating sound occurs, referred to as 'howling' or 'ringing'. This effect is also known as the Larsen effect, after Danish scientist Søren Absalon Larsen, who first discovered the idea behind the problem of frequency feedback.

1.1. Frequency Feedback

Frequency feedback occurs when, in a system that contains at least one recording device and one loudspeaker that are interconnected, create a closed loop, in which the sound from the loudspeaker is fed back to the recording device. For example in a lecture room where the lecturer has a microphone and the speakers reflect from the wall back to the lecturer. In this closed loop, several gains are identified, which are the circuit gain and the free space gain. The circuit gain includes the frequency characteristics of the microphone and the loudspeaker, as well as alterations made to the signal by means of, for example, a mixing setup. The free space gain is also referred to as the room impulse response, or RIR, and indicates how much the frequency components attenuate when fed back. In most spaces, the room impulse response is commonly expected to consist of mostly indirect reflections from walls, obstacles and people.

Figure 1.1: Examples of acoustic equipment. When interconnected, they all experience frequency feedback.

1.2. State of the Art in Acoustic Frequency Feedback

The problem of frequency feedback is a longstanding problem, with publications dating back as far as the 1960's.

Frequency and Phase Shifting (FS and PS) [1] [2] were one of the first solutions to the acoustic feedback problem, with articles dating back to the 1960's. This concept uses phase modulation techniques that shifts the phase and/or frequency of the signal slightly, to shift the rise in amplitude to another frequency, which prevents the components from building up. Doing so smooths the loop gain and makes the maximum stable gain (MSG) depend on the average gain magnitude instead of the peak gain magnitude [2]. An increase of the MSG margin of 14 dB was reported, but for audible after-effects of the FS operation, this is limited to 6 dB [2]. Because FS shifts the frequency components of signals, relative relations between frequency components can change, which is not desired in the case of speech and music signals [2]. A second group of methods makes use of adaptive notch filters [3], called notch-based howling suppression (NHS) [2]. The goal of these methods is to detect howling and use adaptive filters to reduce the gain around the critical frequencies. These filters are by nature reactive, but efforts have been made make proactive versions [4]. Another set of solutions is grouped under the term of Adaptive Feedback Cancellation (AFC). These methods try to estimate the feedback component and subtract it from the received signal. These are generally effective methods and many algorithms exist using this principle [5]. The estimation, however, has a certain bias towards the source signal, which is to be kept. Such a bias is counteracted by accompanying the adaptive filter with a decorrelation block.

1.3. Synopsis

For this project, the AFC method was chosen for reasons that are explained in Ch. 2 , since it is said that FS, PS and NHS do not have much space to work with in terms of increasing the maximum stable gain (MSG), whereas AFC does [2]. There is a lot of freedom in how one can make such an adaptive filter. That is why the group decided on AFC. For this project, the group was split into three subgroups, each of them working on a different part of an AFC filter, which are the Adaptive Filtering, Decorrelation [6] and Postfiltering [7].

Adaptive Filtering includes estimating the RIR and building a cancellation filter based on the RIR, Decorrelation handles decorrelation of the signals coming from the loudspeaker with the signals received by the microphone and Postfiltering is about smoothing of residual peaks that the adaptive filter couldn't fix.

This report will discuss, as mentioned before, the Adaptive Filtering part. In Ch. 2 of the report, the problem at hand is explained in more detail, to give you a good sense of the situation. System requirements that are aimed at to meet are listed in Ch. 3. In chapter 4 a state of the art of existing methods and algorithms to estimate the room impulse response (RIR) will be provided, to analyse and compare the methods proposes in literature. Promising methods will be researched and eventually compared with each other to determine which method will be chosen to be implemented in accordance to our specifications, which is done in. Underlying theory about the different methods will be given to support the choices that are made. In Chapter 5 A simulation setup will be discussed and used to analyse the performance of the methods in multiple scenarios, which will be further discussed in chapter 6 the discussion. At last, a number of considerations and advice will be given on creating implementations based on our work. The report is closed with a final conclusion in Ch. 8.

2

Problem Description

As said before, the acoustic feedback cancellation method can be divided into three parts: Adaptive Filtering, Decorrelation and Post-filtering, with Adaptive Filtering being discussed in this thesis. First, a single-channel system without any feedback cancellation is considered. This system consists of a single microphone and loudspeaker. From the microphone to the loudspeaker is an interconnection, possibly going through a mixer or a computer of sorts, of which the combined gain is given by G. Let $x(t)$ be the signal to be send by the loudspeaker. The signal is sent out and modulated through the room impulse response (RIR) F denoted by signal $y(t)$ combined with the sound signal, for example of somebody speaking into the microphone, $s(t)$ back to the microphone creating a closed loop signal $m(t)$ due to feedback. A system without feedback would be considered to have open loop gain as the output has no influence. A schematic of the closed loop system is given in Fig. 2.1. From this figure, the closed loop gain without AFC can then be given as:

$$H(s) = \frac{G}{1 - GF} \tag{2.1}$$

where $H(s)$ is the transfer from $s(t)$ to $x(t)$.

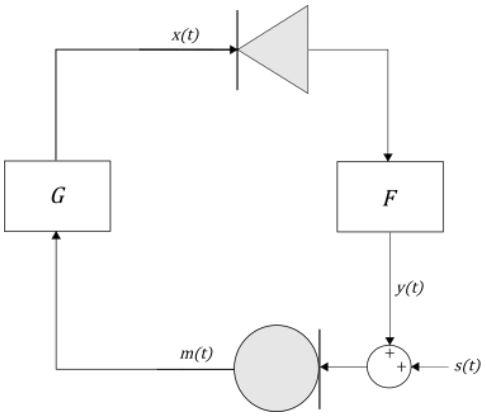


Figure 2.1: Schematic of frequency feedback in acoustic systems. G is the circuit gain, F is the RIR

The idea of adaptive filtering is to add an extra block between the microphone and loudspeaker, as shown in Fig. 2.2 as $\hat{F}$. This block is added, together with blocks for decorrelation D [6], which decorrelates the feedback $y(t)$ and microphone signals $s(t)$ needed for a good estimation of F, and postfiltering P [7], to improve the sound quality after feedback reduction and stabilise the system when howling still occurs. With these additions to the system the closed loop gain changes to

$$H(s) = \frac{DPG}{1 - DPG(F - \hat{F})} \tag{2.2}$$

The goal now is to make an as accurate as possible estimation of F and use it to cancel out its effects by subtracting an expected feedback signal $\hat{y}(t)$ which is the convolution of $x(t)$ and $\hat{F}$. The expected signal without feedback $u(t)$ is further used in the system and amplified. It is seen from equation Eq. 2.2 that when the estimation of F becomes more accurate, the influence of the feedback becomes less, eventually nullifying the effects of the closed loop and leaving only the circuit or open-loop gain in the ideal case, which is when $\hat{F} = F$.

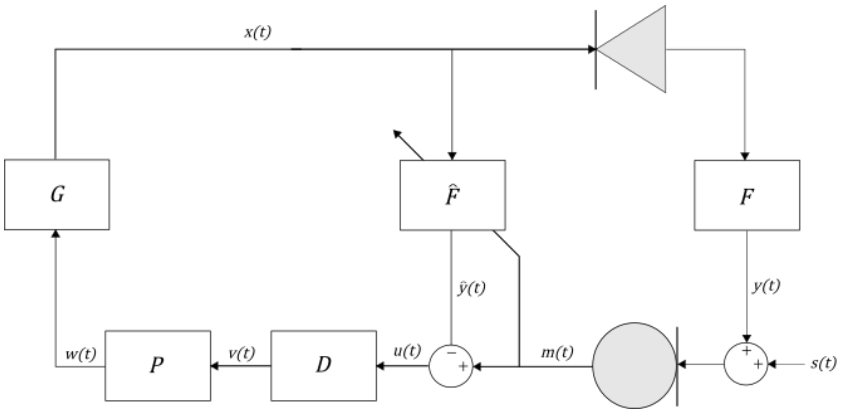


Figure 2.2: Schematic of frequency feedback in acoustic systems. G is the circuit gain, F the RIR, $\hat{F}$ the frequency cancellation filter, D the decorrelation block and P the postfiltering block.

2.1. Least Squares Estimate

One of the most basic estimations, which also forms a basis for nearly all linear adaptive filtering algorithms, is the Least Squares (LS) estimation. For estimation of a filter $\mathbf{f}$ given as

$$\mathbf{f} = [f_0 \; f_1 \; \ldots \; f_M]^T \tag{2.3}$$

the LS estimate of the filter is given by

$$\hat{\mathbf{F}}(t) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{m} \tag{2.4}$$

where $\hat{\mathbf{f}}$ is the filter estimate, $\mathbf{m}$ is the output signal of the filter received by the microphone and $\mathbf{X}$ is defined as

$$\mathbf{X} = [\mathbf{x}(t) \; \mathbf{x}(t-1) \; \ldots \; \mathbf{x}(1)]^T \tag{2.5}$$

with

$$\mathbf{x}(t) = [x(t) \; x(t-1) \; \ldots \; x(t-M)]^T \tag{2.6}$$

for a discrete time range $[1, t]$.

A problem often encountered with this method, is that due to poor excitation of the input signal $x(t)$, the matrix $\mathbf{X}^T\mathbf{X}$ can be ill-conditioned or even singular, also called *ill-posed* problems. [8]. This makes it hard to do computations with its inverse, which is done in Eq. 2.4. Situations like this cause a large variance in the estimation $\hat{\mathbf{F}}$ of the RIR [8].

2.1.1. Bias

What is important to consider is the fact that solutions based on the LS estimate introduce a bias into the estimate, following from

$$E\left\{\hat{\mathbf{F}}(t)\right\} = \mathbf{F}(t) + E\left\{(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{s}\right\}$$

where $E\{\cdot\}$ is the expectation operator. The rightmost term is said to be nonzero in the case of systems with feedback. This causes the source and loudspeaker signals to be correlated, thus introducing the bias given as

$$\text{bias}\left\{\hat{\mathbf{F}}(t)\right\} = E\left\{(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{s}\right\} \neq \mathbf{0}$$

This bias needs to be reduced to estimate the RIR more accurately. This problem is expected to be mainly solved by the Decorrelation subgroup of this project.

2.2. Maximum stable gain

The howling effect occurs when the loop gain is too large, which can be traced back to the Nyquist stability criterion [2] [9] for closed loops, where we include the effects of $\hat{F}$ and exclude those of D and P, which is given as

$$\left|G(\omega,t)\left[F(\omega,t)-\hat{F}(\omega,t)\right]\right| \geq 1 \tag{2.7}$$

$$\angle G(\omega,t)\left[F(\omega,t)-\hat{F}(\omega,t)\right] = n2\pi, \quad n \in \mathbb{Z} \tag{2.8}$$

The maximum stable gain (MSG) follows as

$$\text{MSG}(t)[\text{dB}] = -20\log_{10}\left[\max_{\omega\in\mathcal{P}_{\hat{F}}}\left|J(\omega,t)\left[F(\omega,t)-\hat{F}(\omega,t)\right]\right|\right] \tag{2.9}$$

with

$$\mathcal{P}_{\hat{F}} = \left\{\omega \middle| \angle G(\omega,t)\left[F(\omega,t)-\hat{F}(\omega,t)\right] = n2\pi\right\} \tag{2.10}$$

and $J(\omega,t)$ relates to G as

$$G(\omega,t) = KJ(\omega,t) \tag{2.11}$$

This shows that if $F = \hat{F}$, the MSG should be infinite. For this project, it is assumed that the gain G is the same for all frequencies, which means that $J(\omega,t) = 1$.

2.3. Filter size

The size of the estimated filter depends the time required to properly catch the effect of the room impulse response. A longer filter length is required when the main components of feedback have to travel long distances. A filter time $t = 100$ ms will be assumed to capture enough information about the RIR. A sampling frequency $F_s = 40$ kHz is needed to include the required frequency range. In literature, this is often increased to $F_s = 44.1$ kHz, which will be used in this project as well. Fact is that the same sample frequency is used for CD's. The needed filter size is then given by $M = F_s \cdot t$ resulting in a filter order of $M = 4410$ weights. Large filter orders do come with disadvantages of more computational complexity and possible lower convergence speed.

3

System Requirements

The goal of the project is to create a software implementation of an acoustic feedback cancellation method.

3.1. Group System Requirements

- The system should consist of a program that can be added into a sound system loop and should operate independently of the type of hardware used.
- The system is meant for speech and audio, which means the operating frequency range should be 20 Hz - 20 kHz
- It should increase the Maximum Stable Gain (MSG) by at least 10 dB. The MSG is the maximum gain the circuit can deliver over the input signal before the system becomes unstable. The increase of MSG (ΔMSG) is the increase in gain due to the implemented improvements.
- It should operate automatically without human interaction, besides setting up and starting the system.
- The system should be able to sustain the throughput of incoming and outgoing audio data, not creating gaps between frames. With a frame delay within 50 ms to avoid noticeable or distracting delays in the sound. [10]
- The sound after being altered by the system should still be intelligible.
- It should work with a single channel setup.
- It should work for both music and speech signals.

3.2. Adaptive Filter Estimation System Requirements

- The program should be quick enough in computation to not block other needed operations that are in the loop and interfere with the delay requirement of the complete system
- The estimated filter should converge to stability.
- While converging the filter should not introduce noticeable signal changes due to it not being at the right estimation, for example increasing the feedback temporarily.
- The estimated filter should, when stable, have a misalignment equal to or smaller than -5 dB
- The filter subsystem itself should give an increase of the maximum stable gain ΔMSG of at least 10 dB.
- The system should be stable with multiple speech and music samples for 30 seconds.
- With inputs $x(t)$ and $m(t)$ the program has to pass a signal $u(t)$ which is the microphone input with the estimated feedback subtracted.

4

State of the Art

As mentioned before, the problem of estimating an unknown system is a longstanding problem. Over time, different kinds of solutions have been presented, most of these also having different ways to implement them. There are several groups of methods. The first group is based on minimising the least squares error. The second group is based on minimising the gradient of the error. The third group is the use of Cepstral analysis to estimate the filter, which has recently been proposed, but due to it being very distinct from the other solutions and being rather complex with limited information, this method will not be considered. This leaves a set of more similar algorithms, which are discussed below.

In the following algorithms $m(t)$ denotes the desired signal the filter output should converge to and $x(t)$ the input signal.

4.1. RLS and Fast RLS

To solve the so-called ill-posed problems, mentioned in section 2.1, in which singular matrices are used, regularisation is applied. Regularisation techniques make ill-posed problems into more well-posed ones. Most often seen is the method derived by Tikhonov et al. [11], which adds a scaled identity matrix to the matrix $\mathbf{X}^T\mathbf{X}$. Using this method, an algorithm can be derived called the Recursive Least Squares (RLS) algorithm [8]. Just as the normal LS algorithm, the RLS tries to estimate the coefficients of the RIR $\mathbf{f}$.

RLS is a high-performance algorithm, often at the cost of higher complexity. There exist, however, fast versions of the algorithm. A benefit of RLS is being able to converge very fast [5].

In Horita et al. [12], a method is proposed that uses the normal RLS algorithm, but with the addition of a so-called *forgetting factor*. This gives relatively more weight to more recent samples.

The RLS algorithm [5] is given by A.1, where $\mathbf{R}(t)$ is the time-average correlation matrix of the input signal, $\mathbf{x}(t)$, $\hat{\mathbf{f}}(t)$ is a vector with the estimated filter weights, $y(t)$ is the desired output signal and $\mathbf{c}(t)$ is the *Kalman gain vector*. In an ideal, noise-free case, the RLS algorithm would converge exactly, but since there is the initialisation of the matrix $\mathbf{R}(-1) = \delta\mathbf{I}$ and the vector $\hat{\mathbf{f}}(-1) = \hat{\mathbf{f}}_0 = \mathbf{0}$ gives a certain bias to the estimate [5]. The complexity of RLS algorithms is of order $O(M^2)$.

It is shown in literature, that the previously mentioned Kalman gain vector can be updated in so-called 'fast' schemes [13]. To do so, the FRLS algorithm uses a mechanism using forward and backward predictions. This gives the order of the complexity as $O(7M)$. This means that while having the same properties of the RLS algorithms, its complexity is made comparable to that of the LMS algorithm, which is discussed later.

A drawback of FRLS is the fact that it has poor error performance, caused by the innate instability of the algorithm [13]. There has been quite some research into the instability of the algorithm and some successful solutions have already been presented. These solutions use specific variables and try to estimate these variables as part of a feedback system. These variables lead to another set of variables that include parameters that are chosen through careful selection [13]. Another drawback is the fact that there exists a constraint on the factor λ, which negatively affects the convergence rate and tracking ability of the algorithm [13].

4.2. Stochastic gradient descent methods

A large portion of the available algorithms are based on the stochastic gradient descent method. Convergence is dependent on a step size μ in the direction of the steepest descent of the mean squared error. These methods commonly only use the current error and can categorised as posteriori error algorithms.

The least mean squares (LMS) algorithm, given in A.2, uses a constant step size independent of the signal power. The Normalised LMS (NLMS) algorithm, given in A.3, is a variant that solves the problem of LMS for choosing a step size that is stable by normalising with the input to cancel the effects of signal power deviation [14].

Block processing for adaptive filter design was studied by Clark et al. where the convergence was found to be the same as for continuous LMS [15]. Further reductions in computational complexity can be achieved by realising the LMS algorithm in the frequency-domain (FLMS) and were shown to achieve the optimum filter weights[16]. The advantage of block implementations is that the new weights are only calculated once per block thus reducing complexity.

There are many variants of the Frequency Domain Adaptive Filter (FDAFs). J.J.Shynk [17] gives an overview of several FDAFs. FDAFs generally have lower computational complexity and a higher convergence rate than the time domain alternatives. Circular convolution FDAF, given in A.4, was found to be the most computationally efficient of the single window approaches. Dividing the frame in frequency sub-bands can also improve decorrelation and computational complexity.

MDF (Multi Delay block FDAF), as given in A.6, reduces the size of the FFT blocks by buffering multiple delayed frames. Using multiple smaller FFT blocks reduces the block delay and computational complexity by calculating smaller FFT's for a longer filter estimation order. The convergence speed is, up to a certain extend, increased with the number of blocks. Its complexity is

$$O((4D+6)log_2(N)+8D-(4B+6)log_2(D)$$

where D is the number of blocks and N the frame length which is half the length of the FFT [18].

4.3. APA and Fast APA

A generalised form of NLMS and RLS exists, called the Affine Projection algorithm (APA). As mentioned in [13], APA can be seen as an algorithm that estimates the filter by minimising the error norm

$$c_M(n) = \underset{c_M}{\operatorname{argmin}} \|c_M - c_M(n-1)\|^2$$

with use of the constraint given as

$$Y_L(n) = X_{M,L}^T c_M$$

This algorithm was introduced in [19] as an improvement of NLMS. In there most common form, APA has poor performance when there is noise present in the output signal or if the input signal is non-stationary, as is for most cases of acoustic feedback cancellation. Countermeasures for this are regularisation and using the time-average correlation matrix of the input signal in the algorithm, which makes the APA resemble the RLS algorithm [5].

Fast implementations of APA are of an order comparable to LMS and are not very sensitive to computational noise due to the use of predictions [5].

Fast APA uses the shift-invariant property of the input signal vector and the error vector in combination with derivations done for the Fast RLS algorithm in [13]. From the sliding-window Fast RLS algorithm, back and forward prediction vectors can be computed and used to estimate the variable vector. In Fast APA, this is the error with the unknown filter, rather than the estimated coefficients of the filter [13]. A derivation of the Fast APA is found in [20].

4.4. Frequency-domain Kalman Filter

An adaptation to the Kalman filter for implementation in echo cancellation was successfully developed in the frequency domain (FDKF) [21]. It is based on the use of a first-order Markov process that models the behaviour of the acoustic path which is assumed to change slowly. Further research on the use, performance and improvements for the Kalman filter in acoustic feedback cancellation are very recent

but show promising results [22] [23] without increase of complexity due to the simplicity of the underlying statistical model [21]. Obtaining said underlying models would require additional research that falls outside of the scope of this thesis.

4.5. Comparison

RLS and APA have a much higher complexity than the other options if the filter order is increased and are not viable as high filter orders are required. Most of the used methods are based on the LMS algorithm which is well known and researched. Frequency domain analysis has advantages over time domain due to the options it gives like sub-band analysis and partitioned blocks resulting in FDAF and MDF with reduced complexity due to efficient use of the frequency domain. Cepstral Analysis and the Frequency domain Kalman Filter approaches are not feasible due to their high complexity, as mentioned before.

MDF complies with the specifications as it is efficient with memory, converges to stability, reaches a good estimation and is not computationally complex with a low delay, which makes it the most promising method to use. Tab. 4.1 summarises the options discussed.

Algorithm	Complexity	Stability	Convergence speed
LS	$O(M^3)$	Good	Instant
LMS	$O(2M)$	Poor	slow
NLMS	$O(3M)$	Good	okay
RLS	$O(M^2)$	Good	quick
FRLS	$O(8M)$	Poor	quick
APA	$O(M^2)$	Good	quick
Fast APA	$O(2M + 21P)$	Good	quick
FLMS	$O(10log_2(M) + 8)$	Good	slow
Circular FDAF	$O(3Mlog_2(M) + 8M)$	Good	okay
MDF	$O((4D + 6)log_2(M) + 8B - (4D + 6)log_2(D)$	Good	okay
FDKF	Similar to MDF	Good	okay

Table 4.1: Algorithm characteristics summary. It holds that $P < M$. and D is the number of delay blocks used in the MDF algorithm.

5

Testing & Results

To check the insights and knowledge gained from literature, all methods that are deemed feasible were implemented. Methods that were considered feasible were LMS, NLMS, Circular FDAF and MDF. It was decided not to implement the Fast APA, Fast RLS and Frequency-domain Kalman Filter methods, because of the extent of the theory behind it. APA and RLS were considered unfeasible options, since their high complexity would make them very slow and not satisfy our requirements by blocking continues audio throughput. It was decided that, to match the scope of this project, it would be better to implement multiple smaller algorithms, giving this project a wider approach.

To test and simulate the different methods and algorithms, a simulation environment was built in Matlab, for this subgroup in specific for the easier tracking and control of computations. For the project group Simulink is used for the time based simulation approach.

In the following chapter, powers of 10 are denoted by E, i.e. $2E^{-4}$ means $2 \cdot 10^{-4}$.

5.1. Test Setup

Two simulations are used to test potential algorithms and their performance. The first simulation is an adaptive filter estimation without feedback, therefore creating a open loop system and is commonly used as the setup used to evaluate adaptive filter designs. The mainly used setup for this is illustrated in Fig. 5.1 where the filter is adapting to minimise the error between desired and input. Input is generated by a convolution of a desired input signal (audio, speech or noise) with the filter F that has to be estimated. The simulation is setup in sample time steps of $T_s = 1/F_s$. The estimations of the filter are done in buffered frames to simulate the case when a digital audio processor is used that outputs frames.

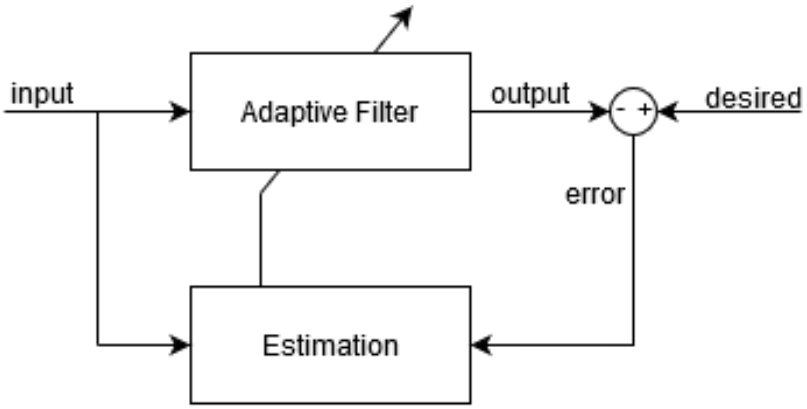


Figure 5.1: Schematic of test setup for general adaptive filter estimation

The second simulation is implemented with feedback to simulate the actual problem at hand and creates a closed loop system. Fig. 2.2 shows the setup for this situation, although for the following

tests, D and P are excluded. The adaptive filter estimation and output function is implemented with the input signal $x(t)$ and the desired signal $m(t)$. The output is the estimated feedback from $x(t)$ in $m(t)$, denoted by $\hat{F}$.

5.1.1. Assumptions

In the feedback simulation maximum signal strengths are implemented. Typical conversion rates for microphones are used to create a base voltage reference. Using the specifications of a typical podium microphone [24] with a sensitivity of 20 mV/Pa. Normal conversation voice sound level is 70 dB [25] which is converted to $s(t) = \pm 1.25$ mV, and maximum microphone range is set to 102 dB or ±50 mV for $m(t)$ and $\hat{y}(t)$ and a maximum of 130 dB or ±1.25V which is the pain threshold for sound for the loudspeaker signal x(t) is used as a maximum. The conversion from pascal to decibel is done by

$$P_{dB} = 20 \cdot log10(P_{Pa}/0.00002)$$

where P_{dB} is the sound pressure in decibel and P_{Pa} the sound pressure in pascal. Then, the voltages can be calculated as

$$U_V = 20 \cdot 10^{-3}(V)$$

It is expect that no unknown delay exist between $u(t)$ and $x(t)$, any delay that is added will be known. From the decorrelation group a delay of 2000 samples at $F_s = 44.1$ kHz has been implemented. A delay is implemented between $u(t)$ and $x(t)$ to simulate the delay effect of the decorrelation sub-group.

The system is only aware of the inputs $m(t)$ and $x(t)$ and no other setting will be supplied to the estimator.

5.1.2. Room impulse response

The room impulse response will be assumed to have typical RIR characteristics. The estimated room impulse response will have a length of $M = 4410$. A measured RIR from the Aachen Impulse Response Database [26] is used as a real example of an aula.

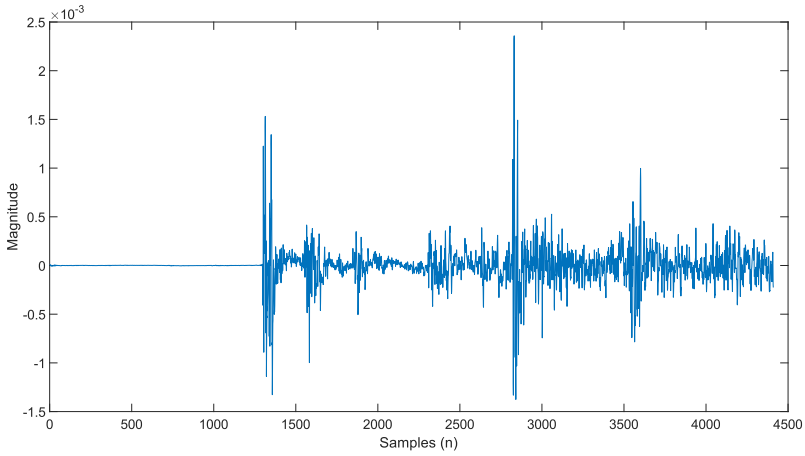


Figure 5.2: Room Impulse Response used in simulations of 4410 samples

5.1.3. Input Signals

One audio sample of 30 seconds will be used. The sound fragment is a cut from *Take On Me* from A-ha [27] where the cut is made to have the singing start at 15 seconds from the beginning of the audio sample.

5.1.4. Procedure

The selected algorithms will be tested first in the open loop simulation. Variations in step sizes will be made by multiplying or dividing by 10. Misalignment, Maximum stable gain and estimation error will be used as will be discussed in Sec. 5.2. The results are used to create a baseline of expected

performance, viable parameters and exclude methods that under perform or of which the calculation time will be to high.

The second simulation is used to test the performance in a feedback environment. Under performing algorithms from the first test will be excluded. Gains of 1, 10, 25 and 100 will be used to analyse the performance of the methods.

5.2. Analysis criteria

Three criteria will be considered to analyse the achieved performance of the algorithms in the suggested simulation setups. These are the misalignment,maximum stable gain (MSG) and the estimation error.

5.2.1. Misalignment

The performance of the system is highly related to the estimation of the adaptive filter. The misalignment mis(k) with k the frame number is calculated as [28]

$$\text{mis}(k) = \frac{\|f - \hat{f}\|}{\|f\|} \tag{5.1}$$

5.2.2. Maximum stable gain

To get an indication of the MSG a different approach was used from Sec. 2.2 due to the lack of critical frequencies the simulation setup. The simplified approach from B. Bispo [28] is used instead.

$$\begin{aligned} \varepsilon(\omega, n) &= |F(\omega, n) - \hat{F}(\omega, n)| \\ \varepsilon(n)[dB] &= 20\log_{10}[\text{max}\varepsilon(\omega, n)] \\ \text{MSG}(n) &= -20\log_{10}[\text{max}\varepsilon(\omega, n)] \end{aligned}$$

5.2.3. Estimation Error

A simple method that was implemented was taking the relative error between the feedback signal $y(t)$ and the estimated feedback signal $\hat{y}(t)$ or the desired signal and estimated signal in the no feedback simulation.

$$E(t) = 10\log_{10}(|y(t) - \hat{y}(t)| - 10\log_{10}(|y(t)|) \tag{5.2}$$

where $E(t)$ is the estimation error.

Local differences come up pretty fast with this method, which is why there was mostly looked at the mean of this error. An average of the errors can be used to quantify the performance.

5.3. Test Results

In the following section, the results of the simulations done are presented in the form of tables, with the methods, simulation parameters and results displayed. Essentially, the only parameter that is adjusted is the step size. Step sizes larger or smaller than the listed values were found to be unstable, show no improvements or even decreased the performance.

5.3.1. Computation time

The simulations are performed in Matlab on a personal computer. The computation times in Tab. 5.1 and 5.2 are factors relative to the lowest method computation time of 1.5ms to indicate the differences in computational complexity. The actual computation time will dependant on the hardware.

5.3.2. Open Loop Simulations

Based on the simulation results shown in Tab. 5.1 it can be seen that a low misalignment of -7.04 dB can be achieved by NLMS with a step size of $\mu = 1$. This however does not reflect immediately in a low estimation error or high MSG compared to lower step sizes. Fig. B.2 shows that for smaller step sized the misalignment reduces slower but do result in a smaller estimation error whilst converging, this is supported by the average error of $\mu = 0.1$ which is higher. Although NLMS shows good results, its computation time is certainly a factor that has to be taken into account.

The lowest error and largest MSG are both achieved by LMS at a step size of $\mu = 1\text{E}^{-3}$. Notable is that increasing the step size breaks the trend of increasing performance, resulting in a unstable system of which the results are excluded from the table.

Circular FDAF shows a lower misalignment than MDF but has a higher average error. In Fig. B.3 it can be seen that for longer simulation the algorithm does not seem to converge further and for higher step size is unstable even. Circular FDAF will not be tested in a feedback loop due to the long computation time and does not meet the requirements.

MDF shows the best results in the case $D = 32, \mu = 1\text{E}^{-6}$ and continues improving in time as seen in Fig. B.5, B.6 and B.4. As expected the performance improves for more delay blocks except for $D = 1$. The computation time increases in the test setup for more delay blocks, this might be caused by the program not having a noticeable difference for shorter FFT lenghts.

Without feedback LMS has the highest MSG, NLMS the lowest MIS and MDF the lowest computation time.

Method	**Step size** μ	**Avg. MIS** [dB]	**Avg. MSG** [dB]	**Avg. error** [dB]	**comp. time factor**
LMS	0.001	-5.86	36.58	-13.77	100
	1E^{-4}	-1.93	29.41	-6.93	
NLMS	1	-7.04	34.15	-9.80	200
	0.1	-4.91	34.81	-12.42	
	0.01	-1.39	28.47	-5.94	
	0.001	-0.33	26.68	-3.17	
Circular FDAF	0.01	27.00	-44.92	27.48	666
	0.001	-1.49	27.77	-1.50	
	1E^{-4}	-1.24	27.95	-1.19	
MDF($D = 1$)	1E^{-5}	-0.70	27.25	-4.46	1
	1E^{-6}	-0.16	26.29	-2.10	
	1E^{-7}	-0.03	26.00	-0.39	
MDF($D = 2$)	1E^{-5}	-0.19	18.18	0.67	2
	1E^{-6}	-0.25	26.52	-2.81	
	1E^{-7}	-0.05	26.00	-0.71	
MDF($D = 4$)	1E^{-6}	-0.39	26.23	-3.50	4
	1E^{-7}	-0.08	25.87	-1.20	
MDF($D = 8$)	1E^{-6}	-0.60	26.00	-4.24	8
	1E^{-7}	-0.14	25.48	-1.87	
MDF($D = 16$)	1E^{-6}	-0.93	26.60	-5.11	16
	1E^{-7}	-0.22	25.55	-2.60	
MDF($D = 32$)	1E^{-6}	-1.44	27.66	-6.21	32
	1E^{-7}	-0.34	25.69	-3.29	

Table 5.1: Simulation results without feedback. With MDF, D indicates the amount of delay blocks. The computation time is given as a factor w.r.t. 0.0015s, which is the lowest achieved loop time.

5.3.3. Closed Loop Simulations

The results of the simulations with feedback are shown in Tab. 5.2. In the table, a row showing the results of a simulation without any feedback cancellation is given, which is used as reference. The highest achieved MSGs are reached by MDF for $D = 1, 2$ and LMS for $\mu = 1\text{E}^{-3}$, both for a gain of $G = 25$. NLMS shows the best filter approximation in terms of the misalignment, with an average misalignment of -7.04 dB, but despite that does not achieve the lowest estimation error or highest MSG.

Method	**Gain** [dB]	**Step size** μ	**Avg. MIS** [dB]	**Avg. MSG** [dB]	**Avg. error** [dB]	**Comp. time factor**
Without Cancellation	1	-	0	25.964	-7E^{-4}	-
	10	-	0	25.964	0.1407	
	25	-	0	25.964	23.2629	
LMS	1	0.1	0.4114	22.6667	2.0061	100
		0.01	8.7244E^{-4}	25.9587	-0.0163	
		0.001	-4.9626E^{-4}	25.9635	-0.0097	
	10	0.1	0.1546	22.2881	2.4654	
		0.01	-0.0844	26.0673	0.0924	
		0.001	-0.0370	25.9769	-0.0097	
	25	0.1	NaN	NaN	4.2604	
		0.01	-0.8585	28.9277	-1.2242	
		0.001	-0.3120	28.0037	-1.1332	
NLMS	1	1	12.0867	-5.1927	17.7868	200
		0.1	10.1755	-3.8759	17.0771	
	10	1	3.0644	13.8862	8.2300	
		0.1	1.2508	15.5903	7.3737	
	25	1	-0.6401	23.1410	3.7690	
		0.1	-1.7930	23.5735	3.3179	
MDF($D = 1$)	1	1E^{-3}	-7.1840E^{-4}	25.9632	-0.0192	1
	10	1E^{-3}	-0.0581	26.0081	-0.4413	
	25	1E^{-3}	-0.4108	28.1994	-1.3933	
MDF($D = 2$)	1	1E^{-3}	-8.9982E-4	25.9621	-0.0324	2
	10	1E^{-3}	-0.0755	26.0480	-0.4361	
	25	1E^{-3}	-0.5088	28.4881	-1.4931	
MDF($D = 4$)	1	1E^{-3}	5.1371E^{-4}	25.7722	-0.0332	4
	10	1E^{-3}	-0.0548	25.8576	-0.2134	
	25	1E^{-3}	Inf	-121.8932	-0.77745	
MDF($D = 8$)	1	1E^{-3}	0.0102	25.2998	0.0430	8
	10	1E^{-3}	0.0771	25.1445	0.2644	
	25	1E^{-3}	NaN	NaN	2.0281	
MDF($D = 16$)	1	1E^{-3}	0.0488	25.3009	0.37623	16
	10	1E^{-3}	0.3345	24.5383	0.97404	
	25	1E^{-3}	NaN	NaN	1.7512	
MDF($D = 32$)	1	1E^{-3}	0.1812	24.9124	1.1390	32
	10	1E^{-3}	0.5769	23.8896	1.7532	
	25	1E^{-3}	NaN	NaN	1.9002	

Table 5.2: Simulation results with feedback for a gain of $G = 1$ and $G = 25$. For MDF, D indicates the amount of delay blocks. The computation time is given as a factor w.r.t. 0.0015 s

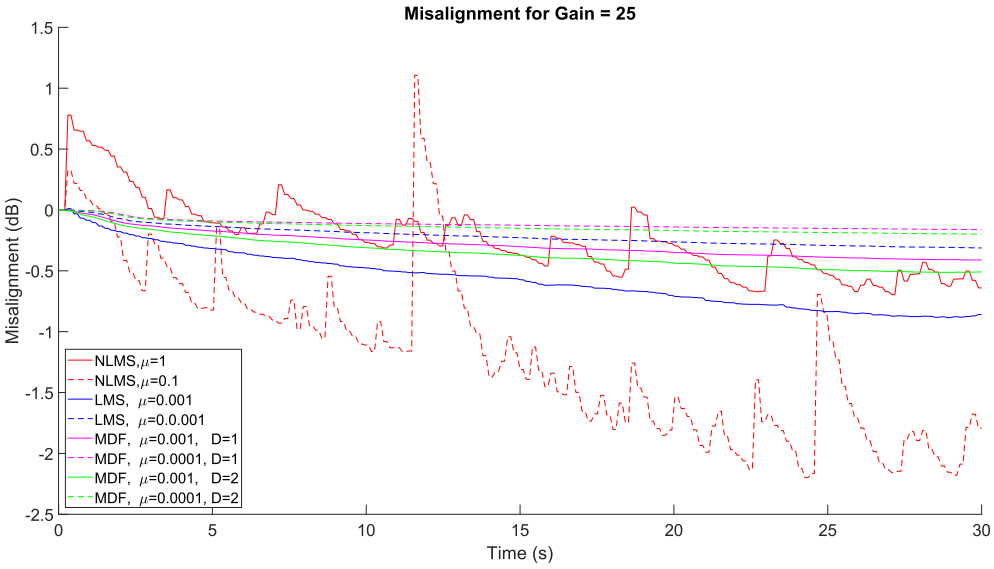


Figure 5.3: Closed loop simulation misalignment results for gain $G = 25$

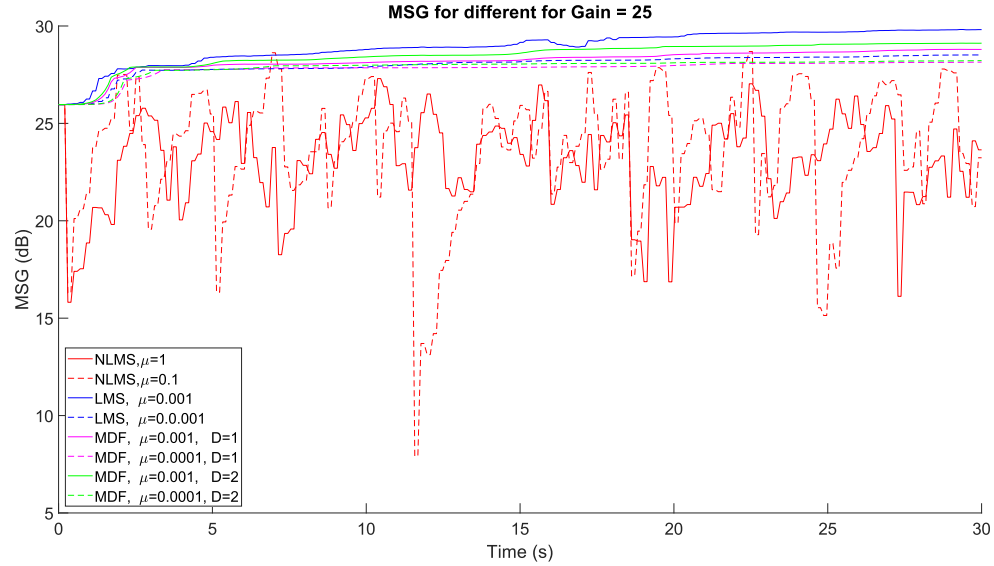


Figure 5.4: Closed loop simulation maximum stable gain results for gain $G = 25$

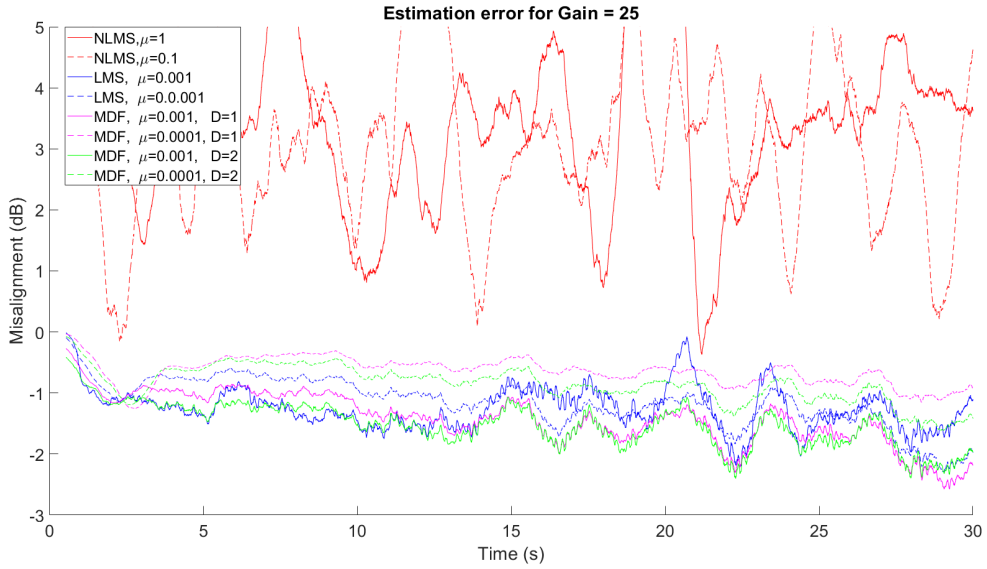


Figure 5.5: Closed loop simulation estimation error results for gain $G = 25$, Estimation error is the average over 44100 samples

6

Discussion

In the following sections, the results of the simulations, both without and with feedback, are discussed. The results will be mentioned again, which will be discussed in further detail, with as goal to find logical explanations behind the achieved results.

6.1. Open Loop Simulations

A difference with the initial expectations is seen from the results, where it is shown the best filter estimation does not result in the lowest estimation error and highest MSG, as is the case with NLMS.

In the case of the open loop estimation with no feedback, The lack of correlation of the misalignment with the MSG with respect to the estimation error could be explained by stating that the misalignment says more about the differences with the RIR in a global way and the estimation error more locally, making the latter also more susceptible for local variations, making it possibly less consistent. In the case of NLMS, bigger steps are taken due to it being normalised from a small signal input, which results in a quick approximation with the addition of errors as a consequence. It could be seen that for MDF a higher step size resulted in a higher MSG, but that after a certain threshold the estimation completely derails. A bigger step size results more quickly in a better estimation of the RIR, and in literature it was already shown that for a higher value of D, the error would decrease as well.

6.2. Closed Loop Simulations

In the simulations with feedback, the misalignment seems to be more in line with the estimation error for all methods. It still holds that the best results in terms of average MSG come from LMS with a gain of $G = 25$ and MDF with $D = 1, 2$. For all methods a higher gain, within the stable range, is beneficial for the three specifications. For low gains and attenuation through the room the magnitude of the feedback is relatively small compared to the voice signal. For LMS and MDF the error and MIS do not change much from the initial state, due to the fixed step size and low signal strength, NLMS however shows more variation. Higher gains see improvements to the estimation and compared to without cancellation the system becomes stable for a gain of $G = 25$.

In the test setup no decorrelation techniques or noise has been added. The length of the RIR used for the simulation could be changed to simulate the effect on the estimation and error when using a shorter filter estimation. Typical and atypical RIR's could be used for the filter estimation testing, as having multiple scenarios would benefit the research.

From the closed loop simulations it can be concluded that LMS or MDF is the most suitable solution, MDF with $\mu = 1E^{-3}$ and $D = 2$ showed the lowest average error and MSG. The misalignment is lower than for LMS $\mu = 0.01$, however a lower error is a more direct link to the feedback cancellation and is considered more valuable. While LMS shows the top results, MDF shows results of about the same order, but more consistently for different values of the gain, while LMS is merely situationally effective. Due to this, the most viable method to be implemented is MDF with $D = 2$ delay blocks. This complies with the comparison made in the Ch. 4.

The specifications set in the requirements 3 are not met by the proposed solution, in terms of the amount of misalignment and increase in MSG. This could be because of several reasons. The first is

that the requirements were unreasonable in the first place, due to misinterpretation of literature, from where most of the expected performance is derived. Secondly, the fault could lie in the simulation environment being imperfect in the sense that not the right parameters were used and conditions were met to achieve the results derived from literature. However, when listening to the output compared to when no feedback cancellation is present, howling was successfully removed by the solution and only a small echo was still audible.

7

Implementation Advice & Considerations

Due to situational restrictions, no physical implementation of the system could be realised. However, in the following sections, some advice and considerations are given for future reference.

One option is to implement the algorithm into software. Common languages that one can use are Python, C and C++. It is even possible to implement the algorithm into Matlab and convert it into C. In addition to this, one would also have to make a graphical interface of sorts, as well as provide the program with the ability to adapt itself to the hardware used, letting the user of the program choose how and where to use the program. Advantages of making a software implementation are that it saves physical space, but the way the program is executed will be greatly dependent on the hardware used. A program also requires a computer with operating system to run.

To implement the solution into real hardware two audio inputs, one audio output and a processor with memory will be required. The audio input and outputs could either work on frame or sample base, although sample base would be preferred to remove block delay from the MDF algorithm when the estimated RIR is used on a single sample base. The delay would be lower if more delay blocks are used resulting in smaller buffers. MDF with $D = 2$ and $M = 4410$ requires a FFT length of $2N = 4410$ and buffer frame length of $N = 2205$ which would result in a delay of $T_{delay} = N/F_s = 50$ ms which is outside of the specifications, when combined with the decorrelation. Increasing the amount of delay blocks would reduce the delay and the FFT size. In the algorithm, a total of $M + 3$ FFT's and $M + 2$ IFFT's are computed, all with the same previously mentioned FFT length. A latency time for calculating the FFT's is realistically between 4.9 and 57 ms for FFT and between 4.9 and 38.9 ms for IFFT as tested on a FPGA for lengths of a 4096[29]. The computation time will be between 44 and 423 ms. The selection of proper hardware that can compute the FFT's within a frame of 50 ms is possible for this solution. Performance can be increased by considering the availability of more efficient Digital Signal Processors with dedicated FFT hardware which are more effective than FPGA's and makes the implementation of the algorithm easily possible within the frame.

8

Conclusion

The goal of this thesis was to implement an adaptive filter design to remove acoustic feedback from a closed loop system. The literature study showed many years of work on the subject in fields such as hearing aids, noise cancelling and echo cancellation. The state of the art listed potential solutions, of which MDF was deemed the most promising. A test setup and simulation were designed to test a selection of the algorithms. Without the possibility to have a real system, research was done to make a realistic simulation by implementing signal hysteresis and relate it to voltage levels. Although it was concluded in the end that the MDF algorithms would be the best to implement of the selected algorithms, the open loop and closed loop simulations showed unexpected results. The performances were lower than expected from literature, which could be a result from the simulation setup, which might not have been designed in a way that fully used the algorithms correctly, variations of systems and room impulse responses, i.e. in length, amplitude, signal levels and parameter settings that have not been fully explored. In the end, we could not meet the set requirements derived from the performance of certain algorithms in literature, although there were clear improvements to the system signals with respect to the case without any feedback cancellation and when listening the audible feedback was clearly reduced.

Quite some research was done on the state of the art and trying to compare the algorithms by literature alone. Since the initial thought was to compare the different algorithms based on literature and past results alone, less time was spent preparing a simulation setup on our own. More time time could have been better spent working on the Matlab implementations rather than theorising. In hindsight, the goals of the project would be better reached if one algorithm from the state of the art was selected based on literature, thoroughly understood and implemented with improvements to robustness and performance. The initial results of the simulations showed a lot of instability for some of the algorithms, against all expectations. This caused for more effort being put into making sure of the trustworthiness of the results and the way the algorithms were implemented, rather than making real improvements to the system with the chosen method.

A

Algorithms

A.1. RLS

$$\begin{aligned}\mathbf{c}(t) &= \lambda^{-1}\mathbf{R}^{-1}(t-1)\mathbf{x}(t)\\ \gamma^{-1}(t) &= 1 + \mathbf{c}^T(t)\mathbf{x}(t)\\ \mathbf{R}^{-1}(t) &= \lambda^{-1}\mathbf{R}^{-1}(t-1) - \mathbf{c}(t)\gamma(t)\mathbf{c}^T(t)\end{aligned}$$

$$\begin{aligned}\epsilon(t) &= m(t) - \hat{\mathbf{f}}^T(t-1)\mathbf{x}(t)\\ \hat{\mathbf{f}}(t) &= \hat{\mathbf{f}}(t-1) + \epsilon(t)\gamma(t)\mathbf{c}(t)\end{aligned}$$

with initialisation conditions

$$\begin{aligned}\mathbf{R}(-1) &= \delta\mathbf{I}\\ \delta &= 0.01\sigma_x^2\\ \mathbf{h}(-1) &= 0\end{aligned}$$

A.2. LMS

The weight equation can be described as

$$\hat{f}_{n+1} = \hat{f}_n - \mu\Delta\varepsilon[n]$$

With ε representing the mean squared error.
The algorithm

$$\begin{aligned}&\text{For} \quad n = 0, 1, 2...\\ &\mathbf{x}(n) = [x(n), x(n-1), ..., x(n-M+1)]^T\\ &e(n) = m(n) - \hat{f}(n)\mathbf{x}(n)\\ &\hat{f}(n+1) = \hat{f}(n) + \mu\, e^*\mathbf{x}(n)\end{aligned}$$

with

$$\begin{aligned}\hat{f}(0) &= 0\\ 0 < \mu &< \frac{2}{\lambda_{max}}\end{aligned}$$

where M is the filter order and μ is the so-called step size. With λ the auto-correlation matrix, its eigen values which are non negative. With maximum convergence speed when

$$\mu = \frac{2}{\lambda_{max} + \lambda_{min}}$$

A.3. NLMS

$$\begin{aligned}
&\text{For} \quad n = 0, 1, 2\ldots \\
&\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-M+1)]^T \\
&e(n) = m(n) - \hat{f}(n)\mathbf{x}(n) \\
&\hat{f}(n+1) = \hat{f}(n) + \frac{\mu e^* \mathbf{x}(n)}{\mathbf{x}^H(n)\mathbf{x}(n)}
\end{aligned}$$

with

$$\hat{f}(0) = 0$$

When there is no interference ($v(n) = 0$) it holds that

$$\mu_{opt} = 1$$

For the general case with interference, assuming $s(n)$ and $x(N)$ are uncorrelated and $y(n)$ and $\hat{y}(n)$ are the outputs of the actual filter and the estimated filter resulting from $x(n)$, the optimal step size is given as

$$\mu_{opt} = \frac{\mathrm{E}[|y(n) - \hat{y}(n)|^2]}{\mathrm{E}[|e(n)|^2]}$$

A.4. Circular FDAF

$$\begin{aligned}
\bar{\mathbf{F}}(0) &= [0..0]^T \\
P_m(0) &= \delta_m, \quad m = 0, \ldots, N-1 \\
X(k) &= diag\mathrm{FFT}[x(kN), \ldots, x(kN+N-1)]^T \\
\mathbf{M}(k) &= \mathrm{FFT}m(k) \\
\mathbf{Y}(k) &= X(k)\hat{F}(k) \\
\mathbf{E}(k) &= \mathbf{M}(k) - \mathbf{Y}(k) \\
P_m(k) &= \lambda P_m(k-1) + \alpha |X_m(k)|^2, \quad m = 0, \ldots, N-1 \\
\mu(k) &= \mu(0) diag(P_0^{-1}(k), \ldots, P_{N-1}^{-1}(k)) \\
\hat{F}(k+1) &= \hat{F}(k) + 2\mu(k)X(k)\mathbf{E}(k)
\end{aligned}$$

A.5. LMS derivation

$$e(n) = d(n) - \hat{\mathbf{f}}_0^H \mathbf{x}(n) \tag{A.1}$$

$$C(n) = E\left\{|e(n)|^2\right\} \tag{A.2}$$

$$-\nabla_{\hat{\mathbf{f}}^H} C(n) = \nabla_{\hat{\mathbf{f}}^H} E\left\{|e(n)|^2\right\} \tag{A.3}$$

$$\nabla_{\hat{\mathbf{f}}^H} C(n) = 2\nabla_{\hat{\mathbf{f}}^H}(e(n))e(n) \tag{A.4}$$

$$\nabla_{\hat{\mathbf{f}}^H}(e(n)) = \nabla_{\hat{\mathbf{f}}^H}(d(n) - \hat{\mathbf{f}}^H \mathbf{x}(n)) = -\mathbf{x}(n) \tag{A.5}$$

$$\nabla_{\hat{\mathbf{f}}^H} C(n) = -2e(n)\mathbf{x}(n) \tag{A.6}$$

$$\hat{\mathbf{f}}^H(n+1) = \hat{\mathbf{f}}^H(n) + \mu \mathbf{x}(n)e(n) \tag{A.7}$$

A.6. MDF

$$\mathbf{X}(m,j) = \text{diag}\left\{\text{FFT}\left[x_0(j-1), x_1(j-1), \ldots, x_{N'/2-1}(j-1), x_0(j), x_1(j), \ldots, x_{N'/2-1}(j)\right]^T\right\} \quad \text{(A.8)}$$

$$\mathbf{X}(m,j) = \mathbf{X}(m-1, j+1), \qquad m = 1, 2, \ldots, M-1 \quad \text{(A.9)}$$

$$\mathbf{y}(j) = \text{last } N'/2 \text{ terms of } \left\{\text{FFT}^{-1}\left[\sum_{m=1}^{M} \mathbf{X}(m,j)\mathbf{W}(m,j)\right]\right\} \quad \text{(A.10)}$$

$$\mathbf{E}(j) = \text{FFT}\left\{\mathbf{0}_{1\times N'/2}, [\mathbf{d}(j) - \mathbf{y}(j)]^T\right\}^T \quad \text{(A.11)}$$

$$\phi(m,j) = \text{first half of } \left\{\text{FFT}^{-1}\left[\mathbf{X}^*(m,j)\mathbf{E}(j)\right]\right\} \quad \text{(A.12)}$$

$$\Phi(m,j) = \text{FFT}\left[\phi(m,j), \mathbf{0}_{1\times N'/2}\right]^T \quad \text{(A.13)}$$

$$\mathbf{W}(m, j+1) = \mathbf{W}(m,j) + M\mu_B\Phi(m,j) \quad \text{(A.14)}$$

B

Figures

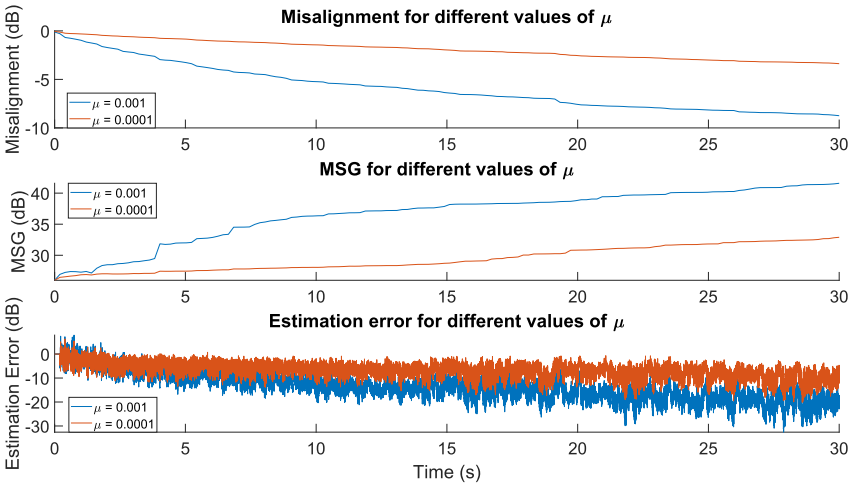


Figure B.1: No feedback simulation results of the LMS algorithm, Estimation error is the average over 100 samples

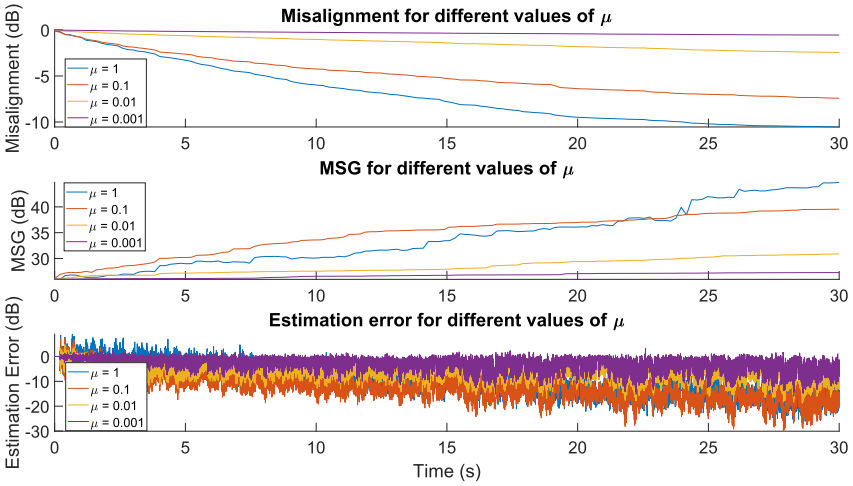


Figure B.2: No feedback simulation results of the NLMS algorithm, Estimation error is the average over 100 samples

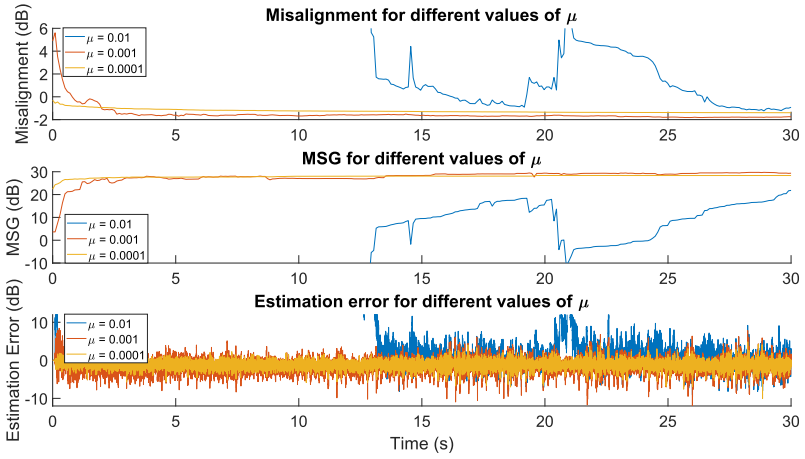


Figure B.3: No feedback simulation results of the Circular FDAF algorithm, Estimation error is the average over 100 samples

B.1. Test Results

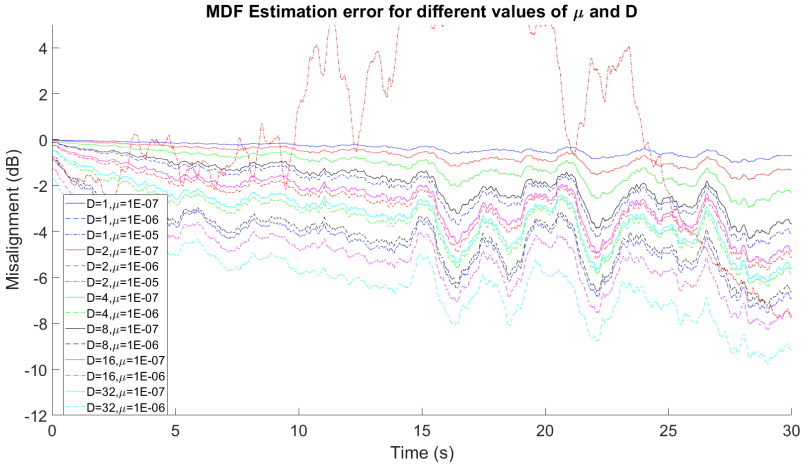


Figure B.4: No feedback simulation estimation error results of the MDF algorithm, Estimation error is the average over 44100 samples

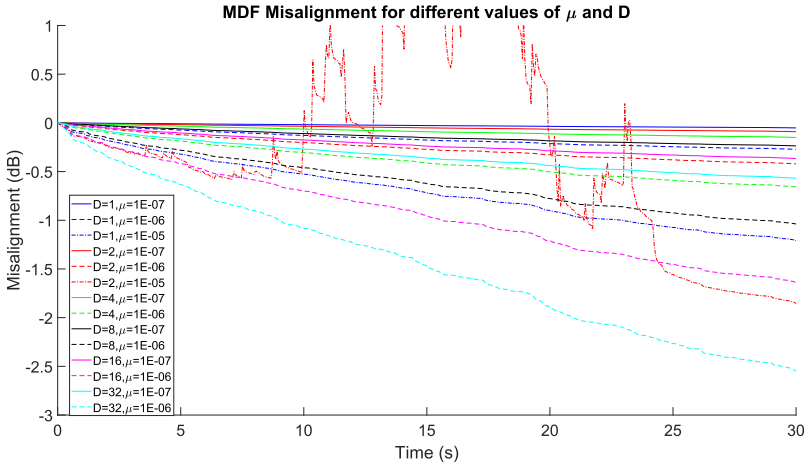


Figure B.5: No feedback simulation MSG results of the MDF algorithm

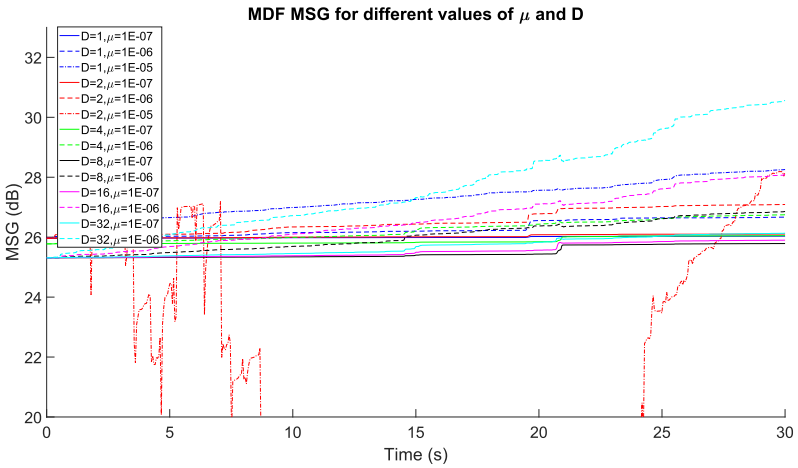


Figure B.6: No feedback simulation MIS results of the MDF algorithm

C

Matlab Code

C.1. Matlab Simulation Environment Without Feedback

```
%{
   Testing enviorment for adaptive filter estimation algorithms
   Authors; Cees Kos, Mathijs Bekkering
%}
clear all
close all

M = 4410; %Estimated filter length
Nf = 4410; %Room Impulse Response (RIR) filter length used for filter
    output

load RIR44100.mat %Load RIR
f = f(1:Nf); %Section to wanted length

%load test audio sample (input) and sampling frequency (Fs)
[input,Fs] = audioread('a-ha - Take On Me.mp3');

input = input(23*Fs:24*Fs,1); %Section 30 seconds
samples = length(input); %Samples in test signal

desired = conv(f,input); %Generate desired signal for testing with RIR
desired = desired(1:samples); %Resize

Methods = ["No" "Perfect" "LS" "LMS" "NLMS" "FNLMS" "APA" "MDF" "
    CircularFDAF" "Overlap_safe_FLMS" "FNLMS" "BLMS" "BNLMS" "RLS"];
selected = ["LMS"];
for method = selected
   disp(method)
   %Initialization of algorithm
   %mu is the stepsize
   switch method
         case "No"
            w = zeros(M,1);
            N = M; %Size of frame
         case "Perfect"
            N = M;
            w = f(1:M);
         case "LS"
```

```
37              N = M;
38              w = zeros(M,1);
39          case ”LMS”
40              mu = 0.01;
41              N = M*2;
42              w = zeros(M,1);
43          case ”NLMS”
44              mu =1;
45              N = M*2;
46              w = zeros(M,1);
47          case ”APA”
48              N = M;
49              mu = 1;
50              w = zeros(M,1);
51              x_N_old = zeros(N,1);
52          case ”FNLMS”
53              N=M;
54              w = zeros(N,1);
55              W = fft(w);
56              mu = 0.0005;
57              P = zeros(M,1);
58          case ”Overlap_safe_FLMS”
59              W = zeros(2*M,1);
60              mu = 0.00001;
61              x_old = zeros(M,1);
62              P = zeros(2*M,1);
63              N=M;
64          case ”CircularFDAF”
65              N=M;
66              mu = 0.1;
67              W = zeros(M,1);
68              P = zeros(M,1);
69              w = zeros(M,1);
70          case ”BLMS”
71              N = 2*M;
72              mu = 1;
73              w_l = zeros(M,1);
74              w = w_l;
75          case ”BNLMS”
76              N = M;
77              mu = 3;
78              w_l = zeros(M,1);
79              w = w_l;
80          case ”RLS”
81              N = 2*M;
82              w_l = zeros(M,1);
83              w = w_l;
84              delta = 0.01*var(input);
85              Phi = delta*eye(M);
86              P = inv(Phi);
87              lambda = 0.999;
88          case ”MDF”
89              DelayBlocks = 32; %Amount of delay blocks
90              Np = 2*M/DelayBlocks; %Size of FFT
91              N = Np/2; %Size of frame
92              mu = 0.00001; %stepsize
```

```
93              x_previous = zeros(Np/2,1);
94              W = zeros(Np,DelayBlocks);
95              X = zeros(Np,DelayBlocks);
96              w = zeros(M,1);
97           otherwise
98              disp("NO case")
99        end
100
101       %Reset simulation enviorment
102       xx = []; %Input
103       yy = []; %Output
104       dd = []; %Desired
105       ww = []; %Estimated Filter
106       buffer_fh = zeros(M,1); %M buffer length for estimated filter length
107       bufferx = zeros(N,1); %N buffer length for frames
108       bufferd = zeros(N,1);
109       buffery = zeros(N,1);
110
111       loopcountN = 0;
112       loopcountW = 0; %Loop count for storing estimated weights
113       simulation_time = tic;
114
115       for n = 1:samples
116          calculationstime = tic;
117          x = input(n);
118          d = desired(n);
119          [y,buffer_fh] = ContinuesFIR(w,x,M,buffer_fh); %Estimated filter
                output
120
121          %Buffers for estimators
122          bufferx = circshift(bufferx,1);
123          bufferx(1) = x;
124
125          bufferd = circshift(bufferd,1);
126          bufferd(1) = d;
127
128          buffery = circshift(buffery,1);
129          buffery(1) = y;
130
131          if loopcountN == N %If buffer is full
132             method_time = tic; %Method time estimation
133             switch method
134             case "No"
135             case "Perfect"
136             case "LS"
137                [w] = LS_Func(M,N,flip(bufferx),flip(bufferd));
138             case "LMS"
139                [w,e] = LMS_Func(bufferd,bufferx,N,M,mu,w);
140             case "NLMS"
141                [w,e] = NLMS_Func(bufferd,bufferx,N,M,mu,w);
142             case "APA"
143                [e,w,x_N_old,X_LN] = AlgorithmAPAloop_rec(N,M,mu,w,x_N_old,
                      flip(bufferx),flip(bufferd));
144             case "FNLMS"
145                [W,~,e] = FNLMS_Basic(flip(bufferx),flip(bufferd),W,mu,N,P)
                      ;
```

```
146                 w = ifft(W);
147             case "Overlap_safe_FLMS"
148                 [x_old,W,e,~] = Overlap_Save_Func(N,x_old,x,W,mu,d,P);
149                 w = ifft(W);
150                 w = w(1:M);
151             case "CircularFDAF"
152                 [W,E,Y,P] = Circular_Conv_Func(flip(bufferx),W,mu,flip(
                        bufferd),P);
153                 w = ifft(W);
154             case "BLMS"
155                 [w_l] = BLMS_Func((bufferd),flip(bufferx),M,N,mu,w_l,(
                        buffery));
156                 w = flip(w_l);
157              case "BNLMS"
158                 [w_l] = BNLMS_Func(flip(bufferd),bufferx,N,M,mu,w_l,flip(
                        buffery));
159                 w = flip(w_l);
160             case "RLS"
161                 [e,w] = AlgorithmRLSloop((bufferx),(bufferd),M,P,lambda,w);
162             case "MDF"
163                 [~,X,W,x_previous,PHI,phi,E] = MDF_1Func(Np,DelayBlocks,mu,
                        flip(bufferx),flip(bufferd),X,W,x_previous);
164                 w = ifft(W);
165                 w = (reshape(fliplr(w(1:Np/2,:)), [DelayBlocks*Np/2,1])); %
                        Reshape to single filter estimation
166             otherwise
167             end
168             loopcountN = 0;
169             method_time = toc(method_time);
170         end
171
172         loopcountN = loopcountN + 1;
173         loopcountW = loopcountW + 1;
174
175         if loopcountW > M %Store filter estimation
176             ww= [ww w];
177             loopcountW = 0;
178         end
179
180         %% save information
181         dd(n) =d;
182         xx(n) =x;
183         yy(n) =y;
184       end
185 end
186
187 %Calculate performance
188 for k = 1:size(ww,2)
189     mis = sqrt(sum((abs((f(1:M)-ww(1:M,k)).^2))/sum(f(1:M).^2)));
190     MIS(k) = mis; %Misalignment
191     msg = calcMSG(1,f,[ww(1:M,k); zeros(length(f)-M,1)]);
192     MSG(k) = msg.M; %Maximum Stable Gain
193
194 end
195
196 Estimation_error = movmean(10*log10(abs(dd-yy)),100)-movmean(10*log10(abs(
```

```
    dd)),100);
197
198 %Plots
199 timeaxis =(0:samples-1)/Fs;
200 frametimeaxis = linspace(0,(samples-1)/Fs,length(MIS));
201
202 figure;
203 subplot(3,1,1)
204 plot(frametimeaxis,10*log10(MIS));
205 title('MIS')
206
207 subplot(3,1,2)
208 plot(timeaxis,Estimation_error)
209 title('estimationerror')
210
211 subplot(3,1,3)
212 plot(frametimeaxis,MSG)
213 title('MSG')
214
215 sgtitle(strcat((method), " M:", num2str(M), " Nf:", num2str(Nf), " u:",
    num2str(mu) ))
```

C.2. Matlab Simulation Environment With Feedback

```
1  %{
2     Testing enviorment for adaptive filter estimation algorithms
3     Authors; Cees Kos, Mathijs Bekkering
4  %}
5  clear all
6  close all
7
8  M = 4410; %Estimated filter length
9  Nf = 4410; %Room Impulse Response (RIR) filter length used for filter
       output
10 G = 50; %Loop gain
11
12 load RIR44100.mat %Load RIR
13 f = f(1:Nf); %Section to wanted length
14
15 %load test audio sample (input) and sampling frequency (Fs)
16 [input,Fs] = audioread('a-ha - Take On Me.mp3');
17
18 input = input(23*Fs:24*Fs,1); %Section 30 seconds
19 samples = length(input); %Samples in test signal
20
21 buffermst =[];
22 bufferxst =[];
23
24 Methods = ["No" "Perfect" "LS" "LMS" "NLMS" "FNLMS" "APA" "MDF" "
       CircularFDAF" "Overlap_safe_FLMS" "FNLMS" "BLMS" "BNLMS" "RLS"];
25 selected = ["APA"];
26 for method = selected
27    disp(method)
28    %s is signal going into microphone from person or music
29    %y is feedback from speaker
30    %y_hat is estimated feedback
31    %x is signal going to speaker
```

```
32      %u is signal after subtraction of extimation
33      %m is signal from microphone
34      %mu is the stepsize
35
36      %Initialization of algorithm
37      switch method
38         case ”Perfect”
39            w = f;
40            N = M; %N is frame size
41         case ”NO”
42            w = zeros(M,1);
43            N = M;
44         case ”LS”
45            M = N/2;
46            f_l = f(1:M);
47            w = f;
48         case ”LMS”
49            mu = 0.001;
50            N=M*2;
51            w = zeros(M,1);
52         case ”NLMS”
53            mu = 0.1;
54            N = M*2;
55            w = zeros(M,1);
56         case ”APA”
57            N = M;
58            mu = 1;
59            w = zeros(M,1);
60            x_N_old = zeros(N,1);
61         case ”FNLMS”
62            mu = 0.02;
63            w = zeros(M,1);
64            W = fft(w);
65            N=M;
66         case ”Overlap_safe_FLMS”
67            W = zeros(2*N,1);
68            mu = 0.00001;
69            x_old = zeros(N,1);
70            P = zeros(2*N,1);
71            M = N;
72            f_l = f(1:N);
73         case ”CircularFDAF”
74            N = M;
75            W = zeros(N,1);
76            mu = 0.001;
77            P = zeros(N,1);
78            w = zeros(N,1);
79         case ”BLMS”
80            M = N/2;
81            f_l = f(1:M);
82            mu = 50;
83            w_l = zeros(M,1);
84          case ”BNLMS”
85            M = N/2;
86            f_l = f(1:M);
87            mu = 30;
```

```
88          w_l = zeros(M,1);
89       case "RLS"
90          f_l = f(1:M);
91          M_RLS = length(f);
92          w_l = zeros(M_RLS,1);
93          w = zeros(M_RLS,1);
94          delta = 0.01*var(input);
95          Phi = delta*eye(M_RLS);
96          P = inv(Phi);
97          lambda = 0.999;
98       case "MDF"
99          DelayBlocks = 1; %Amount of delay blocks
100         Np = 2*M/DelayBlocks;
101         N = Np/2;
102         mu = 0.0001; %stepsize
103         x_previous = zeros(Np/2,1);
104         W = zeros(Np,DelayBlocks);
105         X = zeros(Np,DelayBlocks);
106         w = zeros(M,1);
107      otherwise
108         disp("NO case")
109   end
110   %Reset simulation enviorment
111   xx = [];
112   yy = [];
113   uu = [];
114   mm = [];
115   yy_hat = [];
116   ww = [];
117   ss = [];
118   Delaybuffer = zeros(2000,1); %Delay in loop
119   buffer_f = zeros(Nf,1);
120   buffer_fh = zeros(M,1);
121   bufferx = zeros(N,1);
122   bufferm = zeros(N,1);
123   bufferu = zeros(N,1);
124   loopcountN = 0;
125   loopcountW = 0;
126
127   for n = 1:samples
128      % insert estimation function
129      s = input(n);
130
131      [y,buffer_f] = ContinuesFIR(f,x,Nf,buffer_f);
132      [y_hat,buffer_fh] = ContinuesFIR(w,x_est,M,buffer_fh);
133
134      m = s+y; %Feedback + Input signal
135
136      %Saturation of input signal
137      if m > 0.05
138         m = 0.05 ;
139      elseif m < -0.05
140         m = -0.05 ;
141      end
142
143      %Saturation of estimation signal
```

```
144         if y_hat > 0.05
145            y_hat = 0.05 ;
146         elseif y_hat < -0.05
147            y_hat = -0.05 ;
148         end
149
150         u = m-y_hat; %Feedback cancellation
151
152         %buffers
153         bufferx = circshift(bufferx,1);
154         bufferx(1) = x;
155
156         bufferm = circshift(bufferm,1);
157         bufferm(1) = m;
158
159         bufferu = circshift(bufferu,1);
160         bufferu(1) = u;
161
162         if n > 2*N %Delay to fill buffers with usable data
163         if loopcountN > N %If buffer is full
164            method_time= tic; %Method time estimation
165            switch method
166            case "Perfect"
167            case "NO"
168            case "LS"
169               [w] = LS_Func(M,N,estimation_bufferx,estimation_bufferm);
170            case "LMS"
171               [w,e] = LMS_Func((estimation_bufferm),(estimation_bufferx),N,M
                     ,mu,w);
172            case "NLMS"
173               [w,e] = NLMS_Func(estimation_bufferm,estimation_bufferx,N,M,mu
                     ,w);
174            case "APA"
175               [e,w,x_N_old,X_LN] = AlgorithmAPAloop_rec(N,M,mu,w,x_N_old,
                     flip(estimation_bufferx),flip(estimation_bufferm));
176            case "FNLMS"
177               [W,~,e] = FNLMS_Basic(estimation_bufferx,estimation_bufferm,W,
                     mu,N);
178               w = ifft(W);
179            case "Overlap_safe_FLMS"
180               [x_old,W,e,~] = Overlap_Save_Func(N,x_old,x,W,mu,m,P);
181               w = ifft(W);
182               w = w(1:N);
183            case "CircularFDAF"
184               [W,E,Y,P] = Circular_Conv_Func(flip(estimation_bufferx),W,mu,
                     flip(estimation_bufferm),P);
185               w = ifft(W);
186            case "BLMS"
187               [~,w,e] = BNLMS_Func(m,x,N,M,mu,w);
188             case "BNLMS"
189               [~,w,e] = BNLMS_Func(m,x,N,M,mu,w);
190            case "RLS"
191               [e,w_l] = AlgorithmRLSloop(x,m,M_RLS,P,lambda,w_l);
192               w = flip(w_l);
193            case "MDF"
194               [~,X,W,x_previous,PHI,phi,E] = MDF_1Func(Np,DelayBlocks,mu,
```

```
                flip(estimation_bufferx),flip(estimation_bufferm),X,W,
                x_previous);
195             w = ifft(W);
196             w = (reshape(fliplr(w(1:Np/2,:)), [DelayBlocks*Np/2,1]));
197          otherwise
198             disp("NO case")
199           end
200          method_time = toc(method_time);
201          loopcountN = 0;
202       end
203       end
204       loopcountN = loopcountN + 1;
205
206       x = Delaybuffer(end)*G; %Delay buffer and loop gain G
207       Delaybuffer = circshift(Delaybuffer,1);
208       Delaybuffer(1) = u;
209
210       %Loudspeaker saturation
211       if x > 1.25
212          x = 1.25;
213       elseif x < -1.25
214          x = -1.25;
215       end
216
217       %% save information
218       ss(n) = s;
219       xx(n) = x;
220       yy(n) = y;
221       uu(n) = u;
222       mm(n) = m;
223       yy_hat(n) = y_hat;
224       xx_est(n) = x_est
225
226       loopcountW = loopcountW + 1;
227       if loopcountW > M
228          ww= [ww w];
229          loopcountW = 0;
230       end
231    end
232 end
233
234 %Calculate performance
235 for k = 1:size(ww,2)
236    mis = sqrt(sum((abs((f(1:M)-ww(1:M,k)).^2))/sum(f(1:M).^2)));
237    MIS(k) = mis; %Misalignment
238    msg = calcMSG(1,f,[ww(1:M,k); zeros(length(f)-M,1)]);
239    MSG(k) = msg.M; %Maximum Stable Gain
240 end
241
242 Estimation_error = movmean(10*log10(abs(yy-yy_hat)),100)-movmean(10*log10(
       abs(yy)),100);
243 Signal_error = movmean(10*log10(abs(ss-uu)),100)-movmean(10*log10(abs(ss))
       ,100);
244
245 timeaxis =(0:length(yy)-1)/Fs;
246 frameaxis = linspace(0,(length(yy)-1)/Fs,length(MIS));
```

```
247
248 %% Plot
249 figure()
250 subplot(241)
251 plot(timeaxis,yy);
252 title('yy');
253 subplot(243)
254 plot(timeaxis,ss);
255 title('ss');
256 subplot(242)
257 plot(timeaxis,yy_hat);
258 title('yy_hat');
259 subplot(244)
260 plot(timeaxis,uu);
261 title('uu');
262
263 subplot(245)
264 plot(frameaxis,10*log10(MIS));
265 title('mis');
266 subplot(246)
267 plot(frameaxis,MSG)
268 title('MSG');
269 subplot(247)
270 plot(timeaxis,Estimation_error);
271 title('Estimation_error y_hat-y');
272 subplot(248)
273 plot(timeaxis,Signal_error);
274 title('Signal_error u-s');
275
276 sgtitle(strcat(method," u ", num2str(mu),"G ", num2str(G)))
```

Bibliography

[1] M. R. Schroeder, “Improvement of acoustic-feedback stability by frequency shifting,” *The Journal of the Acoustical Society of America*, vol. 36, September 1964.

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

[3] W. Leotwassana, R. Punchalard, and W. Silaphan, “Adaptive howling canceller using adaptive iir notch filter: simulation and implementation,” in *International Conference on Neural Networks and Signal Processing, 2003. Proceedings of the 2003*, vol. 1, pp. 848–851 Vol.1, 2003.

[4] G. Rombouts, T.van Waterschoot, and M.Moonen, “Proactive notch filtering for acoustic feedback cancellation,” *Proc. 2nd Annual IEEE Benelux/DSP Valley Signal Process*, 2006.

[5] A. Gilloire, E. Moulines, D. Slock, and P.Duhamel, “State of the art in acoustic echo cancellation,” in *Digital Signal Processing in Telecommunications*, pp. 45–91, Springer, 1996.

[6] L. C. A. Huijbregts and M. A. Jongepier, “Decorrelation in adaptive feedback cancellation for pa systems.” B.S. Thesis, Delft Univ. Technol., Delft, 2020.

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

[8] T. van Waterschoot, G. Rombouts, and M. Moonen, “Optimally regularized adaptive filtering algorithms for room acoustic signal enhancement,” *Signal Processing*, vol. 88, pp. 594–611, March 2008.

[9] H. Nyquist, “Regeneration theory,” *The Bell System Technical Journal*, vol. 11, no. 1, pp. 126–147, 1932.

[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] A. N. Tikhonov and V. Y. Arsenin, “Solutions of ill-posed problems,” *Siam Rev.*, vol. 21, no. 2, pp. 266–267, 1979.

[12] E. Horita, K. Sumiya, H. Urakami, and S. Mitsuishi, “A leaky rls algorithm: its optimality and implementation,” *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2924–2936, 2004.

[13] G. Glentis, K. Berberidis, and S. Theodoridis, “Efficient least squares adaptive algorithms for FIR transversal filtering,” *IEEE Signal Processing Magazine*, vol. 16, no. 4, pp. 13–41, 1999.

[14] J. Dhiman, S. Ahmad, K. Gulia, *et al.*, “Comparison between adaptive filter algorithms (lms, nlms and rls),” *International journal of science, engineering and technology research (IJSETR)*, vol. 2, no. 5, pp. 1100–1103, 2013.

[15] G. Clark, S. Mitra, and S. Parker, “Block implementation of adaptive digital filters,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 3, pp. 744–752, 1981.

[16] J. C. Lee and C. K. Un, “Performance analysis of frequency-domain block lms adaptive digital filters,” *IEEE Transactions on Circuits and Systems*, vol. 36, no. 2, pp. 173–189, 1989.

[17] J. J. Shynk, “Frequency-domain and multirate adaptive filtering,” *IEEE Signal Processing Magazine*, vol. 9, no. 1, pp. 14–37, 1992.

[18] J. Soo and K. K. Pang, “Multidelay block frequency domain adaptive filter,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 2, pp. 373–376, 1990.

[19] K.Ozeki and T. Umeda,, “An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties,” *Electronics and Communications in Japan (Part I: Communications)*, vol. 67, no. 5, pp. 19–27, 1984.

[20] S. L.Gay, “A fast converging, low complexity adaptive filtering algorithm,” in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 4–7, IEEE, 1993.

[21] G. Enzner and P. Vary, “Frequency-domain adaptive kalman filter for acoustic echo control in hands-free telephones,” *Signal Processing*, vol. 86, Issue 6, pp. 1140–1156.

[22] G. Bernardi, T. van Waterschoot, J. Wouters, and M. Moonen, “Adaptive feedback cancellation using a partitioned-block frequency-domain kalman filter approach with pem-based signal prewhitening,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 9, pp. 1784–1798, 2017.

[23] F. Yang, G. Enzner, and J. Yang, “Frequency-domain adaptive kalman filter with fast recovery of abrupt echo-path changes,” *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1778–1782, 2017.

[24] E. Audio, “Fm360, fm500, fmr500, fmr600 fmr720.” `https://earthworksaudio.com/products/microphones/flexmic-series/20khz-fm/`. Accessed on 19-06-2020.

[25] E. ToolBox, “Voice level at distance.” `https://www.engineeringtoolbox.com/voice-level-d_938.html`, 2005. Accessed on 19-06-2020.

[26] 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.

[27] A-ha, “Take on me.” `https://www.youtube.com/watch?v=djV11Xbc914`, 1985. Accessed on 19-06-2020.

[28] B. C. Bispo, “Acoustic feedback and echo cancellation in speech communication systems,” 2015.

[29] M. A. Jaber, D. Massicotte, and Y. Achouri, “A higher radix fft fpga implementation suitable for ofdm systems,” in *2011 18th IEEE International Conference on Electronics, Circuits, and Systems*, pp. 744–747, 2011.