Reproduction of Three-Dimensional Audio with Head Tracking

- P6 -

Project Report

Group 642

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

This report was written in LaTeX and has been shared online in-between the group members using Overleaf licensed to all students at Aalborg University.

AALBORG UNIVERSITY

STUDENT REPORT

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

Title:
Reproduction of Three-Dimensional Audio with Head Tracking

Theme:
Digital Signal Processing

Project Period:
Spring Semester 2019

Project Group:
Group 642

Participants:
Frederik Skyt Dæncker Rasmussen
Magnus Bøgh Borregaard Christensen
Max Væhrens
Jakob Krarup Thomsen
Peter Kjær Fisker

Supervisor:
Sofus Birkedal Nielsen

Report Page Numbers: 82

Appendix Page Numbers: 52

Date of Completion:
May 29, 2019

Abstract:

The purpose of this report is to document the design and implementation of a real time system on a Texas Instruments TSM320C553 DSP, that increases immersion in three-dimensional audio by introducing dynamic cues through head tracking. The system is divided into two separate subsystems, one for head tracking and one for audio filtering with HRTF-based digital filters. Head tracking is done by using measurements from a MARG sensor array to calculate a quaternion that describes the instantaneous orientation. The calculation uses the gradient descent method to iteratively determine the orientation together with sensor fusion by using a complementary filter. Audio filtering is done with digital FIR filters created with the HRIRs from the Valdemar database created at AAU. The filters are preprocessed to reduce the amount of coefficients from the 256 coefficients of the HRIRs to 72 coefficients. To remove audible artifacts from switching between filters, crossfading has been implemented. Both subsystems were implemented and tested separately on the DSP. The tests of the head tracking subsystem demonstrated that while the implemented algorithm converged fast enough and had no drift in orientation over time, the achieved accuracy was not suitable for head tracking applications. The tests of the audio filtering subsystem demonstrated that the implementation was able to correctly switch between filters in the horizontal plane and filter audio creating a single virtual sound source. Due to the head tracking subsystem's inadequate performance the two subsystems were never combined and thus the overall system performance of the proposed system could not be tested. However, tests of the designed implementations on a PC suggested that the algorithms designed did function as desired, and thus it is deduced that there are errors in the code implementation on the DSP.

Preface

This report is written as part of a bachelor project in Electronics Engineering at Aalborg University. The project is focused on digital signal processing with a real time implementation and takes place on the sixth semester. As such, some limitations will arise as a result of this being a university student project. Most notably constraints regarding time, budget and scope of the project might be limited. The project took place over a four-month period during winter throughout spring of 2019. In addition to limitations, the product is to be designed for a given DSP. This DSP is the fixed point TSM320C5535 from Texas Instruments. In addition to being the available DSP, it is also the one which is supported by the courses regarding digital signal processing.

The project will make use of a spherical coordinate-system. It should however be noted that the azimuth angle is positive going clockwise, while also being split into ±180°. The center of the azimuth angle is directly in front of the user's head. Theta will denote azimuth angle, phi will denote elevation angle, and r will denote radius (see figure 1). Additionally, the mathematical notation used in the report can be found in table 1.

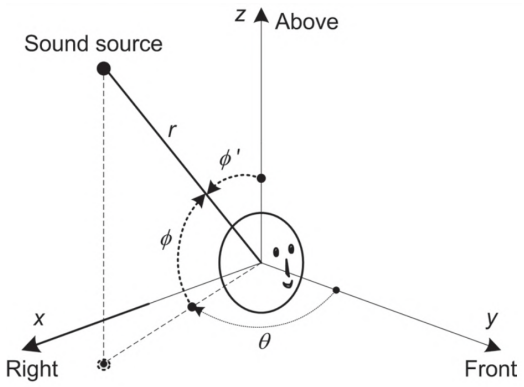


Figure 1: Figure showing spherical coordinate system [1].

Table 1: Table of used mathematical notations.

Overview of Mathematical Notation	
Notation	**Meaning**
$\vec{v}$	A vector, v
$\vec{v}^T$	Transpose of vector v
$\vec{q}^*$	The conjugate of a quaternion, q
$\vec{q}^{-1}$	The inverse of a quaternion, q
$\vec{q} \otimes \vec{p}$	Quaternion product of quaternion q and p
$\dot{f}$	Time derivative of function, f
$\vec{v} \times \vec{u}$	Cross Product of vector v and u
$\|\|\vec{v}\|\|$	2-norm of vector v
$\nabla f\vec{x})$	Gradient of function f
$\boldsymbol{M}$	A matrix, M
$\boldsymbol{J}(\vec{f}(\vec{r}))$	Jacobian of vector function $\vec{f}$
$\boldsymbol{H}(f(\vec{r}))$	Hessian of function f

Contents

I Analysis and Introduction **1**

1 Introduction **2**

1.1 Virtual Three-Dimensional Audio and Attributes of Human Spatial Hearing 2

1.2 Problem Statement 7

2 Requirements and Design Overview **8**

2.1 Limitations 8

2.2 Functional Requirements 8

2.3 Technical Requirements 9

2.4 Digital Signal Processor Hardware 11

2.5 System Design Overview 12

II Head Tracking Design **14**

3 Head Tracking Technology Analysis **15**

3.1 Methods for Tracking Head Movement 15

3.2 Chosen MARG Sensor 16

3.3 Choice of Orientation Algorithm 18

4 Design of Head Tracking Application **19**

4.1 Representation of Orientation in Three-Dimensional Space 19

4.2 Obtaining Orientation from MARG Sensors 19

5 Implementation of Head Tracking **32**

5.1 Numerical Analysis of Dynamic Range 34

5.2 Multiplication 34

5.3 Division 36

5.4 Implementation of Square Root and Reciprocal Square Root. 37

5.5 Fixed-Point Performance Compared to Floating-point Implementation 37

6 Test of Head Tracking Application **39**

6.1 Test of Stability/Drift in Orientation Tracker 39

6.2 Test of Resolution of Orientation Tracker 40

6.3 Test of Accuracy of Orientation Tracker 41

6.4 Test of Convergence Rate of Orientation Tracker 41

6.5 Partial Conclusion 42

III HRTF Filter Design **43**

7 HRTF Analysis and Design **44**

7.1 Introduction to Measured HRTFs 44

7.2 Valdemar HRTF Database 45

7.3 Design of HRTF Filters 46

7.4 HRTF Filter Preprocessing 47

7.5 Switching between HRTFs 52

7.6 Headphone Equalization 56

8 HRTF System Implementation on the DSP **58**

8.1 Orientation-Based Filter Selection 59

8.2 HRTF Filtering on the DSP 60

8.3 HRTF ITD Implementation on the DSP 61

8.4 HRTF Crossfading on the DSP 62

9 Test of HRTF-Based Filtering **63**

9.1 Testing Procedure 63

9.2 Test of the HRTF Preprocessing 63

9.3 Test of the Audio Codec on the DSP 64

9.4 Test of the Filter Selection on the DSP 65

9.5 Test of HRTF Filtering 65

9.6 Partial Conclusion of the HRTF Filtering System 67

IV System Combination and Concluding Remarks **68**

10 System Integration **69**

10.1 Overview of Firmware and Hardware Interfaces 69

10.2 Firmware Structure 70

11 Discussion **75**

11.1 Analysis of Head Tracking System 75

11.2 HRTF Filtering System 76

11.3 Further developments 77

12 Conclusion **79**

Bibliography **80**

Appendix: **83**

A Quaternion Math **83**

B Choice of Method for Determining Step Length in Optimization Algorithm **85**

C Sensor Calibration **92**

D Test journal: Stability/Drift in Orientation Tracker **94**

E Test Journal: Noise in Orientation Tracker **96**

F Test Journal: Accuracy of Orientation Tracker **101**

G Test Journal: Convergence Rate of Orientation Tracker **107**

H Test Journal: Measuring the Transfer Function of the Beyerdynamics DT770 Headphones **111**

I Test Journal: Verifying the Preprocessing Is Correct **115**

J Test Journal: Measurement of the Audio Codec on the DSP **122**

K Test Journal: Verifying the HRTF Filter Selection on the DSP **125**

L Test Journal: Measurement of the ITD in the HRTF Filtering on the DSP **128**

M Test Journal: Verifying the HRTF Filtering Is Correct **132**

Part I

Analysis and Introduction

Chapter 1

Introduction

Most of present-day multimedia experiences uses stereo or surround sound effects to enhance the immersion of the user. However, stereo is not simply having two output channels, and thus it does not sound as intended when played back through headphones. When using headphones, the listener will often experience the sound as appearing inside their head instead of outside it as naturally occurring sounds would. The intended effect of stereo sound is then lost, as only by using properly placed stereo speakers such that both ears will hear both audio sources, with a delay between them, the effect of an entire sound stage can be imitated [1, p. 328].

In addition to this, as mobile phones are supporting more and more types of multimedia experiences people are increasingly consuming entertainment when commuting, relaxing, exercising, etc. More than 70% of YouTube's users watch videos from the smartphone app [2], and the Netflix app has over 500 million installs on Android alone [3]. In these situations, it is not feasible to have the stereo or surround sound setups that is needed to give the intended experience and thus a need for a different solution arises. This solution has to not only solve the problem with loss of the stereo effects but also take into account the fact that contrary to stereo and surround sound setup, the speakers in headphones are fixed to the head of the listener, which gives rise to a set of different problems.

A system that solves this problem must somehow recreate the natural human experience of hearing sound from the surrounding environment. Thus, to get a more detailed picture of this, further investigation is made into the subject of human hearing.

1.1 Virtual Three-Dimensional Audio and Attributes of Human Spatial Hearing

Virtual three-dimensional audio is a simulated effect, the purpose of which is to give a listener the perception of sound coming from different locations in space when, in reality, it is produced in e.g. headphones. In order to do this, virtual three-dimensional audio takes advantage of the human ability of spatial hearing. Spatial hearing is what allows us to discern the direction and distance of a sound source, based on the sound alone. The human auditory system can determine where a sound is coming from, based on what is called spatial attributes in the sound. These attributes are, among others, the difference in arrival time and level of the sound signal at one ear compared to the other, the amount of sound sources, and the effects created by reflections due to the surroundings of the listener [1].

1.1.1 Attributes of Human Spatial Hearing

When localizing a single sound source, two things are determined; the direction of the sound source and the distance of the sound source. Studies have shown that when determining the direction of a sound source, the attributes used includes Interaural Time Difference (ITD) and Interaural Level Difference (ILD) in combination with different dynamic and spectral attributes [4]. A more detailed presentation of how localization works, and a short description of the aforementioned attributes is presented in the rest of this section.

Interaural Time Difference

When the sound waves from a sound source reaches the ears of a listener, they usually do not reach each ear at the same time because one ear is further away from the source than the other. The arrival time difference of the sound between the two ears is the ITD, and since it is closely related to the shape of the head it is different for every individual person [1]. For a given person, the ITD varies as a function of the azimuth angle of the incoming sound wave. When considering sound sources in the horizontal plane, ITD is equal to zero when the source is directly in front of the listener, due to the sound waves reaching both ears simultaneously. However, as the sound source is moved to either the right or left, ITD increases until it reaches its maximum value when the source is directly to the right or left of the listener. Continuing in either direction decreases the ITD until it reaches zero again when the source is directly behind the listener. A sketch illustrating the different distances between the two ears for an incoming wave can be seen in figure 1.1

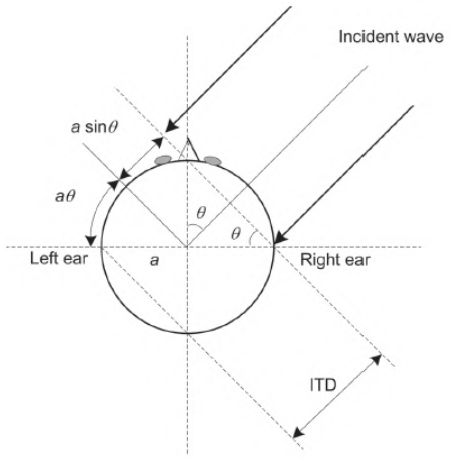


Figure 1.1: Sketch of the different distances between the two ears, when using a spherical head model [1].

Interaural Level Difference

ILD refers to the difference in sound pressure at the right and the left ear respectively. Because of the spherical nature of the head, the sound pressure at the ear furthest away from the sound source is attenuated compared to the other ear, due to a shadowing effect of the head [1]. This attribute is dependent not only on the direction of the sound signal but also on its frequency. For low frequencies, the head shadowing effect is not very prominent and the ILD is nearly negligible at all angles. However, as the frequency increases, the ILD varies in relation to the azimuth angle. The relationship between azimuth angle and ILD for different frequencies, based on a spherical head model, is shown in figure 1.2

As the frequency increases, the ILD does not vary monotonously which results in more than one azimuth angle having the same ILD. This leads to ambiguity when using ILD for localization.

Dynamic and Spectral Attributes

In addition to ITD and ILD, the dynamic and spectral attributes are also important when localizing a sound source. The dynamic attributes are used to counter the ambiguity that comes from using only ITD and ILD. If one only considers the time differences between the two ears, there exists an area in free space referred to as the *cone of confusion*, that propagates outwards from both ears. This is illustrated in figure 1.3

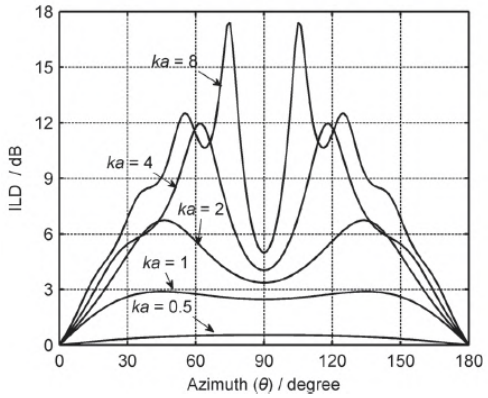


Figure 1.2: Calculated ILD as a function of azimuth at different frequencies, using a spherical head model with radius a =8.75 cm . Here ka is a metric used to describe frequency where $k = 2\pi f c^{-1}$. For reference, $ka = 2$ corresponds to a frequency of 1.2 kHz [1].

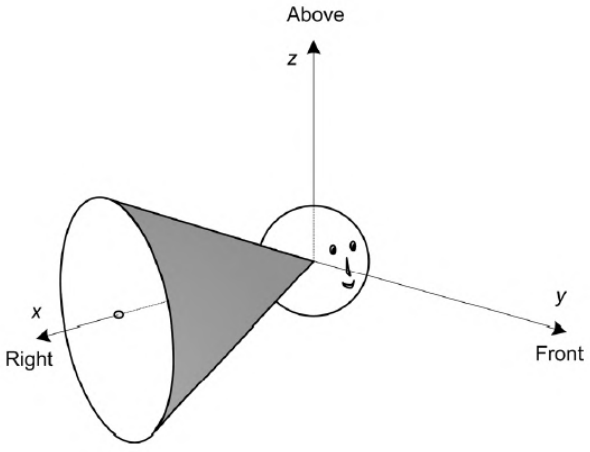


Figure 1.3: Illustration of the *cone of confusion*[1].

At a given distance from the listener, all points on the cone have the same distances to the two ears [1]. This is true, only when the ears are considered as two points in free space, and the spherical head model is discarded. However, for an actual human head with its non-spherical shape, a similar effect exists but no longer as a perfect cone, but a complex contour instead [1].

The way the cone of confusion is countered is by small head movements as this changes the ITD and ILD, this aspect is called dynamic cues. For example, by rotating the head horizontally in either direction, the time and the level differences changes because one ear is closer to the source and the other is further away. This eliminates the ambiguity and localization is possible.

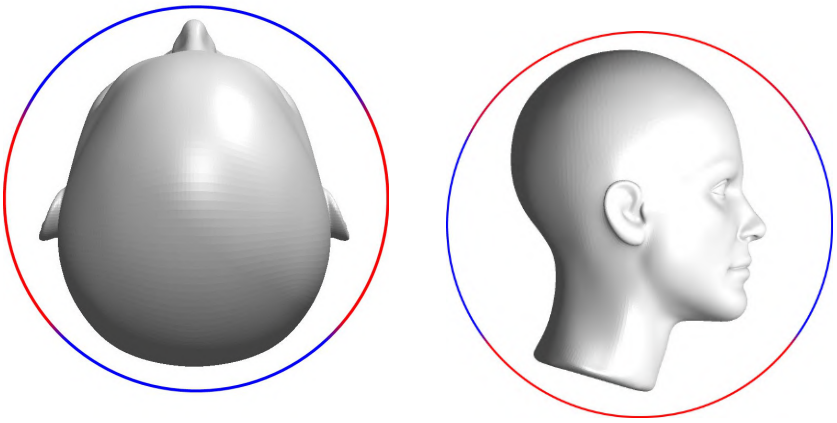
The last attribute described here is the spectral attribute, which is an effect of the pinna. When sound reaches the ear canal, it does so both as a direct component and as a reflected component as a result of the way the pinna is shaped. As the pinna is shaped irregularly, the reflected component of the sound wave varies with the direction of the sound, and thus it contributes to the localization of the sound source[1].

All the above describes how humans naturally perceives and determines the location of a sound source in free space, which is what virtual three-dimensional audio tries to reproduce. This works because when the right spatial attributes are present in a sound signal, it is possible to convince the brain that a sound source is present somewhere in free space when in fact the sound is delivered by two speakers

in e.g. headphones. To create the effect of sound originating somewhere in free space, the sound signal needs to be processed in a way that inserts the right spatial attributes. This is usually done by digital filtering of the original sound signal, based on a transfer function corresponding to the wanted location of the sound in free space. This transfer function is known as a head-related transfer function (HRTF) and there exists an infinite amount of these, corresponding to all continuous directions and distances. These HRTFs describes the filtering effect of the transmission of a sound signal from its source to each of the two ears. Thus, the HRTFs include the effects of all the spatial attributes created by anatomical structures such as head shape and pinnae.

1.1.2 Psychoacoustic Capabilities

In addition to the physical properties and attributes described earlier, there exists some psychoacoustic effects that influence our perception of sounds. Firstly, an important measure is the Minimum Audible Angle (MAA) which describes the ability of a listener to determine a change in position of a sound. The MAA is defined as the smallest change in angular distance between two identical sounds that can reliably be determined [5]. The MAA relies on the direction of the sound source, both horizontally and vertically. In the horizontal plane the lowest MAA is directly to the front and back of the listener and the highest MAA at 90 degrees to either side [5]. One study found that horizontal MAA ranges from 1° directly in front of the listener, to 10° at the side of the head [5]. In the vertical plane the MAA is also lowest directly in front of, and to the back of, the listener and increases with elevation. Figure 1.4 gives an idea of the distribution of the MAA around the head.



(a) Top view. **(b)** Side view.

Figure 1.4: Illustration of the non-uniform distribution of the minimum audible angle at various locations around the head. The red area illustrates areas of high MAA, and blue areas indicate a low MAA.

Next is two effects that both revolve around the case where an auditory event is coupled together with a visual event. The first is an effect often referred to as the *ventriloquism effect*. This effect is a classic example of how, when two senses, here sight and hearing, provide contradicting information, the brain is still able to create a perception of unity in the information [6]. Herbert et al. showed in [7] that when the visual position of a sound source is moved away from the actual position of the sound source, here by using a prism, the perceived origin of the sound tends to move towards the visual position. An example of this effect, and the reason for its name, is how it seems like sound is coming from a ventriloquist's puppet due to the movement of its lips, when in fact it comes from the ventriloquist. The second effect is the ability to accept a slight asynchronicity between visual and auditory events that are expected to be synchronized.

A study by R. Steinmetz et al has investigated how much audio can be out of sync with accompanying video and still be perceived as acceptable for the user [8]. This showed that audio can still be perceived as "in sync" with delays of up to 80 ms between the video and the sound. This is similar to the International Telecommunication Unions official suggestions of 95 ms [9].

When implementing virtual three-dimensional audio these properties and limitations must be accounted for.

1.1.3 Limitations of Virtual Three-Dimensional Audio

Even though it is possible to almost perfectly simulate the effect of the human spatial hearing with the use of HRTFs, some limitations to the technique remain. Some of these limitations arise due to the way the HRTFs are measured. For example, a deviation comes because most of the spatial attributes rely on unique anatomical properties that differs from person to person. Thus, to perfectly replicate the effect for a given person, a personal set of HRTFs must be used. As it is not feasible to create a complete set of HRTFs for every person in the world, instead an approach where a generalization of the HRTFs is used, at the expense of some deviation from each person's natural perception with their own ears.

Although not a limiting factor but something which is worth considering in a virtual three-dimensional audio system, is the signal path from where the sound input is filtered by a HRTF and all the way through to where the sound pressure generated by a pair of headphones reaches the ear drums. For a virtual auditory display to be able to reproduce what is essentially natural occurring sounds, then the D/A conversion, amplification and headphones should have as flat a frequency response as possible. If these parts of the signal chain do not have a flat frequency response, then it must be compensated for in an equalization filter [10].

Additionally, when creating HRTFs, usually some kind of model of a human head and torso is used [1, p. 33-34]. The HRTFs are then recorded inside the head of the model as the model rotates while sound is playing from different locations. However, if the head is not made to rotate separately from the torso, the resulting HRFTs will not correspond to the natural head movements used for example to get the dynamic attributes. This gives some deviation from the natural occurring spatial attributes.

Finally, as the HRTFs represent the transfer functions of a stationary head, the ambiguity of sounds originating from the cone of confusion cannot be located reliably without some additional information about the orientation of the head.

1.1.4 Orientation Tracking for Virtual Three-Dimensional Audio

As discussed in section 1.1 the brain uses a variety of cues, such as dynamic cues, to determine the direction of incoming sound. However, as dynamic cues are tied to head movement, these cannot be described using only HRTFs. To create a similar effect with headphones, the HRTFs must change according to the wearers head movement in real time. To achieve this, the orientation of the head must be tracked in real time to change the HRTFs accordingly. By changing the HTRFs according to head movement the illusion of sound sources in three-dimensional space will be greatly improved, thus enhancing the immersion of the audio experience. [11].

Orientation is defined as the horizontal and vertical angle of the user's head in relation to a fixed reference orientation; e.g. 0° in both the elevation and azimuth planes. If the user moves their

head 20° to the left, the system must move the audio sources opposite that, in order to stay still in three-dimensional space.

1.2 Problem Statement

From the analysis it can be concluded, that mobile devices are extensively used for consuming entertainment. An important part of entertainment is the ability to immerse oneself in it. To increase the immersion of the user, technology such as virtual three-dimensional audio can be used.

However, as virtual three-dimensional audio is based on HRTFs to create the illusion of sounds originating from all around the user, the information gained from dynamic cues is lost. Without the possibility of using dynamic cues, which are very important when determining the direction of sounds, the listener only gains a limited amount of immersion.

Thus, to further increase immersion a system is proposed that uses head tracking to determine the orientation of the head, in combination with virtual three-dimensional audio, thereby adding dynamic cues. To accommodate the wide use of smartphones, it is proposed that the system is developed as an embedded system that can be used with headphones.

Chapter 2

Requirements and Design Overview

In this chapter the requirements for the system are established. The requirements are divided into two groups, functional requirements and technical requirements. Finally, an overview of the system that is to be designed is presented. However, first a small discussion about the scope and limitations of the project.

2.1 Limitations

As stated in the preface this report focuses on the implementation of a real time system on a specific DSP, the TMS320C5535 from Texas Instruments. This means that no other DSP will be considered when trying to implement the system. The detailed hardware restrictions that the DSP imposes upon the design of the system is described in section 2.4. Furthermore, when designing the three-dimensional audio application, only sound sources in the far field, defined as sources at least 1 meter from the head, will be considered. This is due to the complexity in the models of sound sources in the near field, where the exact distance of the sound source has a major influence as opposed to the far field where distance has no influence [1].

2.2 Functional Requirements

The functional requirements specifying the desired functionality of the system are listed in table 2.1. All the functional requirements are prioritized by their importance. The highest priority requirements represent the most basic functionalities of the system and without these the system would not be able to fulfill its intended purpose in any capacity.

Table 2.1: Table of functional requirements and their justifications.

#	Prio.	Specification	Justification
F.1	1	Filter virtual sound sources, using HRTFs exclusively based on the sound source's azimuth coordinate, while assuming a constant elevation angle of 0° as well as a fixed distance to the origin.	As human sensitivity to sound directionality is most profound in the 0° elevation plane [12], reproducing virtual three-dimensional audio for this region is prioritized. Additionally, the distance to the origin is kept constant, as the distance to a sound source does not include information about the directionality [1, p.19].
F.2	1	Perform virtual three-dimensional audio calculations in real time on a DSP.	This is required by the study curriculum for the semester.
F.3	1	Simulate three-dimensional audio so that it accounts for the user's head orientation.	Modelling head orientation allows the user to use dynamic cues to estimate the directionality of sound much more accurately [13]. In turn, the dynamic cues add another layer of immersion to the virtual experience.
F.4	1	Dual channel output for over-ear headphones.	Two channels are required for recreating authentic three-dimensional audio in headphones. Over ear headphones are used, as opposed to in-ears, because much of the literature on the subject primarily considers over-ear [10] [14].
F.5	2	Filter virtual sound sources, using HRTFs exclusively based on the sound source's azimuth and elevation coordinate, with respect to the origin, while assuming a fixed distance to the origin.	While the azimuth region is prioritized, simulating three-dimensional audio for audio sources placed arbitrarily on a sphere around the head will improve immersion. The distance to the origin is kept constant, as the distance to a sound source does not include information about the directionality [1, p.19].
F.6	2	Tuned for a specific audio amplifier and headphones model.	This ensures that the entire audio reproduction chain is accounted for, thus making it possible to improve audio quality by compensating for non-ideal behavior of analog subsystems.
F.7	3	Filter virtual sound sources, using HRTFs based on the sound source's azimuth, elevation and distance coordinate, with respect to the origin.	This should improve immersion and localization of the sound source, as the sound sources exact position is accounted for.
F.8	4	Include capabilities to simulate environmental reflections.	Nearly all occurring sounds contain some form of reflections. These reflections are used in binaural hearing to improve the localization of a sound source for both its direction and distance [1, p. 28-32].

2.3 Technical Requirements

The technical requirements, listed in table 2.2, specify the performance criteria which are used to evaluate the final system. As it is intended that the desired system is to be used by humans, the presented requirements are largely based on psychoacoustic phenomena.

As sound perception is largely a subjective experience based on a variety of factors, such as room acoustics and visual cues, it is difficult to quantify the minimum specifications required to authentically replicate three-dimensional audio. Nevertheless, the technical requirements are formulated to try and encapsulate the key parameters that are applicable to most humans for the purpose of experiencing credible interactive three-dimensional audio.

Table 2.2: Table of technical requirements and their justifications.

#	**Requirement**	**Justification**
T.1	A minimum of two simultaneous sound sources	Two separate sound sources are required to simulate a stereo speaker sound stage including both left and right audio channels [15, p. 17-18]
T.2	A minimum resolution of • 1° for the horizontal forward direction (−75° to 75° azimuth). • 1° for the horizontal backward direction (105° to −105° azimuth). • 10° for the horizontal 75° to 105° and −105° to −75° azimuth. • 1° for the vertical direction 0° to 80° elevation. • 4° for the remaining 80° to 90° elevation. • 1° for the vertical direction 0° to −80° elevation. • 4° for the remaining −80° to −90° elevation. for the HRTF filters	The requirements for the horizontal resolution is based on the minimum audible angle of humans which has been tested by A. W. Mills [5] and verified by Grantham et al. [16]. Similar for the vertical direction, which has been verified by Perrot and Saberi [12]. Information regarding minimum resolution in the vertical directions below the head has not been found. Thus, it is assumed that it mirrors that of the upward direction.
T.3	A minimum resolution of 1° for the orientation tracker.	As the highest resolution for the HRTFs are 1°, the resolution for all orientations must be the same to accommodate this.
T.4	A maximum of 26 ms of delay between head movement, and the relative change of the position of the audio source in the audio playback.	In [13] Jesper Sandvad's system had a minimum latency of 26 ms, and at 96 ms of latency it introduced significant localization errors for test subjects.
T.5	The system must add a maximum of 95 ms of delay to the audio signal(s).	Recommendations show that if the audio from e.g. a movie is lagging behind the video with more than 95 ms people will notice that the audio is delayed [9]. Thus, this needs to be adhered to if the system is to be used for multimedia purposes.
T.6	A maximum of 11° angular displacement error between the subjects heads actual orientation, and its measured orientation.	Based on the minimum perceivable angular displacement between a TV film and corresponding sound source [6, p. 121].

2.4 Digital Signal Processor Hardware

In this section the hardware used in this project is described. As mentioned in section 2.1 the development platform for this project is the TMS320C5535 DSP, as the use of this platform is specified by the semester description as part of the learning objectives for the semester. The TMS320C5535 which is used is on a development board from Digital Spectrum, namely the eZdsp USB stick Development Kit. First the DSP is covered, and thereafter the on-board peripherals are described.

2.4.1 Texas Instrument TMS320C5535 Digital Signal Processor

The TMS320C5535 (C5535 for short) is a low-power 16-bit fixed-point processor with a 100 MHz clock rate. This allows the processor to perform up to 100 million instructions per second (MIPS). The C5535 also features dual Multiply-and-Accumulate (MAC) units which allows up to 200 million MACs to be performed per second. The two MAC units consist of two 17-bit multipliers [17]. There is also one 40-bit ALU, four 40-bit accumulators and a barrel shifter which supports 32 bits right shift and 31 bits left shift. The C5535 has built-in 320 kB RAM which is composed of 64 kB Dual-Access RAM (DARAM) and 256 kB Single-Access RAM (SARAM) [17].
The C5535 also supports a variety of communication protocols, such as SPI, I2C, I2S and UART. Additionally, it supports a wide range of other common DSP features such as DMA and hardware support for circular buffers [17].

2.4.2 On-board Peripherals

The eZdsp development is a small PCB which houses the TMS320C5535 DSP together with a suit of peripherals and required circuitry. The board contains stereo input and output in the form of 3.5 mm jack ports. These are connected to an AIC3204 which is a low power stereo audio codec which contains both an ADC and DAC . The codec communicates input and output via the I2S interface; however, it is controlled via the I2C protocol [18, p. 10].
The board also contains an 8 MB flash chip which communicates with the DSP using the SPI protocol and a Micro SD card connector [18, p. 9]. Finally, the board also has a large expansion edge connector with 58 pins. The 58 pins consists of a variety of GPIO, SPI, I2C, I2S and UART pins as well as power supply and ground pins [18, p. 22]

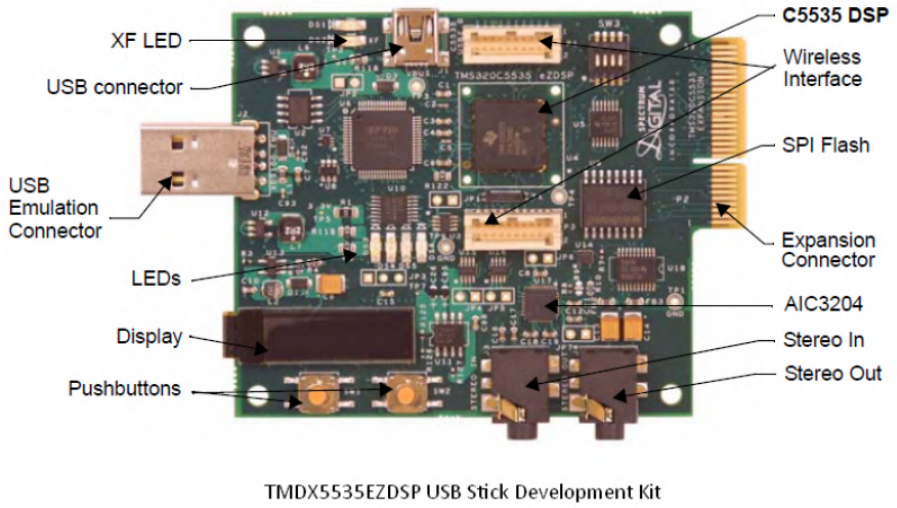


Figure 2.1: Image of the eZdsp USB Stick Development Kit by Digital Spectrum [18, p. 8].

2.5 System Design Overview

Now that the system requirements have been defined, it is possible to embark upon the system design. Figure 2.2 presents an overview of the key functionalities that must be present on the desired system:

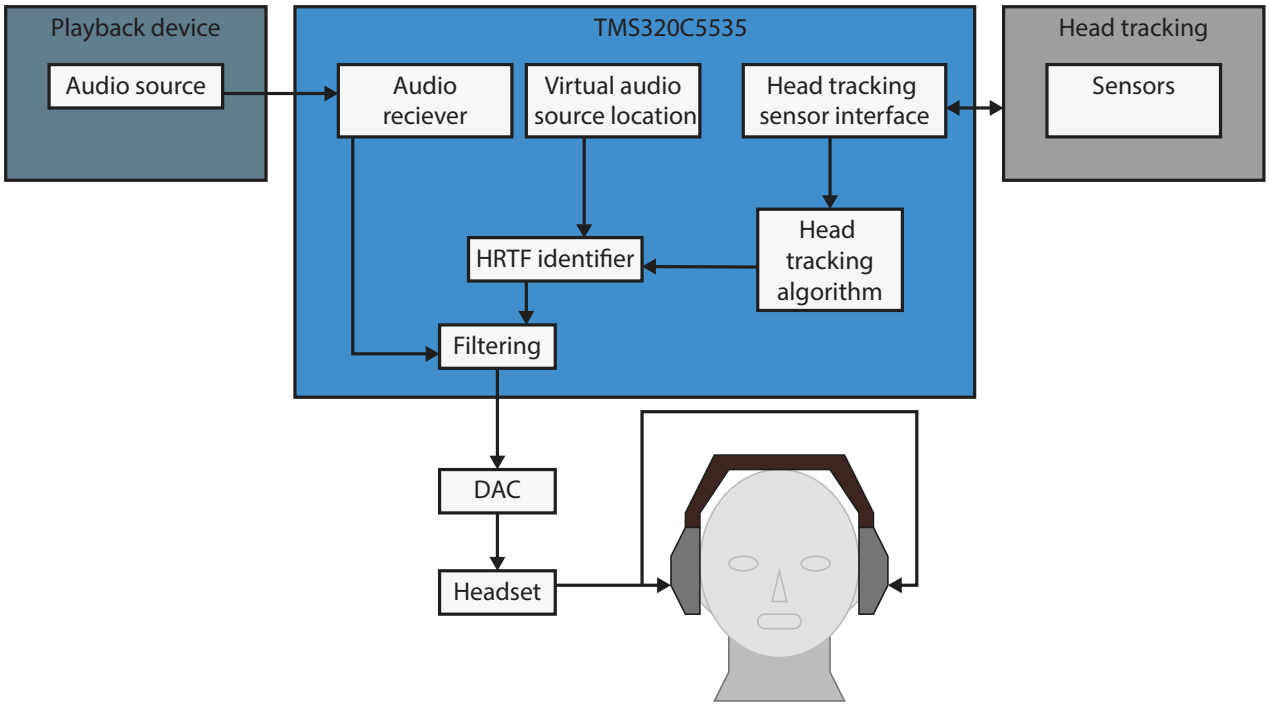


Figure 2.2: Block diagram of the desired system.

To create three-dimensional audio, an audio playback device generates a stream of one or more audio channels which are streamed to the DSP. Each audio channel is associated with a virtual location, generated inside the DSP. Collectively each audio channel and its corresponding virtual location is referred to as an audio object. A visualization of how three virtual audio objects might be placed is shown in figure 2.3:

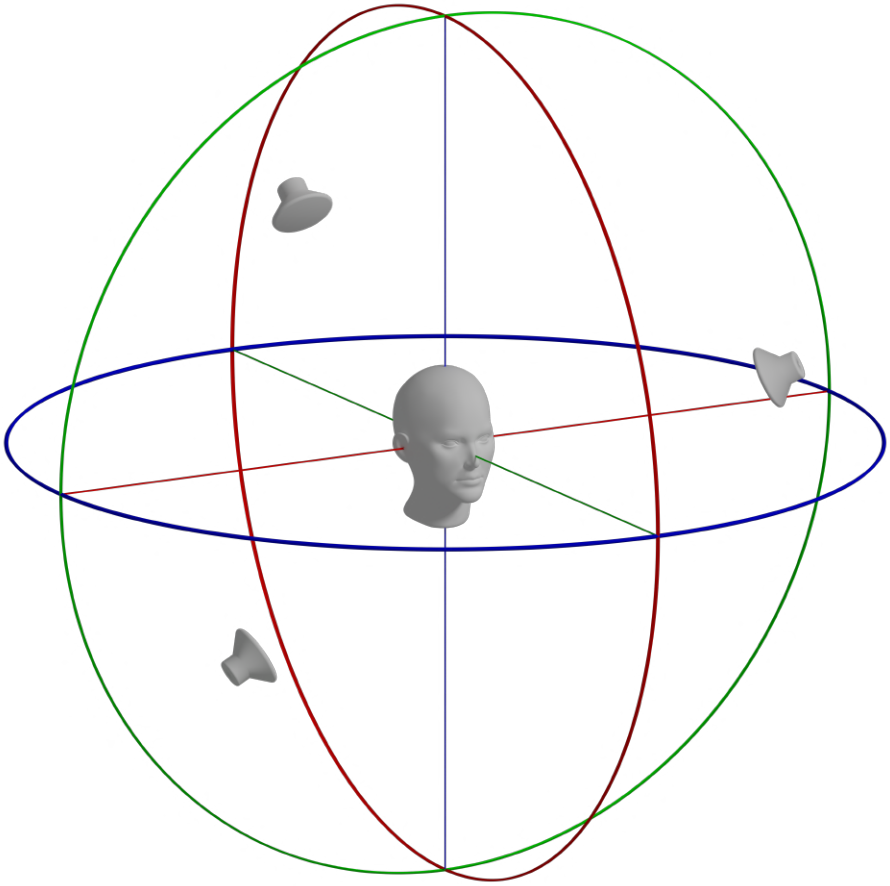


Figure 2.3: Illustration of the virtual three-dimensional space with three virtual sound objects.

Besides audio, the DSP also receives real time data from sensors monitoring the user's head. This sensor data is then used by the DSP to estimate the instantaneous orientation of the head. The estimated orientation is then used in combination with each audio objects virtual position to select a pair of suitable HRTFs (one for each ear) for each audio object. These HRTFs are then applied to each of their corresponding audio streams. Finally, the audio stream is converted to an analog signal and sent to the user's headphones.

From the system overview it can be seen that the system can be divided into two independent subsystems; Head tracking and HRTF filtering. The development of these two subsystems is described in the following two parts. The analysis, development, implementation, and testing of the head tracking can be found in part II. Likewise, the corresponding chapters concerning HRTF filtering can be found in part III.

Part II

Head Tracking Design

Chapter 3

Head Tracking Technology Analysis

This chapter will investigate the different technologies currently used in head tracking, with the goal of finding the technology best suited for the system described in chapter 2.5. Additionally, a more thorough description of the chosen technology will be presented together with an introduction to the principles of determining orientation.

3.1 Methods for Tracking Head Movement

There are multiple ways to track the position and orientation of a head in three-dimensional space, all of which have their own pros and cons. Many of these technologies are used in the modern virtual/augmented reality systems. However, there are different requirements for head tracking when tracking the head for visuals compared to audio purposes. More specifically, the requirements for tracking of the head for visual media has much higher requirements for achieving full immersion [19]. Thus, many techniques used for head tracking have better accuracy than required for audio. Additionally, as the distance to an audio source is less relevant for immersion in far field audio applications, it is more important to track orientation, as opposed to position [1][p. 18].

Camera Tracking

Camera tracking is performed using one or more cameras. Usually these are combined with markers for the camera to track. Such a setup requires that there are cameras in place where the tracking is to be done. This type of tracking can be very precise, as it has no drift and can measure both position and orientation. The primary disadvantages of such a system is that it requires cameras to be set up in a stationary position, and its computational demands might make it difficult to implement on small low powered embedded systems. However, depending on the setup, it is possible to achieve a tracking resolution with sub-millimeter accuracy [20].

Tracking with Magnetic, Angular Rate, and Gravity Sensors

Tracking with Magnetic, Angular Rate, and Gravity Sensors (MARG) is done using an accelerometer, a gyroscope and a magnetometer. For the purpose of position tracking, measured accelerations can theoretically be double integrated to obtain the instantaneous position. However, due to sensor biases and drift, double integration is prone to unbounded growth, and as such the estimated position will quickly become too erroneous for any realistic use cases. On the other hand, orientation tracking can be much more accurate, as orientation algorithms relies on the orientation of static fields, such as Earth's magnetic field as well as gravitational acceleration, to infer the sensors own orientation in space.

While MARG tracking is not suitable for tracking position, it does offer a few advantages. For example, since MARG tracking is not dependent on external sensors (such as cameras), there are few limitations as to where the tracking can take place. Additionally, MARG's can be relatively cheap devices[21] compared to other systems[22][23], thus making them suitable for consumer devices. Finally, the required tracking algorithm can often be implemented on embedded processors due to their relatively low computational requirements [24].

Sender-Receiver setups

There are other ways to track objects, of which many use a sender and a receiver. When multiple distances are known between the senders and receivers, the position can be tracked using triangulation. Examples of such systems are electromagnetic, laser and ultrasound tracking. An electromagnetic setup such as the Polhemus Flock of Birds can achieve a tracking resolution of less than one millimeter and rotational resolution of less than a tenth of a degree [25]. However, like camera tracking this system also require a setup of external equipment.

3.1.1 Choice of Tracking Method

Based on the above analysis of different technologies MARG tracking is deemed the best choice. While both camera tracking and sender-receiver setups allows for the possibility of greater accuracy and precision, both methods are fundamentally incompatible with the problem statement, as they require the use of external base stations to function.

3.2 Chosen MARG Sensor

The specific MARG chosen in this project is the MPU9250 from InvenSense. Figure 3.1 shows the breakout board used to contain the chip.

Figure 3.1: Picture of the MPU9250 breakout board from Sparkfun [26].

This chip was chosen as it contains all three MARG sensors; an accelerometer, a gyroscope and a magnetometer, in a single package. Additionally, it supports data transfer over both SPI and I2C, which allows for greater flexibility in the development process. Following is a short description of each of the internal sensors, what they measure and how they aligned.

3.2.1 Accelerometer

An accelerometer is a sensor that measures acceleration by measuring the displacement of a proof mass. The MPU-9250's 3-Axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially [27]. The MPU-9250 accelerometer can be configured to support acceleration ranges of $\pm 2g$, $\pm 4g$, $\pm 8g$, or $\pm 16g$ [27]. As each signal is sampled using a 16-bit ADC,

a higher range also results in a lower resolution. The accelerometer supports a sample rate of up to 4 kHz [27].

3.2.2 Gyroscope

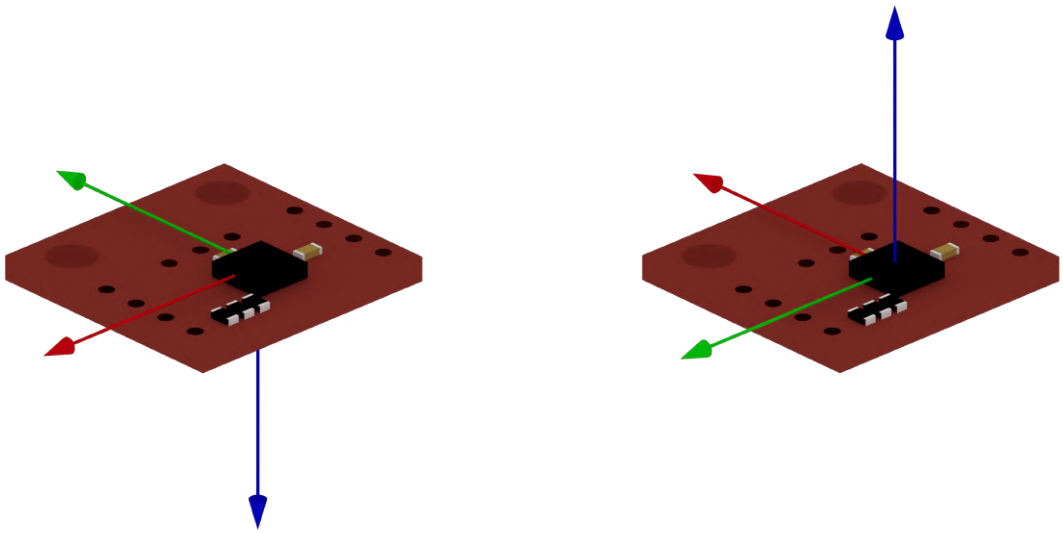
The MPU-9250 consists of three independent vibratory Micro Electro-Mechanical Systems (MEMS) rate gyroscopes, which detect rotation about the X-, Y-, and Z- Axes. When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff [27]. The gyroscope supports ranges of ±250, ±500, ±1000, or ±2000 degrees per second [27]. Additionally, the gyroscope supports sample rates up to 32 kHz [27].

3.2.3 Magnetometer

The 3-axis magnetometer uses highly sensitive Hall sensor technology to measure the magnetic flux density of Earth's magnetic field [27]. It only supports one range configuration: ±4800 µT. Compared to the accelerometer and gyroscope, the magnetometers maximum sample rate of 100 Hz is quite low.

3.2.4 Axis Layout

The axes of the three internal sensors of the MPU-9250 are not aligned. The accelerometer and gyroscope share the same axis definition. However, compared to them, the definition of the magnetometer Z-axis is inverted, and X and Y axes are switched. This is illustrated in figure 3.2.



(a) Placement of axes on the magnetometer.

(b) Placement of axes on the accelerometer and gyroscope.

Figure 3.2: Comparison of magnetometer and accelerometer/gyroscope axis alignment.

Thus, the axes must be aligned in software if the orientation algorithm is to work as expected.

Another prerequisite for calculating orientation is that the sensors are calibrated properly. For a detailed explanation of how this is performed, see appendix C.

3.3 Choice of Orientation Algorithm

There exist many methods to calculate complete three-dimensional orientation [28]. However, many of these are computationally intensive and require high numerical precision. These computational demands may prove difficult to overcome on the C5535 DSP 100 MIPS 16-bit fixed point architecture. For example, it is possible to calculate orientation using an extended Kalman filter [28, p.37]. However, to implement this algorithm matrix inversion is required. Not only is this computationally expensive [28, p.35], it may prove difficult to accurately calculate the matrix inverse of ill-conditioned matrices on the 16-bit fixed point DSP [29, p.94]. Instead, a more computationally simple and numerically robust algorithm is sought.

An orientation algorithm by Madgwick et al [30] fulfills these requirements. This algorithm was designed to be computationally efficient. Furthermore, unlike methods like the Kalman filter, it does not require matrix inversion. Thus, this algorithm will form the basis for the orientation algorithm in this project. It should be noted that this algorithm was designed for floating point, and thus special care must be taken when implementing it in fixed point.

Chapter 4

Design of Head Tracking Application

From the analysis it was determined that for the purposes of the intended system a MARG sensor array combined with sensor fusion in the form of a complementary filter would be the preferred choice. In this chapter the design of the head tracking application based on this is described. To get an introduction of the problem to be solved a brief description of how orientation can be represented in three-dimensional space is found below.

4.1 Representation of Orientation in Three-Dimensional Space

There exist many methods for representing orientation in three-dimensional space. Two of these are *Euler Angles* and *Quaternions*. Euler Angles uses a set of three rotations, ϕ, θ and ψ, to represent an orientation. This set specify three successive rotations around three different axes in a very specific order. The reason that the order of the rotations is so important is because rotations in three-dimensional have an important property: they do not commute. What this means, is that if rotation 1 is done before 2 the resulting orientation is different than if rotation 2 is done before rotation 1 [31].

Quaternions are a number system, that extends the complex numbers. As such, a quaternion is a 4-dimensional number with 3 complex parts. Quaternions are especially useful for representing orientations and rotations in three-dimensional space, because they never give an ambiguous expression for an orientation, as e.g. Euler Angles can do under certain circumstances. Due to this, quaternions are chosen to represent orientation in the head tracking application. A more thorough description of quaternions and how they are used in calculations can be found in appendix A.

4.2 Obtaining Orientation from MARG Sensors

To get the orientation of a body in three-dimensional space, measurements from a tri-axis MARG sensor array can be used. In the following sections methods to get orientation from these sensors are described. All quaternions in this section are unit quaternions as they are describing pure rotations.

4.2.1 Orientation from Gyroscope Data

The tri-axis gyroscope measures the angular velocity in three perpendicular directions; x, y and z. These measurements can be arranged as a pure quaternion, ω, as seen in equation (4.2.1).

$$\vec{\omega} = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix}$$

In order to get an orientation from angular velocity, integration is needed. The measured angular velocity tells something about how the orientation of the sensor changes over time. The change in orientation over time is defined as the time derivative of the orientation. If the orientation is represented by a quaternion, $\vec{q}$, then the time derivative can be noted as $\dot{\vec{q}}$. By integrating $\dot{\vec{q}}$, the instantaneous orientation can be acquired. Thus, an expression of $\dot{\vec{q}}$ described by ω must be found.

Orientation is related to the position of a vector, and thus a relation from physics comes to mind when trying to find its derivative. For a three-dimensional vector, $\vec{s}(t)$, rotating with angular velocity, $\vec{\omega}(t)$, the velocity at the tip is given as:

$$\frac{\mathrm{d}\vec{s}}{\mathrm{d}t} = \vec{\omega} \times \vec{s} \tag{4.1}$$

Now, because the angular velocity vector is perpendicular to $\vec{s}(t)$, and thus their dot product is zero, the equation can be written in quaternion form:

$$\frac{d\vec{s}}{dt} = \vec{\omega} \otimes \vec{s} \tag{4.2}$$

where both quaternions are pure quaternions and the vector parts are $\vec{\omega}(t)$ and $\vec{s}(t)$ respectively. This is seen to be true by inspecting the definition of quaternion multiplication in equation (A.3). If then, the instantaneous position of the vector $\vec{s}(t)$ is represented as a rotation performed by a quaternion, $\vec{q}$, on an initial vector $\vec{s}_0$, equation (4.2) becomes:

$$\frac{\mathrm{d}\vec{s}}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\left[\vec{q} \otimes \vec{s}_0 \otimes \vec{q}^{\,*}\right] = \vec{\omega} \otimes \vec{s} \tag{4.3}$$

As with normal calculus, the product rule of derivatives holds for quaternion products as well, however, the non-commutativity of quaternion multiplication should be kept in mind when doing the calculations [31]. The derivative can then be calculated as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\vec{q} \otimes \vec{s}_0 \otimes \vec{q}^{\,*}\right] = \dot{\vec{q}} \otimes \vec{s}_0 \otimes \vec{q}^{\,*} + \vec{q} \otimes \vec{s}_0 \otimes \dot{\vec{q}}^{\,*}$$

now if this is combined with the equation in (4.3), the following relation between $\dot{q}$ and $\vec{\omega}$ can be found.

$$\dot{\vec{q}} \otimes \vec{s}_0 \otimes \vec{q}^{\,*} + \vec{q} \otimes \vec{s}_0 \otimes \dot{\vec{q}}^{\,*} = \vec{\omega} \otimes \vec{s} \tag{4.4}$$

However, the last term, the derivative of the conjugate, is not defined so far. Instead of trying to evaluate that expression directly, it has proven to be simpler to find a relationship between $\dot{\vec{q}}$ and $\dot{\vec{q}}^{\,*}$ [31]. As $\vec{q}$ is of unit length the quaternion product $\vec{q} \otimes \vec{q}^{\,*}$ is equal to one. If the derivative of this product is taken and the product rule is used the following relationship is found:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left(\vec{q} \otimes \vec{q}^{\,*}\right) &= \frac{\mathrm{d}}{\mathrm{d}t}1 \\ \dot{\vec{q}} \otimes \vec{q}^{\,*} + \vec{q} \otimes \dot{\vec{q}}^{\,*} &= 0 \\ -\vec{q}^{\,*} \otimes \dot{\vec{q}} \otimes \vec{q}^{\,*} &= \dot{\vec{q}}^{\,*} \end{aligned} \tag{4.5}$$

By substituting for $\dot{\vec{q}}^{\,*}$ in equation (4.4) and expressing $\vec{s_0}$ in terms of $\vec{s}$ it is possible to get the following relation:

$$\dot{\vec{q}} \otimes \vec{q}^{\,*} \otimes \vec{s} - \vec{s} \otimes \dot{\vec{q}} \otimes \vec{q}^{\,*} = \vec{\omega} \otimes \vec{s} \tag{4.6}$$

Further algebra is needed to simplify this expression; however this has been omitted in this report. A more thorough description of this can be found in [31].
From equation (4.6) a differential equation describing the relationship between $\dot{\vec{q}}$ and $\vec{\omega}$ can be expressed as:

$$\dot{\vec{q}} = \frac{1}{2}\vec{q} \otimes \vec{\omega} \tag{4.7}$$

Now, this expression describes the continuous-time relationship, but in order to implement this on the DSP it needs to be discretized. Assuming a high sampling rate the difference between two consecutive quaternions should be very small. This allows us to substitute the quaternion, $\vec{q}$, in equation (4.7)

with the calculated quaternion from last iteration as seen in equation (4.8). Then, by using Euler's method, equation (4.9) gives an expression for the orientation at time t.

$$\dot{\vec{q}}_{\omega,t} = \frac{1}{2}\vec{q}_{t-1} \otimes \vec{\omega}_t \tag{4.8}$$

$$\vec{q}_{\omega,t} = \vec{q}_{t-1} + \dot{\vec{q}}_{\omega,t}\Delta t \tag{4.9}$$

where the subscript ω indicates that the quaternion was estimated by using angular velocity.

4.2.2 Orientation from Accelerometer and Magnetometer

Another way to get an estimate of the orientation of the sensor, is by using measurements from the tri-axis accelerometer and magnetometer. These sensors measure the direction of a field in the Earth frame, the gravitational and the magnetic field respectively. As only the directions of the fields are of interest, measured vectors are always normalized. The sensors measure relative to the sensor frame and thus if the direction of the field in the Earth frame is known, the relative orientation of the sensor frame in the Earth frame can be calculated. However, measurements from one sensor alone cannot give a unique representation of the sensor's orientation. For a given orientation of the sensor, the same measurements would be measured if the sensor were to be rotated around the axis of the direction of the field that it measures.

Thus, to get a unique orientation, a combination of the measurements from the two sensors are needed. A unique solution can be found by finding the quaternion, $\vec{q}$, that rotates the measured directions of the two fields in the sensor frame to their respective reference directions in the Earth frame. The reference direction of gravity is defined as directly downwards and the reference direction of the magnetic field is defined as the inclination of the field at Aalborg. At the time of writing, the inclination in Aalborg is approximately 70° to the horizontal [32].
The general case of finding the difference between a measured sensor direction, $\vec{d}$, rotated by a quaternion, $\vec{q}$, and the reference direction, $\vec{s}$, can be described using the following base function:

$$\vec{f}\left(\vec{q}, \vec{d}, \vec{s}\right) = \vec{q}^{\,*} \otimes \vec{d} \otimes \vec{q} - \vec{s} \tag{4.10}$$

where:

$$\vec{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}$$
$$\vec{d} = \begin{bmatrix} 0 & d_x & d_y & d_z \end{bmatrix}$$
$$\vec{s} = \begin{bmatrix} 0 & s_x & s_y & s_z \end{bmatrix}$$

To identify the quaternion that rotates the measured direction, $\vec{d}$, into the reference direction, $\vec{s}$, equation 4.10 must be set equal to a zero vector and solved with respect to $\vec{q}$. However, as $\vec{q}$, and in extension $\vec{f}\left(\vec{q}, \vec{d}, \vec{s}\right)$, are four dimensional entities, it is difficult to derive an analytical solution. Instead it is more practical to solve this problem numerically, by means of an optimization method.

As optimization algorithms attempts to maximize or minimize functions, the function to be minimized, called the objective function, must map the output to a one-dimensional real number. This is required as inequalities are not defined for multidimensional number. For example, the following statement is void:

$$\begin{bmatrix} 0.1 \\ 2 \end{bmatrix} < \begin{bmatrix} 0.8 \\ 1 \end{bmatrix} \tag{4.11}$$

One common way to map the output to $\mathbb{R}$ is to apply a vector norm. There exist infinite types of norm functions. However, for this problem it intuitively makes sense to use the 2-norm (also called the Euclidean norm), to estimate the error between $\vec{q}^{\,*} \otimes \vec{d} \otimes \vec{q}$ and $\vec{s}$. To simplify computation and mathematical derivations required later, the squared 2-norm is used. This simplifies the expression, as it eliminates the use of square roots. Thus, the objective function mapping equation 4.10 from $\mathbb{R}^4$ to $\mathbb{R}^1$ is defined as:

$$F\left(\vec{q}, \vec{d}, \vec{s}\right) = \left|\left|\vec{f}\left(\vec{q}, \vec{d}, \vec{s}\right)\right|\right|^2 \tag{4.12}$$

The quaternion, $\vec{q}$, can then be found as the solution to:

$$\min_{\vec{q} \in \mathbb{R}^4} F\left(\vec{q}, \vec{d}, \vec{s}\right) \tag{4.13}$$

As stated before, each of the sensors, by themselves, cannot give a unique solution for the true orientation. The solutions for the objective functions with only measurements from one sensor lies along a line, however if the objective functions are combined as seen in equation (4.14), the solution lies in a single point.

$$\vec{f}_\nabla\left(\vec{q}, \vec{g}, \vec{a}, \vec{b}, \vec{m}\right) = \begin{bmatrix} \vec{f}_g\left(\vec{q}, \vec{g}, \vec{a}\right) \\ \vec{f}_b\left(\vec{q}, \vec{b}, \vec{m}\right) \end{bmatrix} \tag{4.14}$$

$$F_\nabla\left(\vec{q}, \vec{g}, \vec{a}, \vec{b}, \vec{m}\right) = \left|\left|\vec{f}\left(\vec{q}, \vec{g}, \vec{a}, \vec{b}, \vec{m}\right)\right|\right|^2 \tag{4.15}$$

where $\vec{g}$ and $\vec{b}$ are the reference directions of Earth's gravity and magnetic field respectively, $\vec{a}$ and $\vec{m}$ are the measured directions of the same fields respectively and $\vec{q}$ is the quaternion that rotates the measured directions of the field into their respective references. The orientation obtained from the optimization problem is denoted as $\vec{q}_{\nabla,t}$.

4.2.3 Solving the Optimization Problem

To minimize the expression:

$$\min_{\vec{q} \in \mathbb{R}^4} \left|\left|F_\nabla\left(\vec{q}, \vec{g}, \vec{a}, \vec{b}, \vec{m}\right)\right|\right|^2 \tag{4.16}$$

an optimization algorithm is required. Ideally this algorithm should be easy to compute, fast to converge, and provide an accurate solution to the minimization problem. To select the best algorithm for this specific problem requires a deep and thorough understanding of theoretical optimization. As this approach would likely require years of mathematical study, it is instead decided to select a variety of common optimization algorithms, and then identify the best one using empirical means. While this approach is only able to identify the best algorithm out of the ones tested, it is relatively simple to perform.

For this experiment it is decided to compare the performance of five different algorithms. Four of these algorithms attempt to find the minimum of the objective function using a steepest descend approach. However, each algorithm identifies the step size of each iteration by different means. A fifth algorithm attempts to solve the optimization problem using a Gauss Newton approach. These algorithms were benchmarked with respect to their running time, as well as how well they minimized the objective function. From these results, the best algorithm was chosen. Further descriptions and results of this test can be found in appendix B.

Optimization Overview

For a given continuous function, $f(x)$, at a point, x, belonging to the domain of f there exists a number of directions, $\vec{d}$, along which the function f is decreasing, if x is not an extremum point. In mathematical terms, for some step, $\alpha \geq 0$, and a descending direction, $\vec{d}$, the following holds [33, p. 121]:

$$g\left(\vec{x}\right) > g\left(\vec{x} + \alpha\vec{d}\right) \tag{4.17}$$

For example, a contour graph of the function $x_1^2 + 2x_2^2$ is shown in figure 4.1. On this graph, the point A is shown. For this point the shaded region and red vectors illustrate the infinite number of decreasing directions that may be chosen. The step size, α, for a given direction, must be such that the new point on the graph decreases the function output [33, p. 123].

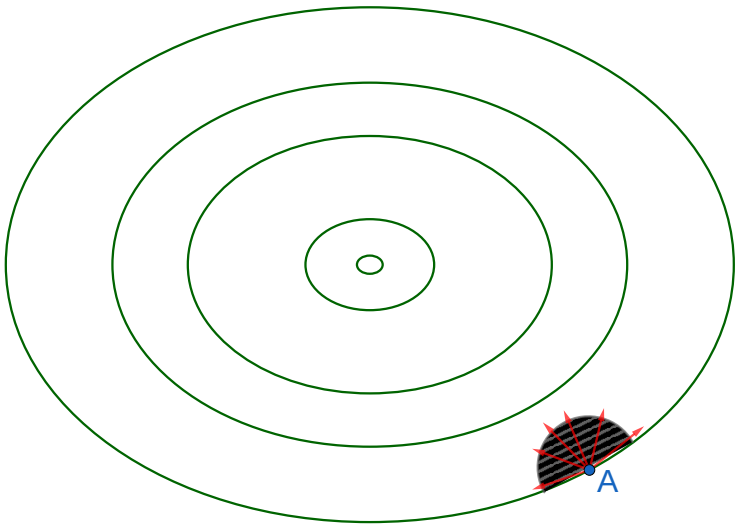


Figure 4.1: Illustration showing the infinite decreasing directions at the point A.

If $\vec{d}$ points in the direction of the minimizer, then it is possible to find a value of α which solves the optimization problem. However, as this direction is not easy to determine, an iterative method must instead be used, where multiple directions and steps sizes are determined in a successive manner [33, p. 123]. This principle is illustrated in figure 4.2.

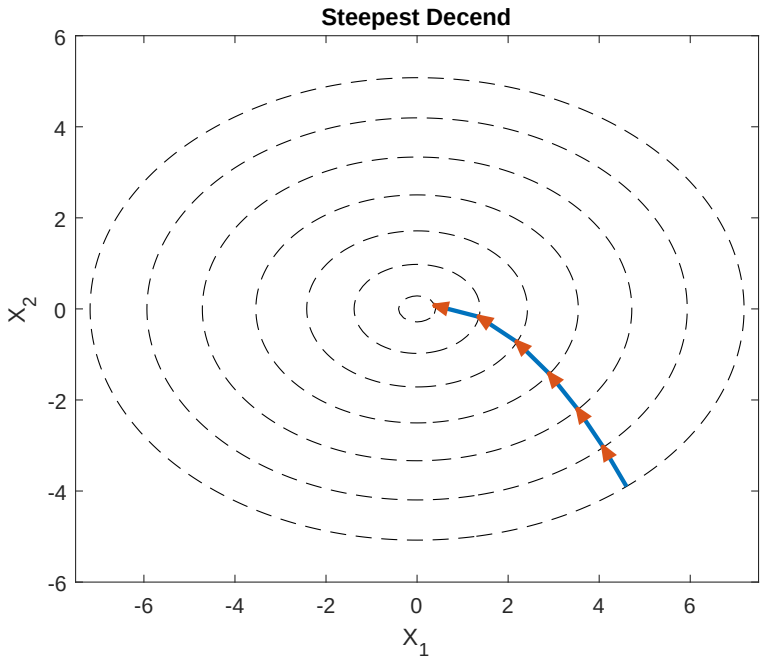


Figure 4.2: Illustration showing how multiple iterations of the gradient descend method converges to the local minimum.

Each successive point shown in the figure, is found using the following method:

$$\begin{aligned} \vec{x}_1 &= \vec{x}_0 + \alpha_0 \vec{d}_0 \\ \vec{x}_2 &= \vec{x}_1 + \alpha_1 \vec{d}_1 \\ &\vdots \\ \vec{x}_{k+1} &= \vec{x}_k + \alpha_k \vec{d}_k \end{aligned} \tag{4.18}$$

In the Steepest Descend method the direction is defined as the negative of the gradient of the function, as this represents the direction of steepest descend [33, p. 121]. Thus:

$$\vec{d}_k = -\nabla g\left(\vec{x}_k\right)$$

What differentiates each of these methods, is the way their step size, α_k, is found. If it is desired to use the fewest possible iterations to solve a minimization problem using a steepest descend approach, then each step will be orthogonal to each other[33, p. 123-124]. This is illustrated in figure 4.3.

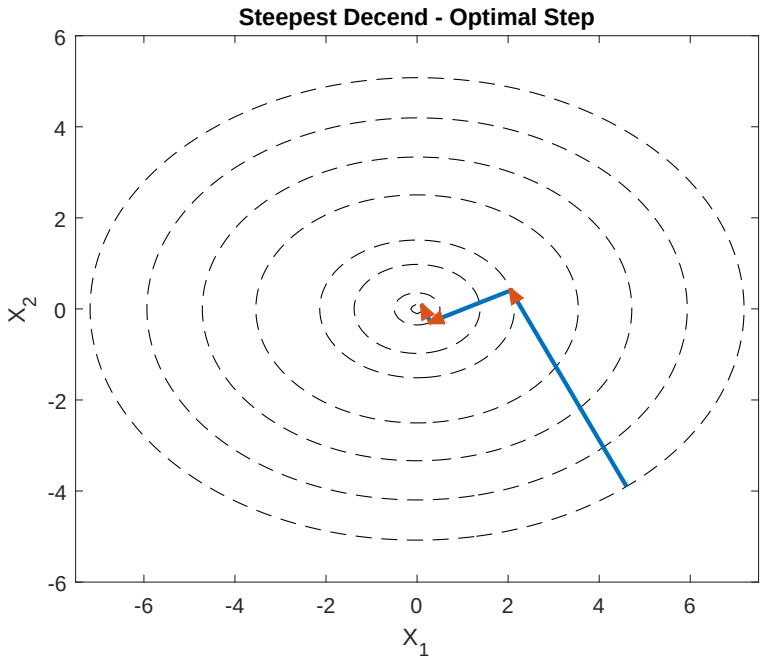


Figure 4.3: Illustration of gradient descend using an ideal step size.

However, to realize this, one needs to find the optimal value of α_k for each iteration. This can be a computationally expensive task, and as such it may prove advantageous to use a method requiring more iterations to solve the problem but where each iteration is easier to compute. Four different methods have been chosen for analysis. The details regarding the optimal algorithm identified in appendix B is described in the following sections. Details regarding the competing algorithms are found in appendix B.

Finding the Step Size Using a Second Order Approximation

To find an analytic solution to the optimal value of α_k the first step is to derive an expression for $g\left(\vec{x}_k + \alpha_k \vec{d}_k\right)$. Then the derivative of the expression with respect to α_k is found. Finally, by equating the derivative to zero and solving, the minimum of the expression can be found.

To approximate $g\left(\vec{x}_k + \alpha_k \vec{d}_k\right)$, a second order approximation is derived. The second order Taylor approximation of $g\left(\vec{x}\right)$ near $\vec{x}_k$ is formulated as:

$$g\left(\vec{x}\right)|_{\vec{x}_k} \approx g\left(\vec{x}_k\right) + \nabla g\left(\vec{x}_k\right)^T \left(\vec{x} - \vec{x}_k\right) + \frac{1}{2}\left(\vec{x} - \vec{x}_k\right)^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right)\left(\vec{x} - \vec{x}_k\right) \tag{4.19}$$

When evaluating the function at $\vec{x} = \vec{x}_k + \vec{h}$ the expression becomes[33, p. 29]:

$$\begin{aligned} g\left(\vec{x}_k + \vec{h}\right)|_{\vec{x}_k} &\approx g\left(\vec{x}_k\right) + \nabla g\left(\vec{x}_k\right)^T \left(\vec{x}_k + \vec{h} - \vec{x}_k\right) + \frac{1}{2}\left(\vec{x}_k + \vec{h} - \vec{x}_k\right)^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right)\left(\vec{x}_k + \vec{h} - \vec{x}_k\right) \\ &\approx g\left(\vec{x}_k\right) + \nabla g\left(\vec{x}_k\right)^T \vec{h} + \frac{1}{2}\vec{h}^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right)\vec{h} \end{aligned}$$

Substituting $\vec{h}$ with $\alpha_k \vec{d}_k$, it follows that:

$$g\left(\vec{x}_k + \alpha_k \vec{d}_k\right)|_{\vec{x}_k} \approx g\left(\vec{x}_k\right) + \alpha_k \nabla g\left(\vec{x}_k\right)^T \vec{d}_k + \alpha_k^2 \frac{1}{2}\vec{d}_k^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right)\vec{d}_k$$

To find the point where α_k minimizes the function, the derivative with respects to α_k is evaluated [33, p. 125]:

$$\frac{\mathrm{d}}{\mathrm{d}\alpha_k} g\left(\vec{x}_k + \alpha_k \vec{d}_k\right)|_{\vec{x}_k} \approx \nabla g\left(\vec{x}_k\right)^T \vec{d}_k + \alpha_k \vec{d}_k^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right) \vec{d}_k$$

Now the expression is set equal to 0 and solved for α_k to get an expression for its minimizer[33, p. 125]:

$$\begin{aligned} \nabla g\left(\vec{x}_k\right)^T \vec{d}_k + \alpha_k \vec{d}_k^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right) \vec{d}_k &= 0 \\ \alpha_k \vec{d}_k^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right) \vec{d}_k &= -\nabla g\left(\vec{x}_k\right)^T \vec{d}_k \\ \alpha_k &= \frac{-\nabla g\left(\vec{x}_k\right)^T \vec{d}_k}{\vec{d}_k^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right) \vec{d}_k} \end{aligned}$$

Substituting $\vec{d}_k$ with $-\nabla g\left(\vec{d}_k\right)$ the expression for the step size becomes:

$$\begin{aligned} \alpha_k &= \frac{\nabla g\left(\vec{x}_k\right)^T \nabla g\left(\vec{x}_k\right)}{\nabla g\left(\vec{x}_k\right)^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right) \nabla g\left(\vec{x}_k\right)} \\ &= \frac{||\nabla g\left(\vec{x}_k\right)||^2}{\nabla g\left(\vec{x}_k\right)^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right) \nabla g\left(\vec{x}_k\right)} \end{aligned}$$

Substituting this expression for α_k into equation 4.18, the final equation for finding $\vec{x}_{k+1}$ is found:

$$\vec{x}_{k+1} = \vec{x}_k + \frac{||\nabla g\left(\vec{x}_k\right)||^2}{\nabla g\left(\vec{x}_k\right)^T \boldsymbol{H}\left(g\left(\vec{x}_k\right)\right) \nabla g\left(\vec{x}_k\right)} \nabla g\left(\vec{x}_k\right)$$

Substituting $g\left(\vec{x}_k\right)$ with the objective function, F_∇, shown in equation 4.15 and solving with respect to $\vec{q}$, results in an iterative solution for $\vec{q}_\nabla$ (arguments of F_∇ omitted to enhance readability):

$$\vec{q}_{\nabla,t} = \vec{q}_{t-1} + \frac{||\nabla_q F_\nabla||^2}{\nabla_q F_\nabla^T \boldsymbol{H}_q\left(F_\nabla\right) \nabla_q F_\nabla} \nabla_q F_\nabla \tag{4.20}$$

However, it can be difficult to obtain an analytic expression for the Hessian and therefore a numerical approximation is made. To get to the Hessian, first the gradient is investigated.
To find the gradient, note the following relation:

$$\begin{aligned} F_\nabla &= \left|\left|\vec{f}_\nabla\right|\right|^2 \\ &= \vec{f}_\nabla^T \cdot \vec{f}_\nabla \\ &= \sum_{i=1}^{8} \vec{f}_{\nabla,i}^2 \end{aligned}$$

From this the partial derivatives making up the gradient can be found using the chain rule [33, p. 138]:

$$\frac{\partial F_\nabla}{\partial \vec{q}_j} = \sum_{i=1}^{8} 2\vec{f}_{\nabla,i} \frac{\partial \vec{f}_{\nabla,i}}{\partial \vec{q}_j} \tag{4.21}$$

for $j = 1, 2, 3, 4$.
The complete expression of the gradient can then be written as [33, p. 138]:

$$\nabla_q F_\nabla = 2\boldsymbol{J}_q^T \vec{f}_\nabla$$

Similarly, from equation (4.21), the components of the Hessian can be expressed as [33, p. 139]:

$$\begin{aligned}\frac{\partial^2 F_\nabla}{\partial \vec{q}_k \partial \vec{q}_j} &= \sum_{i=1}^{8} 2\left(\frac{\partial \vec{f}_{\nabla,i}}{\partial \vec{q}_k}\frac{\partial \vec{f}_{\nabla,i}}{\partial \vec{q}_j} + \vec{f}_{\nabla,i}\frac{\partial^2 \vec{f}_{\nabla,i}}{\partial \vec{q}_k \partial \vec{q}_j}\right) \\ &\approx \sum_{i=1}^{8} 2\frac{\partial \vec{f}_{\nabla,i}}{\partial \vec{q}_k}\frac{\partial \vec{f}_{\nabla,i}}{\partial \vec{q}_j}\end{aligned}$$

for $k, j = 1, 2, 3, 4$.
As the algorithm approaches the minimizer, $\vec{f}_{\nabla,i}$ goes towards zero, and thus the accuracy of the approximation increases. The above approximation is useful as it results in a relatively simple expression for the Hessian [33, p. 139]:

$$\boldsymbol{H}_q\left(F_\nabla\right) \approx 2\boldsymbol{J}_q^T\boldsymbol{J}_q$$

Determining Required Number of Optimization Iterations

Now that an expression for finding an iterative solution of $\vec{q}_\nabla$ has been derived, it is important to explore how many iterations are required to obtain adequately precise results. To do this, a stress test was performed where the sensor was violently shaken while logging data. The datasets were processed and analyzed in Matlab by applying equation (4.20) on each sample of the data set. The error of each of the calculated instantaneous orientations were then found using the objective function shown in equation (4.12). Finally, the average of each of the calculated orientation errors was found. This whole process was then repeated for a range of differing number of iterations of equation (4.20).

As the error term represents the Euclidean distance between the calculated orientation and the true value is identical to the chord length between the two points. This chord length is converted to an arc distance to provide a better intuition of the error term.

The average arc error over the entire data set versus number of performed iterations (from 1 to 20) is shown in figure 4.4:

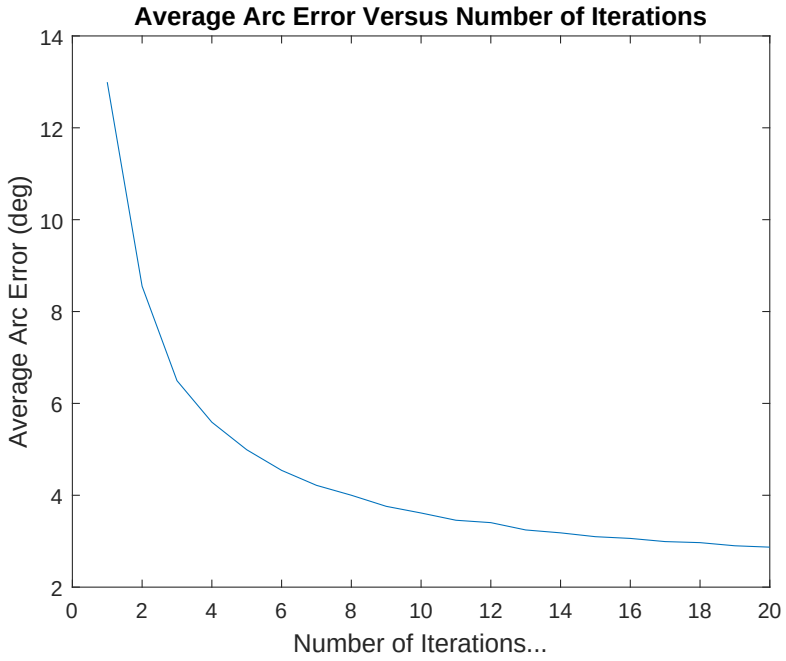


Figure 4.4: Graph of average arc error versus number of iterations calculated per sample.

As it is assumed that the dynamic forces exerted on the sensor during the test were much greater than what normal head movement entails, it is deemed fair to assume that the average arc error under normal use will not be greater than shown in the graph. Thus, according to the graph, to obtain the desired accuracy of 11° at least two iterations are needed.

4.2.4 Sensor Fusion by Complementary Filter

By using the aforementioned methods, two different approximations, q_ω and q_∇, of the true orientation can be obtained. However, these approximations still suffer from the weaknesses of the different sensors. As the gyroscope measurements are integrated over time, the orientation tends to drift over time. The accelerometer measures not only the static acceleration from gravity, but also the dynamic accelerations that comes from head movement making the orientation signal noisy. From this it can be gathered that the gyroscope has poor low frequency characteristics and the accelerometer has poor high frequency characteristics. One could then imagine that if the approximated orientation from the gyroscope is high pass filtered and the orientation obtained from the accelerometer and magnetometer is lowpass filtered before adding the two together, a result without these weaknesses could be achieved. This is exactly what a complimentary filter does. Figure 4.5 illustrates this concept:

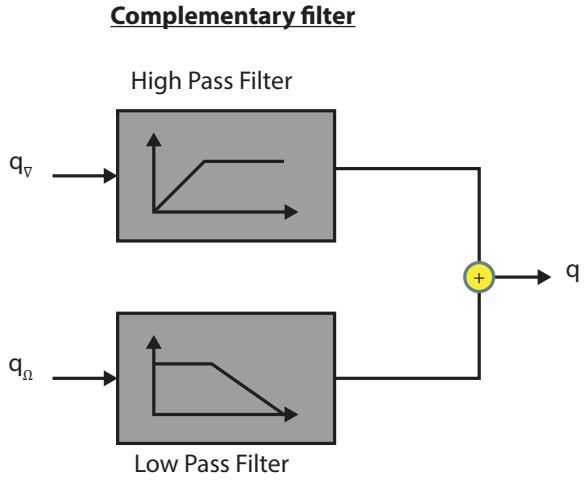


Figure 4.5: Illustration of complimentary filter. Here q_∇ is low pass filtered, and q_ω is high passed filtered.

The expression in the frequency domain for the approximated orientation is then as follows:

$$q_{\text{approx}} = \frac{1}{1+Ts}\vec{q}_\nabla + \frac{Ts}{1+Ts}\vec{q}_\omega = \frac{\vec{q}_\nabla + Ts\vec{q}_\omega}{1+Ts} \tag{4.22}$$

where T alters the cutoff frequency of the filters. To implement this in the orientation algorithm, the expression must be discretized. To transform an expression from the s-domain to the z-domain, different approximations of the relation between s and z can be used. As an analysis of every approximation would be to extensive for this report, only two different approximations are considered. The first is the *Backwards Difference Method*, as it is widely used in the literature for complementary filters with MARGs [30][28]. This method is then compared to the more general *Bilinear Transform*.

Backwards Difference Method

Starting with the backwards difference method, it based upon Euler's method for solving ordinary differential equations and is given as:

$$s \to \frac{1 - z^{-1}}{\Delta t} \tag{4.23}$$

where Δt is the sampling period. This transform maps the $j\omega$-axis from the s-domain to a circle with center in $[0.5, 0]$ and radius 0.5 in the z-domain, with $s = 0$ being mapped to $z = 1$ and $s = \inf$ mapping to $z = 0$ as seen in figure 4.6. This means that higher frequencies in the s-domain maps inside the unit circle in the z-domain, instead of onto it, resulting in some artificial dampening at higher frequencies. The left half plane of the s-domain gets mapped into the circle, which makes it possible that unstable poles in the s-plane can be mapped to stable poles in the z-domain. Additionally, as the transform is not linear, there also exists some amount of frequency warping when using the transform. Thus, if some characteristics are desired at a specific frequency in the z-domain, a different frequency must be used if designing in the s-domain. This is referred to as *prewarping*. In conclusion, the transform does not alias, as the expression is one-to-one, it keeps stability and creates two kinds of warping.

Transforming equation (4.22) to the discrete time domain by using the backwards difference method gives:

$$q_{\text{approx},t} = \alpha \left(q_{\text{approx},t-1} + \vec{q}_{\omega,t} \right) + (1 - \alpha) \vec{q}_{\nabla,t}, \quad 0 \leq \alpha \leq 1 \tag{4.24}$$

where $\alpha = \frac{T}{\Delta t} / \left(1 + \frac{T}{\Delta t} \right)$

Bilinear Method

Next, considering the bilinear method, it is based on the trapezoidal method for numerical integration and is given as:

$$s \to \frac{2}{\Delta t} \frac{1 - z^{-1}}{1 + z^{-1}} \tag{4.25}$$

where Δt is the sampling period. This transform maps the $j\omega$-axis to the unit circle in the z-domain, as seen in figure 4.6, with the left half plane being mapped inside the unit circle. As this expression is also one-to-one, the transform does not alias and due to the nonlinearity of this transform as well, frequency warping occurs. Thus, compared to the backwards difference method, the main difference is the artificial dampening that occurs when using the backwards difference method. However, when considering poles at low frequencies, the two methods give very similar results.

Transforming equation (4.22) to the discrete time domain by using the bilinear method gives:

$$q_{\text{approx},t} = (1 - \alpha)(\vec{q}_{\nabla,t} + \vec{q}_{\nabla,t-1}) + \frac{\alpha}{2}(\vec{q}_{\omega,t} + \vec{q}_{\omega,t-1}) - (1 - 2\alpha) q_{\text{approx},t-1}, \quad 0 \leq \alpha \leq 1 \tag{4.26}$$

where $\alpha = \frac{2T}{\Delta t} / \left(1 + \frac{2T}{\Delta t} \right)$

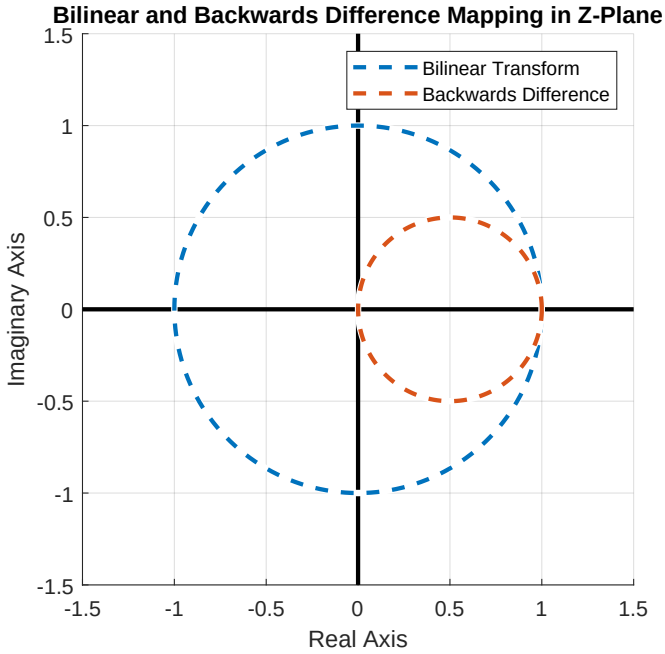


Figure 4.6: Image of the $j\omega$-axis in the z-plane. Dotted line is from bilinear transform and solid line is from backwards difference.

Now to determine which of the methods to use, an analysis of where the desired cutoff frequency of the filter is placed. As mentioned before, the goal of using the complimentary filter, is to use the low frequency characteristics from the magnetometer and accelerometer and the high frequency characteristics from the gyroscope. Additionally, the inherent noise in the sensors should also be subdued. As the drift from the gyroscope is very low frequency, the cutoff frequency should not need to be very high to ensure that the drift is filtered out. To gain more insight into where the cutoff frequency should be, a test to determine the inherent noise of the sensors was performed. Thus, some measurements were made while the MARG was kept stationary on a table. Afterwards an FFT analysis of the measurements was made, to see how the noise was distributed among the sensors. The results from this test can be seen in figure 4.7.

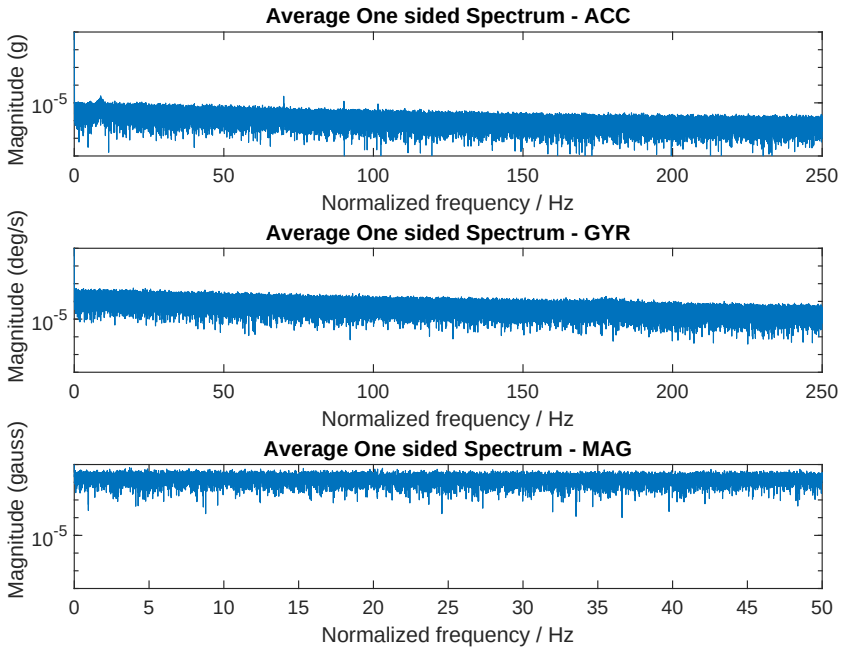


Figure 4.7: Frequency content of the sensors in steady state, averaged over the three axes of the sensors. The accelerometer and gyroscope were sampled at 500 Hz, and the magnetometer was sampled at 100 Hz. The sensor was configured to low pass filter the gyroscope and accelerometer data. The corner frequency was set to 42 Hz.

It is clearly seen that the noise is primarily wide band noise, with the magnetometer being most noisy. As the magnetometer and accelerometer gets lowpass filtered, the cutoff frequency of the complementary filter should be as low as possible to limit the bandwidth of the noisy magnetometer.

Based on the test there is no obvious specific cutoff frequency to be chosen, only that it should be at a low frequency. At these frequencies both the backwards difference and the bilinear method should be viable. As the expression for the backwards difference is somewhat simpler than that for the bilinear method, and it is the one used in the literature, it is chosen as the discretization method. To determine the exact cutoff frequency, an empirical experiment is performed in Matlab where the value of α varied over a range. From this it was found that an α between 0.9 and 0.99 was a good range for producing quaternions with low noise drift. Thus 0.95 is chosen as the final value of α.

Chapter 5

Implementation of Head Tracking

Now that the specific equations have been derived for head tracking, these need to be combined and implemented as an algorithm on the DSP. However, during initial design, the algorithm was implemented in Matlab, and later x86 C compiler on a PC, to enable quick verification of its functionality. Matlab inspired pseudocode of the final algorithm used to calculate the orientation from a given sample of the accelerometer, magnetometer, and gyroscope is shown in figure 5.1:

```
%%%%% Variables %%%%%
rate     % Sample rate in Hertz
iter     % Number of iterations to perform gradient decent
alpha    % Complimentary filter parameter
mag      % Normalized magnetometer measurements
acc      % Normalized accelerometer measurements
gyr      % Current gyroscope measurements
b        % Stores current magnetic reference
q        % Stores previously calculated rotation quaternion
q_omega  % Quaternion found using the gyroscope
q_nabla  % Quaternion found using the accelerometer and magnetometer
f        % Base function
J        % Jacobian Matrix of vector function f
G        % Gradient Vector of the objective function
H        % Hessian Matrix of the objective function
step     % Step size for the gradient decent algorithm

%%%%% Code %%%%%
%Note that ' means transposed
%Multiplication between matrices/vectors imply matrix multiplication
q_omega = q + 0.5 * quaternion_product(q,gyr) * rate

q_nabla = q

%Perform gradient decent
for i = 1:iter
   f       = base_function(q,acc,mag,b)
   J       = jacobian_of_f(q,b)
   G       = 2*(J'*f)
   H       = 2*(J'*J)
   step    = (G'*G)/(G'*H*G)
   q_nabla = q_nabla - step*G
   q_nabla = norm(q_nabla)
end
q = (1-alpha) * q_nabla + alpha * (q_omega + q) % Complimentary Filter
```

Figure 5.1: Matlab inspired pseudocode of the algorithm used to calculate a rotation from a sensor sample.

When implemented in C, the above code translates to a collection of tediously implemented linear

algebra. This is exemplified by figure 5.2 showing how the Jacobian matrix is found (corresponds to line 29 of figure 5.1):

```
1  /***** Definition of Jacobian Matrix *****/
2  double J_11 = -2 * q3;
3  double J_12 =  2 * q4;
4  double J_13 = -2 * q1;
5  double J_14 =  2 * q2;
6  double J_21 =  2 * q2;
7  double J_22 =  2 * q1;
8  double J_23 =  2 * q4;
9  double J_24 =  2 * q3;
10 double J_31 =  0;
11 double J_32 = -4 * q2;
12 double J_33 = -4 * q3;
13 double J_34 =  0;
14 double J_41 = -2 * b4 * q3;
15 double J_42 =  2 * b4 * q4;
16 double J_43 = -4 * b2 * q3 - 2 * b4 * q1;
17 double J_44 = -4 * b2 * q4 + 2 * b4 * q2;
18 double J_51 = -2 * b2 * q4 + 2 * b4 * q2;
19 double J_52 =  2 * b2 * q3 + 2 * b4 * q1;
20 double J_53 =  2 * b2 * q2 + 2 * b4 * q4;
21 double J_54 = -2 * b2 * q1 + 2 * b4 * q3;
22 double J_61 =  2 * b2 * q3;
23 double J_62 =  2 * b2 * q4 - 4 * b4 * q2;
24 double J_63 =  2 * b2 * q1 - 4 * b4 * q3;
25 double J_64 =  2 * b2 * q2;
```

Figure 5.2: Definition of Jacobian matrix as written in C using double precision floating-point.

The implemented Matlab and C algorithm were made using double precision floating-point. However, as the DSP has no dedicated floating-point unit, this implementation cannot be used as is. This is because the floating-point logic would have to be simulated in software, which would drastically increase the algorithms computational requirements.

For example, on the DSP it has been observed that the four basic arithmetic operators: addition, subtraction, multiplication, and division each respectively require up to 250, 280, 300, and 2500 clock cycles to calculate in floating-point. Due to the real time nature of the system, as well as the algorithm's use of arithmetic, it is impractical, if not impossible, to use floating-point.

Thus, a fixed-point implementation is sought instead. With a fixed-point implementation, the calculations can be performed a lot faster, however it comes at the loss of dynamic range and precision (when dealing with small numbers).

To represent decimal numbers, the Q format is used. Unlike pure integers, this format specifies both a fractional and an integer part [34]. For example, a Q4.12 number is represented by a 16-bit integer where the first four bits are used to represent the integer part of the number. In this report this includes the sign bit. The last 12 bits are then used to represent the fractional part. To be able to represent negative numbers, two's complement is used. In this example the resolution and range can

be shown to be:

- Resolution: $2^{-12} = 2.44 \cdot 10^{-4}$
- Range: $[-(2^3); 2^3 - 2^{-12}] = [-8; 7.9998]$

In general, the resolution and range of a Qm.n number can be found using the following equations [34]:

- Resolution: 2^{-n}
- Range: $[-(2^{m-1}); 2^{m-1} - 2^{-n}]$

Thus, in the algorithm m and n must be chosen such that overflow cannot occur, while simultaneously ensuring that adequate resolution remains to represent the rotation quaternion. To do this, a numerical analysis of the required range is made.

5.1 Numerical Analysis of Dynamic Range

To ensure that overflow does not occur the largest possible value of all intermediate results needs to be determined. This is a very difficult problem to solve. Instead of doing it analytically, it was decided to use numerical methods on a subset of the possible intermediate results to find an approximation of the largest possible value. This choice was made in the interest of saving time. To perform the numerical analysis the equations for the selected intermediate equations were inserted into Matlab and given random initial conditions. These functions were than maximized, using Matlab's own optimization function. As it may be possible for local maxima to exist for the selected equations, it is not sufficient to perform the maximization with one set of initial conditions. Thus, over a 12-hour period, the above maximization was repeated 300 times, each with random initial conditions. While this cannot be guaranteed to provide the true value of the maximum, it provides a good basis for which to implement the fixed-point algorithm upon. From this experiment, the biggest value was found to be exactly 2^{16}.

This requires that at least 18 bits must be allocated for the integer part. This more than the DSPs 16-bit architecture is designed to handle efficiently. While performance is degraded, the C compiler is capable of natively managing both 32-bit and 40-bit data types. While the use of 40-bit values would allow for higher accuracy than 32-bit, they also require more clock cycles to perform arithmetic operations. Thus, in the interest of minimizing the computational load of the algorithm it is decided to use 32-bit data types. As this allows for the use of 14 fractional bits, the Q18.14 format is used.

5.2 Multiplication

As it is not possible to directly multiply two 32-bit numbers on the DSP, the calculation has been split up into parts, each operating on 16-bit numbers. The idea is to split each 32-bit number into two 16-bit numbers, representing the upper and the lower 16 bits respectively. The upper parts can be calculated as the 32-bit number right-shifted by 16 bits which algebraically corresponds to:

$$A_U = (A - A_L) \cdot \frac{1}{2^{16}}$$

The lower parts can be calculated as the 32-bit number bitwise AND'ed with `0xFFFF` or algebraically as:

$$A_L = A - A_U \cdot 2^{16} \tag{5.1}$$

Now, if the 32-bit number is isolated in one of the equations, the product $A \cdot B$ can be represented as a sum of 16-bit multiplications as seen in equation (5.2)

$$\begin{aligned} A \cdot B &= (A_U \cdot 2^{16} + A_L) \cdot (B_U \cdot 2^{16} + B_L) \\ &= A_U \cdot B_U \cdot 2^{32} + (A_L \cdot B_U + A_U \cdot B_L) \cdot 2^{16} + A_L \cdot B_L \end{aligned} \tag{5.2}$$

Now, as this expression does not consider that the numbers has 14 fractional bits, the entire expression must be shifted to the right by 14 bits to give the correct result. To understand the necessity of this operation, consider the following example in decimal notation. In a given notation, the decimal number 1000 is set to represent the value 1 thus meaning that in this notation there are 3 fractional decimals. Now when multiplying 1000 with 1000 normally, the algebra would give the result 1000000. However, the correct result in the given notation is 1000 as $1 \cdot 1 = 1$. Thus, the algebraic result needs to be shifted to the right with the number of fractional decimals, to give the correct result in the chosen notation.

If this operation is applied to the expression in equation (5.2) the multiplication is given by:

$$\begin{aligned} A \cdot B &= 2^{-FB} \cdot (A_U \cdot B_U \cdot 2^{32} + (A_L \cdot B_U + A_U \cdot B_L) \cdot 2^{16} + A_L \cdot B_L) \\ &= A_U \cdot B_U \cdot 2^{32-FB} + (A_L \cdot B_U + A_U \cdot B_L) \cdot 2^{16-FB} + A_L \cdot B_L \cdot 2^{-FB} \end{aligned} \tag{5.3}$$

Where FB is the number of fractional bits.

These equations were then implemented as the following algorithm in C:

```
/***** Function for multiplying two fixed-point numbers *****/
int32_t fixedpt32_mul(int32_t A, int32_t B, uint16_t FBITS)
{
    //Uses C55x intrinsic to calculate the Absolute value of A and B
   uint32_t A_abs = _labss(A);
   uint32_t B_abs = _labss(B);

   uint16_t A_upper = A_abs >> 16;
   uint16_t A_lower = A_abs & 0xffff;
   uint16_t B_upper = B_abs >> 16;
   uint16_t B_lower = B_abs & 0xffff;

   uint32_t ALoBLo = (uint32_t)A_lower * (uint32_t)B_lower;
   uint32_t AHiBLo = (uint32_t)A_upper * (uint32_t)B_lower;
   uint32_t ALoBHi = (uint32_t)A_lower * (uint32_t)B_upper;
   uint32_t AHiBHi = (uint32_t)A_upper * (uint32_t)B_upper;

   int32_t result = (ALoBLo >> FBITS)
   result += ((AHiBLo + ALoBHi) << 16 - FBITS)
   result += (AHiBHi << (32 - FBITS));

   if (A < 0 ^ B < 0)
      result = -1 * result;

   return result;
}
```

Figure 5.3: Definition of unoptimized fixed-point multiplication function.

However, when implemented on the DSP, profiling shows that this multiplication algorithm requires up to 97 clock cycles to complete. Given that the gradient descend algorithm alone requires about 300 multiplications to complete a single iteration, the algorithm becomes very computationally expensive to run. Thus, the multiplication algorithm is a prime candidate for optimization endeavors. Setting the compiler to '-o3' optimization resulted in the algorithm only requiring about 50 clock cycles to complete. However, the greatest gains were achieved by hand writing a Q18.14 multiplication assembly function taking advantage of the DSP's parallel computational capabilities. With this function, multiplication only requires 35 clock cycles, an almost 90% decrease compared to floating-point multiplication.

5.3 Division

True 32-bit division is a very difficult algorithm to implement in fixed-point on a 16-bit processor. However, unlike multiplication, division is only required once per gradient descent iteration. This implies that it is not nearly as critical for it to be optimized for performance. Thus, the simple solution is to cast the fixed-point divisor and dividend as floating-point numbers, perform the floating-point division, and then convert the result back to a fixed-point number. The function for this is shown in figure 5.4:

```
/***** Function for dividing two fixed-point numbers *****/
int32_t fixedpt32_div(int32_t A, int32_t B, uint16_t FBITS)
{
    float a_f = (float) A;
    float b_f = (float) B;

    float result = a_f / b_f;

    //Function to convert floating-point number to a Q18.14 representation
    int32_t result_fixed = fixedpt32_float_to_fixedpt(result, FBITS);
    return result_fixed;
}
```

Figure 5.4: Definition of fixed-point division function.

While this is not a computationally efficient way to do fixed-point division, it is considered "good enough" for this application.

5.4 Implementation of Square Root and Reciprocal Square Root.

In the head tracking algorithm, the norm needs to be repeatedly found to ensure that vectors and quaternions have unit length. This usually consists of a division operation, which is very time consuming and therefore a method that uses multiplication instead is desired. This can be done by calculating the reciprocal of the square root and then multiplying instead of calculating the square root of a number and then dividing.

To calculate the reciprocal square root Newtons method is used. Newtons method approximates the value numerically through an iterative process. The reciprocal square root, y, of number, x, can be found as:

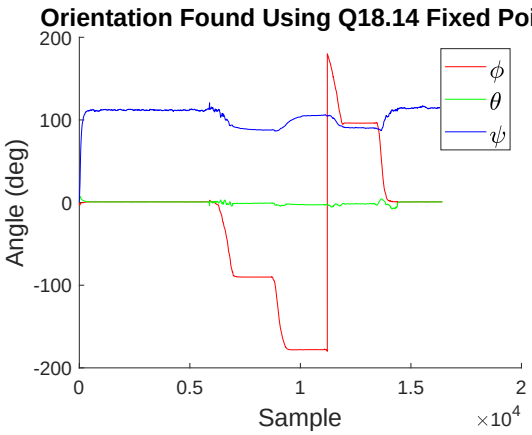
$$y_{n+1} = y_n \cdot \frac{1}{2}\left(3 - xy_n^2\right) \tag{5.4}$$

As Newtons method works by first making an initial guess to the solution and then iterating towards the true solution, a look-up table with initial guesses, calculated in Matlab, is created as well.

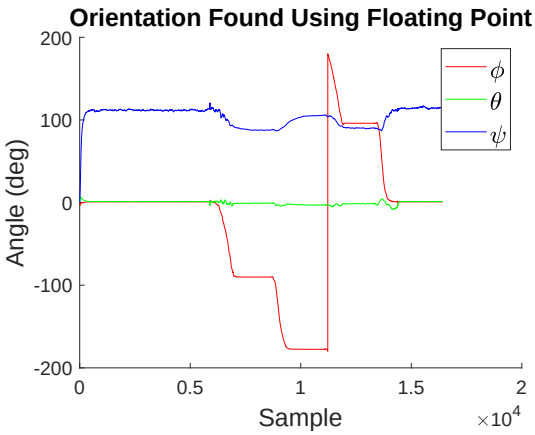
5.5 Fixed-Point Performance Compared to Floating-point Implementation

As the implemented Q18.14 fixed-point format must necessarily produce different results than when using double precision floating-point, it is of interest to evaluate the numerical differences between these two implementations. To do this, both algorithms where provided with sample input data generated for appendix B. For each sample, the output orientation quaternion of each algorithm is used to rotate a test vector. The arc distance between the two rotated vectors can then be found. This metric does not inform how well either algorithm approximates the true orientation. However, as it is fair to assume that the floating-point implementation calculates a better approximation, the arc difference does indicate how much precision is lost when using the fixed-point implementation.

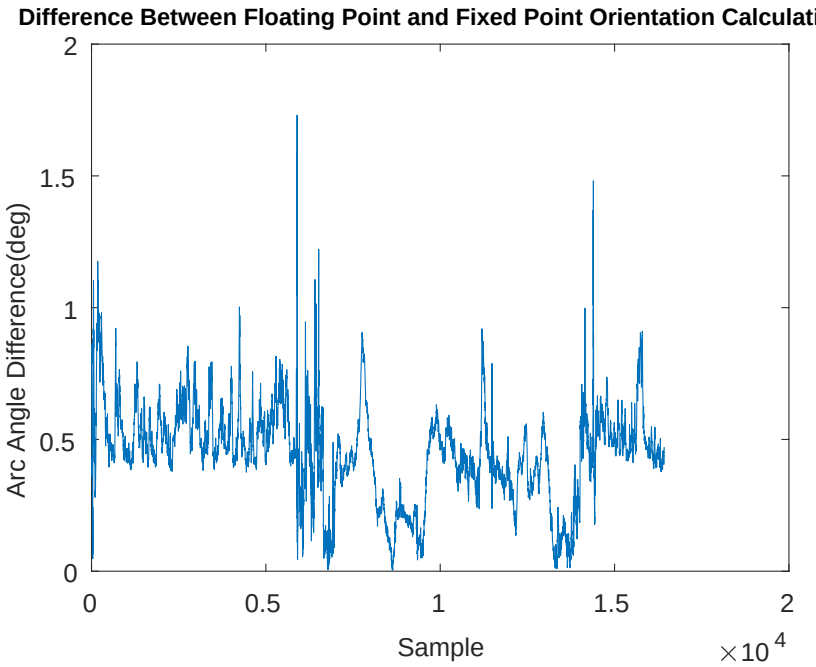
Figure 5.5 illustrates the difference in the calculated orientation between the fixed and floating-point implementation.



(a) Orientation calculated using Q18.14.



(b) Orientation calculated using double precision floating-point.



(c) Arc difference between the calculated floating-point and fixed-point orientation.

Figure 5.5: Figure (a) and (b) illustrates the calculated orientation derived from counterclockwise rotation around the Y axis, as measured in appendix B. Orientation is represented using ZYX Euler angles. Figure (c) graphs the arc difference between (a) and (b).

Chapter 6

Test of Head Tracking Application

In the previous chapters the head tracking application has been designed and an implementation for the DSP been proposed. To determine if the implementation is able to meet the requirements stated in chapter 2, a series of tests has been performed.

For the head tracking subsystem, the relevant technical requirements regarded the resolution, accuracy and convergence rate of the subsystem. To be able to determine the resolution, accuracy and convergence rate, first it must be ensured that the orientation tracker is able to measure an orientation without drift over time. As the orientation algorithm uses gyroscope values, there is a risk that the calculated orientation will drift over time. Thus, a test of the drift in the orientation tracker is performed for verification purposes.

6.1 Test of Stability/Drift in Orientation Tracker

The orientation tracker does not drift over time. As described in the test journal found in appendix D, it was verified that in one hour the calculated orientation does not change at all, if the system is kept stationary. The logged values of the quaternion elements during the test can be seen in figure 6.1.

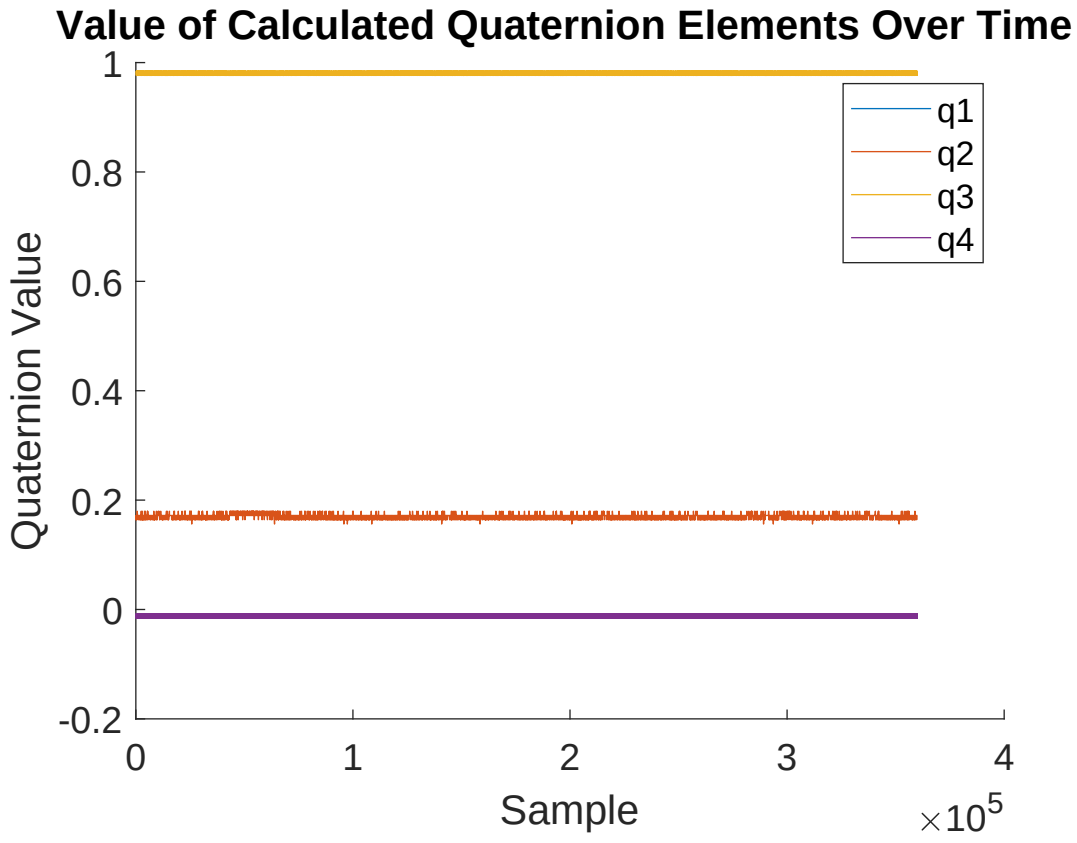


Figure 6.1: Drift as measured over one hour. Note that q1 is barely visible as it is placed closely to q4.

Thus, it should be possible to determine the resolution, accuracy and convergence rate through tests.

6.2 Test of Resolution of Orientation Tracker

The resolution for the orientation tracker did not meet all the requirements proposed. As described in the test journal found in appendix E it was determined that the calculated orientation would always fluctuate upwards of $\approx 1.6°$, when stationary. An example of these fluctuations can be seen in figure 6.2

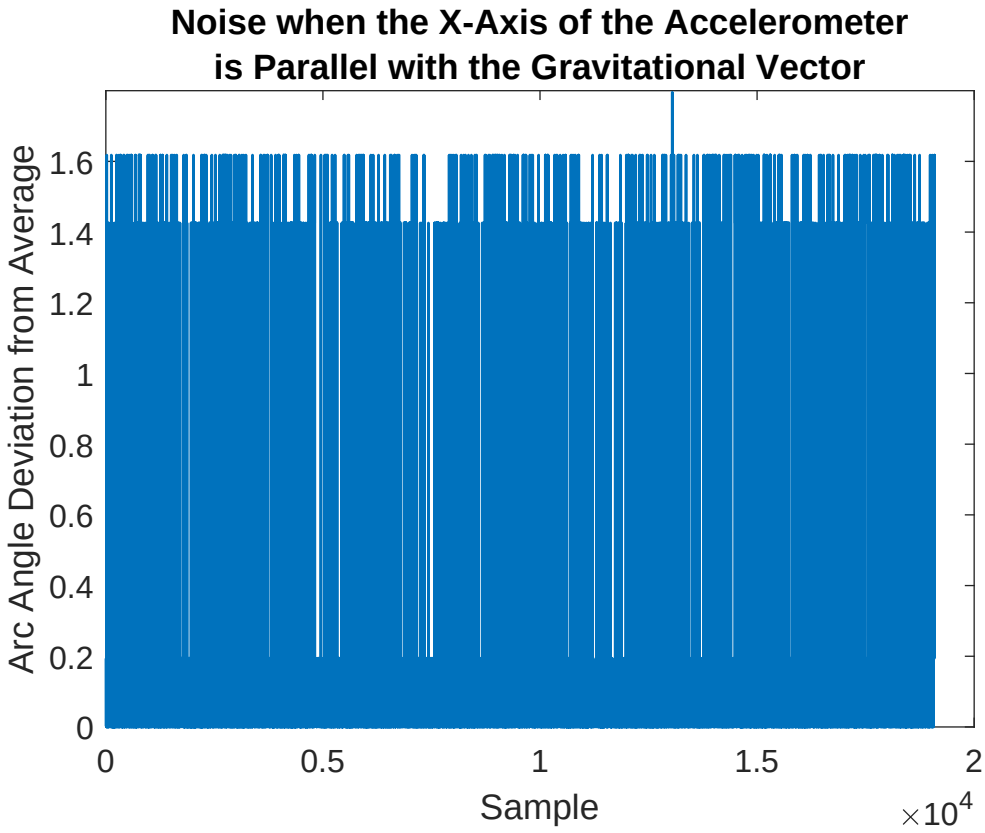


Figure 6.2: Figure from the noise test. The figure show the fluctuations in the measured orientation when the system is kept stationary.

Thus, it is not possible to achieve a resolution of 1° as stated in requirement T.2. However, this resolution is still deemed acceptable as the MAA requirements were based on a study where subjects listened to pure tone pulses in very controlled environments.

The complexity of the signals in usual multimedia entertainment sources such as speech or music is a lot higher than in the study and thus, for these usages the MAA is likely to be somewhat larger. Additionally, when presented with visual cues as well, the ventriloquism effect suggests that the human auditory system will be able to compensate for small errors in resolution.

6.3 Test of Accuracy of Orientation Tracker

The accuracy of the orientation tracker did not meet all the requirements proposed. As described in the test journal found in appendix F the accuracy of the orientation tracker varied a lot with the axis of rotation. The only axis that had an accuracy that met the required 11° from requirement T.6 was the z-axis, where the test showed a calculated rotation of 92.13° around the z-axis. Thus the orientation tracker should enable the complete system to calculate the azimuth coordinate when the z-axis is parallel with Earth's gravitational field. The projection of the vectors from before and after rotation around the z-axis can be seen in figure 6.3.

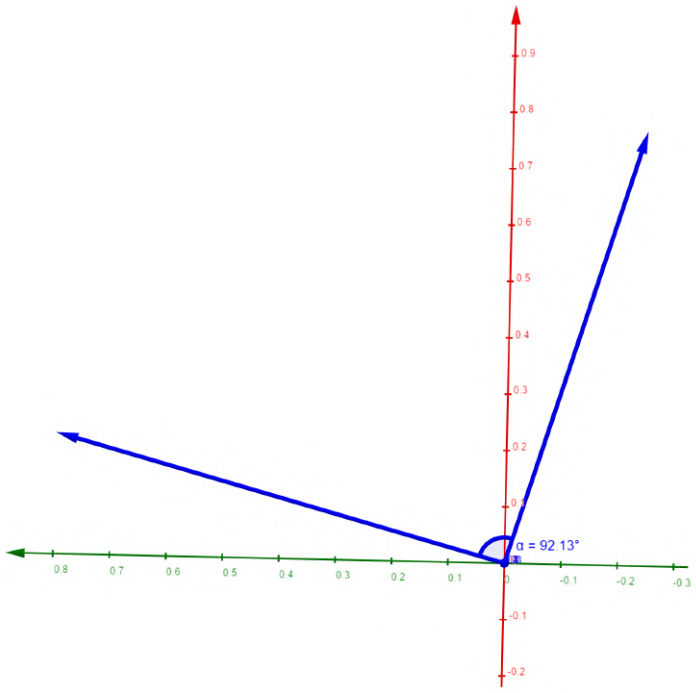


Figure 6.3: 2D-view of projections onto the XY-plane of the measured orientation before and after the rotation around the z-axis.

For rotation around the two other axes, the accuracy is somewhat harder to describe. Here the results showed that, while the orientation tracker would measure a rotation, the direction of the rotation would not correspond to the axis of rotation in the test. Additionally, when looking at the projections onto the plane perpendicular to the axis of rotation, the actual rotation around the axis of rotation from the test would be far from 90°. For the x-axis the rotation was 11.52° and for the y-axis the rotation was 157.27°.

6.4 Test of Convergence Rate of Orientation Tracker

The convergence rate of the orientation tracker did not meet the requirements either. As described in the test journal found in appendix G, the orientation was determined to be converged for the x- and z-axis as soon as the gyroscope measured that the system was stationary. However, for the y-axis the convergence rate was 40 ms to 50 ms. The quaternion elements, together with the gyroscope measurements, from the test with rotation around the y-axis can be seen in figure 6.4

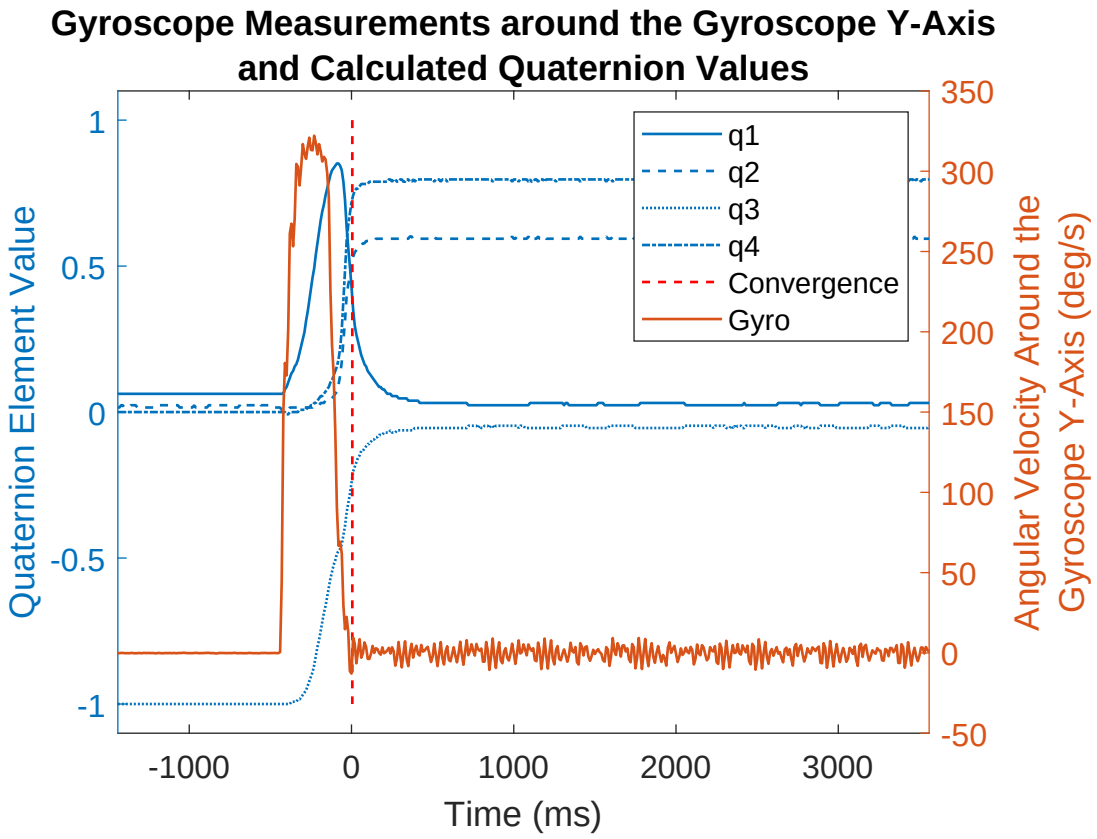


Figure 6.4: Figure from the convergence rate test. It is here easy to see the noisy gyroscope signal after the rotation, which might have impacted the convergence rate.

This is above the maximum delay stated in requirement T.4 of 26 ms, however not by very much. It has not been possible to explain definitively why the orientation converged slower for rotation around the y-axis. However, a detail worth mentioning is that during the tests the servo motor at times had trouble holding the setup completely still. This was especially observed during the test with rotation about the y-axis. In figure 6.4 it can be seen from the gyroscope measurements after the rotation, that some additional noise was introduced which in turn might have had an impact on the convergence rate.

6.5 Partial Conclusion

From the tests of the orientation tracker it can be concluded that the subsystem in general cannot fulfill the requirements to the system. The resolution of 1.6° is close enough to the requirement that it is assumed that it will not have a major impact on the functionality of the system. However, the accuracies and convergence rates determined through the tests are so far from the requirements that it is expected to have major influence on the performance of the system, and thus makes the tracking system not usable for head tracking purposes.

Part III

HRTF Filter Design

Chapter 7

HRTF Analysis and Design

To reproduce three-dimensional audio, a subsystem which makes use of HRTF-based digital filters is to be designed and implemented on the DSP. The primary feature of this system is to filter the audio input using HRTFs, however it must also be able to switch between different filters to account for the change in orientation calculated by the head tracking application.

To understand how to create digital filters from HRTFs a brief introduction to HRTFs and the AAU Valdemar database of HRIRs is presented.

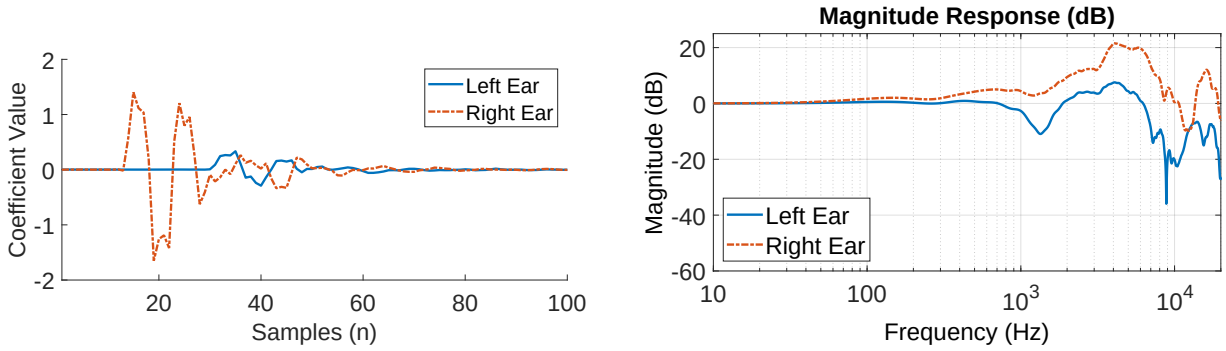
7.1 Introduction to Measured HRTFs

As described in section 1.1, when a sound source in free space emits sound to a pair of ears, all of the localization cues needed to localize the position of the sound source is described by a unique pair of HRTFs, corresponding to the sound sources' location in space. However, section 1.1 only describes the aspects of human spatial hearing, this section introduces measured HRTFs. Since this report is not about measuring HRTFs, the technique is not described in this section. However, the topic of measuring HRTFs is briefly described in H.2.1.

7.1.1 Measured HRTFs

When measuring HRTFs on human subjects or artificial heads, either the transfer function or impulse response is measured, resulting in HRTFs or Head Related Impulse Responses (HRIRs) respectively.

Obtained from the Valdemar HRTF database figure 7.1(a) shows the impulse responses of the HRTF at 45° azimuth and 0° elevation, note that this figure only shows the first 100 coefficients, however the database consists of 256 coefficients. On this impulse response it can be noted that the first 12 samples correspond to the arrival time delay, this is the time it takes for sound to reach the first ear which here is the right ear. E.g. for the 0° azimuth and 0° elevation the arrival time delay would be the same for both ears. Again, for figure 7.1(a), between the two ears there is the ITD which consists of approximately 17 samples of the response. In turn, that means the arrival time delay for the left ear is approximately 29 samples. Note that here the ITD is measured in samples, this can easily be converted to time, usually measured in µs, by multiplying the number of samples with the sampling rate which is 48 kHz.



(a) Impulse response of the HRTF.

(b) Frequency response of the HRTF.

Figure 7.1: Impulse and frequency responses from the Valdemar database for HRTF location 45° azimuth and 0° elevation.

Similarly, on figure 7.1(b) the frequency response at the same 45° azimuth and 0° elevation from Valdemar is plotted. On this figure it can be noted that the right ear has the highest amount of gain throughout the plot, this is comparable with the impulse response, and it is because the HRTF's location is closest to the right ear.

7.2 Valdemar HRTF Database

The Valdemar HRTF database is a collection of HRIRs measured using the Valdemar dummy head [35]. The dummy is constructed with lifelike pinnae, head and torso, allowing for realistic measurements of human hearing. This allows for measurement of HRTFs, for use in three-dimensional audio as well as other applications. The pinnae chosen for the dummy are the ones that yielded the best result with three-dimensional audio, using a wide array of test subjects [35]. As a result, the HRTF database is suited for use by a broad spectrum of users, which has both positive and negative consequences. While the database works for most people, it is not nearly as good as one which is custom made for the individual user. However due to HRTF measurements being both complex and time consuming, this trade-off is necessary.

The Valdemar database is chosen for this project, because it is readily available from the university, has coverage of the entire spherical coordinate system, as well as being well documented (see [36]) and it has the highest spatial resolution of any HRTF databases [1, p. 65].

The database consists of a three-dimensional array containing the filter coefficients, as well as the correlating elevation and azimuth angles. The highest azimuth resolution is 2°, while the lowest is 360°. The difference in resolution is due to the measurements being in a uniform sphere, resulting in a lower azimuth resolution as the elevation angle increases or decreases from 0° elevation. For example, at 90° only one point is needed, since any azimuth rotation results in the same point. The different resolutions can be seen in table 7.1. Each unique elevation and azimuth index contain 256 coefficient values, thus, making the array three-dimensional.

Table 7.1: Table showing the different resolutions of the Valdemar database.

Elevation (+/-)	Resolution	No. of measurements
0° - 48°	2°	180
50° - 66°	3°	120
68° - 70°	5°	72
72° - 76°	6°	60
78°	9°	40
80° - 82°	10°	36
84° - 86°	18°	20
88°	45°	8
90°	360°	1

These intervals result in a sphere of uniformly distributed points, a segment of which can be seen in figure 7.2.

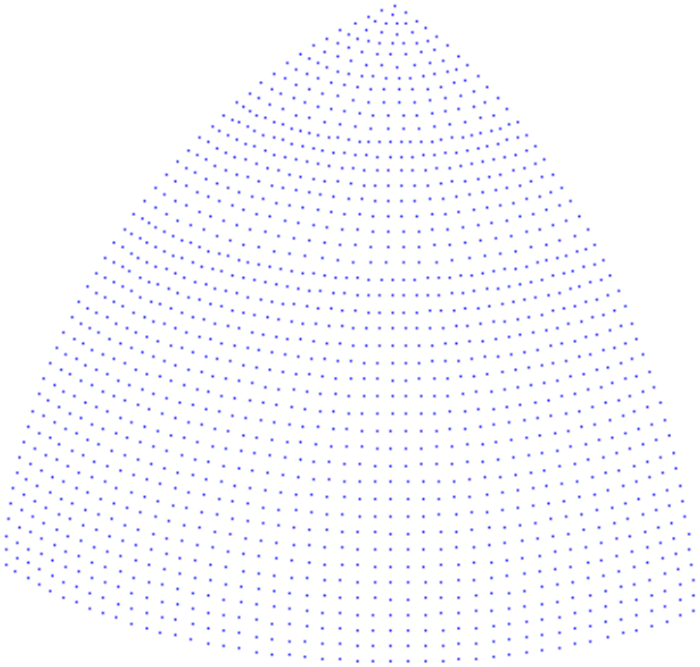


Figure 7.2: Illustration of the uniform point sphere of the Valdemar database [35].

Due to having different resolutions in certain intervals, the indexing of this three dimensional-array is not entirely trivial, and as such requires a conversion between angle and index. This conversion process will be described in section 8.1.

7.3 Design of HRTF Filters

The transmission of sound from a source to a pair of ears can be regarded as a Linear Time-Invariant (LTI) process, this means that HRTFs can be considered as LTI systems [1, p. 20]. Since HRTFs can be regarded as LTI systems they can be implemented by digital filters and the process of designing digital filters from HRTFs will be covered in this section.

When designing a digital filter, the first step when choosing a filter structure is to decide whether

it should be recursive or non-recursive. In other words, if it is an Infinite Impulse Response (IIR) or Finite Impulse Response (FIR) filter, respectively. First it is important to assure that the filter structure can represent the data in the HRTFs correctly. The peaks and notches seen in the frequency response of HRTFs (see figure 7.1(b)) can appropriately be represented by poles and zeros, this means that IIR filters with poles and zeros can represent HRTFs [1, p. 140]. An FIR filter with only zeros can also appropriately represent HRTFs because the peaks and notches in the frequency response are relatively smooth. [1, p. 140]. Thus, both filter types can be used, and other parameters are needed to determine the type of filter. There are several similarities and differences between IIR and FIR filters, these will be considered before choosing which filter type is going to be implemented.

7.3.1 Considerations when Choosing Between IIR and FIR Filters

As mentioned previously, the fundamental difference between IIR and FIR filters are that one is recursive while the other is non-recursive. However, there are other differences which are just as important to consider when selecting the filter design. These are for example the storage required for current and past values of both input and output, the computation time and the stability of the filter.

It is known that in general that an IIR filter requires much fewer coefficients compared to an FIR filter to produce the same filter response. This makes IIR filters much more efficient when considering both the computation time and the storage of values in memory. However, this does not entirely hold true for filters designed for HRTFs. Sandvad and Hammershøi compared IIR and FIR filters of different lengths in listening experiments in [37]. The goal was to find the most efficient way of representing filters from HRTFs, with no loss of sound quality. The results show the probability that a subject could detect a test filter from two reference filters. For the tested FIR filters, 72-order filter has a probability of 0.08 to be detected, while for the IIR filters a 48-order (48 poles and 48 zeros) filter has a probability of 0.03 to be detected [37, p. 15-16]. These results show the minimum required filter length for an FIR and IIR filter which makes the filter indistinguishable from the reference 256 tap filter. Additionally, the FIR filter has the advantage of being much easier to design and the filter itself will always be stable, even when the coefficients undergo quantization when transformed to a fixed-point system. Based on the above study and the fact that FIR filters can be derived directly from HRIRs compared to the long process of converting HRIRs to stable IIR filters, it is decided to work exclusively with FIR filters. However, before the HRIRs can be implemented as filters on the DSP, they must undergo different stages of preprocessing to improve the performance of the resulting filter when they are implemented on the DSP.

7.4 HRTF Filter Preprocessing

Before the HRTF database can be implemented as filters on the DSP some preprocessing of the filters is necessary. The objective of this preprocessing is to reduce the number of coefficients by means of removing the arrival time delay and extracting the ITD in the impulse responses followed by truncation. Additionally, because the DSP is limited to fixed point the subject of quantizing the filter coefficients are also covered.

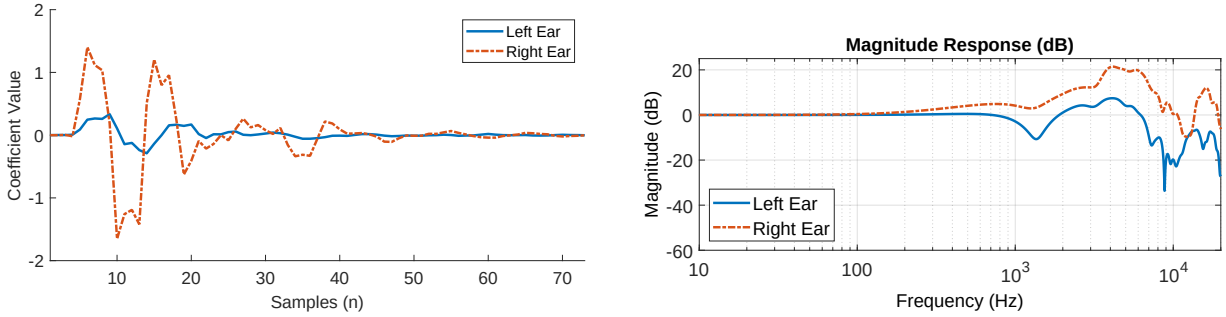
7.4.1 Reducing the Length of the HRIRs

It was mentioned in section 7.2 that the Valdemar database consists of HRIRs with 256 coefficients. If these full-length impulse responses were to be implemented as FIR filters, they would require a lot of

storage and computation time to process on the DSP. To solve this problem, some of the coefficients in the HRTFs can be omitted without impacting the localization information in the HRTF, however the audio quality will be compromised [38, p. 6]

As mentioned in section 1.1 the ITD is the arrival time difference between the two ears for a given sound. The highest ITD for any HRTF is found at 90° to either side of the head at 0° elevation, and here it is 30 filter coefficients for the Valdemar database. As the ITD only refers to the time difference from one ear relative to the other, the time it takes for the sound to reach both ears possesses no information of the direction of the sound. This time is referred to as the arrival time delay. For the Valdemar database, this delay is 12 samples and it can entirely be omitted as long as the ITD is still correctly implemented in the filtering process. Meanwhile the ITD value can be extracted and saved, and then the entire initial group delay can be removed from the HRIRs. Then, on the DSP the ITD can instead be implemented via the built-in time delay hardware or as ring buffers.

In addition to extracting the ITD, the HRIR can be truncated because the energy of the impulse response is concentrated at the beginning of the impulse response when the ITD and arrival delay are removed [1, p. 141]. As it was shown in section 7.3.1, a 72 coefficients FIR filter is indistinguishable from a reference FIR filter with 256 coefficients. Therefore, after the delay has been extracted, the impulse response can be truncated with a rectangular window, as this window has proven best in preserving the sharper notches in the transfer function [37, p. 4]. In figure 7.3(a) the impulse responses of a pair of left and right ear HRTFs with ITD and arrival delay removed and truncated to 72 coefficients can be seen. In figure 7.3(b) the corresponding frequency responses can be seen.



(a) Impulse response of the HRTF.

(b) Frequency response of the HRTF.

Figure 7.3: Impulse and frequency responses of the HRTF with arrival time delay and ITD removed for HRTF location 45° azimuth and 0° elevation.

Although the 72 taps FIR filters are enough to convey the needed localization cues, they also inflict a significant reduction in frequency resolution. This effect is especially noticeable at low frequencies where the response is much smoother which is evident when comparing figure 7.1(b) and 7.3(b) [38, p. 6]. However, it can be seen when comparing the two frequency responses that the higher frequencies remain the same despite the truncation and removal of the two-time delay. Although not apparent from the frequency response on figure 7.3(b), a downside of truncating HRTFs is that their DC gain might not remain at 0 dB. This topic of DC gain adjustment will be covered after the following section where the ITD is extracted from the HRIRs.

7.4.2 ITD Extraction from HRIR Coefficients

The ITDs are extracted and removed during preprocessing using a Matlab script. In order to identify which coefficients are relevant to the HRTF, the script finds the highest absolute coefficient, and compares the rest of the coefficients with this, starting at the first coefficient. If a coefficient is higher

than 5% of the maximum coefficient's value [37, p. 5], then that value is deemed significant and the next 72 coefficients are saved and used as a filter.

There are two ways of reacquiring the ITD, which is either by saving it during preprocessing or calculating it separately in the DSP. If storage space is limited, the ITD would ideally be calculated on the DSP, since saving ITD values for all measurement points and both channels requires a lot of memory. However, since the database itself is too big to fit onto the internal memory of the DSP, an external memory module is required either way. This means the ITD values can be saved together with the filter coefficient data structure, thus ensuring the correct ITDs for the database are used in the filtering on the DSP.

During preprocessing, the script counts how many coefficients are removed, and adds the absolute difference between the two channels as the first index of the FIR coefficients. For example, if the right channel has an arrival time delay of 20 samples and the left has 45, the first index of the FIR coefficients is set to be $45 - 20 = 25$ which is the ITD measured in samples. The ITD value is only stored in the filter channel with the highest arrival time delay, this is because it is only the ITD which is important, and the arrival time delay is negligible in the filter implementation. The other channel simply has zero added as its first coefficient index. This value is then read during filtering and used in the DSP's built in delay functionality or ring buffer, which will be described during implementation in chapter 8.4. As it was mentioned earlier, due to the truncation of the HRIR it is likely that the DC gain must be adjusted, this will be described and implemented in the following section.

7.4.3 DC Gain Adjustment of Filter Coefficients

As a standard, the Valdemar database has had the DC value of the HRTFs set to unity gain [36, p. 93]. However, due to the changes in the impulse response as a result of the truncation, it is likely that the resulting HRTFs no longer have unity gain at DC [38, p. 6]. This can be seen when looking at the frequency response of a HRTF before and after it was truncated, however in the example given on figure 7.1(b) and 7.3(b) it is very hard to see because the difference is so small. Unity DC gain is desired because if there exists a large difference between the DC gain of the two headphone channels then it will also exist between the two ears. For real binaural sounds, this large difference between the two ears does not exist and therefor it will feel unpleasant for the user of the binaural synthesis if suddenly there is a large difference between the two ears [14, p. 22].

To correct the DC value back to unity gain (0 dB) the sum of the filter taps in the time domain must be equal to one [38, p. 6]. In order to achieve unity DC gain, the filter coefficients must be shifted the opposite way of the non-unity gain. To implement this, first all filter coefficients for a specific filter are summed, which results in the DC gain. This value is then divided by the total number of coefficients, and finally subtracted from all filter coefficients. The result of this is that each coefficient is shifted opposite the DC gain, thus eliminating it. Figure 7.4 shows the effect the DC gain adjustment has on the HRTF for 45° azimuth and 0° elevation.

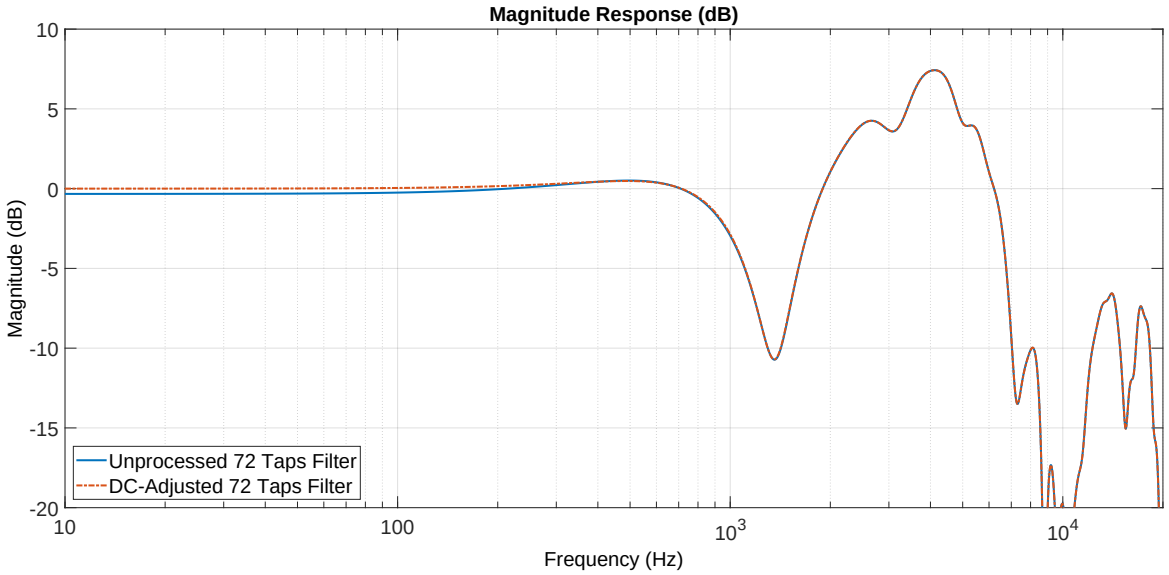


Figure 7.4: Frequency response of the truncated filter with and without DC-adjustment.

7.4.4 Fixed Point and Quantization Errors

Since the DSP uses 16-bit fixed point representation, the filter coefficients must be converted from floating point to fixed point. This is done after all other preprocessing as this process introduces quantization errors. Although these errors are unavoidable when working with fixed point numbers it is still something which must be considered during design. The easiest way to convert the coefficients to fixed point, is to find the largest possible value a coefficient can be, and scale this to be the largest value the 16-bit DSP can process without hitting overflow. The largest value in all filters is slightly larger than two, and thus Q3.13 is needed, since Q2.14 cannot represent values larger than two. To scale the coefficients to Q3.13, all values must be multiplied by $2^{13} = 8192$, and then rounded. This scaling and rounding are also carried out in a Matlab script. Now that all preprocessing has been detailed the final result can be presented in the following section.

7.4.5 Resulting Impulse and Frequency Responses

With all preprocessing applied to the HRTFs, the final filters are compared with the unprocessed filters. First the impulse response is compared for the filter at 45° azimuth and 0° elevation, this can be seen in figure 7.5.

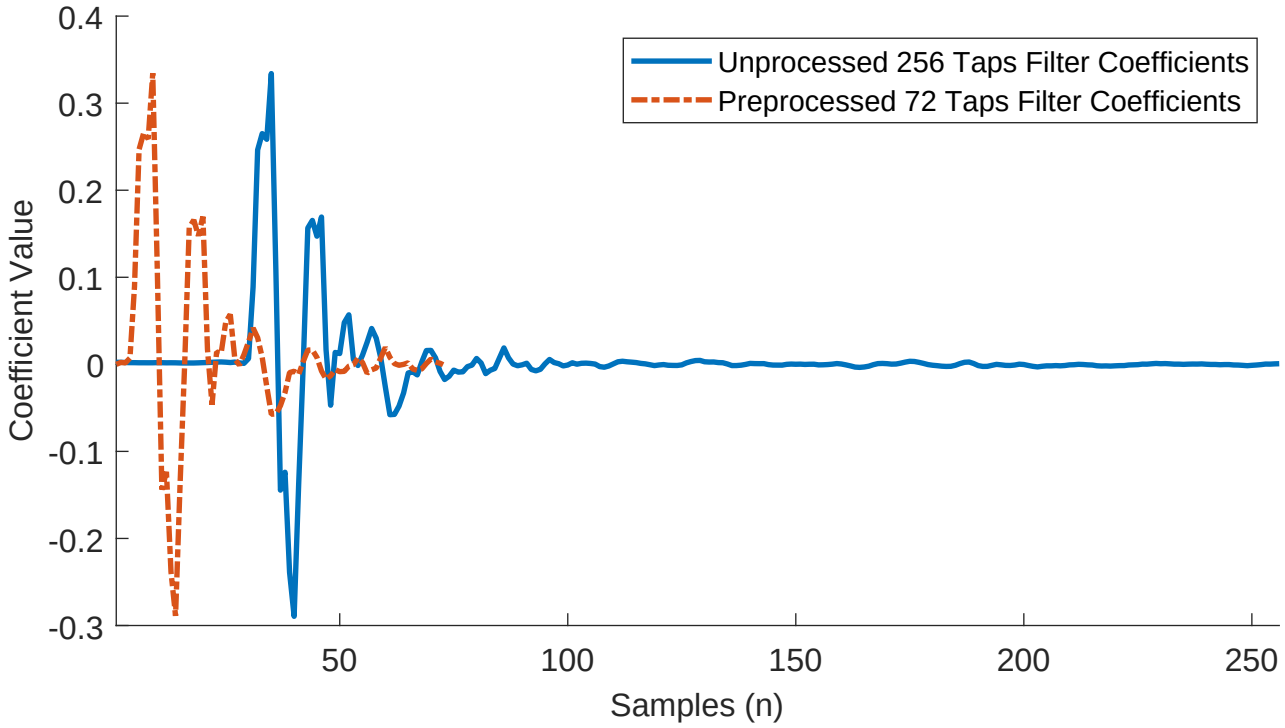


Figure 7.5: Impulse response comparison of unprocessed and preprocessed filter coefficients for 45° azimuth and 0° elevation.

The main difference between the filters, are that the preprocessed filter lacks the arrival time delay, and HRIR pairs do not have any ITD. In addition to this, the number of coefficients is drastically reduced, as is expected due to the truncation.

Next, the frequency responses are compared for the same filter and the result can be seen in figure 7.6. Here it is clear to see that the reduction in coefficients has reduced the frequency resolution, which mostly affects rapid transitions in frequency.

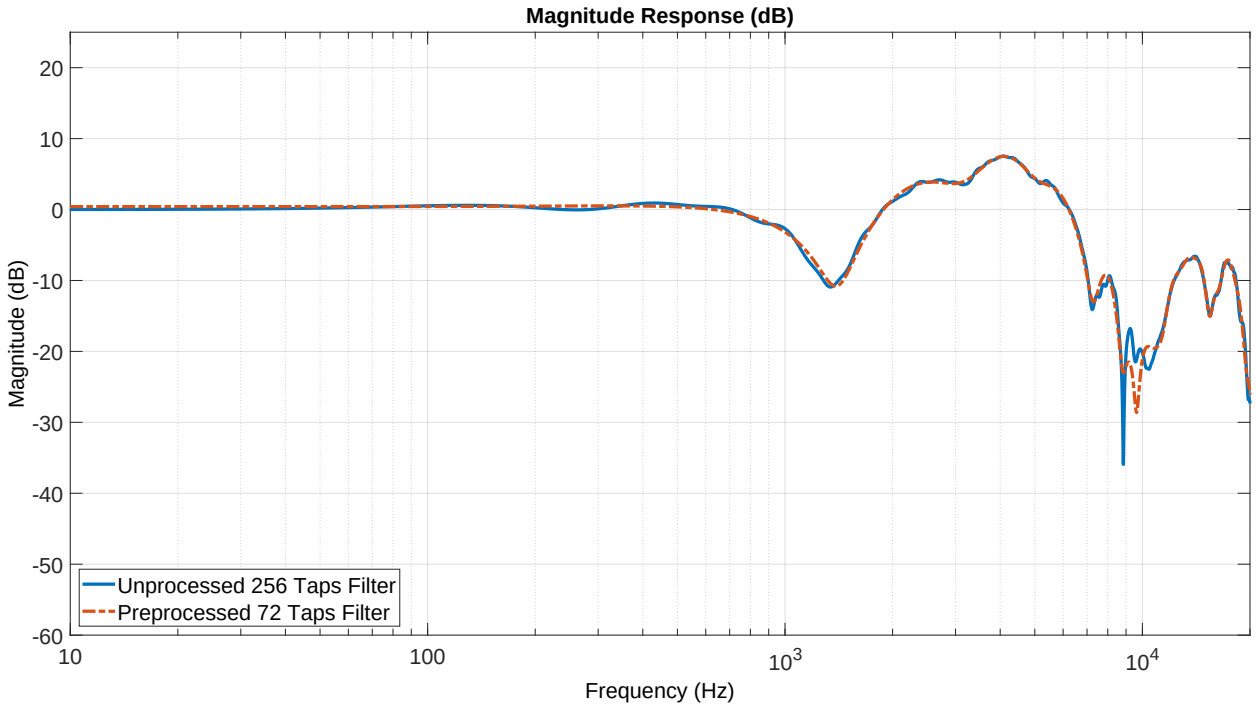


Figure 7.6: Frequency response comparison of unprocessed and preprocessed HRTF for 45° azimuth and 0° elevation.

7.4.6 Converting the HRIR Database to a One-Dimensional Array

Since the Valdemar database is three-dimensional, it has to be converted to a one-dimensional array. This is due to having to save the coefficients on the DSP's flash chip. This is done by saving all filters, from each azimuth angle, to an increasing one-dimensional array. This process is done for all elevations, starting at the bottom at $-90°$ and ending at the top at $90°$.

This results in an array of 872350 entries which is very large, therefore it must be saved outside the DSP's memory. The array is accompanied by an array that contains all start indexes for each elevation incrementation. This array has the value 0 in location 0 in the array, 73 in location 1, 658 in location 2 and so on. This comes as a result of the different elevations having different azimuth resolutions. The way this array is indexed on the DSP is described in section 8.1.

7.5 Switching between HRTFs

To achieve the dynamic cues related to head movement the system has to be able to switch between different HRTFs corresponding to the changing orientation of the head. However, when switching between the HRTFs, audible artifacts are created in the output of the filtered signal. It is desired to avoid the artifacts for a better subjective experience. This section will give an overview of the causes of these artifacts and how to avoid or remove them.

Causes of Filter Switching Artifacts

The artifacts are introduced because the HRTFs are measured discretely and thus discontinuities appear when switching directly between two different HRTFs. These discontinuities can appear in two

different ways; as instantaneous changes in magnitude due to differences in the spectral properties of the HRTFs and instantaneous changes in phase due differences in group delay between the HRTFs. The changes in group delay are because of the different ITDs saved in the filters, as these are implemented as pure delays on the DSP [39]. The most audible of the two types is the one due to changes in group delay [39]. Both types of discontinuities are illustrated on figure 7.7.

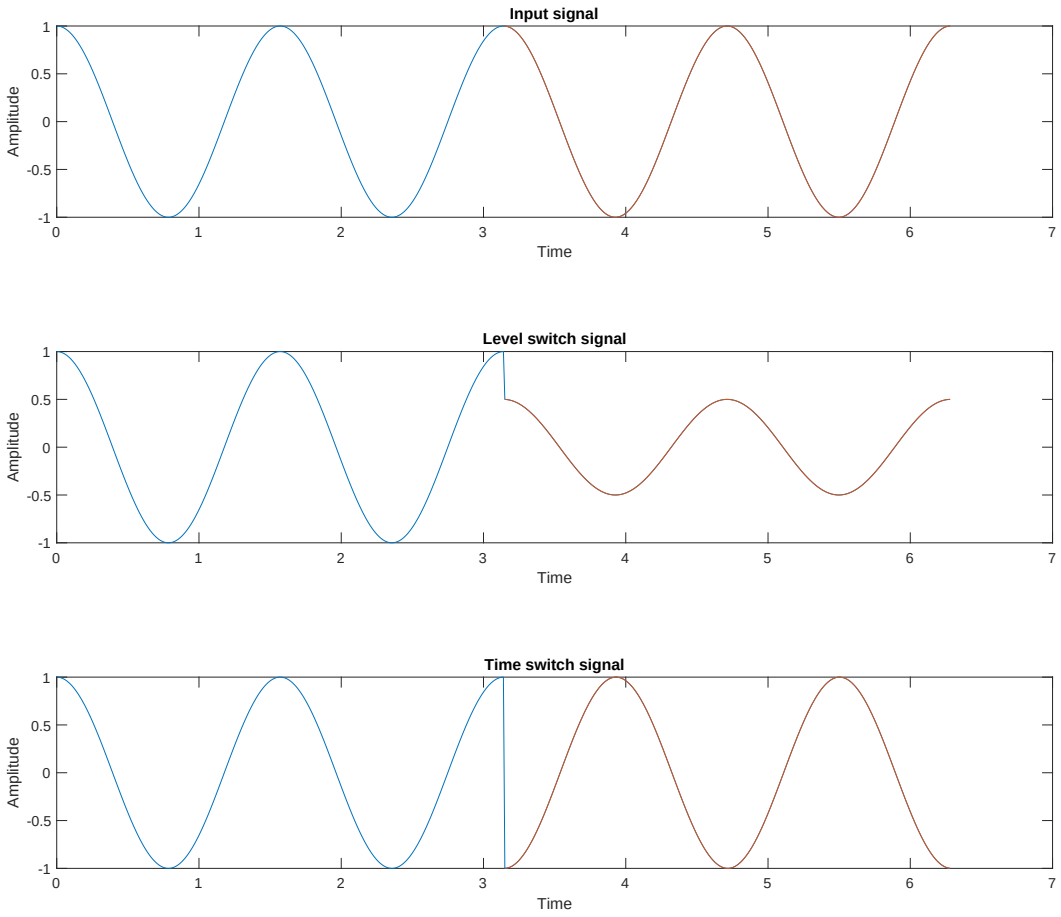


Figure 7.7: Illustration showing examples of the discontinuities created by switching between two filters.

The discontinuities are commonly heard as audible "clicks", and their audibility is most probably proportional to the magnitude of the difference between the HRTFs. As the differences between two HRTFs increases as the distance between them increases, it should be possible to only have inaudible artifacts if switching between HRTFs is sufficiently close to each other. The smallest steps that results in audible switching artifacts have been determined in [39][MIII]. There it was determined that the smallest step between two HRTFs that resulted in audible artifacts due to spectral properties was 5°. As the resolution of the Valdemar database is 2°, no artifacts due to spectral properties will appear. However, for changes in group delay, audible artifacts will appear at differences of 6 µs of delay between two HRTFs. Thus, since a delay of one sound sample will result in a delay of 20 µs due to the sampling frequency of 48 kHz, audible artifact will appear in the output signal. Therefore, something must be implemented to avoid these filter switching artifacts. To avoid these artifacts being audible, the steps can either be made small enough that the artifacts cannot be heard or the transition between the filters can be 'smoothed' to decrease the discontinuities.

Creating Smaller Steps between Filters

The audible artifacts can be avoided if the steps are small enough that the 'clicks' are inaudible. As the existing resolution of the HRTFs is high enough to not create spectral level artifacts, only the steps in delay need to be considered. As the delay equal to one sample is 20 µs fractional delays are required to achieve delay steps below 6 µs. This can be done with a combination of whole sample delays and fractional-delay-filters. If filters for 1/4-, 2/4- and 3/4 sample fractional delay are available, then delay steps down to 5 µs can be achieved, which is sufficient to avoid audible artifacts.

Smooth Switching of the Filters

Instead of avoiding audible artifacts by implementing sufficiently low delay steps, the audible artifacts can be reduced in the output to the point where they are inaudible. This can be done by crossfading the outputs of the initial filter, with the new filter [40]. In crossfading the original output signal, $f(t)$, is multiplied with a time varying window function, $h(t)$, which output will monotonically decrease from one to zero. Meanwhile, the output of the filter, which is being crossfaded to, $g(t)$, is being multiplied with another window function $u(t)$ which monotonically increases from zero to one. These two products, $f(t)h(t)$ and $u(t)g(t)$, are than added together to produce the crossfaded output:

$$s(t) = f(t)h(t) + u(t)g(t) \tag{7.1}$$

where $s(t)$ is the crossfaded output.

However, to crossfade properly, one important condition must be met. As perceived volume is a function of signal power, it is critical that the window functions, $h(t)$ and $u(t)$, are power preserving [41]. This ensures that the perceived volume is not artificially raised or dampened throughout the crossfade duration.

The power preservation of the window functions are dependent on the correlation of $f(t)$ and $g(t)$. If $f(t)$ and $g(t)$ are completely uncorrelated, i.e. they are orthogonal to each other, $h(t)$ and $u(t)$ must adhere to the following relationship to be power preserving [41]:

$$h(t)^2 + u(t)^2 = 1 \tag{7.2}$$

If on the other hand that $f(t)$ and $g(t)$ are perfectly correlated, then they must satisfy [41]:

$$h(t) + u(t) = 1 \tag{7.3}$$

Figure 7.8 illustrates how the windowing function $h(t)$ and $u(t)$ could be constructed using domain restricted cosine and sine functions. It is also shown how the power varies over the crossfading in both the correlated and uncorrelated case. In the correlated case $h(t)$ and $u(t)$ are simply summed together, while in the uncorrelated case it is the sum of the squares of $h(t)$ and $u(t)$.

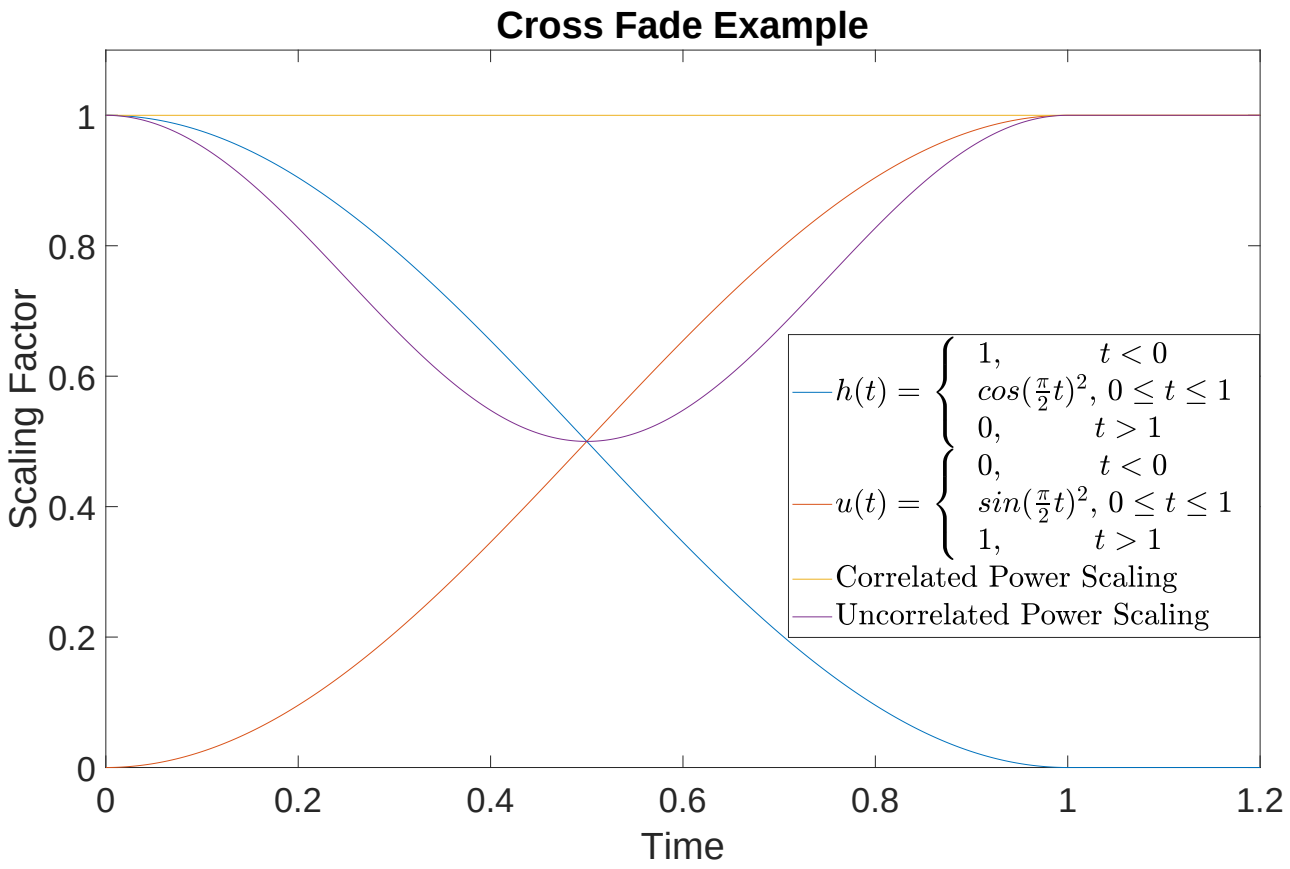


Figure 7.8: Comparison between three frequency responses for headphone equalization filtering.

However, as correlation between signals can continuously vary between one and negative one, equation (7.2) and (7.3) only apply in the case of the correlation being one or zero. Thus, if the correlation between $f(t)$ and $g(t)$ is known, the window functions could theoretically be adjusted to preserve power for arbitrary $f(t)$ and $g(t)$. This is not a trivial task, as it requires that the covariance of $f(t)$ and $g(t)$ must be found [41]. Thus, to simplify the implementation and save valuable computational resources, it is decided to crossfade either according to (7.2) or (7.3). As $f(t)$ and $g(t)$ are filtered versions of the same input signal, the correlation between them is exclusively determined by their differing filtering characteristics. If the two filtering characteristics are very similar, such as in the case of two HRTF filters located near each other in space, then the correlation of $f(t)$ and $g(t)$ will be high. If on the other hand that the two filters are located far apart on the sphere, then their transfer functions can be significantly different, and thus, their correlation is likely low. Since the change of HRTF filters is supposed to be directly tied to head movement, the physical limitations of head movement limit how much the filter can instantaneously change. Assuming a high sampling rate of the user's head orientation, then it is fair to assume that drastic changes in HRTF filters will not occur. Thus, it will be assumed that the signals are correlated, and the windowing functions will be constructed according to equation 7.3. While there exists an arbitrary number of different windowing functions, some perform better than others. A study by Kudo et al tested the perceivable differences between a number of different crossfading windows [40]. From this it was found that while crossfading performed using Fourier series produced the least amount of wave discontinuity, there was no subjective perceivable difference between crossfading using either Fourier series, cosine/sine functions, or a square root function [40].

$$h(t) = cos(t)^2 \tag{7.4}$$
$$u(t) = sin(t)^2 \tag{7.5}$$

Observe that this is identical to the example in figure 7.8.

7.6 Headphone Equalization

For binaural synthesis a pair of headphones is the preferred playback transducer because they deliver the best channel separation between the two ears [10, p. 1]. However, a pair of headphones introduces error into the binaural synthesis, this is due to the characteristics of both the headphones and the ears. The headphones electro-acoustic properties combined with the sound pressure transmission between the headphones and the eardrum can be modeled as a transfer function. Because the HRTFs have very specific frequency responses, any added change in this response can diminish the localization cues in the HRTFs.

To counter the effects of the transfer function from the headphone to the ear, the characteristics can be measured to obtain a transfer function which can then be implemented as an equalization in the binaural synthesis. Once the frequency response of the headphones and ear characteristics are measured it can be turned into an inverse FIR filter based on the impulse response coefficients which is the basis of the equalization filter [14, p. 12]. However, because the headphones cannot reproduce frequencies which are below the audible range, below 20 Hz, then the filter does not need to filter these frequencies and therefore the response can be 0 dB. Conversely, at higher frequencies, e.g. above 7 kHz, the placement of the headphones on the user's head have a major impact on the frequency response's characteristic. This is due to reflections which occur when the sound pressure reaches different parts of the ear. Therefore, these frequencies should be conservatively equalized by the filter, for simplicity the response above 7 kHz is set to 0 dB [14, p. 12]. Optionally, to free computation time the equalization filters can be deconvolved with every HRTF for the respective ears as part of preprocessing of the HRTFs [14, p. 13]. This would result in no additional filtering process on the DSP for the equalization. However, it would also limit the whole HRTF filtering system to only one unique model of headphones. Because of this very strict limit to just one pair of headphones, it is opted to process the equalization filter as a separate filter on the DSP.

In the case of the Valdemar database, the transfer function for the headphones and ear can be obtained by placing the headphones onto Valdemar (the artificial head) and then measuring the impulse response through Valdemar's ears for each ear and corresponding headphone channel. These measurements were completed for the Beyerdynamics DT770 and the test journal can be found in appendix H.
The results of the measurements show a fairly flat frequency response in the low to mid frequency range (20 Hz to 2 kHz) where the maximum gain peaks at 5 dB. And between 3.5 kHz and 7 kHz there is a much larger peak at up to 15 dB which definitely require equalization.

As mentioned previously, ideally the equalization filter would be implemented as an FIR filter because this would be simple to design. This design would consist of the inverse impulse response, which was measured in the test, with However, because the sampling frequency is set to 48 kHz it would require a large number of coefficients to properly represent the headphone and ear response at low frequencies. This effect can be seen on figure 7.9, where the full response (blue line) is plotted and compared with both a 52 taps (red, line and dots) and a 500 taps (yellow line with spaces) representation of the response.

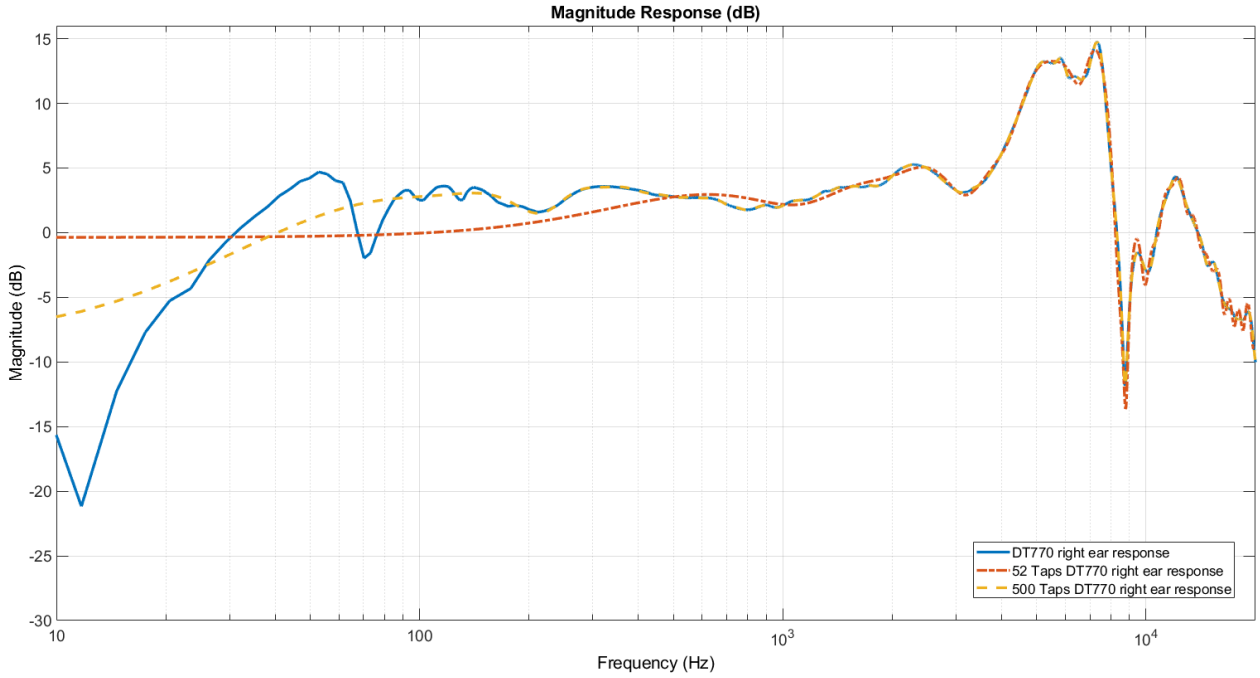


Figure 7.9: Comparison between three frequency responses for headphone equalization filtering.

The 500 taps FIR filter accurately represents the full impulse response, even at low frequencies. However, it would be very unrealistic to implement two 500 taps FIR filters, one for each ear, in succession with the HRTF filtering system on the DSP. For the 52 tap FIR filter, it represents the full response adequately down to 500 Hz however below that it is not able to equalize the low frequencies very well.
For HRTFs there are virtually no spectral detail embedded in the low frequency range [14, p. 23]. For sounds below 1.5 kHz it is the ITD which is the primary localization cue [1, p. 17]. Therefore, for the pure purpose of reproducing localization cues in the HRTF filtering system the 52 taps inverse filter would be a suitable implementation.

Although the headphone's response has been measured and the equalization filter technique has been proposed it is opted for it to not be implemented. Due to time restrictions and the low priority of headphone equalization, it is opted to not proceed and create the inverse filter of the headphone response.

Chapter 8

HRTF System Implementation on the DSP

In the previous chapter it has been discussed in part what is required to implement HRTF filtering on the DSP. This chapter will explain how it is implemented on the DSP. In figure 8.1 is an overview of the data flow of the HRTF filters.

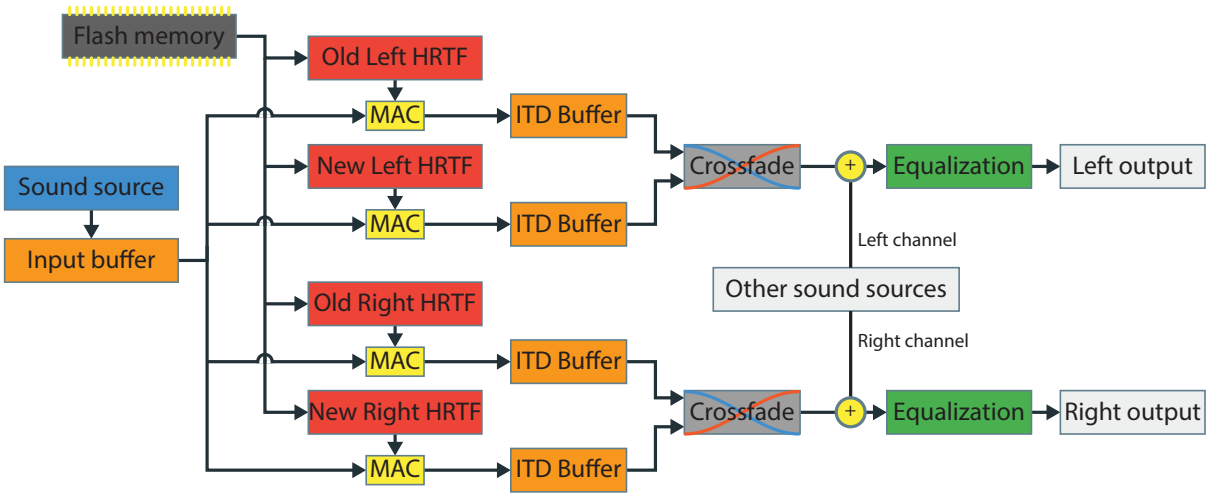


Figure 8.1: Overview of the implemented data flow.

As seen on the overview the filter coefficients are stored on the flash and loaded into the red HRTF buffers. Once a set of filter coefficients are loaded from flash to memory it can be used directly for the MAC operations. The samples from the sound source is first added to the beginning of the circular input buffer. Then the old and new HRTFs are MAC'ed with the input buffer in parallel, first for the left and then right channel. Once a channel has completed its MACs the 'old' and 'new' filter output is added to a circular output buffers. The outputs are then delayed based on the ITD. This is done by saving the latest output of the filter to a circular buffer and then sending an older output to the crossfading section. Crossfading is then done by weighting the outputs from both ITD buffers. Once the outputs have been crossfaded, they can be mixed with other sound sources. And finally, after all signals sources has been mixed, equalization could be performed. Do note that on the DSP, equalization filtering has not been implemented, and the current HRTF filtering system does not support multiple sound sources.

Since the filtering has to be completed before the next signal is sampled to maintain a real-time implementation, the filter is implemented in assembly to use as few instructions as possible. This allows for optimization of the DSP's computing capabilities over what would otherwise be possible in pure C code. Using a 100 MHz clock frequency on the DSP with a 48 kHz signal sampling rate, there is $10^8/48000 = 2083$ clock cycles between each sample. The filtering function requires approximately 400 clock cycles per sound source. This means that if the DSP exclusively performed filtering, it would be possible to simulate three-dimensional audio for five audio sources simultaneously. However, more importantly it means that it is possible to filter two sources, as specified in the requirements,

while using less than half of the available computational power of the DSP. This leaves adequate computational power for calculating head orientation.

The following sections from 8.1 to 8.4 will provide a more detailed explanation of the implementation of the individual parts of the HRTF filtering system.

8.1 Orientation-Based Filter Selection

In order to index the Valdemar database properly, the input angles must be converted to the closest index. The elevation index can be found by checking the remainder when dividing the angle by two, flooring it and adding 45.

$$el_{index} = \left\lfloor \frac{el}{2} \right\rfloor + 45$$

This rounds to the nearest elevation index, which then must be checked against the different elevation ranges to find the azimuth resolution. The elevation ranges and their corresponding azimuth lengths can be seen in figure 8.2. With the azimuth resolution known, the azimuth angle must be rounded to nearest index value, which depends on the elevation. This is done the same way as with elevation, excluding the addition of 45.

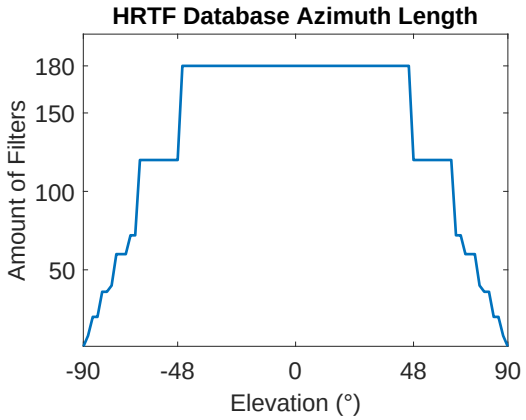


Figure 8.2: Graph showing different amounts of filters of the Valdemar database according elevation.

The elevation index is used to index the elevation start index array, which contains a start index for the HRTF array stored in flash. The start index plus the azimuth index multiplied by 73 (the length of a filter plus ITD), identifies the desired filter's location in flash. For an overview of these arrays, see figure 8.3. For example; an elevation of 12°and an azimuth of 90° returns the indices 51 and 135 respectively. The elevation index gives the address 508446, which is relative to the start of the filter coefficient memory block in the flash array. The actual filter is then located at $508446 + 135 \cdot 73 = 518301$ and 72 addresses forward.

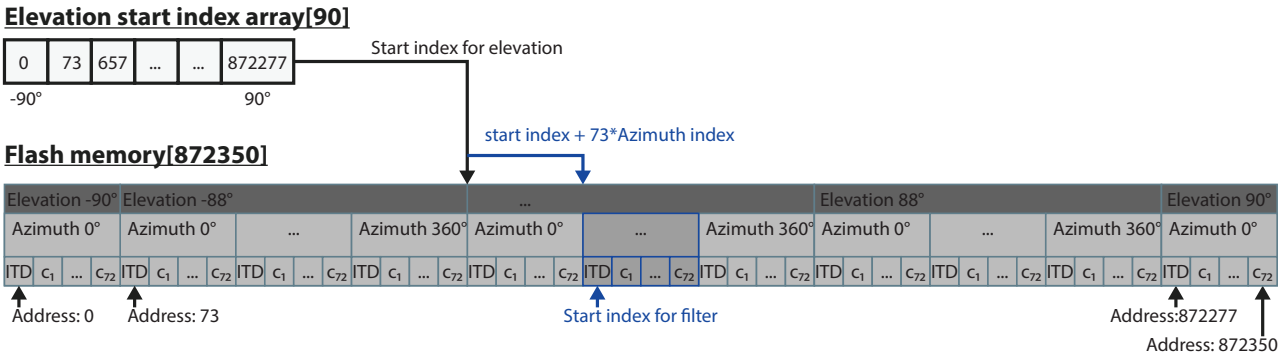


Figure 8.3: Figure showing how the elevation index is used to indicate the start of the filters for each elevation.

To load the filter coefficients onto the flash a program was designed to copy a specified section of an SD-Card directly into the flash. The filters were converted to 16-bit signed integers and then converted to hexadecimal to be easily written to the first 2 MB of the SD-Card. Then the aforementioned program was used to copy the first 2 MB of the SD-Card to the first 2 MB of the flash. It should be noted that the flash use byte addressing. Since the used word-size is 16-bit, the word address must be multiplied by 2 to get the byte address that is used on the flash.

8.2 HRTF Filtering on the DSP

Once the index of the start of the filter is located, the filter is loaded via SPI from the flash into an array that is not currently in use. The HRTF filter arrays can be seen on figure 8.4. The filter is loaded asynchronously. Once it is fully loaded into memory it is set as the new filter. Meanwhile the old 'new' filter is set as the old filter, and the buffer array used for the old 'old' filter is freed to be used when the next filter is to be loaded from the flash. Once the input is sent to the filter, the index of the circular input buffer is decremented by one and the oldest value is overwritten by the newest value which is then set as the start of the buffer.

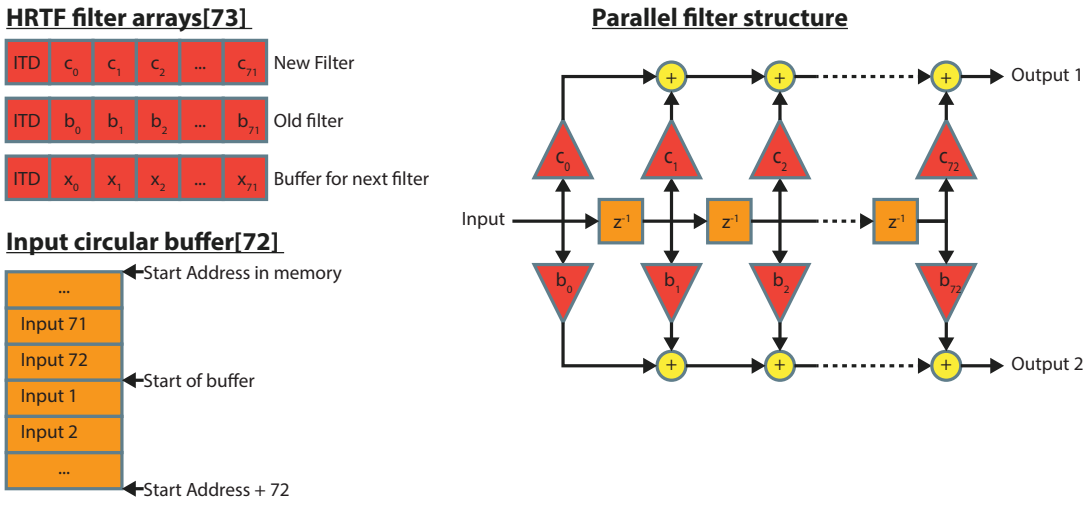


Figure 8.4: Data structures used for the filtering.

The filtering is then executed in parallel as seen on figure 8.4. The filtering is performed using the

DSP's two separate MAC units. The MAC's share the input buffer as one of their inputs, while the new and old filter coefficients are used as their other input. Filtering is first done for left channel and then subsequently for the right channel, both using the same input buffer. The filtering has been implemented for one sound source but could easily be extended for multiple sources.

8.3 HRTF ITD Implementation on the DSP

As the ITD has been extracted from the filters, it has to be added back into the filtering process. The delay that has been extracted from the filters is saved with the filters as the first entry in the HRTF filter arrays, the structure of one of these can be seen on figure 8.5. To perform a sample delay, the past outputs have to be saved. For this purpose, a circular buffer has been implemented as seen on the figure 8.5.

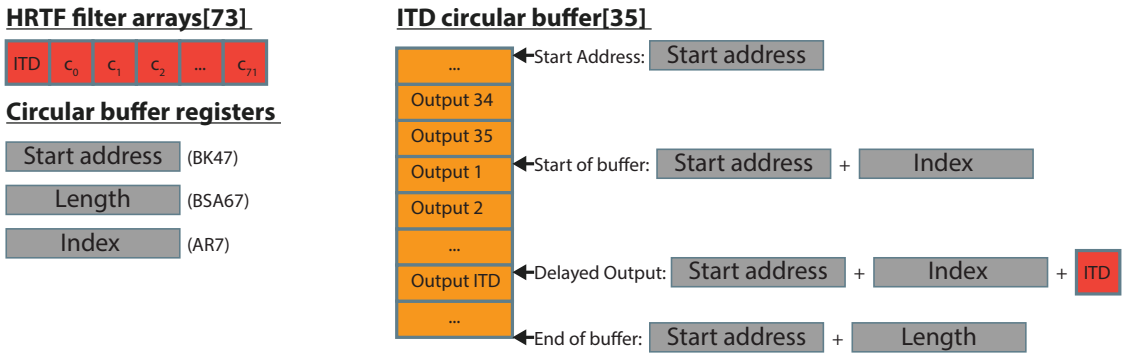


Figure 8.5: Data structures used for adding delay for ITD.

To setup the circular buffer the intrinsic circular addressing was used. The DSP support circular addressing if setup properly. There are a number of registers on the DSP that are used to keep track of the circular addressing. These can be seen on figure 8.5. The assembly to setup the circular buffer and delays the signal corresponding to the ITD can be seen in figure 8.6.

```
1  ;Output buffers ITD L1.
2      bit(ST2 , #ST2_AR7LC ) = #1          ;Enable circular addressing for AR7
3      BK47 = #IM                           ;Set circular buffer length to IM=35
4      BSA67 = *(_output_pointer_l1)        ;Point circular buffer to start of circular
5                                           ;    buffer
6      AR7 = *(_output_index_pointer_l1)    ;Load index from memory
7      (*AR7-)=(*AR7)                       ;Decrement index
8      (*(_output_index_pointer_l1))=AR7    ;Update Buffer start index in memory
9      AR0 = *(_filter_pointer_l1)          ;Get pointer to start of filter array,
10                                          ;    where ITD is located.
11     T0=*AR0                              ;Load ITD
12     (*(AR7+T0))=AC0                      ;Save filter output from AC0 to the buffer
13                                          ;    and add ITD to pointer
14     AC0=*AR7 << #16                      ;Load delayed output
```

Figure 8.6: The implemented assembly used setup the circular buffer and delay the signal.

8.4 HRTF Crossfading on the DSP

To avoid the artifacts from switching filters as described in 7.5, crossfading has been implemented. The crossfading is done by weighting the new and old filters after each filtering process. The weights are taken from a look-up-table (LUT) with 256 discrete steps of the function $cos(x)^2$ where $0 \leq x \leq pi/2$. The data structure of the crossfade LUT can be seen on figure 8.7.

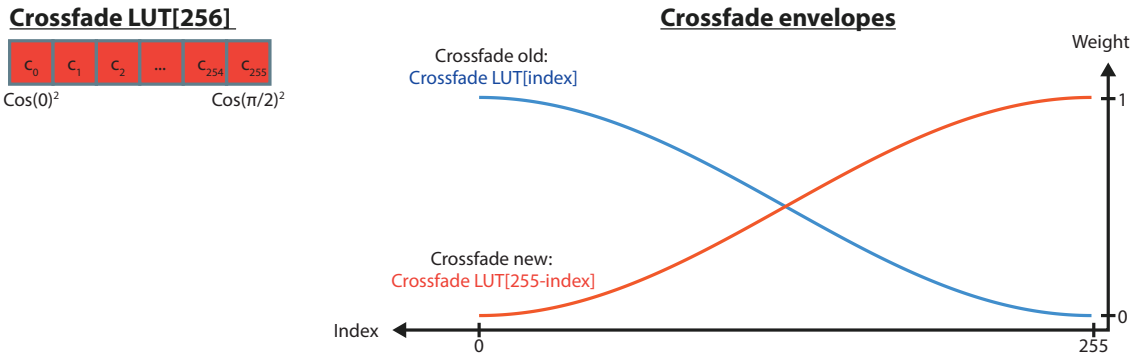


Figure 8.7: Data structures used for crossfading.

After a filter is changed there is a delay of 35 samples before the crossfading starts, this is to ensure that the ITD buffers have been filled with outputs from the new filter. After the delay the crossfading starts by incrementing the index for crossfading every sample. The weights from starts from opposite end of the crossfade LUT, the values of this LUT is illustrated in figure 8.7.

Chapter 9

Test of HRTF-Based Filtering

For the sound filtering subsystem the relevant functional requirements are those regarding the ability to reproduce virtual sound sources around a sphere, filtering for both headphone channels, process the filtering in real time, the degrees of freedom for the sound sources, flat frequency response for the reproduction chain, environmental reflections and distance variation of the sound source. The relevant technical requirements are two simultaneous sound sources and the minimum HRTF resolution around the sphere.

These requirements will be evaluated in the tests in the sections 9.2 to 9.5. However, as some of these aspects of the system were not implemented the following requirements are excluded from the testing: headphone equalization, environmental reflections and distance variation. The following section will explain the testing procedure.

9.1 Testing Procedure

To verify that the HRTF filtering system is working as intended and fulfill the requirements it must be tested thoroughly. To facilitate the testing process as effectively as possible, several tests will be completed at the same time or in succession due to the similarity of the tests. To ease the processing and comparison of the test results a set of HRTF locations with certain intervals on the sphere will be used in several of the tests.

The first test which is going to be carried out is the testing of the preprocessed HRTFs. This is done to verify that the preprocessing is done correctly. To verify this, the original 256 taps HRTFs are compared with the preprocessed 72 taps filters in the frequency domain. Also, the impulse response of the 256 taps HRTFs will be plotted so that the ITD can be compared with the ITD which is extracted for the preprocessed HRTFs. Thereafter, the DC gain adjustments are verified by plotting and comparing the preprocessed HRTFs and the 72 taps HRTFs without the adjustment. Finally, the quantization of the filter coefficients is compared with the original HRTFs.

Assuming that the results of the preprocessed HRTFs show that the preprocessing is done correctly, then these HRTFs are implemented and tested on the DSP. The HRTFs which are used on the DSP will then be measured in the tests and finally be compared with the preprocessed HRTFs to verify that everything is implemented properly on the DSP. The test measurements on the DSP are similar to those carried out for the preprocessing however they will be real world measurements compared to computer computations. Finally, during these tests on the DSP the filter selection system will also be tested.

9.2 Test of the HRTF Preprocessing

This test is conducted in Matlab, where unprocessed HRTFs have been compared with preprocessed HRTFs to see what impact the preprocessing has on the filters. Additionally, the HRIRs are also checked, since the goal is to remove the ITD and shorten them. The test journal can be found in appendix I and it encompasses all of the tests conducted for preprocessing.

All of the preprocessing test results are very satisfactory and therefore it is concluded that the preprocessed HRTFs should function in the HRTF filtering system if they are implemented on the DSP correctly. To see if the implementation of the system on the DSP is done correctly it is tested in section 9.4 to 9.5.

9.3 Test of the Audio Codec on the DSP

As mentioned in functional requirement F.6, the whole audio reproduction chain must be equalized if it does not have an ideally flat frequency response. The ADC and DAC combo (in the AIC3204 codec) on the DSP board must therefore be tested. The frequency response of the codec has been measured for both channels and the test can be found in appendix J. The results of this test show that the right channel has a completely flat frequency response between 100 Hz and 20 kHz, however the left channel only has a flat response between 100 Hz and 2 kHz. It can be seen on figure 9.1, both of the channels have a similar roll-off below 100 Hz. Above 2 kHz there is a lot of fluctuation on the response for the left channel, this fluctuation is very odd compared to the right channel.

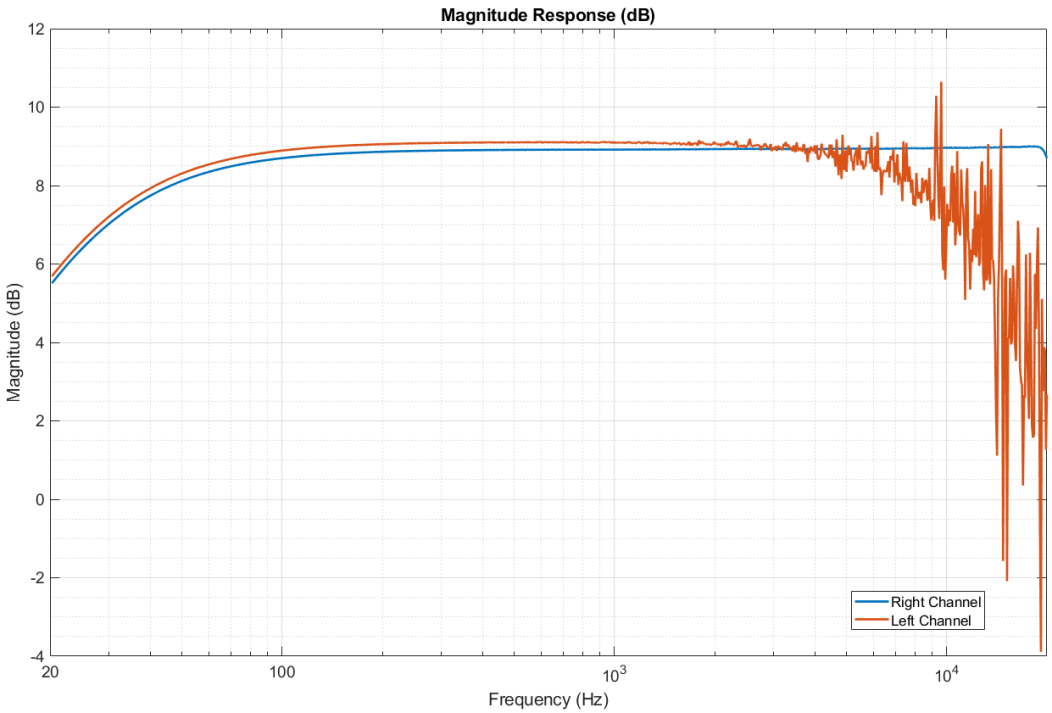


Figure 9.1: Frequency response of both codec channels on the DSP.

As described in section 7.6, an attempt was made to implement an equalization filter for the headphone and ear characteristics, but this implementation has not been completed for the HRTF system. As neither the codec nor the headphones have been equalized the F.6 requirement has not been fulfilled.

9.4 Test of the Filter Selection on the DSP

The filter selection which was implemented in section 8.1 has been tested and the test journal can be found in appendix K. As it was written in the test journal, the DSP cannot process HRTFs between 1° and 90° elevation for the tested azimuth locations because the coefficients are all −1. Therefore requirement F.5 is not fulfilled. However, it was possible to select HRTFs in the opposite elevation, between −1° and −90°. This was done for the HRTF at −45° azimuth and −45° elevation the response of this can be seen on figure 9.2. The HRTF which was tested for negative elevation was not the same as the preprocessed HRTF for the same location, as was shown in the test.

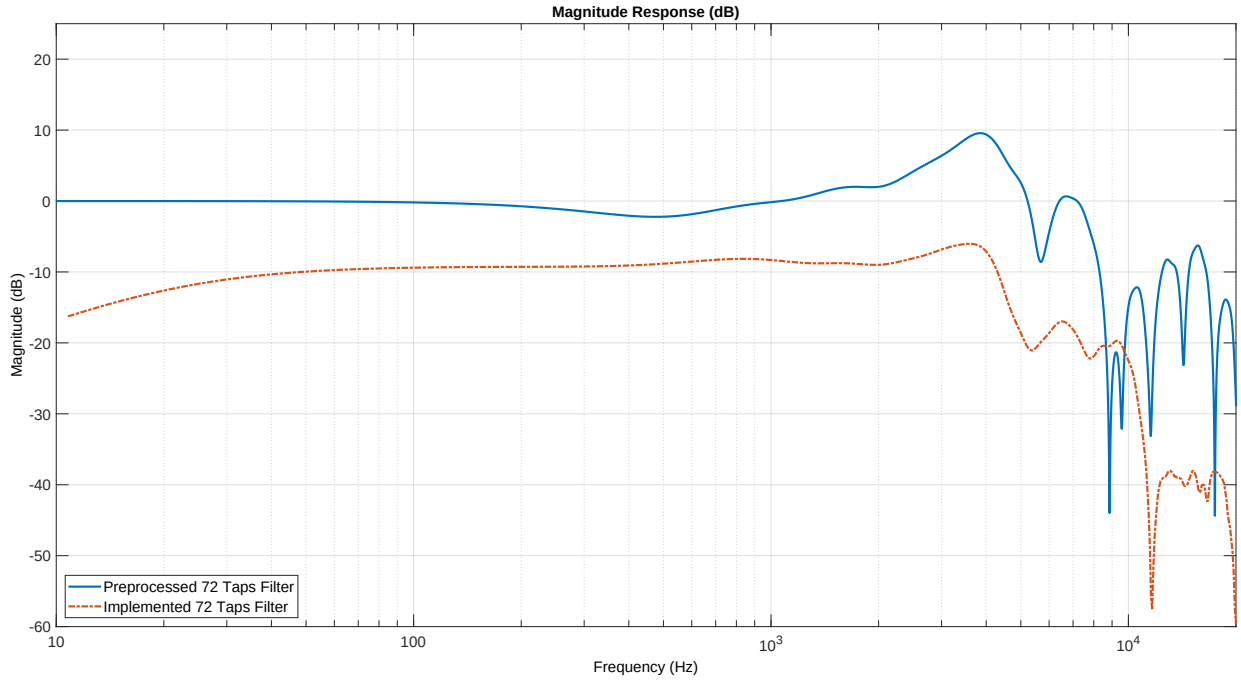


Figure 9.2: Frequency response comparison between the measured HRTF and the preprocessed HRTF for −45° and −45° elevation.

Finally, filtering HRTFs in 0° elevation did prove successful in the test, and several of the HRTFs which were filtered can be seen in figures 9.4 and M.1 through M.4 where they are compared with the preprocessed HRTFs. Therefor, requirement F.1 is deemed to have been fulfilled.

9.5 Test of HRTF Filtering

The ITD has been measured on the DSP and the test journal can be found in appendix L. In this test journal it was proven that the ITD is properly implemented on the DSP, an example of this can be seen for the measured HRTF at −45° azimuth and 0° elevation which is shown on the impulse response in figure 9.3. Here the resulting ITD is equal to 401 µs which is equal to approximately 19 samples. It was also shown in the test journal, that the left channel is delayed by 20 µs, or 1 sample, compared to the right channel when the HRTF at 0° azimuth and 0° elevation was measured.

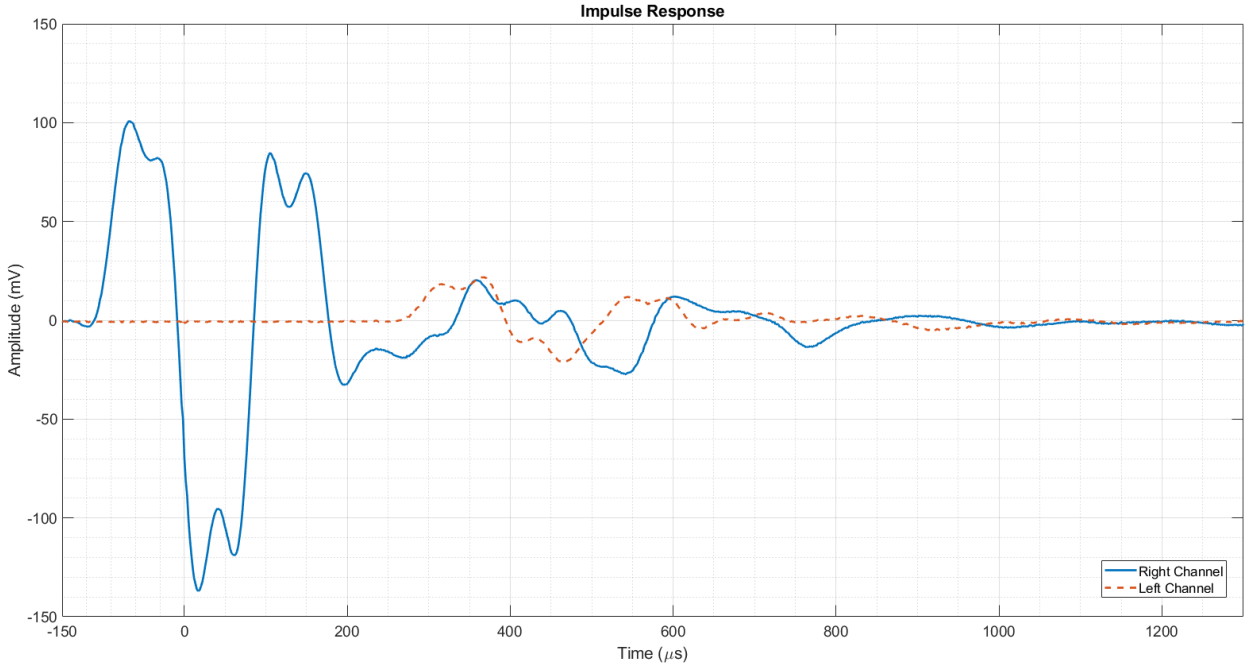


Figure 9.3: Impulse response measured for both channels for 45° azimuth and 0° elevation

The HRTF filtering test has been completed and can be found in appendix M. The results show that the HRTF filtering implementation on the DSP is a success for 0° elevation. An example of this is given for 0° azimuth and 0° elevation which can be seen on figure 9.4. For elevations except 0° the filtering does not work as it was explained in the previous section.

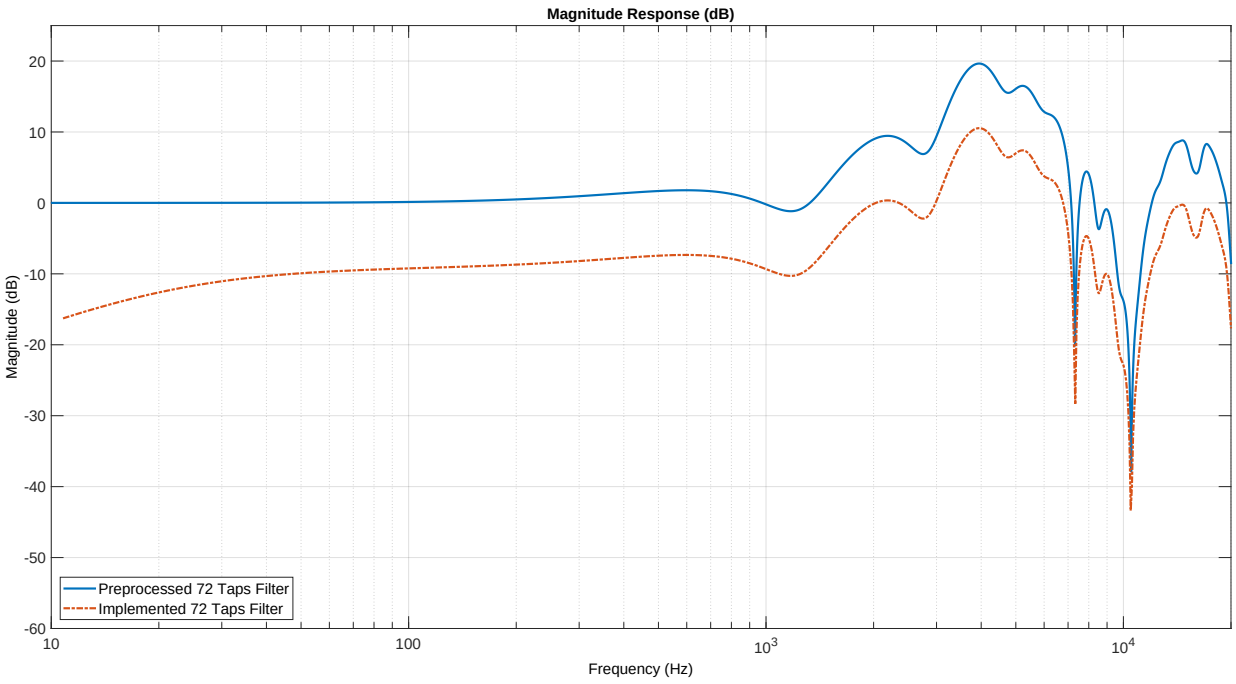


Figure 9.4: Frequency response comparison between the measured HRTF on the DSP and the preprocessed for 0° azimuth and 0° elevation.

9.6 Partial Conclusion of the HRTF Filtering System

As with the orientation tracker the HRTF filtering system cannot fulfill all the relevant requirements stated in chapter 2. Requirement T.1 cannot be achieved because the system is currently only able to filter a single sound source. At 0° elevation, the resolution of the filters in the Valdemar database is only 2° as opposed to the required 1° in T.2. The similar requirement for elevations between 0° to 80° and 0° to −80° is also not fulfilled. The final technical requirement T.5 for the delay between input and output has not been tested and therefore it cannot be concluded whether the requirement is met. This test was deemed less important, and due to time constraints, it was not performed in time. However, based on the completed tests summarized in section 9.2 through 9.5 some requirements have been met. These are the following requirements: F.1, F.2 and F.4. Therefor it can be concluded that the HRTF filtering system can adequately reproduce three-dimensional audio in the most important area, the area where stereo speaks should be located, which is 0° elevation.

Part IV

System Combination and Concluding Remarks

Chapter 10

System Integration

At this point both of the two primary sub systems, head tracking and HRTF filtering, should be merged and integrated into one unified system. Because it has not been possible to implement a functional version of the head tracking subsystem, this merger has not been performed. Thus, this chapter does not document how the sub systems are combined, but rather how the complete system is intended to function.

10.1 Overview of Firmware and Hardware Interfaces

The firmware contains one main loop which calculates the orientation of the user's head, indexes the appropriate HRTFs, and initiates filter extraction from the external flash.

This loop can be interrupted by any one of the system's three interrupt sources. These include the MARG sensor array, codec, and the DSP's internal SPI controller. The MARG sensor array periodically activates its interrupt pin when it has new measurements available. This interrupt signal enables the sensor INterrupt Service Routine (ISR) which starts the process of data extraction. To facilitate this data transfer, an SPI ISR is required. Whenever read data appears in the SPI controllers input registers, an interrupt is raised to activate the SPI ISR. This ISR determines whether a sensor data transfer or a filter coefficient transfer is ongoing. Based on the determined transfer type, the data in the register is then moved to a new location in memory. Finally, if the data transfer is incomplete, the SPI ISR initiates a new transfer to fetch the next data. The final interrupt source is the audio codec. Like with the MARG sensor array, the codec raises an interrupt when it has new audio samples available. When this interrupt is raised the I2S ISR is called. As all audio processing occurs in this ISR, it is system critical that this ISR terminates before the next audio samples are available. A failure to do so can result in lost samples, which would result in distortion and thus negatively impacting the overall audio experience. To ensure that the I2S ISR completes within its given deadline, it is assigned the highest interrupt priority, and it is given the ability to preempt any other executing interrupt. Figure 10.1 illustrates the interaction between these firmware and hardware systems:

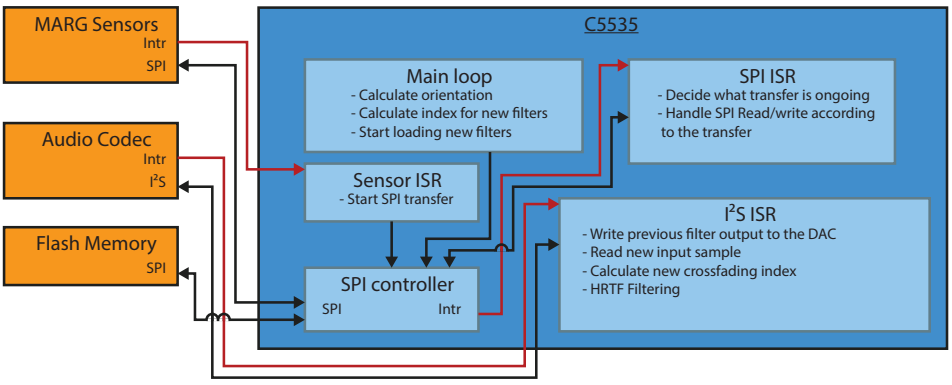


Figure 10.1: Overview of the firmware's interaction with external peripherals. Red lines indicate interrupts, while black lines represent data paths.

10.2 Firmware Structure

This section expands upon the previously presented firmware sections. First the main loop is covered, followed by a detailed description of the content of each of the ISRs.

10.2.1 Main Loop

A flow chart highlighting the key aspects of the main loop's code is shown in figure 10.3. This function contains the default execution pattern, which will be running when there are no active ISRs. If no new MARG data is available the code will run in an infinite loop, continuously checking if new data is made available. When this occurs, the data will first be processed and then subsequently used to estimate the new orientation. Following this, a check is performed to determine when the HRTF filters were previously updated. If some number of orientation updates has passed (arbitrarily set to 25 in the flow chart), an updated set of HRTF filters will be fetched from the flash. Otherwise the CPU will repeat the process of waiting for new MARG data. This artificial restriction of the HRTF update rate is implemented as the orientation calculation may be performed many hundred times a second, and an equivalent filter update rate would exceed what is necessary for authentic reproduction of three-dimensional audio [13]. Thus, if the rate of these updates is limited, CPU resources will be conserved.

10.2.2 MARG ISR

The MARG ISR is simplest of the ISRs. When activated, it initiates the transfer of MARG sensor data from the MARG sensor array to the DSP. Note that this ISR only starts the transfer, and that the the SPI controller ISR is required to complete it. Figure 10.2 illustrates the ISR's execution sequence.

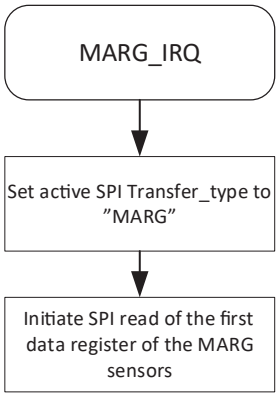


Figure 10.2: Flowchart of the MARG sensor's interrupt service routine.

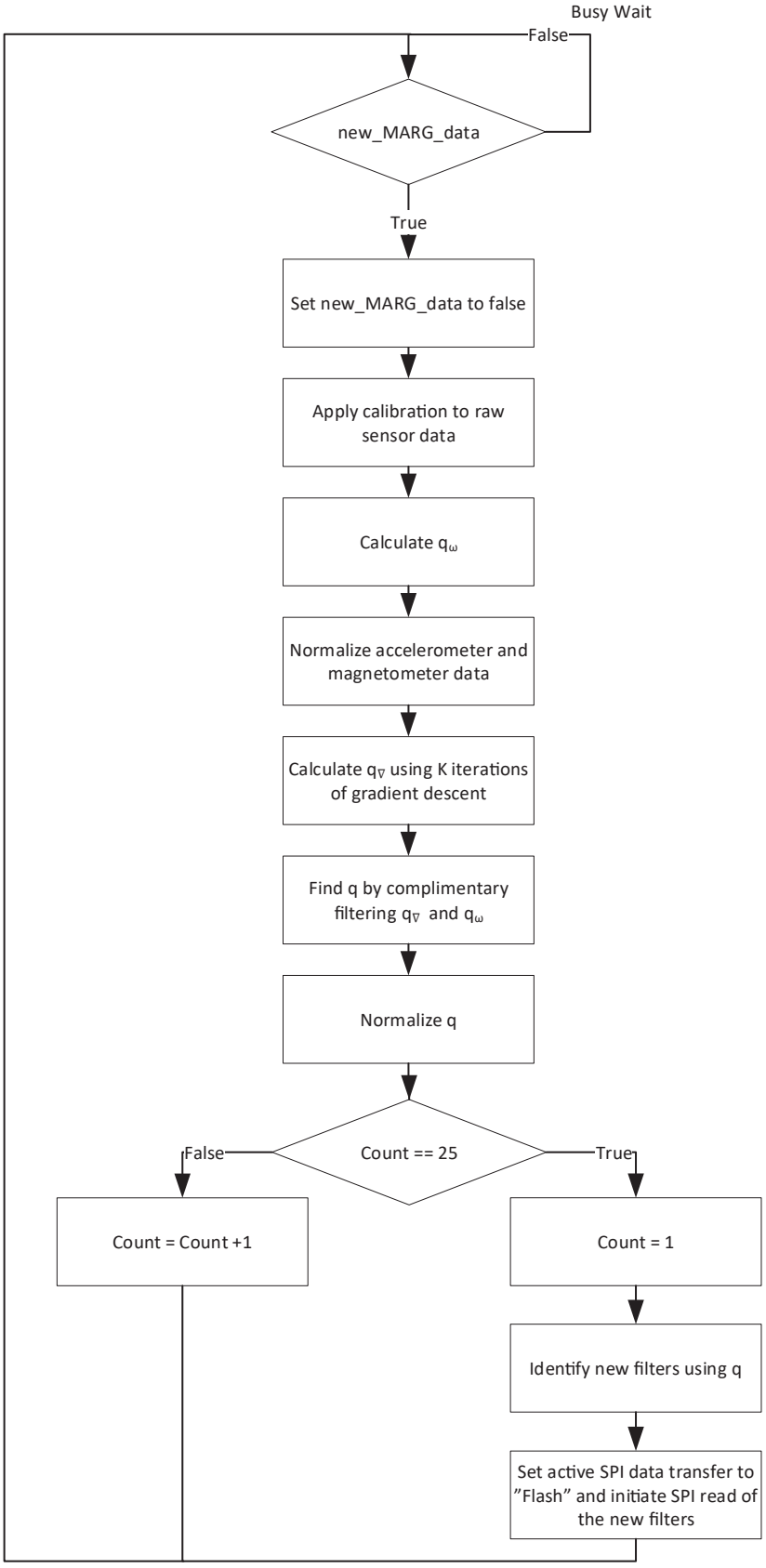


Figure 10.3: Flowchart of the main loop.

10.2.3 SPI Controller ISR

The SPI Controller ISR is activated whenever the SPI controller has read new data into its registers. The ISR then moves the data from the register into memory. The specific memory location is dictated by the type of data transfer being performed. If sensor data is read from the MARG sensor array, then the data is moved into a temporary data array. On the other hand, if the filter coefficients are being read from the flash, then the data is transferred into the inactive filter array(s). If the data transfer is incomplete the ISR instructs the SPI controller to read the next data byte. When all data has successfully been transferred the ISR informs the relevant functions of the event. In the case of MARG data transfer completion, the main loop is notified and thus allowed to calculate a new orientation using the new data. Following FIR coefficient data transfer completion, the pointers used to index the HRTF filters are updated so that the newly imported filters become the "new" filters, and the current filters become the "old" filters. Additionally, the crossfading indices are reset, and the crossfading state is set to "init". It should be noted that this implementation assumes that it is impossible for a flash and a sensor data transfer to occur simultaneously. As flash data fetching can only be initiated immediately following an orientation calculation, then the sensor sampling period must be higher than total execution time required to calculate the orientation, fetch the filters coefficients from the flash, and any intermediary audio processing. If this is not possible, then a more complicated SPI management system must be constructed. A flow chart of this ISR is provided in figure 10.4.

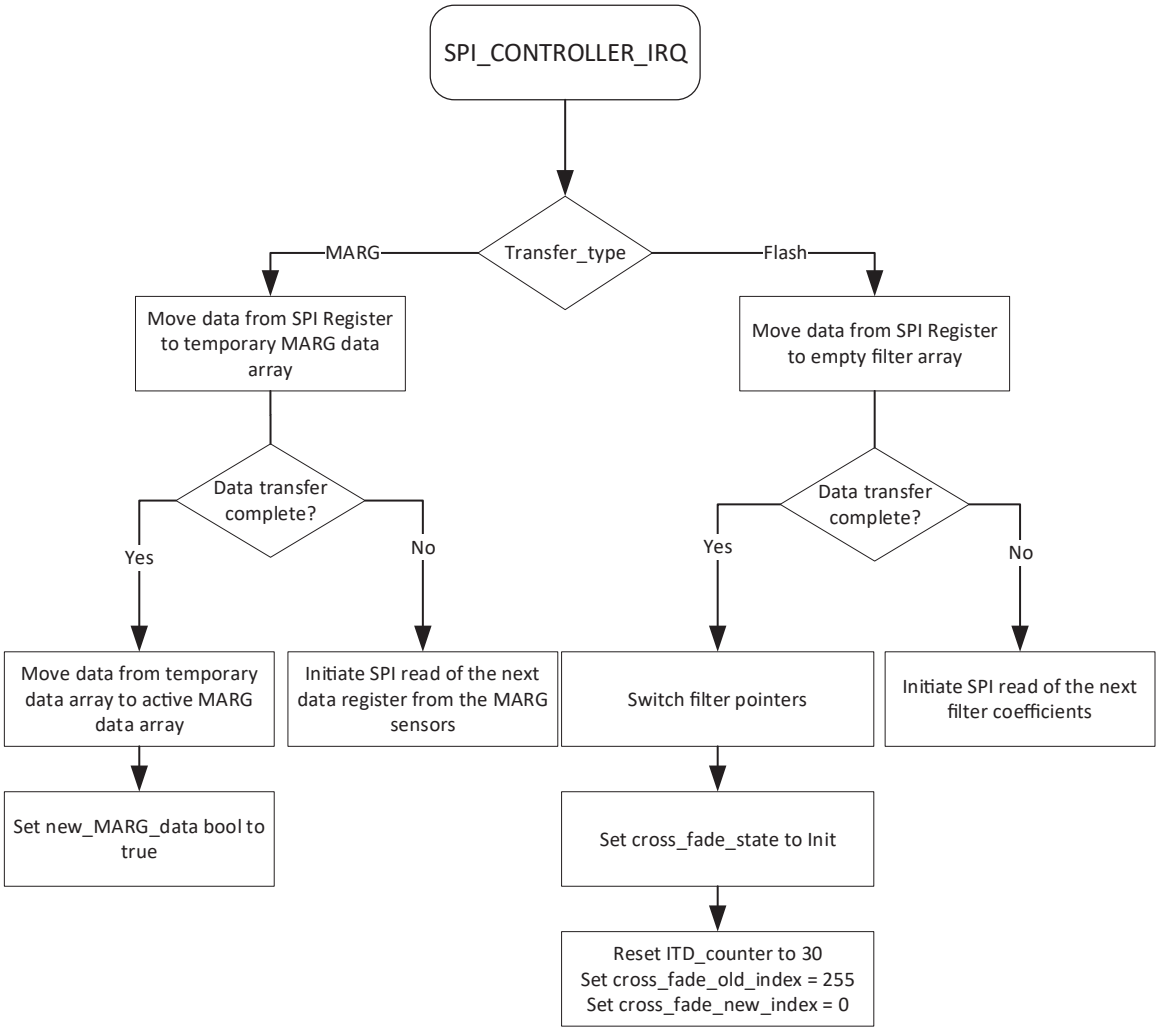


Figure 10.4: Flowchart of the SPI Controller's interrupt service routine.

10.2.4 Codec ISR

When the codec interrupt is raised, the codec ISR should preempt any other code being executed. The ISR starts by writing the previously calculated output samples to the DAC. Following this, the crossfading behaviour is determined. If new filters were previously loaded into memory, then the crossfade state must be set to "init". In this case no crossfading is performed (as the crossfading indices are kept constant), since the "new" filters's ITD buffers are being filled with new output samples. Crossfading remains in the init state for maximum ITD, set to 30 samples in this system. This ensures that the ITD buffer will be full for all possible HRTFs before crossfading is performed. After the maximum ITD, the 30th sample, the crossfading state is changed to "Running". In this state the ISR updates the crossfading indices every sample and crossfading is thus performed. Once the ISR has iterated over the entire crossfading window the crossfading state is set to "idle", where the crossfading indices will no longer be updated, thus completing the crossfade.

After the crossfading behaviour is determined, the audio signals will be filtered, crossfaded, summed, and finally equalized, according to the principles described in chapter 8.4. Figure 10.5 illustrates the ISR's functionality

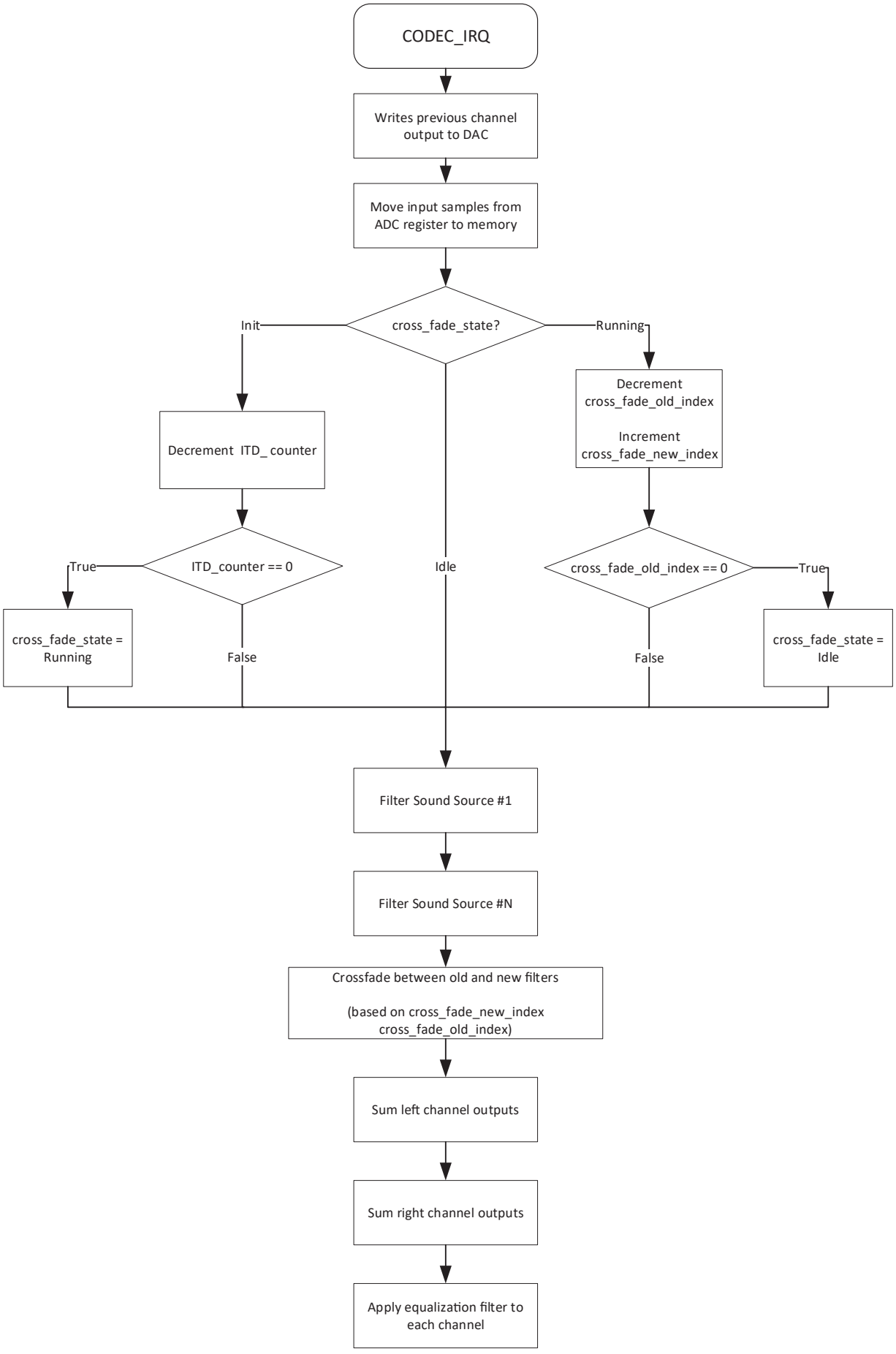


Figure 10.5: Flowchart of the codec's interrupt service routine.

Chapter 11

Discussion

In this chapter the process of combining the two main subsystems will be discussed further, as it was not possible to implement a combined system to be tested as a whole. Lastly, some of the problems encountered both during and after the design of the system and hypothetical solutions to these, will be discussed.

As stated in chapter 10 the proposed combination of the two developed subsystems were not implemented on the DSP, due to inadequate performance of the head tracking system. It was therefore decided that instead of spending time on further development of the head tracking system, effort would be put into analyzing why it did not work as intended.

11.1 Analysis of Head Tracking System

As concluded in chapter 6 the orientation tracker system did not fulfill any of the requirements stated in chapter 2. Even with more thorough tests this conclusion is not assumed to change and thus it makes sense to discuss what may have caused the system to perform at the level documented.

One possibility is that the calibration of the sensors was not accurate enough, as it has been discovered that calibration is crucial for the algorithm. However, even though the calibration of the sensors was performed with care it still depends on the equipment and knowledge available. More advanced calibrations would require an understanding of more complex models of the sensors than those inside the scope of this project. Additionally, if dedicated calibration equipment had been available, the process of calibrating the sensors could have been more structured and repeatable. An example of such equipment would be a device that could accurately rotate an element in three degrees of freedom. This could help with calibrations regarding orthogonality errors in the accelerometer and magnetometer, as well as the tests performed.

Another possibility is an error in the implementation of the orientation tracking algorithm on the DSP. Before the algorithm was implemented on the DSP, it was implemented on a PC to compare the fixed point performance to floating point as seen in section 5.5. The data used to compare the two implementations came from a test similar to the tests in appendix G and F. In the comparison both implementations on the PC showed better accuracy than the one from the tests on the DSP while the convergence rates were somewhat slower. Thus it is assumed that the algorithm itself is working as intended.

11.1.1 Thoughts and Comments Regarding Tests

With regards to the tests performed to determine the behavior of the orientation tracker, some insights were gained towards the end of the project that results in a different interpretation of the results. It was discovered that some of the tests performed were not extensive enough to make decisive conclusions. The main problem was testing an insufficient number of combinations of orientation and axes of rotation. To give an example, during the accuracy tests the rotation for a given axis were only performed at a single orientation. However, to ensure that the measured accuracy of the system is correct each axis should be placed in three perpendicular orientations, to make sure that the result

was independent of both orientation and axes of rotation. Additionally, all tests were only performed once and thus the accuracy of the tests might be inadequate. Instead, the same tests should have been repeated a number of times and their results compared to each other to give more validity to the formed conclusions based on the tests. As these insights were achieved towards the end of the project it was chosen not to repeat all the previous tests in this manner due to the time constraints of the project.

In addition, to the analysis of the head tracking system, it is worth looking into the results and performance of the HRTF filtering system as well, this is done in the following section.

11.2 HRTF Filtering System

Although the HRTF implementation is somewhat successful, as it is shown in chapter 9, there is still room for improvements, further development and additional tests. There are also some crucial flaws in the implementation which were discovered in the test journals in appendix J and K. These different topics related to the HRTF filtering system will be discussed in this section.

11.2.1 Incomplete and Omitted Implementations

As it is detailed in chapter 9, there are several relevant requirements which were not fulfilled for the HRTF filtering system. This section seeks to discuss the implemented systems which were either left incomplete or omitted in the implementation process.

The first and most important of these is that only one sound source was implemented. Having two sound sources is absolutely necessary to be able to implement a stereo sound stage [1, p. 23]. However, because the filtering process required to add any number of additional sound sources is identical to the single sound source filtering process, it should prove easy to implement two sound sources. The only limiting factor which prevents the addition of extra sound sources is the required computation time and internal memory. This is because the whole filtering process must be completed before a new sample of audio is received to maintain real-time audio processing. It is deemed that there is still enough room to implement an extra sound source, however there was not enough time left within the project to implement and test this addition.

Secondly, there is the resolution of the HRTF database which has been implemented. For this database, the highest resolution does not fulfill the requirements. This is in part due to a decision to not interpolate between the HRTFs which would result in the required resolution. It was opted to not interpolate because the new database would require a lot more memory on the DSP, it would not be a uniform resolution for the whole sphere, which would make it harder to index, and interpolating the HRTFs to obtain a new sphere would be a very time consuming task. The other reason why it was decided against fulfilling this requirement is because it is based on the very edge cases of psychoacoustic capabilities of human hearing. For example, the minimum audible angle is based on tests where the test subjects are asked to discriminate between pure-tone pulses [5, p. 3]. These pure-tone pulses are a lot easier for humans to discriminate very accurately, compared to broad frequency audio like music. Based on this discussion it is deemed that the resolution of the Valdemar database is more than adequate for the use case of reproducing three-dimensional audio.

As it has been shown in appendix K, the filter selection implementation is faulty for all elevations except 0°. This result is likely because the HRTF database was incorrectly loaded into the flash memory on the DSP, which would explain why positive elevations returned the value -1 for every filter

coefficient. For negative elevations it is unclear whether the mistake is due to faulty storing of the database in the flash memory or the indexing function. Troubleshooting this would be a high priority if more time was allocated for the project.

The audio codec used on the DSP is another subject of error which was discovered in appendix J. The measured frequency response of the codec showed a lot of error at high frequencies on the left channel of the DAC's output. However, no further tests were able to be completed so only speculation about these results can be made. It is assumed that this error is due to an abnormal sampling rate for the left channel. This might be the case because the system is receiving two interrupts from the codec when only one is required by the program code. One aspect of this test that is interesting to look at, is that the DSP's codec was programmed to send the input of the ADC directly to the DAC. In a different test, which can be found in appendix M, the DSP was programmed to filter the input before sending it to the DAC. This test also showed signs of the same errors at higher frequencies for the left channel. It can also be noted that the response of the left channel did not match the frequency response of the corresponding preprocessed HRTF at higher frequencies as precisely as the right channel did. However, the left channel HRTFs would likely still be usable for filtering three-dimensional audio.

The final incomplete HRTF filtering implementation is the headphone equalization. This was not completed due to its low priority. However, during preliminary testing, it was also deemed that the frequency response of the Beyerdynamics DT770 is adequately flat for reproduction of three-dimensional audio without any equalization. These headphones were also used in preliminary tests with a Matlab script, which reproduces three-dimensional audio, these tests had surprisingly good results and thus they will be detailed in the next section.

11.2.2 HRTF Filtering Implementation in Matlab

As the HRTF filtering system was being designed, it was primarily developed in Matlab where both preprocessing, crossfading and filter processing were implemented to imitate the exact system that would be implemented on the DSP. This Matlab implementation is programmed to filter the HRTFs exactly like the fixed-point DSP would. This made way for a Matlab demo of the same system which is implemented on the DSP. During primitive listening tests made within the project group, we experienced a better three-dimensional audio reproduction in the Matlab demo, compared to the implemented demo on the DSP. Due to the primitive nature of these listening tests, we are unable to quantify the difference between the two demos, in the sense of what was implemented differently in the systems.

Furthermore, some proposals for further developments to the system has been found, which could potentially increase the functionality and performance of the implemented system.

11.3 Further developments

To further increase the immersion some additions to the system could be made. Firstly, the addition of another orientation tracker could increase the quality of the three-dimensional audio experience. By placing an orientation tracker on the body of the listener, two improvements could be seen. Firstly, it would make it possible to determine the head's orientation in relation to the body. If this information is combined with the right set of HRTFs, the filtered sound could be made to sound more realistic, as the effects of reflections from, e.g. the shoulders, could be modelled more accurately. Additionally,

it would also create a reference frame wherein to place the simulated sound sources, which would allow the user to move around while using the system. By placing the sound sources in the sensor frame of the body sensor, a sound placed "in front of" the user would keep this relation as the listener walks around inside a room or on in the streets. However, this will remove some of the immersion achieved with head tracking alone, as the sound sources once again move in space in relation to body movement. Thus, this addition should be optional and to be used when the situation is right for it. As the system is already intended to be used together with smartphones and similar devices which by default possess a MARG sensor array, a reasonable solution would be to use these to track the orientation of the body.

11.3.1 Optimization of Implemented Algorithms on the DSP

To achieve faster calculation results, some optimization of the program on the DSP is necessary which should be possible. Not enough effort has been put into the individual small parts of the currently implemented algorithm. The only part where an effort has been put into optimizing is the way multiplication is performed, as this operation is performed many times in the algorithm. However, even this operation could possibly be optimized even further. Other parts that could possibly be optimized is when calculating and using the Hessian matrix. Through the tests in Matlab it was determined that the Hessian matrix is always a symmetric matrix. However, each element is calculated separately in the implementation on the DSP, which means that approximately half of the calculations for calculating the Hessian could be removed. Optimization of parts like these would result in faster iterations of the algorithm and in turn faster convergence rates or make it possible to add multiple sound sources as more computation time would be available.

Chapter 12

Conclusion

This report has sought to find out if adding the element of head tracking to existing technologies for three-dimensional audio would increase the immersion perceived by the listener. A system consisting of real time head tracking and sound filtering on a DSP was proposed in section 2.5 together with system requirements based on human psychoacoustic phenomena and functions derived from the analysis in chapter 1.

The two developed subsystems were tested individually to determine if the relevant requirements regarding each of them were met. The tests of the orientation tracker showed that while the system is without drift and can accurately calculate an orientation within the given time constraint, the resolution and accuracy of the calculated orientations is insufficient to reliably use for head tracking. The subsystem for filtering audio with HRTF-based filters proved more successful, as it can create a three-dimensional effect with a single sound source placed in free space. However, this effect could only be created in the horizontal plane as it was not possible to switch to filters corresponding to sound sources with an elevation outside 0°.

From the tests it can be gathered that with the implemented subsystems it is possible to meet all of the priority functional requirements and the technical requirement for resolution of the HRTF filters has been met where the required resolution is below 2°. The rest of the requirements for the system has not been met.

However, as the number of tests of the orientation tracker were not enough to definitively conclude if the observed performance was descriptive for the entire system or only regarding an edge case, the validity of the designed implementation cannot be determined. In fact, as the tests of the implementation on a PC showed much higher accuracy in similar tests, it is likely that the designed algorithm for orientation tracking is working properly and that the error is a result of an errorous implementation on the DSP. Additionally, the inability of the HRTF filtering subsystem to switch to filters with an elevation is also contributed by a faulty implementation on the DSP, as the designed algorithm for filter selection has showed to work properly in a combination of Matlab and C.

In conclusion it has been possible to design and implement algorithms for orientation tracking of the head with fixed-point arithmetics and audio filtering with preprocessed HRTF-based digital filters on a PC. Additionally, the audio filtering with preprocessed HRTF-based digital filters has been converted into a real time implementation on a DSP, with the ability of changing the position of a sound source in the horizontal plane.

Bibliography

[1] B. Xie, *Head-related transfer function and virtual auditory display.*

[2] Google, "Youtube Press Information Page," https://www.youtube.com/intl/da/yt/about/press/.

[3] ——, "Google Play Store Site for Netflix Application," https://play.google.com/store/apps/details?id=com.netflix.mediaclient.

[4] J. Blauert, *Spatial hearing : The psychophysics of human sound localization.*

[5] A. W. Mills, *On The Minimum Audible Angle.* Acoustical Society of America, 1958.

[6] J. Blauert, *Communication Acoustics.* Springer, 2005.

[7] H. L. Pick, D. H. Warren, and J. C. Hay, *Sensory conflict in judgments of spatial direction*, July 1969.

[8] R. Steinmetz and C. Engler, "Human perception of media synchronization," pp. 737–750, August 2001.

[9] ITU Radiocommunication Assembly, "Relative timing of sound and vision for broadcasting," 1999.

[10] H. Møller, D. Hammershøi, C. B. Jensen, and M. F. Sørensen, "Transfer characteristics of headphones measured on human ears," 1995.

[11] P. Minnaar, J. Plogsties, S. Olesen, F. Christensen, and H. Møller, *The importance of head movements for binaural room synthesis - a pilot experiment*, 2000, dAGA 2000, March 20-24, 2000, Oldenburg, Germany DAGA 2000, March 20-24, 2000, Oldenburg, Germany.

[12] D. R. Perrott and K. Saberi, "Minimum audible angle thresholds for sources varying in both elevation and azimuth," 1990.

[13] J. Sandvad, "Dynamic aspects of auditory virtual environments," 1996.

[14] H. M. Dorte Hammershøi, "Binaural technique - basic methods for recording, synthesis and reproduction," 2005, published in Communication Acoustics (pp. 223-254).

[15] M. H. Usmani, "Improving headphone spatialization for stereo music," May 2015.

[16] D. W. Grantham, B. W. Y. Hornsby, and E. A. Erpenbeck, "Auditory spatial resolution in horizontal, vertical, and diagonal planes," 2003.

[17] T. Instrument, "Tms320c553x datasheet," 2014, sPRS737C.

[18] Digital Spectrum, "Tms320c5535 ezdsp technical reference," 2011.

[19] A. Lotto and L. Holt, "Psychology of auditory perception," July 2010.

[20] OptiTrack, "General FAQs," https://optitrack.com/support/faq/general.html, 2019.

[21] TDK, "MPU-9250," https://store.invensense.com/ProductDetail/MPU-9250-InvenSense-Inc/487537/pid=1135, 2019.

[22] Veronica Romero, "Can low-cost motion-tracking systems substitute a Polhemus system when researching social motor coordination in children?" https://link.springer.com/article/10.3758/s13428-016-0733-1, 2017.

[23] TDK, "Prime 13," https://optitrack.com/products/prime-13/, 2019.

[24] H. Ferdinando, H. Khoswanto, and D. Purwanto, "Embedded Kalman Filter for Inertial Measurement Unit (IMU) on the ATMega8535," pp. 1–5, July 2012.

[25] Volodymyr V. Kindratenko, "A survey of electromagnetic position tracker calibration techniques," http://www.ncsa.illinois.edu/People/kindr/papers/vr00_paper.pdf, 2000.

[26] Sparkfun, "Picture of MPU9250," https://cdn.sparkfun.com//assets/parts/1/1/3/0/6/13762-00a.jpg, 2017.

[27] ——, "MPU9250 Datasheet," https://www.invensense.com/wp-content/uploads/2015/02/MPU-9250-Datasheet.pdf, 2017.

[28] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation."

[29] L. N. Trefethen and D. Bau, *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics, 1997.

[30] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–7, 2011.

[31] A. Narayan, "How to Integrate Quaternions," https://www.ashwinnarayan.com/post/how-to-integrate-quaternions/, 2017.

[32] "Magnetic declination in aalborg, denmark." [Online]. Available: http://www.magnetic-declination.com/Denmark/Aalborg/679014.html

[33] A. Antoniou and W. Lu, *Practical Optimization*.

[34] E. Oberstar, "Fixed-Point Representation & Fractional Math Revison 1.2," August 2007.

[35] B. Bovbjerg, F. Christensen, P. Minnaar, and X. Chen, "Measuring the head-related transfer functions of an artificial head with a high directional resolution."

[36] F. Christensen, "Binaural technique with special emphasis on recording and playback," 2001.

[37] J. Sandvad and D. Hammershøi, "Binaural Auralization. Comparison of FIR and IIR Filter Representation of HIRs," March 1994.

[38] D. Toledo and H. Møller, "Issues on Dummy-Head HRTFs measurements," 2009.

[39] P. F. Hoffmann, "Characteristics of Head-Related Transfer Functions (HRTFs) - Discrimination and Switching between Adjecent Directions," August 2008.

[40] A. Kudo, H. Hokari, and S. Shimada, "A study on switching of the transfer function focusing on sound quality," November 2014.

[41] M. Fink, M. Holters, and U. Zölzer, "Signal-matched power-complementary cross-fading and dry-wet mixing," 09 2016.

[42] L. Euler, "General formulas for any translation of rigid bodies."

[43] B. Stephen and L. Vandenberghe, *Convex Optimization*, 2009.

[44] T. Ozyagcilar, "One example of tests which are omitted in the system at hand is accuracy verification over temperature and humidity." Nov 2015.

[45] H. Vikne, E. Bakke, K. Liestøl, S. Engen, and N. Vøllestad, "Muscle activity and head kinematics in unconstrained movements in subjects with chronic neck pain; cervical motor dysfunction or low exertion motor output," November 2013.

[46] H. Zangemeister, S. Lehman, and L. Stark, "Simulation of head movement trajectories: model and fit to main sequence," February 1981.

[47] M. C. M. Wright, *Lecture Notes on the Mathematics of Acoustics*.

Appendix A

Quaternion Math

In this appendix the basics of quaternions and how they are used in mathematics in relation to rotations are described.

A quaternion, q, can represented as a 4-dimensional vector:

$$q = [w\ x\ y\ z] = w + xi + yj + zk$$

This can also be viewed as a scalar, w, added to a vector $v = [xi\ yj\ zk]$. A quaternion with scalar equal to 0 is called a pure quaternion. For quaternions two binary operations are defined; addition and quaternion multiplication.

A.0.1 Addition and Quaternion Multiplication

Addition is defined as it is for normal vectors and is thus both commutative and associative.

$$q_1 + q_2 = [w_1 + w_2 \quad x_1 + x_2 \quad y_1 + y_2 \quad z_1 + z_2] = q_2 + q_1$$

As multiplication of vectors are not defined, a special operation called quaternion multiplication is defined. The quaternion product of two quaternions, q_1 and q_2, is defined as:

$$q_1 \otimes q_2 = \begin{bmatrix} w_1 & x_1 & y_1 & z_1 \end{bmatrix} \otimes \begin{bmatrix} w_2 & x_2 & y_2 & z_2 \end{bmatrix} \tag{A.1}$$

$$= \begin{bmatrix} w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \\ w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2 \\ w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2 \\ w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2 \end{bmatrix}^T \tag{A.2}$$

This operation can be simplified by using the scalar and vector notation, such that expression becomes:

$$q_1 \otimes q_2 = [w_1w_2 - \vec{v}_1 \cdot \vec{v}_2 \quad w_1\vec{v}_2 + w_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2] \tag{A.3}$$

Quaternion multiplication is distributive however it is not commutative.

A.0.2 Norm, Conjugate and Inverse of a Quaternion

In addition to the binary operations, a few other definitions often prove useful. These are; the norm of a quaternion:

$$\|q\| = \sqrt{w^2 + x^2 + y^2 + z^2} \tag{A.4}$$

the conjugate, which satisfies the property $q \otimes q^* = \|q\|^2$:

$$q^* = \begin{bmatrix} w & -x & -y & -z \end{bmatrix} \tag{A.5}$$

and the inverse, that has the property $q \otimes q^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$:

$$q^{-1} = \frac{q^*}{\|q\|^2} \tag{A.6}$$

A.0.3 Rotating Vectors Using Quaternions

As stated in chapter 4, quaternions can be used to represent rotations, and thus be used to find the orientation of a body in three-dimensional space. Euler's rotation theorem states that any rotation, or set of rotations, of a rigid body about a fixed point, corresponds to a single rotation of θ degrees around a fixed axis that runs through the fixed point [42, p.19]. This axis can be represented as a unit vector which brings us to how quaternions can be used for rotation. The rotation of θ degrees around a unit vector can be represented with a quaternion by using an extension of Euler's formula. Thus, for a unit vector:

$$\hat{\vec{u}} = (u_x, u_y, u_z) = u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k}$$

the quaternion that represents a rotation of θ degrees is defined as:

$$\hat{\vec{q}} = \cos\frac{\theta}{2} + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\theta}{2}$$

Then the way a quaternion is used to rotate a vector is through the quaternion rotation operation.

$$\begin{bmatrix} 0 & \vec{r}' \end{bmatrix} = \hat{\vec{q}} \otimes \begin{bmatrix} 0 & \vec{r} \end{bmatrix} \otimes \hat{\vec{q}}^* \tag{A.7}$$

Where $\vec{r}$ is the vector to be rotated, q is the quaternion representing the rotation and $\vec{r}'$ is the vector rotated by q. [31] As Euler stated, every set of rotations can be reduced to a single rotation around a fixed axis, rotations made by a set of quaternions can be represented by a single quaternion by multiplying them together. The product $q_2 \otimes q_1$ thus corresponds to first rotating using q_1 and then q_2.

Alternatively, equation (A.7) can be expressed as:

$$\hat{\vec{q}}^* \otimes \begin{bmatrix} 0 & \vec{r}' \end{bmatrix} \otimes \hat{\vec{q}} = \begin{bmatrix} 0 & \vec{r} \end{bmatrix} \tag{A.8}$$

Appendix B

Choice of Method for Determining Step Length in Optimization Algorithm

As discussed in section 4.2.3, five different algorithms have been tested to identify the best one at minimizing the expression:

$$\min_{\vec{q}\in\mathbb{R}^4} \left|\left| F_\nabla \left(\hat{\vec{q}}, \hat{\vec{g}}, \hat{\vec{a}}, \hat{\vec{b}}, \hat{\vec{m}} \right) \right|\right|^2 \tag{B.1}$$

In section B.1 there is a description of each of the algorithm tested. Proceeding this is a description of the experiment set up, execution, data processing, and conclusion.

B.1 Optimization Algorithms

Following is a description of the tested algorithms. The algorithms described from section B.1.1 to B.1.3 solely deal with finding a step size for the steepest descent method. A description of the steepest descent method can be found in section 4.2.3. Section B.1.4 describes an alternative method to calculate the step direction using a second order approximation. A description of what turned out to be the best performing algorithm, using gradient descent and finding the step size using a second order approximation, is omitted from the following sections as it is described in detail in section 4.2.3.

B.1.1 Finding the Step Size Using a Second Order Approximation without the Hessian

If the hessian function is computationally difficult to calculate, it may prove advantageous to instead use an estimate of it. This estimate can be shown to change the expression for α_k to [33, p. 125]:

$$\alpha_k = \frac{\alpha_{k-1}^2 \|\vec{d}_k\|^2}{2(f(\vec{x}_k + \alpha_{k-1}\vec{d}_k) - f(\vec{x}_k) + \alpha_{k-1}\|\vec{d}_k\|^2)} \tag{B.2}$$

$$= \frac{\alpha_{k-1}^2 \|\nabla f(\vec{x}_k)\|^2}{2(f(\vec{x}_k + \alpha_{k-1}\nabla f(\vec{x}_k)) - f(\vec{x}_k) + \alpha_{k-1}\|\nabla f(\vec{x}_k)\|^2)} \tag{B.3}$$

B.1.2 Assign the Step Size a Fixed Value

A much simpler approach to selecting a step size is to merely use a constant fix size for all iterations. While this approach will require more iterations to converge, each iterations will require fewer computations to calculate, as only the direction (i.e. the negative gradient) needs to be found. Care should be taken when selecting a value for the step size, as too small a value will result in slow convergence, while too large a value may result in overshoot. These phenomena are shown in figure B.1 and B.2.

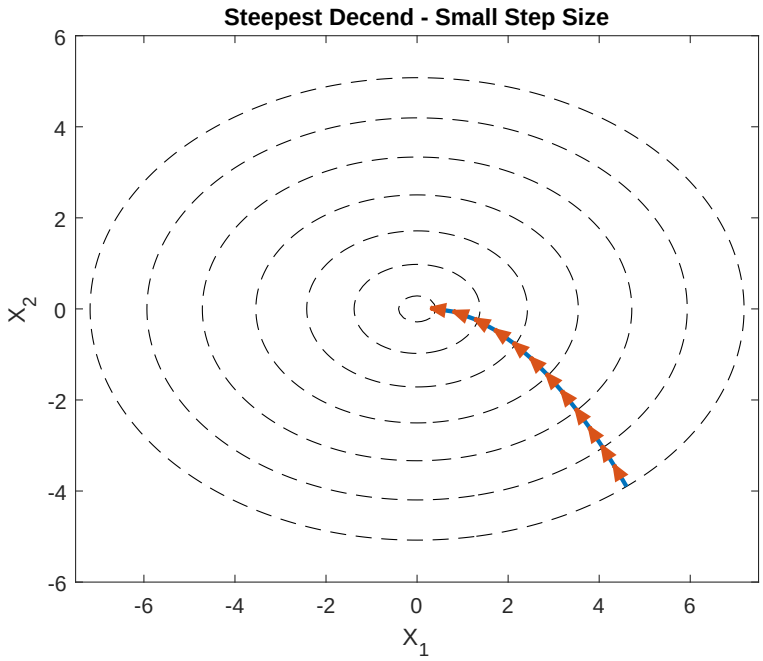


Figure B.1: Illustration of slow convergence when using a small fixed step size in a gradient descend algorithm.

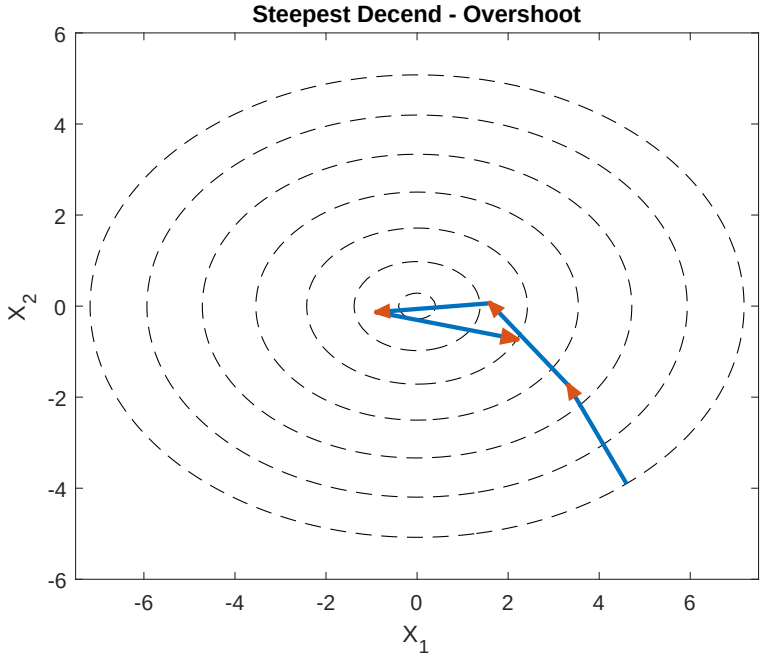


Figure B.2: Illustration of significant overshoot when using a large fixed step size in a gradient descend algorithm.

B.1.3 Finding the Step Size Using a Line Search

In line search the direction is iterative searched for the ideal step size. Specifically, backtracking line search has been chosen as the line search algorithm of choice. In this scheme a large step size is initially chosen, and this is interactively reduced by a factor of β until a decrease in the function output of at least factor α is observed. Further mathematical discussion of the approach is outside the scope of this report, but more details can be found in [43, p. 464].

It should be noted, that compared to the previously discussed methods for finding the step size, line search is an iterative procedure. Thus, the computational time to perform it is nondeterministic. This is generally a disadvantage, as it must be guaranteed that all orientation calculations can be performed between each sensor sample. Thus, if this method is used, and upper bound most be found for the required number of operations.

B.1.4 A Second Order Approximation of the Direction (Gauss Newton)

The Gauss Newton method is not a gradient descend method, but rather a variation of the Newton Raphson method. Unlike the gradient decent method, the Gauss Newton method is a second order approximation which uses both the gradient and the hessian to find the direction.

The expression for $\vec{d}$ is found using a similar method to that of the second order approximation of α in section 4.2.3. Starting with equation 4.2.3, the optimal direction can be found by differentiating with respect to $\vec{h}$, setting the result equal to zero, and then finally solving for $\vec{h}$ [33, p. 129]:

$$\frac{\mathrm{d}}{\mathrm{d}\vec{h}} f(\vec{x}_k + \vec{h})|_{\vec{x}_k} \approx \nabla f(\vec{x}_k) + \boldsymbol{H}(\vec{x}_k)\vec{h} = 0 \tag{B.4}$$

$$\vec{h} = -\boldsymbol{H}(\vec{x}_k)^{-1}\nabla f(\vec{x}_k) \tag{B.5}$$

Thus $\vec{d}$ is:

$$\vec{d} = -\boldsymbol{H}(\vec{x}_k)^{-1}\nabla f(\vec{x}_k) \tag{B.6}$$

Note that this definition requires that the hessian is invertible. If this is not the case, the hessian must be modified to become invertible [33, p. 131]. The methods and algorithms used to perform this operation is outside the scope of this report, however more information can be found in [33, p. 138].

To find the optimal step size for this algorithm it is chosen to use line search. Like in gradient descent, other step size methods exist. However, to limit the scope only line search is explored. This method is chosen as it appears to be the most popular method in the cited literature [33][43].

B.2 Experiment Setup

A rectangular box was created, and an MPU9250 9-DOF breakout board was placed in its center. The MPU9250 samples the accelerometer and gyroscope with 500 Hz, and the magnetometer with 100 Hz (in practice this meant that the magnetometer value was send five times to match the sampling rate of the other sensors). This data is relayed to a PSOC 5LP microcontroller which in turn relays the data to an external computer through a USB UART channel. This computer logs all received samples. An image of the box is shown in figure B.3.

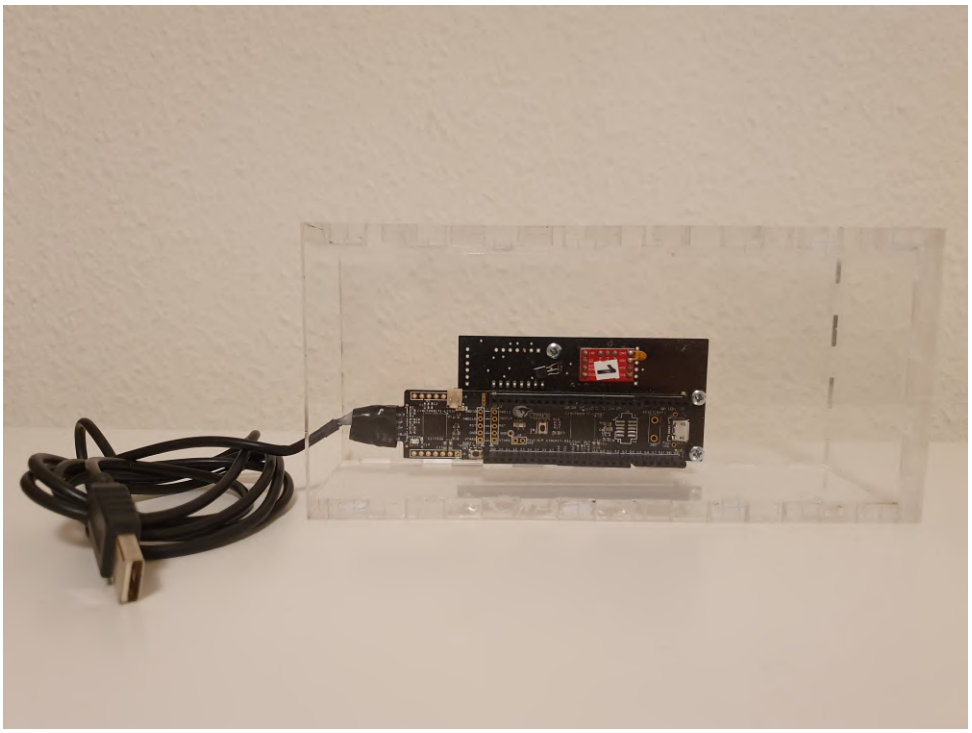


Figure B.3: Image of the acrylic box containing the MPU9250 sensor board and PSOC microcontroller.

B.3 Experiment Execution

To obtain data regarding a wide variety of orientations and motions, the box was rotated around each individual axis in four 90-degree intervals, for a total of 360 degrees. Each 90-degree rotation was followed by approximately five seconds of being stationary, before the next rotation was performed. This was done to gain insight into the steady state behavior of each algorithm. Each rotation test was performed for clockwise and anticlockwise rotations. Thus, in total, six different tests were performed, each of them is listed below:

- Clockwise rotation around the X Axis
- Clockwise rotation around the Y Axis
- Clockwise rotation around the Z Axis
- Counterclockwise rotation around the X Axis
- Counterclockwise rotation around the Y Axis
- Counterclockwise rotation around the Z Axis

B.4 Data Processing

Each of the six collected data sets was fed into a Matlab implementation of each of the six algorithms.

For the algorithm requiring the selection of parameters, such as α and β in line search, their values were determined empirically, by running the algorithm multiple times, and slightly altering the parameters each run, until optimal values were found.

For each algorithm each calculated quaternion was inserted into the objective function to find an error term quantifying how much the output deviated from the desired goal of zero. After calculating all the quaternions for a given data set the mean and median of the error terms was calculated. As all of the selected algorithms should converge to the precise solution given enough time, each of the algorithm were artificially limited to run a finite number of iterations. This makes the performance comparison more realistic, as the algorithm implemented on the DSP will have a finite amount of clock cycles available to estimate the orientation. It should be noted that since all calculations in Matlab are performed on an x86 processor using floating point, the computational time required to run each algorithm in Matlab does not necessarily correspond to the performance of a fixed-point implementation on the C5535 DSP. However, the algorithmic complexity between the two implementations are the same, thus it is assumed that some kind of relationship exists.

A list of tables quantifying the performance of each algorithm for each data set can be found in section B.6. Each of these tables list the computational run time, as well as the mean and median error of the instantaneous orientation produced by each respective algorithm.

Table B.1 condense the information shown in the tables in section B.6. Here the total run time for all data sets is displayed. Additionally, the averages of the mean and median errors displayed in the section B.6 tables are also listed.

Table B.1: Summary of all data presented in section B.6. Lists the total run time of each algorithm over all data sets. The averages of the mean and median errors computed by each algorithm on each data set are also shown.

Summarized Results			
Algorithm	*Total Run Time*	*Mean of Mean Errors*	*Mean of Median Errors*
Steepest Decent, Hess	6.119 s	3.255E-04	5.796E-06
Steepest Decent, Hess Approx	7.681 s	1.290E-03	2.891E-05
Steepest Decent, Fixed Step	7.306 s	6.469E-03	1.139E-03
Steepest Decent, Line Search	6.475 s	4.815E-04	3.909E-05
Steepest Decent, Lagrangian	9.479 s	1.425E-01	1.518E-01
Gauss Newton Method	14.97 s	5.767E-02	2.247E-02

B.5 Conclusion and Justification of Algorithm Choice

According to the data presented in table B.1, the steepest decent algorithm using the Hessian to identify the step size is unambiguously the best algorithm. This algorithm was the fastest to compute and produced the most accurate results. Another well performing algorithm, was the Steepest Descend algorithm using line search to identify the step size. It is likely that if implemented in a limited precision fixed point algorithm, that the differences in accuracy between these two algorithm would be even less profound. However, due to the non deterministic nature of performing line search it may prove troublesome to implement in a real time system. Thus the steepest decent algorithm using the hessian will be implemented.

B.6 Data

This section lists the results of each algorithm, for each data set.

Table B.2: X Axis Counterclockwise Algorithms.

X Axis Counterclockwise			
Algorithm	*Time To Complete (s)*	*Mean Error*	*Median Error*
Steepest Decent - Hessian	0.8759	7.907E-04	5.783E-06
Steepest Decent - Hessian Approx.	1.0748	1.098E-03	1.945E-05
Steepest Decent - Fixed Step Size	1.0353	8.160E-03	1.331E-03
Steepest Decent - Line Search	0.9106	1.168E-03	3.833E-05
Steepest Decent - Lagrangian	1.2963	1.368E-01	1.513E-01
Gauss Newton	1.2277	1.816E-03	5.394E-06

Table B.3: X Axis Clockwise Algorithms.

X Axis Clockwise			
Algorithm	*Time To Complete (s)*	*Mean Error*	*Median Error*
Steepest Decent - Hessian	0.9896	3.664E-04	4.668E-06
Steepest Decent - Hessian Approx.	1.1984	1.485E-03	1.892E-05
Steepest Decent - Fixed Step Size	1.1877	8.933E-03	9.480E-04
Steepest Decent - Line Search	1.0281	4.684E-04	3.542E-05
Steepest Decent - Lagrangian	1.4657	1.322E-01	1.508E-01
Gauss Newton	1.3706	8.279E-04	4.511E-06

Table B.4: Y Axis Counterclockwise.

Y Axis Counterclockwise			
Algorithm	*Time To Complete (s)*	*Mean Error*	*Median Error*
Steepest Decent - Hessian	1.0027	3.502E-04	1.102E-05
Steepest Decent - Hessian Approx.	1.3712	1.563E-03	4.517E-05
Steepest Decent - Fixed Step Size	1.2366	7.633E-03	1.639E-03
Steepest Decent - Line Search	1.1198	4.959E-04	4.658E-05
Steepest Decent - Lagrangian	1.5486	1.374E-01	1.510E-01
Gauss Newton	1.9782	6.544E-03	5.811E-06

Table B.5: Y Axis Clockwise.

Y Axis Clockwise			
Algorithm	*Time To Complete (s)*	*Mean Error*	*Median Error*
Steepest Decent - Hessian	1.0751	3.811E-04	9.985E-06
Steepest Decent - Hessian Approx.	1.3837	3.336E-03	4.238E-05
Steepest Decent - Fixed Step Size	1.2611	6.038E-03	1.096E-03
Steepest Decent - Line Search	1.1142	4.464E-04	4.280E-05
Steepest Decent - Lagrangian	1.5726	1.398E-01	1.504E-01
Gauss Newton	2.6133	7.023E-02	7.509E-06

Table B.6: Z Axis Counterclockwise.

Z Axis Counterclockwise			
Algorithm	*Time To Complete (s)*	*Mean Error*	*Median Error*
Steepest Decent - Hessian	1.1691	2.919E-05	1.445E-06
Steepest Decent - Hessian Approx.	1.4371	1.251E-04	2.133E-05
Steepest Decent - Fixed Step Size	1.3944	3.758E-03	9.573E-04
Steepest Decent - Line Search	1.2057	1.424E-04	3.062E-05
Steepest Decent - Lagrangian	1.8685	1.539E-01	1.532E-01
Gauss Newton	3.2823	6.577E-02	4.323E-06

Table B.7: Z Axis Clockwise.

Z Axis Clockwise			
Algorithm	*Time To Complete (s)*	*Mean Error*	*Median Error*
Steepest Decent - Hessian	1.0065	3.557E-05	1.877E-06
Steepest Decent - Hessian Approx.	1.2153	1.325E-04	2.622E-05
Steepest Decent - Fixed Step Size	1.1907	4.290E-03	8.636E-04
Steepest Decent - Line Search	1.0965	1.679E-04	4.078E-05
Steepest Decent - Lagrangian	1.7273	1.549E-01	1.540E-01
Gauss Newton	4.4933	2.008E-01	1.348E-01

Appendix C

Sensor Calibration

C.1 Introduction

Sensors are not ideal devices. Imperfections, such as offset errors, nonlinear properties, lack of orthogonality between axis, and many more, can among others be attributed to limits in manufacturing capabilities and stress exerted on the sensors during assembly. To be able to reliably and accurately use sensors for their intended purpose, they must be calibrated to counteracts these defects.

As it is possible to model a number of these defects, for practical reasons only significant causes of error are modeled. One example of calibrations which are omitted in the system at hand is accuracy verification over different temperatures.

C.2 Accelerometer and Gyroscope Calibration

For the three axis Accelerometer and Gyroscope, only offset errors are accounted for. This is assumed to be sufficient. However, to perform a complete calibration, further steps would be required [28, p. 59].

To identify the magnitude of these offsets, the sensors are placed at rest, and the mean of a large number of samples is found. Note that for the accelerometer it is important that it is placed on a level surface, with one of the axis being aligned with the gravitational field, for the duration of the measurements. For this axis, the magnitude of the gravitational field must be subtracted from the calculated average. To calibrate the sensor and remove this offset, the offset is simply subtracted from the sensor measurements:

$$\begin{bmatrix} X_{\text{Calibrated}} \\ Y_{\text{Calibrated}} \\ Z_{\text{Calibrated}} \end{bmatrix} = \begin{bmatrix} X_{\text{Not Calibrated}} \\ Y_{\text{Not Calibrated}} \\ Z_{\text{Not Calibrated}} \end{bmatrix} - \begin{bmatrix} \text{Bias}_X \\ \text{Bias}_Y \\ \text{Bias}_Z \end{bmatrix} \tag{C.1}$$

C.3 Magnetometer Calibration

Magnetometer calibration is a bit more involved than that of the gyroscope and accelerometer. Like with the gyroscope and accelerometer, it is desirable to remove offset errors. However, for the magnetometer it also desirable to alleviate other sources of errors. These are soft iron errors caused by ferromagnetic components on the sensor PCB, differing gain in the three axis, and imperfect orthogonality among the axis [44, p. 3].

Ideally, a magnetometer moved through all possible orientations in space would result in a spherical cloud of measurements when plotted in a 3-dimensional-xyz-plot.

However, due to the above-mentioned errors the sphere is warped into an ellipsoid. It is desirable to transform this into a sphere centered at the origin. To perform this, the ellipsoid must first undergo

a translation moving it to the origin. Then it must be transformed into a sphere using a 3×3 transformation matrix, $\boldsymbol{A}^{-1}$. This results in the following expression for the calibrated X Y and Z values [44, p. 4]:

$$\begin{bmatrix} X_{\text{Calibrated}} \\ Y_{\text{Calibrated}} \\ Z_{\text{Calibrated}} \end{bmatrix} = \boldsymbol{A}^{-1} \left(\begin{bmatrix} X_{\text{Not Calibrated}} \\ Y_{\text{Not Calibrated}} \\ Z_{\text{Not Calibrated}} \end{bmatrix} - \begin{bmatrix} \text{Bias}_X \\ \text{Bias}_Y \\ \text{Bias}_Z \end{bmatrix} \right) \tag{C.2}$$

To find transformation matrix $\boldsymbol{A}^{-1}$, the following intuition is provided:

A unit sphere centered at the origin can be described as:

$$x_1^2 + x_2^2 + x_3^2 = 1 \tag{C.3}$$

or equivalently in vector form:

$$\vec{x}^T \vec{x} = 1 \tag{C.4}$$

Similarly, an ellipsoid is described as:

$$ax_1^2 + bx_2^2 + cx_3^2 + dx_1x_2 + ex_1x_3 + fx_2x_3 + gx_1 + hx_2 + ix_3 = 1 \tag{C.5}$$

or equivalently in vector form:

$$\vec{x}^T \boldsymbol{A} \vec{x} = 1 \tag{C.6}$$

Where $\boldsymbol{A}$ is a 3×3 transformation matrix. To transform an ellipsoid to a unit sphere, the inverse of $\boldsymbol{A}$ must be found. This can then be multiplied onto any given point of the ellipsoid, to transform it into its corresponding coordinate on the unit sphere:

$$\boldsymbol{A}^{-1} \vec{x} \tag{C.7}$$

$\boldsymbol{A}$ is found by storing magnetometer samples measured in a wide range of different orientations. An ellipsoid can then be fitted to the measure points using linear least squares. Mathematical details of least square fitting are outside the scope of this report; however, the important takeaway is that this operation will return an approximate of $\boldsymbol{A}$, which can then be used to calculate $\boldsymbol{A}^{-1}$. This fitting can also be used to find the center of the ellipsoid, which can be used to find the Bias vector from equation (C.2).

Appendix D

Test journal: Stability/Drift in Orientation Tracker

D.1 Introduction

As stated in the technical requirements in chapter 2, the system must determine the orientation of the user with an error of maximum 11°. To determine if this requirement is upheld, the stability/drift of the orientation tracking is tested.

D.2 Test frame

Theoretical background

The measured orientation is dependent of integrated measurements from a gyroscope, which are known to drift over time. The orientation tracking algorithm uses a combination of MARG sensors to handle this drift, however if the cutoff frequency in the complementary filter is not placed correctly, some drift might still occur. To determine the amount of drift, if any, the system is kept stationary and the orientation measured regularly over a long period of time. From the use case of the system, it is assumed that during use the system is not kept stationary for more than 1 hour and thus the test will run for that period. As the drift from the gyroscope should be present on all three axes at once, the test will not be repeated with different orientations.

Test setup and test procedure

For this test the setup consists of the DSP-board together with the MARG sensor array and a computer. The system is placed on a table and connected to the computer via USB-cable. On the computer the Python script "—" is running in order to log the calculated quaternion from the system. The setup is then left stationary for 1 hour, sampling with 500 Hz and logging with 100 Hz. The resolution for the MARG sensors is set to $\pm 2\,g$ for the accelerometer, $\pm 1000\,°/s$ for the gyroscope and $4800\,\mu T$. For each sample the quaternion is calculated through 6 iterations of the optimization algorithm.

D.3 Test results and data processing

A total of 360000 quaternions were logged during the test and the values for each element in the quaternions can be seen in figure D.1.

To determine the drift of the orientation over time, the orientation of a reference vector rotated by the first quaternion can be compared to the orientation of the same vector rotated by the last quaternion. To reduce the impact of noise, the mean of the first 1000 samples is used as the first quaternion and the mean of the 1000 last samples are used for the last quaternion. The arc difference between the two resulting vectors are $25 \cdot 10^{-3}$ degrees. This result, together with the figure of the

quaternion elements, leads to the conclusion that there is no drift in the calculated orientation. The arc difference is determined to be so small, that it is most likely due to noise in the system rather than drift.

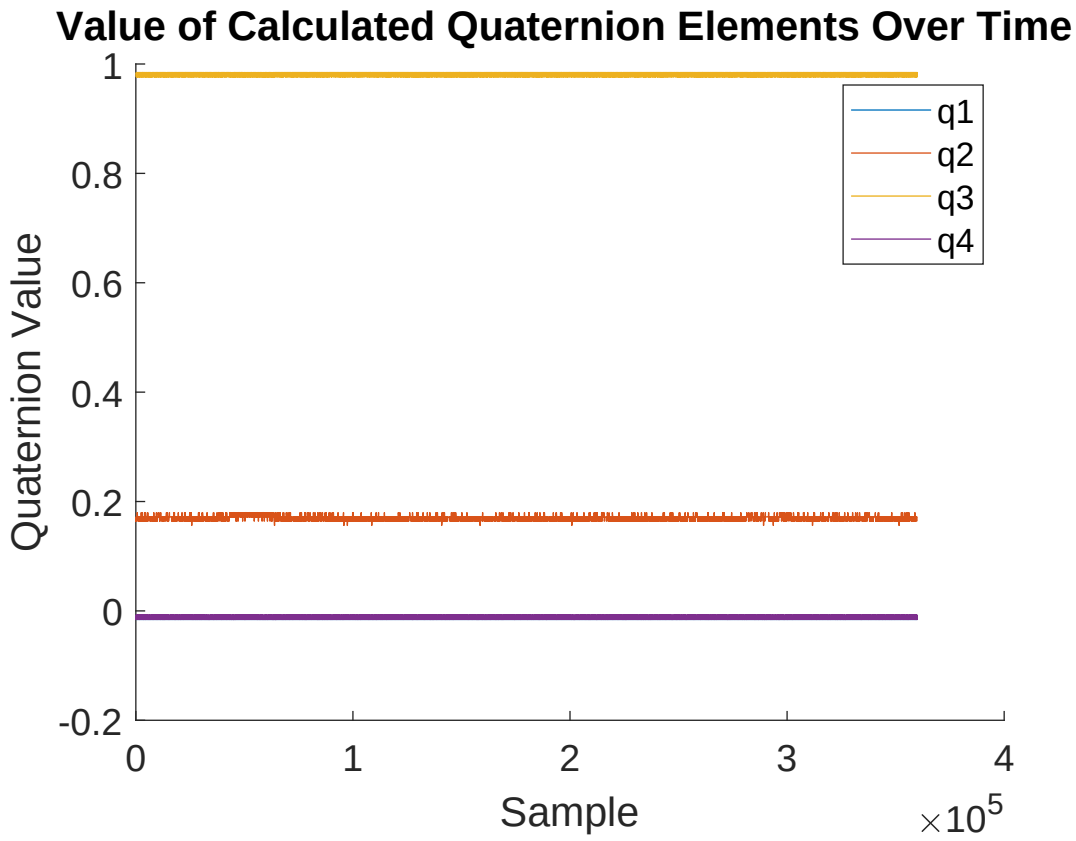


Figure D.1: Drift as measured over one hour. Note that q1 is barely visible as it is placed similarly to q4.

D.4 Sources of error and other uncertainties

The arc difference between the mean of the first thousand orientations and the last thousand orientations could have been influenced by unforeseen disturbances in the sensor.

Appendix E

Test Journal: Noise in Orientation Tracker

E.1 Introduction

As stated in the technical requirements in chapter 2, the system must determine the orientation of the user with a resolution of 1°. To determine if this requirement is upheld, the noise of the orientation tracking is tested.

E.2 Test frame

Theoretical background

As the measured orientation from the orientation tracker depends on measurements from the MARG sensor array, it is assumed that the measurement has some amount of noise. This noise results in some fluctuations in the measured orientation, even when the system is stationary. As it is not possible to measure reliably inside these fluctuations, the resolution of the orientation tracker directly depends on them. To measure the noise, measurements of orientations are performed while the system is kept stationary. To ensure that the measurements are independent of the orientation, the test will be repeated at various different orientations.

Test setup and test procedure

For this test, the system is mounted inside a rectangular box and placed on a table. The system is then powered on and given 1 minute to ensure that the quaternion has converged from its initial guess to the measured orientation. Then the calculated quaternion is logged on a computer for 3 minutes. During the test, the calculated quaternion together with the gyroscope measurement is sampled with 500 Hz and logged with 100 Hz on a computer. The resolution for the MARG sensors is set to $\pm 2\,g$ for the accelerometer, $\pm 1000\,°/s$ for the gyroscope and $4800\,\mu T$. For each sample the quaternion is calculated through 6 iterations of the optimization algorithm. This procedure is then repeated twice to get measurements for rotations around all three axes of the system. The setup for these tests is made by flipping the box on its side thus rotating it 90°.

E.3 Test results and data processing

A total of 18000 quaternions were logged during each of the three tests. Each quaternion is applied as a rotation to a reference orientation and the arc-angle deviation between each of the calculated orientations and the mode orientation is calculated. The deviations can be seen in figures E.1, E.3 and E.5. To see the distribution of the deviations, a histogram is made. These can be seen in figures E.2, E.4 and E.6.

The tests showed that the highest amount of deviation occurred when the y-axis of the accelerometer was parallel with the gravitational vector. However, the deviations occurred more often when the x-axis was parallel with gravity, compared with the rest. From both the test with the x-axis parallel with gravity and the test with the y-axis parallel with gravity a few samples showed a deviation above 1.6°. However, the majority of the deviations were between 1.4° and 1.618°.

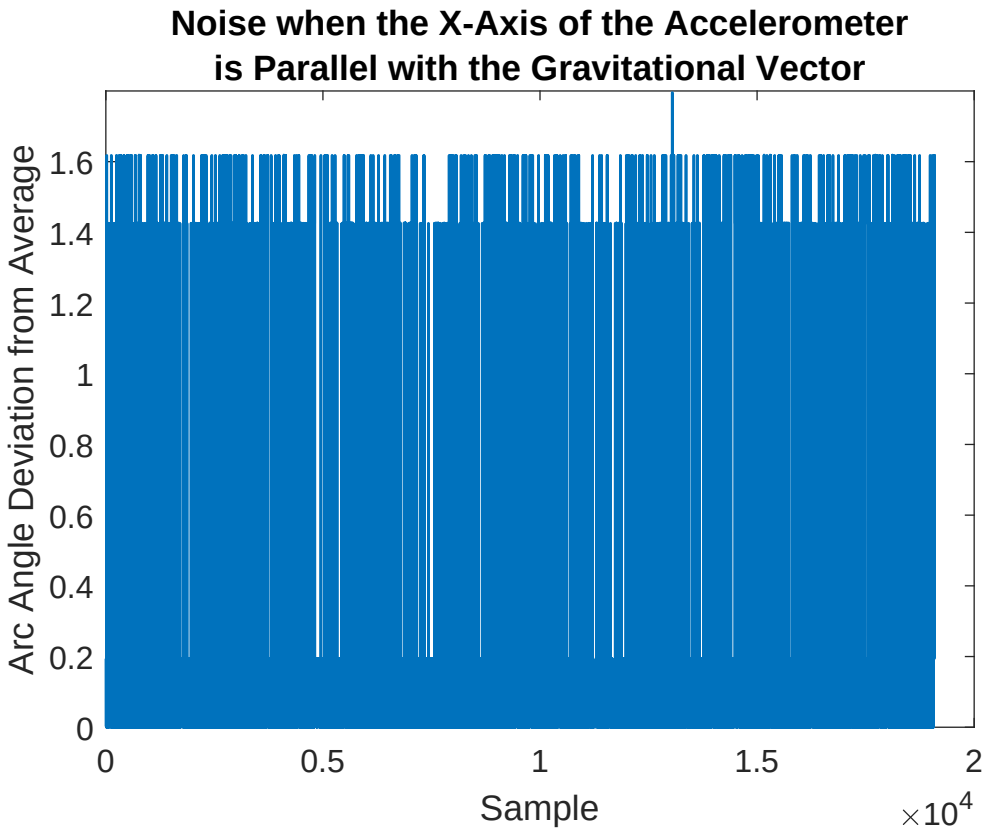


Figure E.1

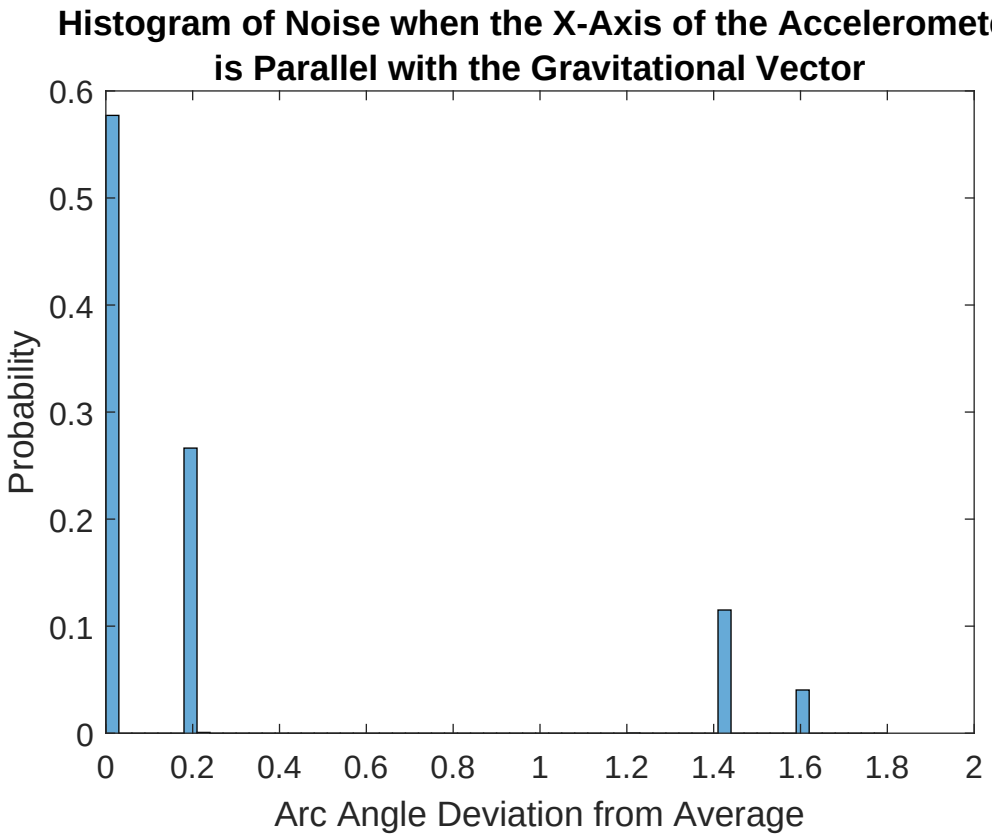


Figure E.2

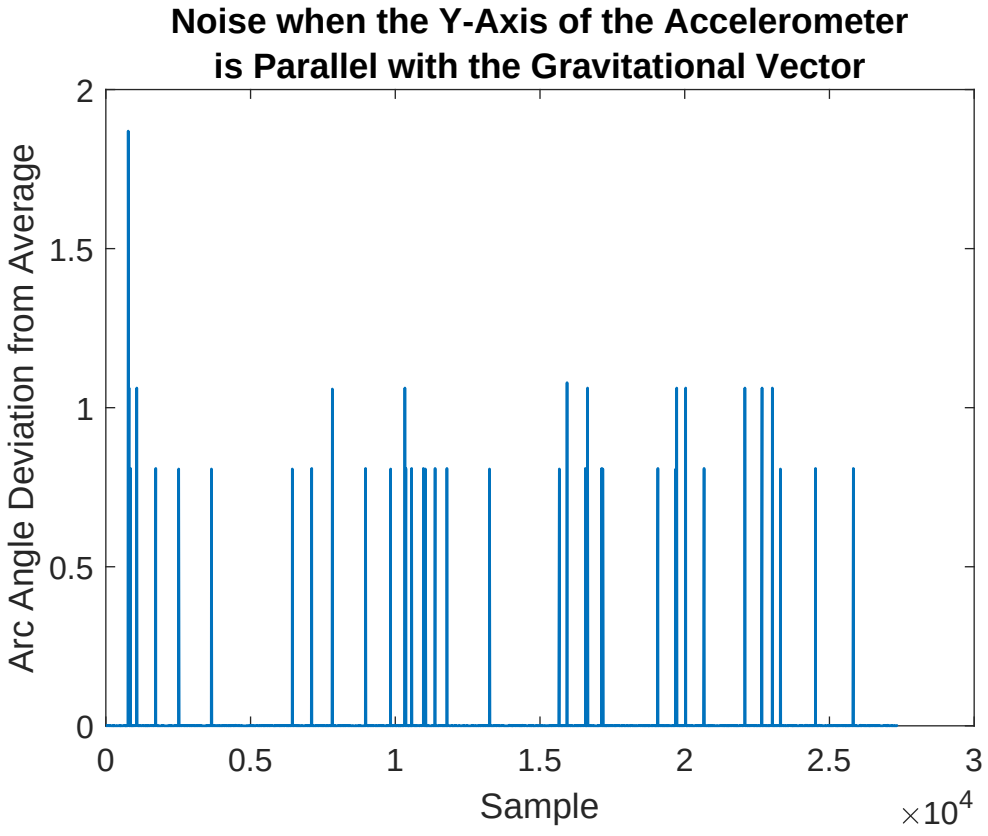


Figure E.3

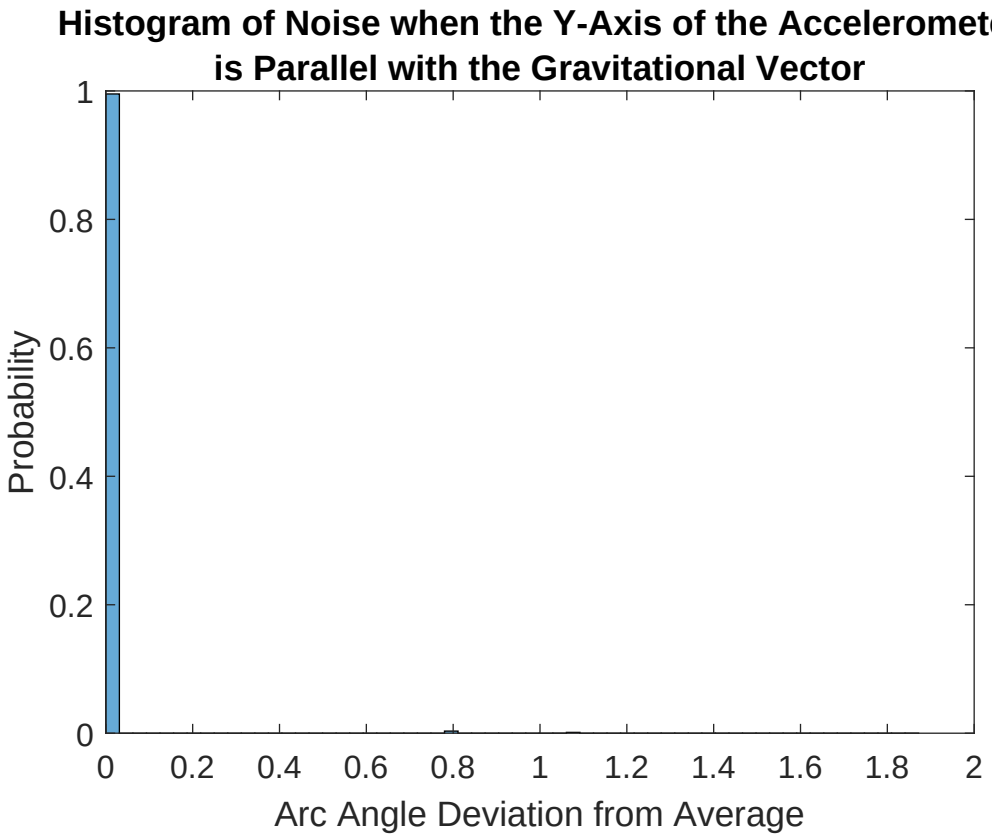


Figure E.4

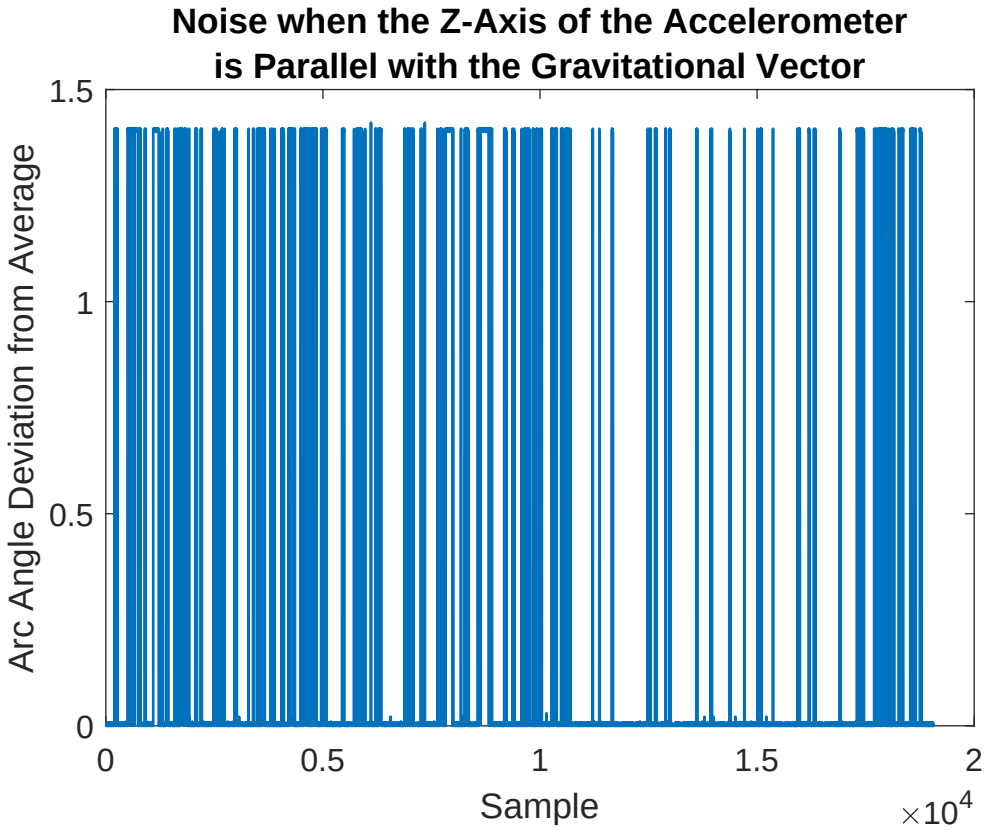


Figure E.5

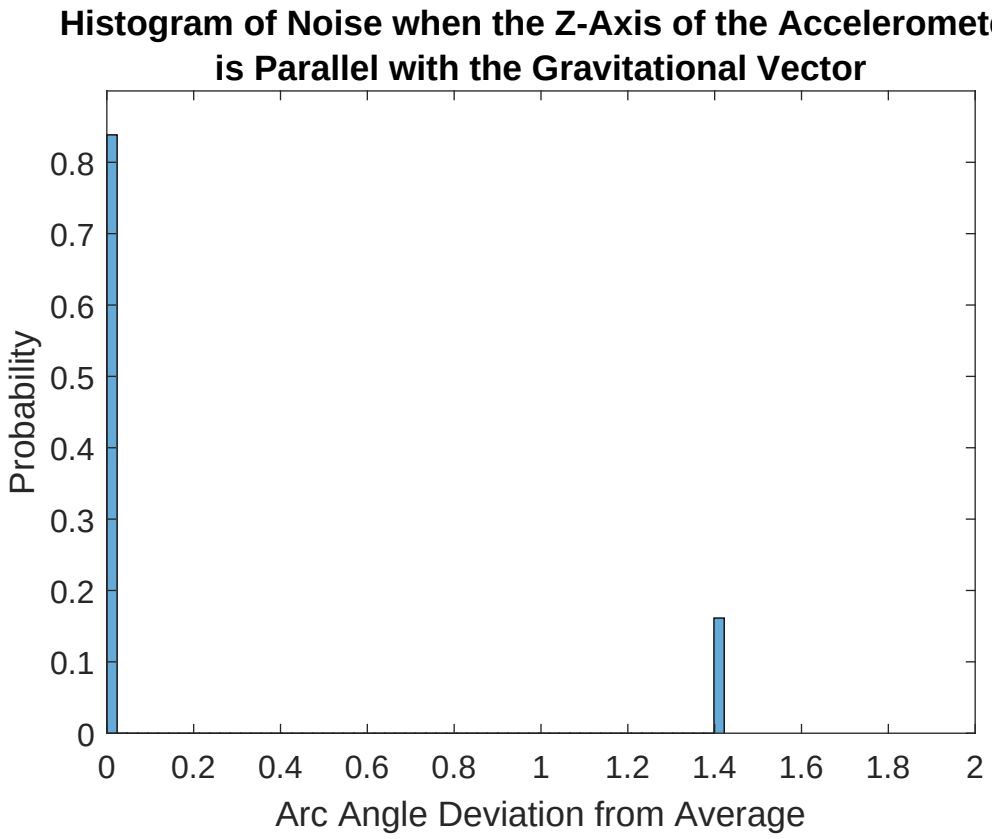


Figure E.6

E.4 Sources of error and other uncertainties

Movement around the test setup by people conducting the test might have made the system move unintentionally.

Appendix F

Test Journal: Accuracy of Orientation Tracker

F.1 Introduction

As stated in the technical requirements in chapter 2, the system must determine the orientation of the user with an error of maximum 11°. To determine if this requirement is upheld, the accuracy of the orientation tracking is tested.

F.2 Test frame

Theoretical background

When the listener moves their head, the orientation tracker should be able to measure the new orientation of the head. However, it is assumed that there will be a small error between the measured orientation and the true orientation due to noise in the sensors and the limited precision on the DSP. This error might be different depending on the direction of the rotation and thus the test should be performed with separate rotation axes.

Test setup and test procedure

For this test the setup consists of the DSP-board together with the MARG sensor array, a computer and a servo motor connected to an Arduino Uno. The servo motor is placed in a vise, powered with 6 V DC, and the system is bolted to the servo motor. On the computer the Python script "—" is running in order to log the measured orientation from the system.

The test is performed by setting the servo to rotate 90° at 300 °/s and meanwhile logging the orientation. During the test, the calculated quaternion together with the gyroscope measurement is sampled with 500 Hz and logged with 100 Hz on a computer. The resolution for the MARG sensors is set to ±2 g for the accelerometer, ±1000 °/s for the gyroscope and 4800 µT. For each sample the quaternion is calculated through 6 iterations of the optimization algorithm. This procedure is then repeated with rotations for each of the three axes of the system.

On figure F.1 one of the test setups can be seen.

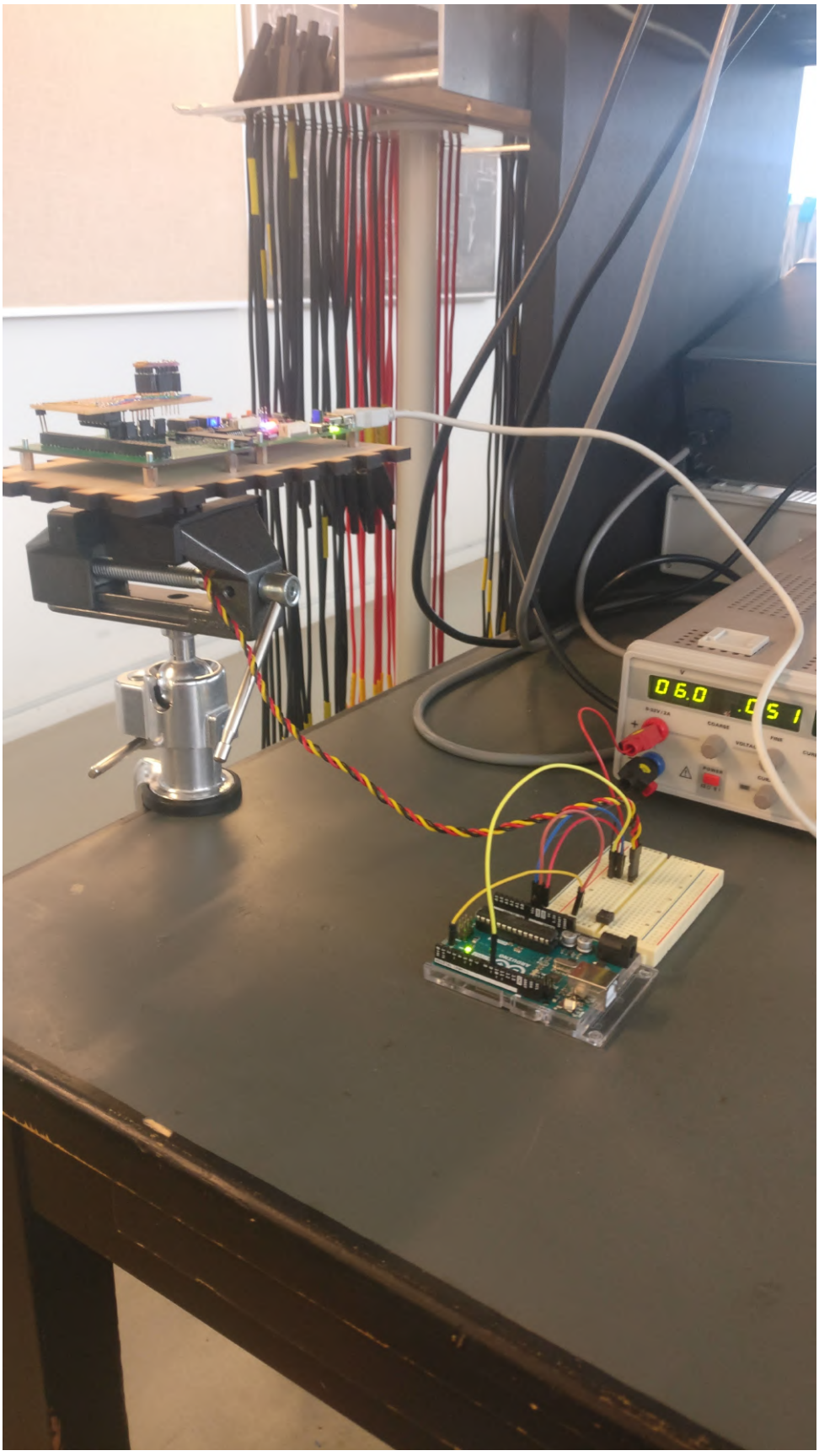


Figure F.1: Picture of one of the test setups used for this test.

F.3 Test results and data processing

A total of 500 quaternions were logged during each test. The measured orientation of the system before and after the rotation can be seen plotted as vectors in figures F.6, F.4 and F.2. To better visualize the change in angle measured by the system, figures F.7, F.5 and F.3 shows the projection of the vectors onto the plane perpendicular to the axis of rotation.

From the figures in shows that the orientation tracker does not track the orientation very accurately. A rotation that should be exclusively around one axis is measured as consisting of rotations of other axes as well and the measured angles does not correspond to the expected angles. An exception seems to be rotation around the z-axis, as this measured a rotation of 92.13°.

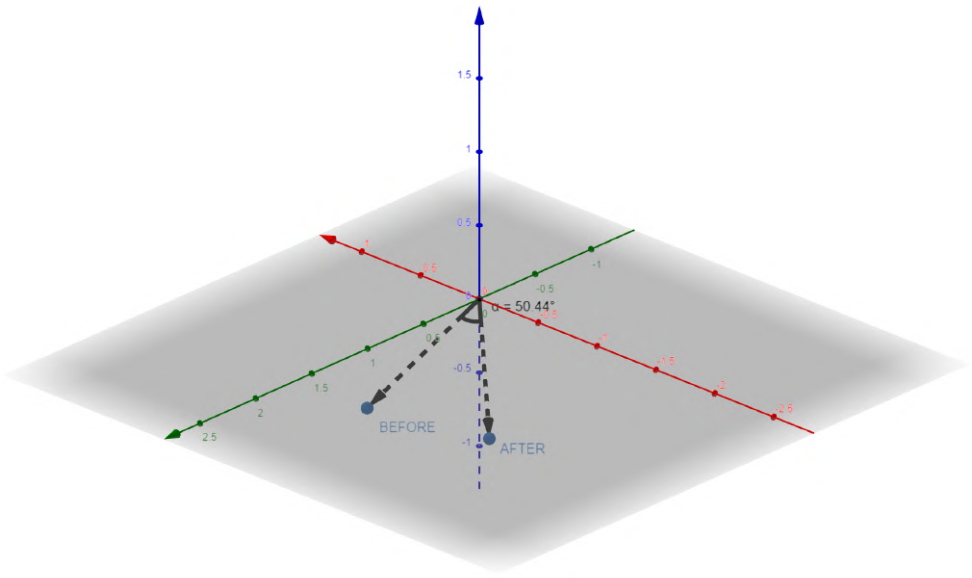


Figure F.2: 3D-view of the measured orientation before and after the rotation around the x-axis. The orientations are represented as vectors.

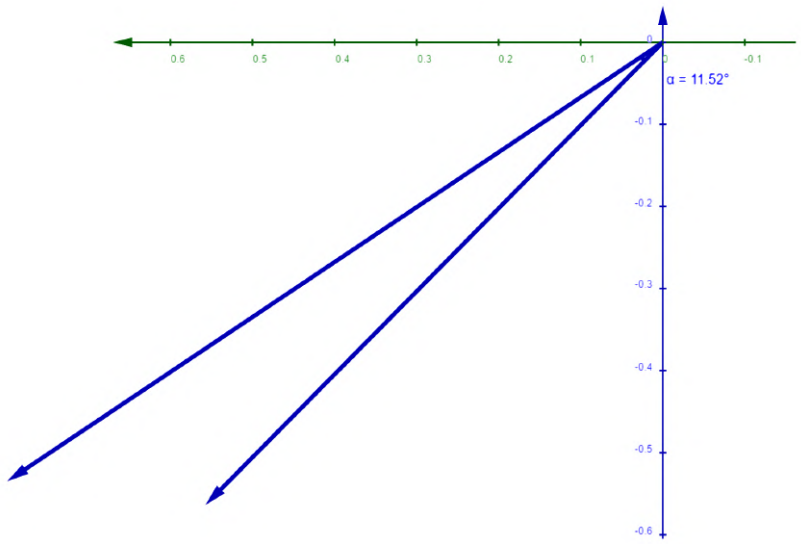


Figure F.3: 2D-view of projections onto the YZ-plane of the measured orientation before and after the rotation around the x-axis.

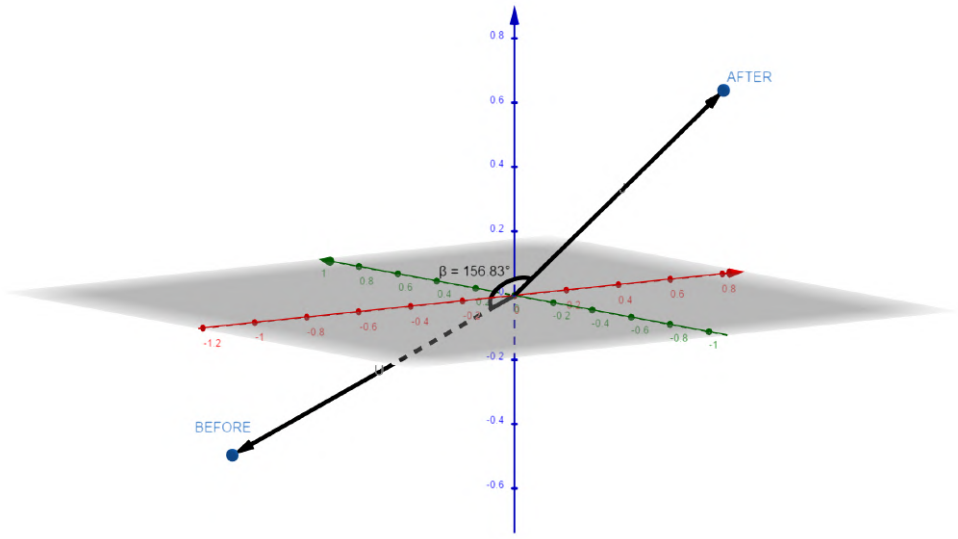


Figure F.4: 3D-view of the measured orientation before and after the rotation around the y-axis. The orientations are represented as vectors.

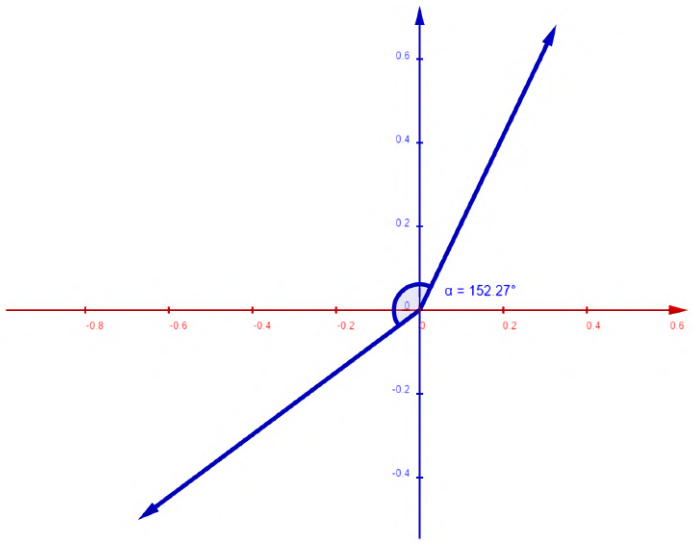


Figure F.5: 2D-view of projections onto the XZ-plane of the measured orientation before and after the rotation around the y-axis.

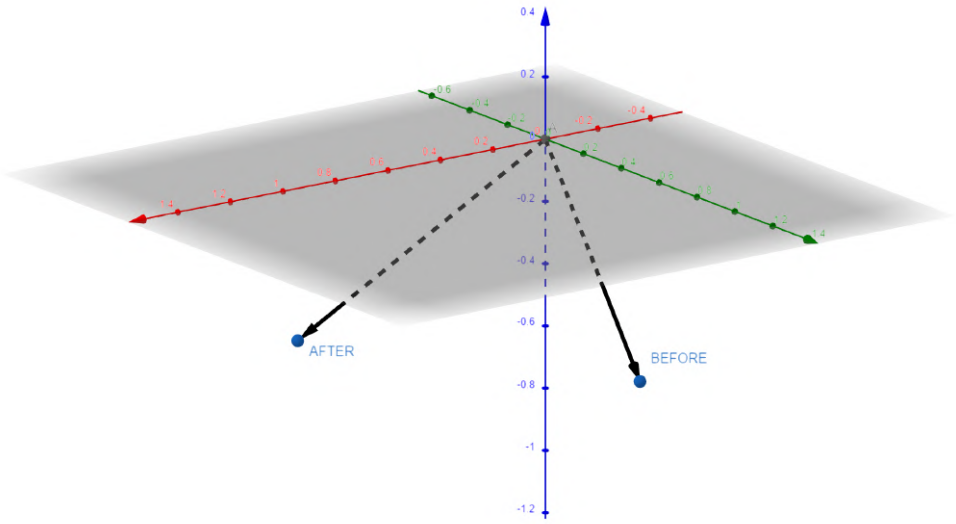


Figure F.6: 3D-view of the measured orientation before and after the rotation around the z-axis. The orientations are represented as vectors.

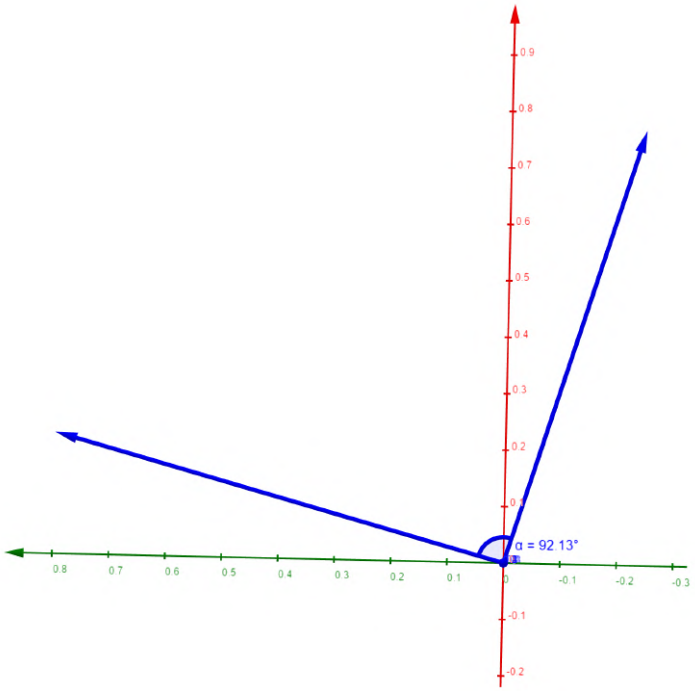


Figure F.7: 2D-view of projections onto the XY-plane of the measured orientation before and after the rotation around the z-axis.

F.4 Sources of error and other uncertainties

Appendix G

Test Journal: Convergence Rate of Orientation Tracker

G.1 Introduction

As stated in the technical requirements in chapter 2, the system can have a delay of maximum 26 ms between head movement and orientation update. To determine if this requirement is upheld, the convergence rate of the orientation tracking is tested.

G.2 Test frame

Theoretical background

When the user moves the head the orientation tracker updates the instantaneous orientation in real time. As the orientation is calculated through an iterative process, it is important that the process converges quickly or a delay between head movement and sound placement is perceived by the listener. As the head movement gets faster, the more detectable the delay gets, thus the delay has to be small even for quick head movements. It is hard to find definitive results for the maximum head movement speed for humans, but one study measured movement speeds upwards of 300 °/s for vertical movement [45]. However, another study measured maximum speeds of upwards approximately 700 °/s for horizontal movements, although for movements of 60° or less the speeds were approximately 300 °/s [46]. Thus, the test is performed at 300 °/s.

Test setup, test procedure and equipment

For this test the setup consists of the DSP-board together with the MARG sensor array, a computer and a servo motor connected to an Arduino Uno. The servo motor is placed in a vise, powered with 6 V DC, and the system is bolted to the servo motor. On the computer the Python script "—" is running in order to log the measured orientation from the system.

The test is performed by setting the servo to rotate 90° at 300 °/s and meanwhile logging the orientation. During the test, the calculated quaternion together with the gyroscope measurement is sampled with 500 Hz and logged with 100 Hz on a computer. The resolution for the MARG sensors is set to ±2 g for the accelerometer, ±1000 °/s for the gyroscope and 4800 µT. For each sample the quaternion is calculated through 6 iterations of the optimization algorithm. This procedure is then repeated with rotations for each of the three axes of the system.

Test Equipment		
Name / Description	*Type*	*AAU-ID*
HS-645MG	Servo motor	-

G.3 Test results and data processing

A total of 500 quaternions together with 500 gyroscope samples were logged during each test. The quaternion elements from each test, together with the gyroscope measurements, can be seen in figures G.1, G.2 and G.3. The sample axis is set such that sample 0 is placed when the gyroscope is at rest after the rotation to better discern the convergence rate.

To get a convergence rate it must be defined when the orientation is considered to have converged. Here the orientation is defined as converged when the calculated orientation is within 11° of the average orientation of the last 50 samples. This is based on requirement T.6 that states that displacements become noticeable when above 11°. The tests showed that the orientation had converged as soon as the gyroscope had stopped registering a rotation, except for the y-axis where convergence happened somewhere between the samples for 40 ms and 50 ms after the rotation. However, some unexpected behavior can be seen on the data for the y-axis, where the quaternion element, q1, is first seen to be rising but then going back to its initial value. Additionally, the gyroscope measurement was very noisy after the rotation. The exact convergence rates calculated from each test can be seen in table G.1

Table G.1

Calculated Convergence Rates	
Axis of rotation	*Convergence rate*
X-axis	0 ms
Y-axis	0 ms
Z-axis	40-50 ms

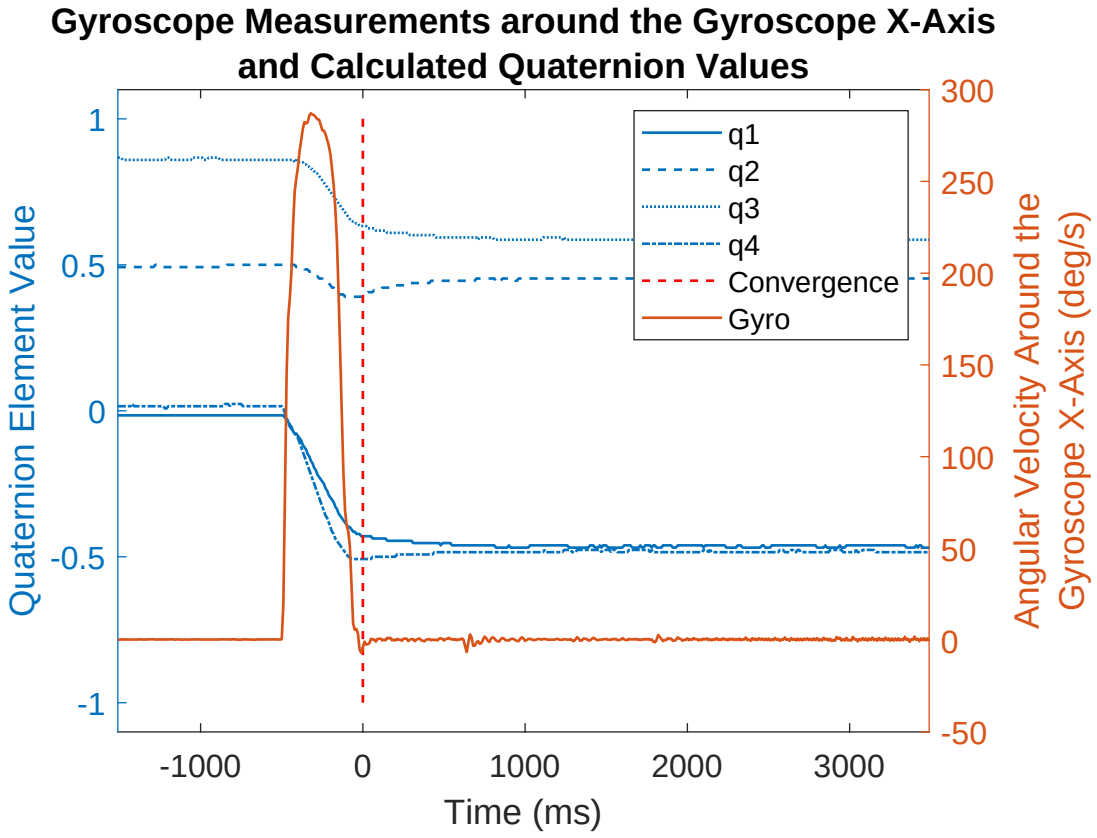


Figure G.1

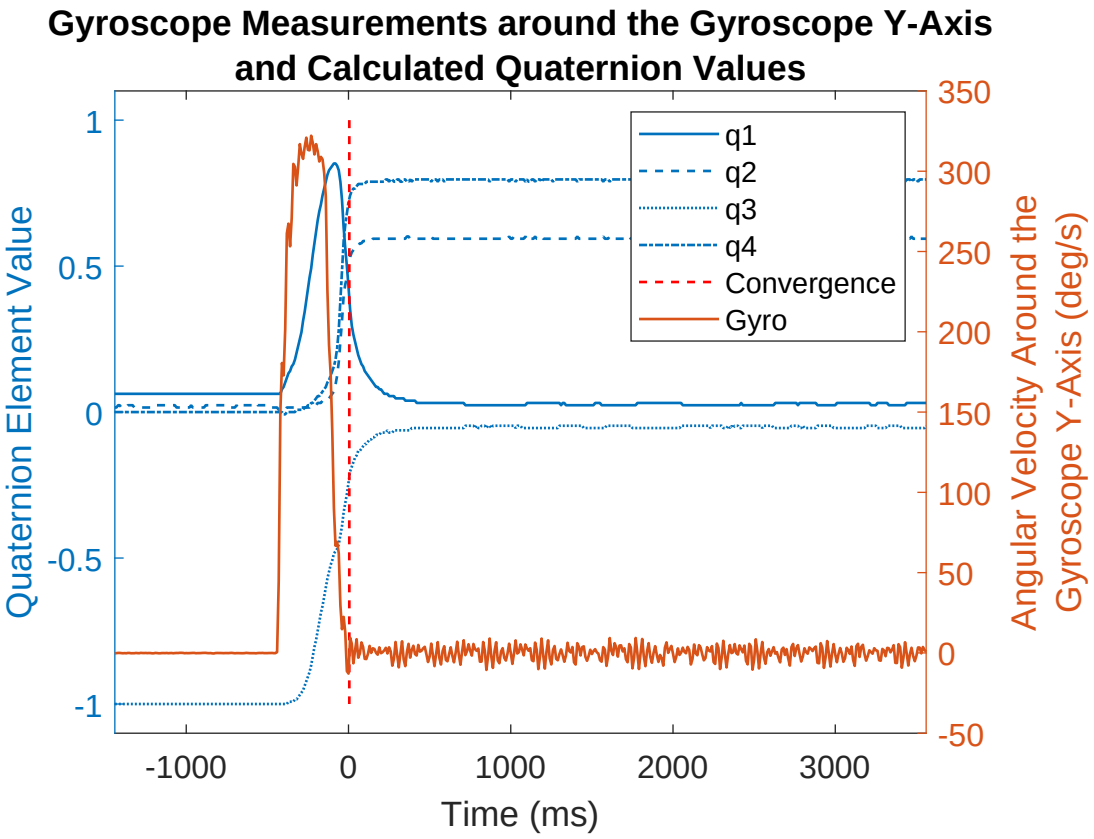


Figure G.2

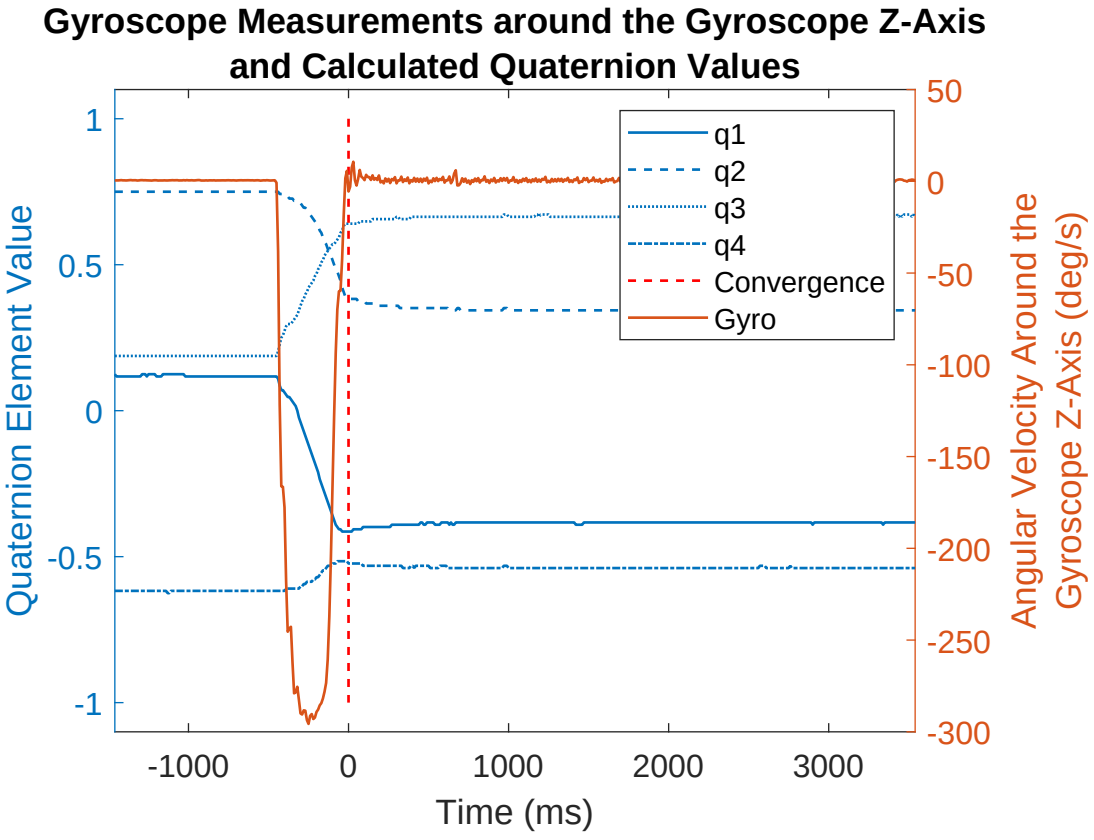


Figure G.3

G.4 Sources of error and other uncertainties

Appendix H

Test Journal: Measuring the Transfer Function of the Beyerdynamics DT770 Headphones

H.1 Introduction

Headphones do not always have a flat frequency response which is ideal for binaural synthesis. Therefore, a given pair of headphones must have their frequency response corrected by the means of equalization. However, the frequency response does not only rely on the specific pair of headphones but also the person which wears them [10]. In the case of using the Valdemar database for the binaural synthesis, the best way to obtain an equalization which will fit most users is to measure the headphones response when Valdemar wears the headphones.

H.2 Test Frame

H.2.1 Test Setup and Procedure

A general illustration of a measurement system for obtaining impulse responses can be seen on the block diagram on figure H.1.

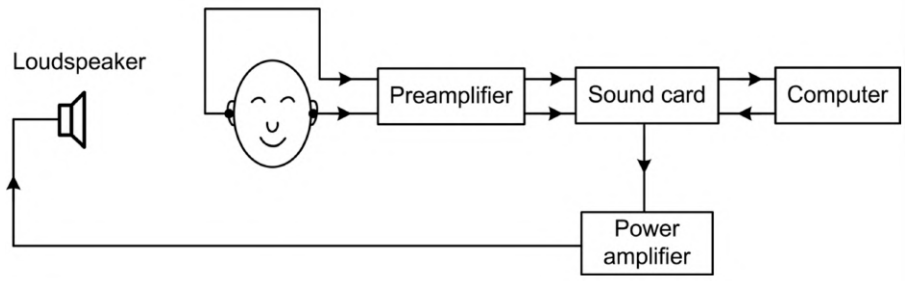


Figure H.1: Block diagram of a HRTF measurement system [1, p. 54].

Here, a sound card controlled by a computer sends a measuring signal through a D/A conversion to a power amplifier and then to a loudspeaker. Then, when the output of the loudspeaker arrives at the ears of either a human subject or an artificial head, it is picked up by a pair of matching microphones each placed at the entrance of a blocked ear canal. Thereafter, both signals recorded by the microphones undergo an identical amplification followed by an A/D conversion where the signals are sampled by the sound card [1, p. 53-54]. Finally, the measured signal can be deconvolved with the signal sent to the loudspeaker and the impulse response is then obtained [1, p. 51-52].

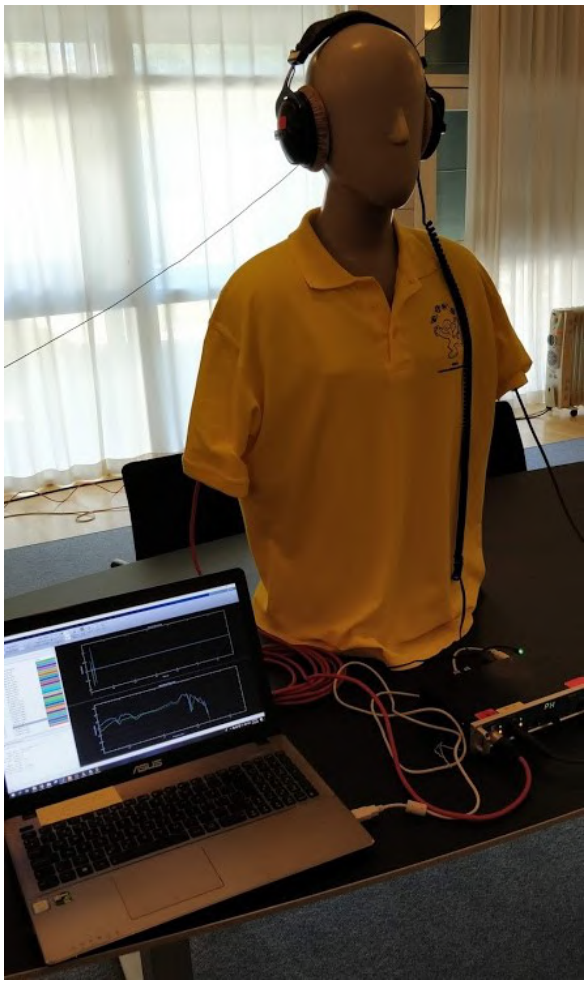


Figure H.2: Test setup of the headphone impulse response measurement in the VR laboratory.

The measurement setup for this test is similar to that on figure H.1, however the loudspeaker is replaced by a pair of headphones. The sound card is an RME Fireface UCX sound card which is able to run full duplex mode and it communicates with the computer via an USB interface. The Computer is a standard laptop with Windows 10 and the measurement program used on the computer is Matlab R2018b. In Matlab the 'impulseResponseMeasurer' application is used to measure the headphones. In this application the following settings are used:

- Audio Device: ASIO Fireface USB
- Sample Rate: 48 000 Hz
- Player Channel: six - 6 mm Stereo Analog Output on the FireFace sound card which the headphones are connected to
- Recorder Channel: one or two for the right and left ear respectively - these are two XLR input ports for microphones which both contain microphone pre-amplifier with phantom power.
- Method: Maximum Length Sequences (MLS)
- Number of Runs: ten
- Duration per Run: 1 Second
- Excitation Level (dBFS): -20

The method of maximum length sequences (MLS) is chosen because it is practically immune to noise while being quicker than other LTI measurement techniques [47, p. 254]. Although the MLS technique can be used with just two runs, one demo run and one measurement run, ten is chosen instead to provide a better measurement average of the frequency response [47, p. 257] The MLS technique also provides a very high signal to noise ratio, this is because it has a crest factor of 1, which means that the measuring signal can use the maximum available headroom when measuring [47, p. 257]. To use the maximum available headroom, the RME TotalMix FX software for Windows 10 was used to adjust the gain on the sound card to make sure the signal never clipped the microphone input range.
In the following section, the exact measurement equipment is detailed.

H.2.2 Measurement Equipment

The experiment was performed in the AAU VR Laboratory (room B3-103) and the following measurement equipment was used:

Test Equipment		
Name / Description	*Type*	*AAU-ID*
RME Fireface UCX	Full-duplex sound card	108248
Two microphone cables	5 m XLR cable	-
Beyerdynamics DT770 Professional	Over-ear headphones	2037-23
Stereo jack converter	3.5 mm to 6 mm	-
USB B to USB A Cable	USB cable	-
DC Power Supply	Power supply for sound card	-
"Valdemar Sejr"	Artifical head	2150-00
Matlab 2018b	-	-

H.3 Test Results and Data Processing

The frequency response of both headphone channels has been measured and they are plotted in the following figure H.3.

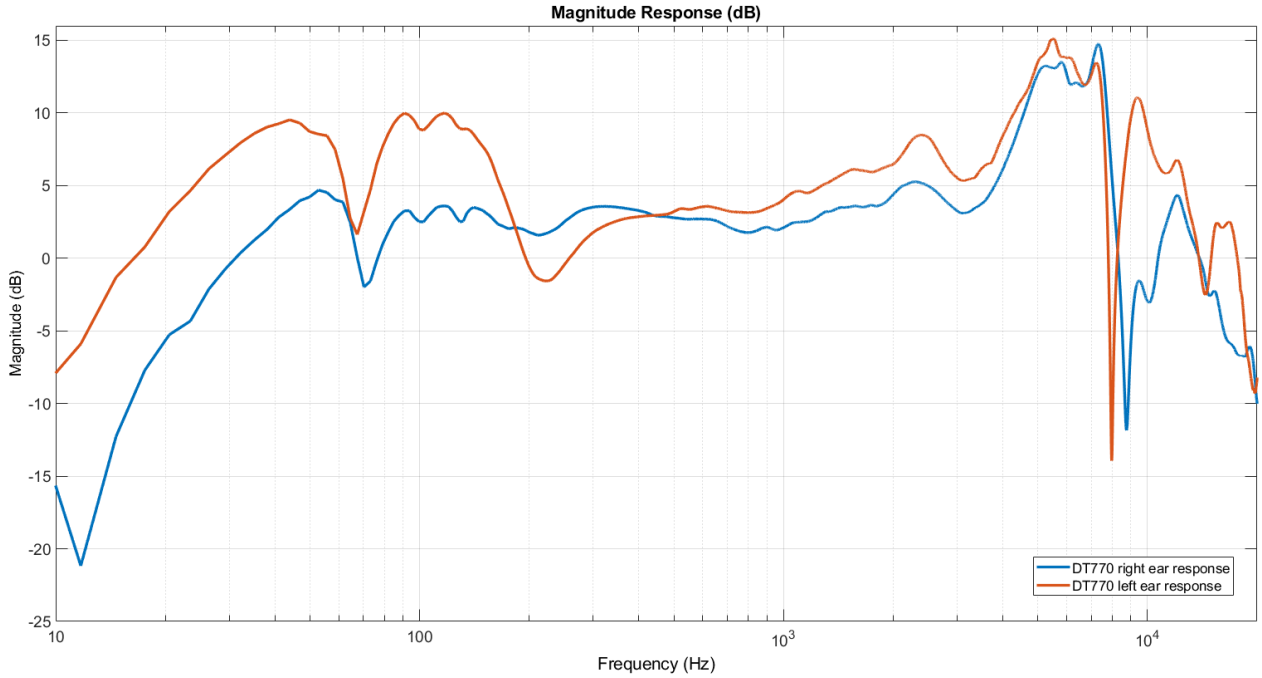


Figure H.3: Frequency response of the Beyerdynamics DT770 measured on Valdemar.

The results of the measurements show a fairly flat frequency response in the low to mid frequency range (20 Hz to 2 kHz) where the maximum gain peaks at 5 dB. And between 3.5 kHz and 7 kHz there is a much larger peak at up to 15 dB which definitely require equalization. There are also bigger individual spikes, both above and below 0 dB, above 7 kHz.

H.4 Sources of Error

The application used in Matlab was unable to measure the microphones in both ears simultaneously. Therefore, first the right ear and then the left ear were measured. However, this error should be mitigated due to both headphone channels playing the same MLS signal during each 10 sequences for both left and right measurement.

Appendix I

Test Journal: Verifying the Preprocessing Is Correct

I.1 Introduction

In this test the preprocessed HRTFs are going to be measured, specifically the extracted ITD, the gain at DC and the scaling and quantization are going to be measured. Additionally, the final preprocessed HRTFs will also be compared with the original 256 coefficient HRTFs. This test is completed to verify that the ITD, DC gain adjustment and scaling and quantization are implemented correctly.

I.2 Test frame

I.2.1 Test Setup and Procedure

This test is being completed purely on a laptop with Matlab installed. Here, the preprocessed filters which are going to be used on the DSP are tested to verify if they will filter sound sources correctly. To test the preprocessing, a number of preprocessed HRTFs for several positions on the sphere for both ears are selected, then the following tests will be carried out:

- To verify the extracted ITD which is saved with the coefficients is correct, the extracted number is compared with the impulse response of the original HRTF for both ears.
- To verify the DC gain is correct the frequency response of the filter is plotted and evaluated. To see if there is any effect at all from the adjustment, it can be compared with the truncated HRTF with no adjustment.
- To see the effect of the scaling and quantization the frequency response of the HRTF can be compared before and after the processing is implemented.
- To evaluate the quality of the complete HRTF preprocessing, the frequency response is plotted and compared for both the original HRTF and the fully preprocessed HRTF for the same locations.

I.2.2 Measurement Equipment

Test Equipment		
Name / Description	*Type*	*AAU-ID*
Windows 10 computer	PC	-
Matlab 2018b	Software	-

I.3 Test Results and Data Processing

As mentioned in the previous section, four different measurements were completed. The first test is the measurement of the ITD value. Figure I.1 shows the original 256 coefficient HRIR for both ears, the black lines at 10 and 40 samples mark where the left and right impulse response start if they had their arrival delay and ITD extracted respectively. These markers can be used to measure the ITD for this HRTF pair (−90° azimuth and 0° elevation) which is equal to 30 samples. By means of inspecting the preprocessed HRTF it is found that the corresponding HRTF has the number 30 saved together with the coefficients for the right filter.

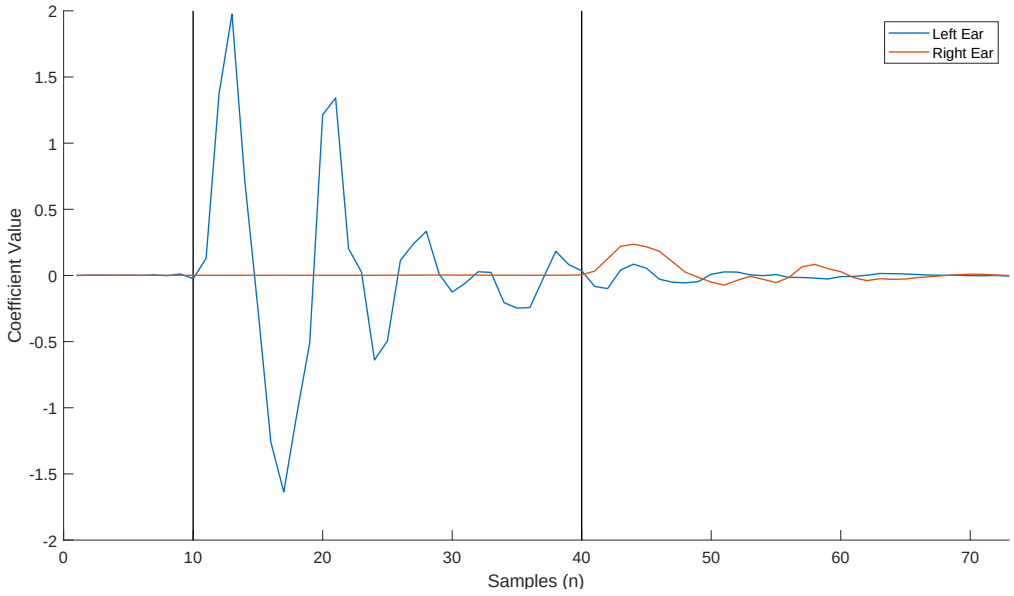


Figure I.1: Impulse Response of the unprocessed HRTF for both ears for −90° azimuth and 0° elevation.

Figure I.2 shows the relationship between the extracted ITD which is saved with the preprocessed filters for every 180 azimuth indexes at 0° elevation. Azimuth index is different from the spherical coordinate system, where index 0 to 90 corresponds to azimuth 0° to 180° and index 91 to 180 corresponds to azimuth −180° to 0°, with steps of 2° for every index. Recall that only the positive value of ITD is saved, which means that when the sound source is the right half circle (positive azimuth degrees) then the ITD is saved for the left ear. Conversely when the sound source is in the left half circle (negative azimuth degrees), then the ITD is saved in the filters for the right ear, as illustrated on figure I.2.

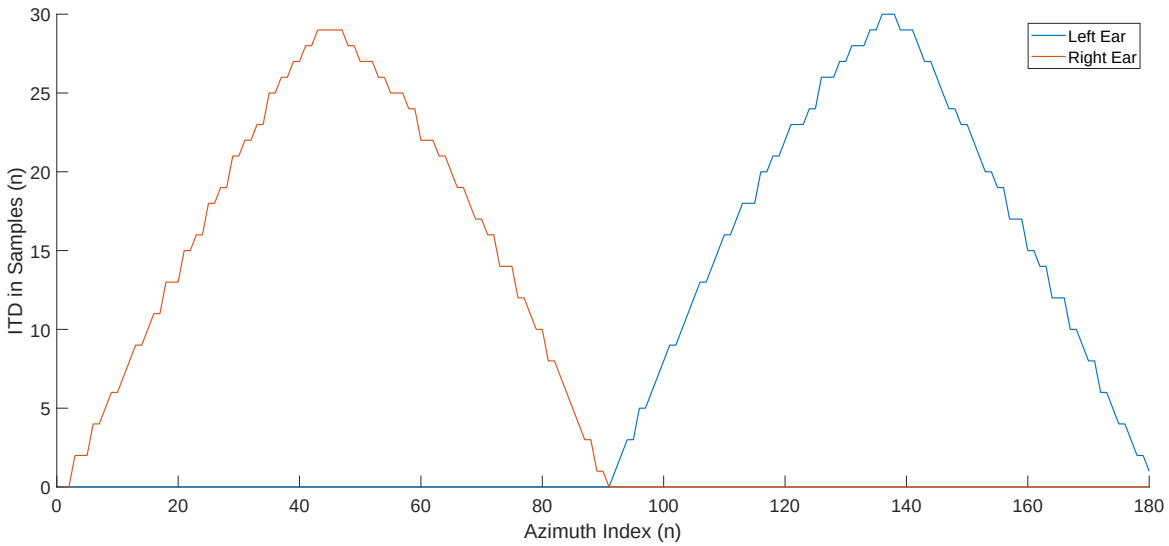


Figure I.2: Graph which shows the relationship between the ITD (measured in samples) and azimuth index for 0° elevation

The second test is where the DC gain adjustment is measured. Here, on figure I.3, the frequency response of the truncated HRTF at 58° azimuth and 70° elevation is compared with its counterpart with DC gain adjustment. The frequency response shows that the DC gain has correctly been adjusted to 0 dB, while the rest of the response is not affected dramatically by the response.

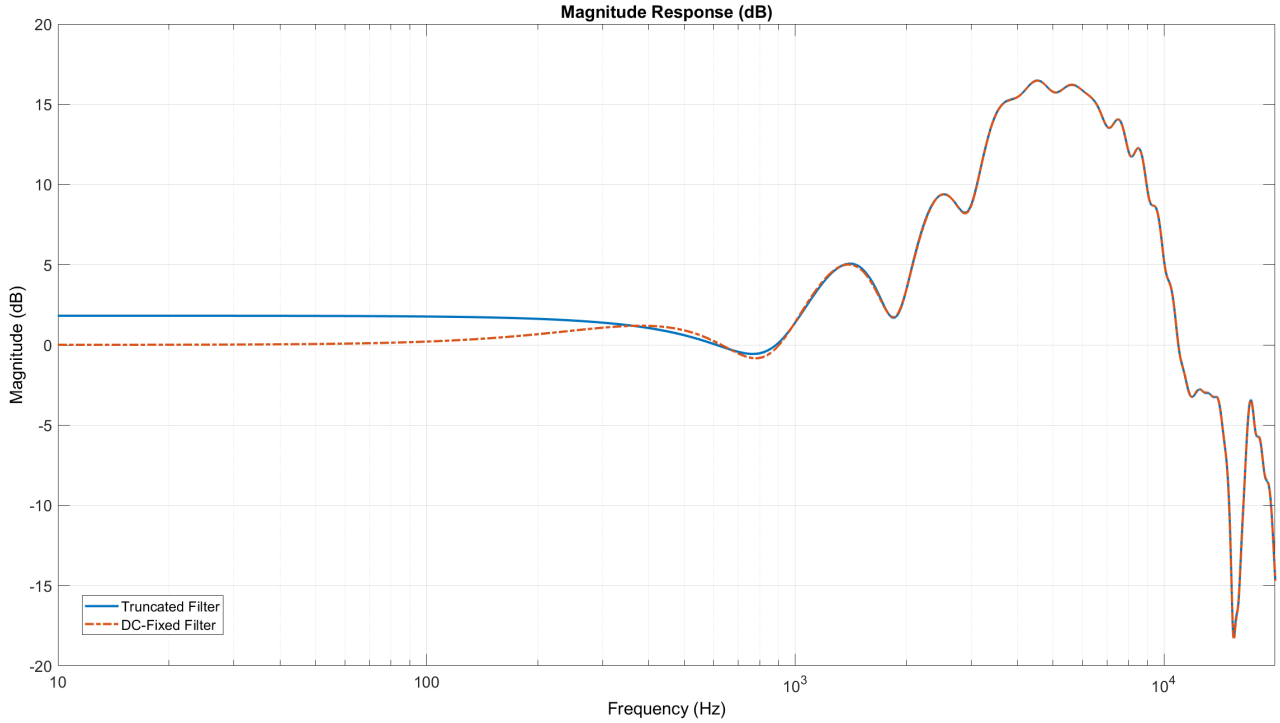


Figure I.3: Frequency response for the filter with highest DC gain (58° azimuth and 70° elevation), before and after DC gain adjustment.

The third test compares an unscaled floating-point filter to the same filter, but where the second filter is scaled and quantized so that it can be implemented on the DSP. Figure I.4 shows the frequency response for the filter with the highest magnitude filter coefficient, before and after scaling and quantization. The test shows that the filter is not affected negatively by scaling, although it is noted that the gain

is increased. This is expected, since 2^{13} results in a gain of $\approx 78dB$, and it is not an issue when used in the DSP correctly.

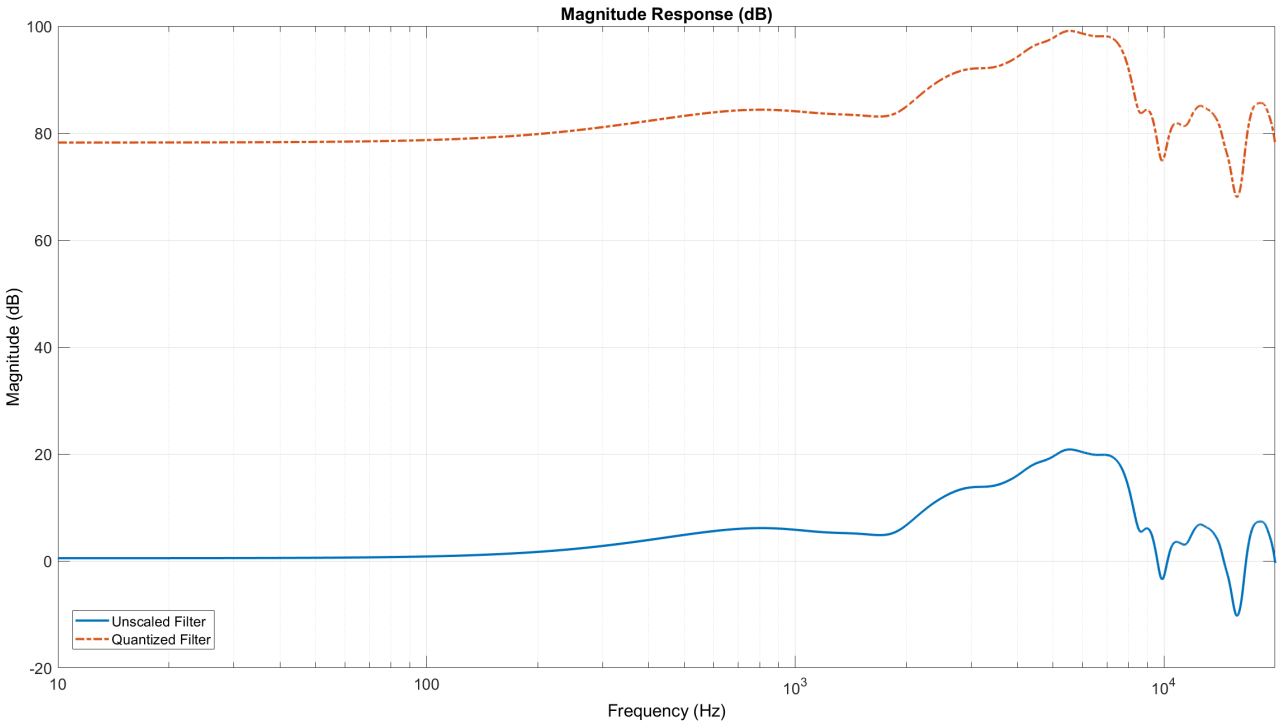


Figure I.4: Frequency response for the filter with the highest coefficient, before and after scaling and quantization.

The final and fourth test is where the fully preprocessed HRTF is compared with the original 256 coefficient HRTF from the Valdemar database. The test result for this test can be seen on figures I.5 through I.10, it should be noted that for the sake of comparison the preprocessed HRTF is not scaled because that would add a significant magnitude offset as seen on figure I.4. It is clear from the frequency response that some frequency resolution is lost when truncating the filter from 256 to 72 coefficients, however it does not seem to affect the HRTF very much. It is also clear that the DC adjustment has an attenuating effect all the way up to 250 Hz. However, because there are no localization cues in frequencies below 1.5 kHz [1, p. 17] and a large difference in DC value between the two ears is very undesirable [14, p. 22], it is deemed that the adjustment is more than satisfactory. Figure I.5 shows the same filter as in figure I.3, however it shows the difference between a fully preprocessed filter and one which has zero processing done. It should be noted that the scaling is removed, to better compare the two graphs. It is clear that some resolution is lost when truncating the filter from 256 to 72 taps, however the overall filter is unaffected. Further comparisons between the original HRTFs and the preprocessed HRTFs have been made and these are shown in figure I.6 to I.10.

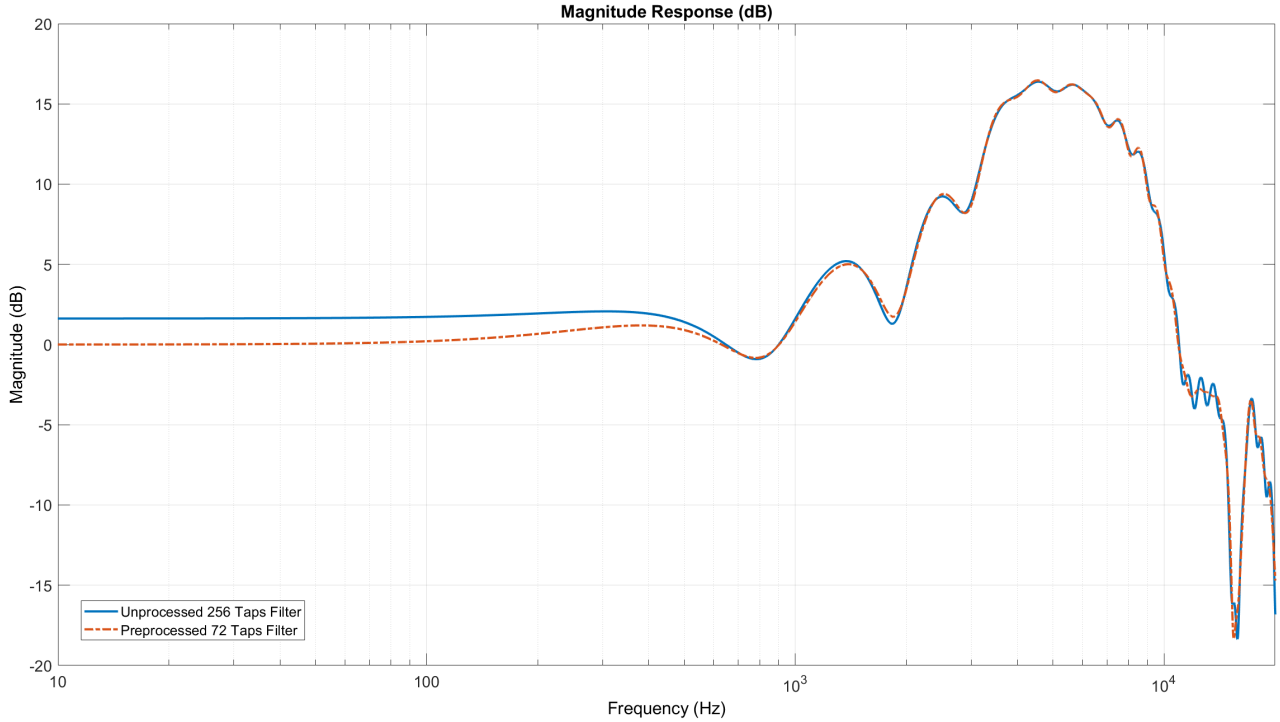


Figure I.5: Frequency response for the comparison between the original HRTF for 58° azimuth and 70° elevation and the fully preprocessed HRTF for the same location.

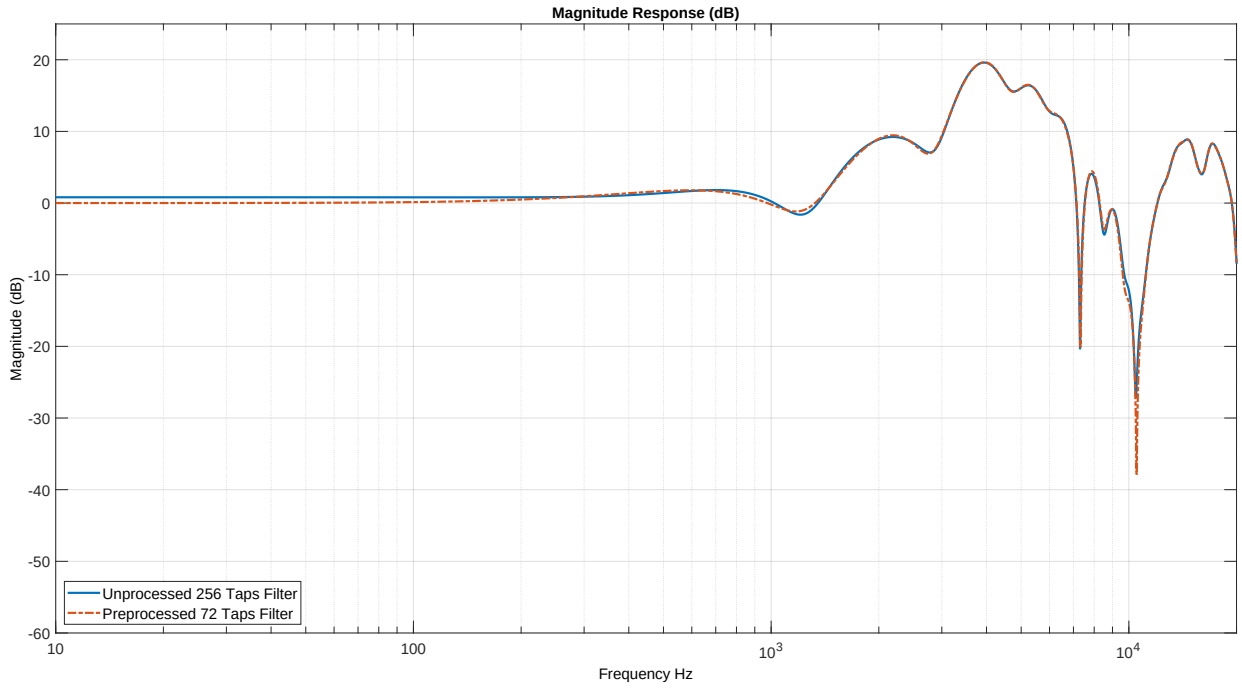


Figure I.6: Frequency response for the comparison between the original HRTF for 0° azimuth and 0° elevation and the fully preprocessed HRTF for the same location.

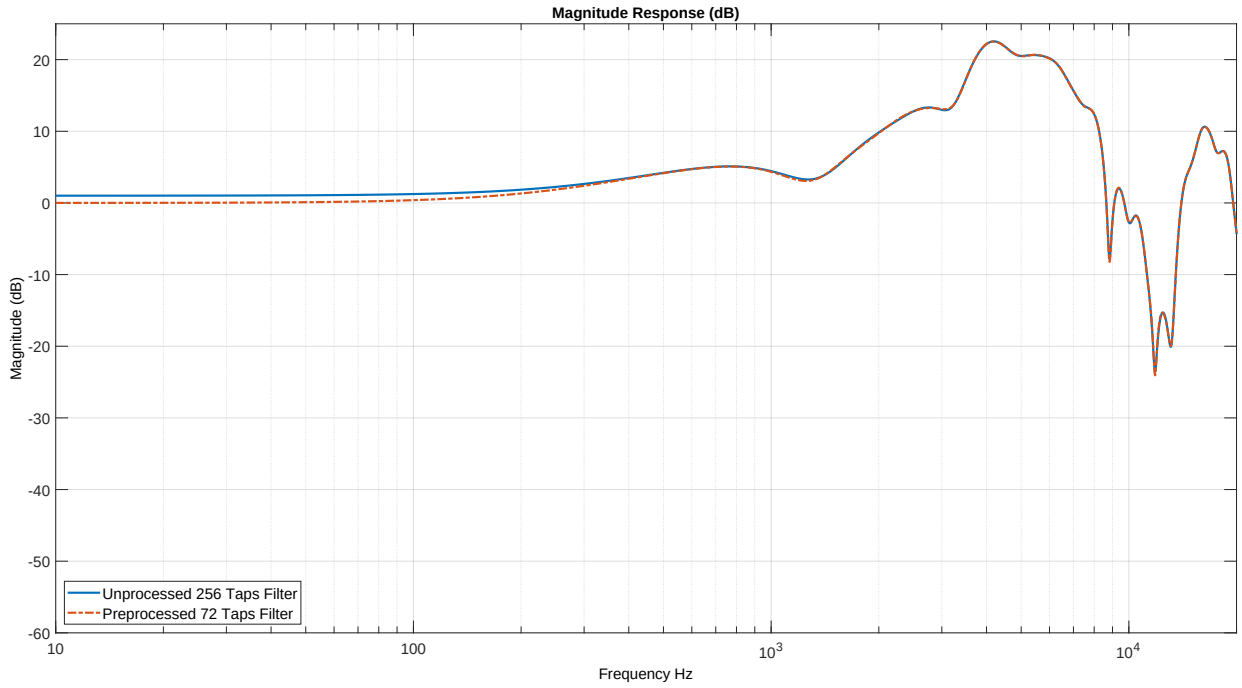


Figure I.7: Frequency response for the comparison between the original HRTF for 45° azimuth and 0° elevation and the fully preprocessed HRTF for the same location.

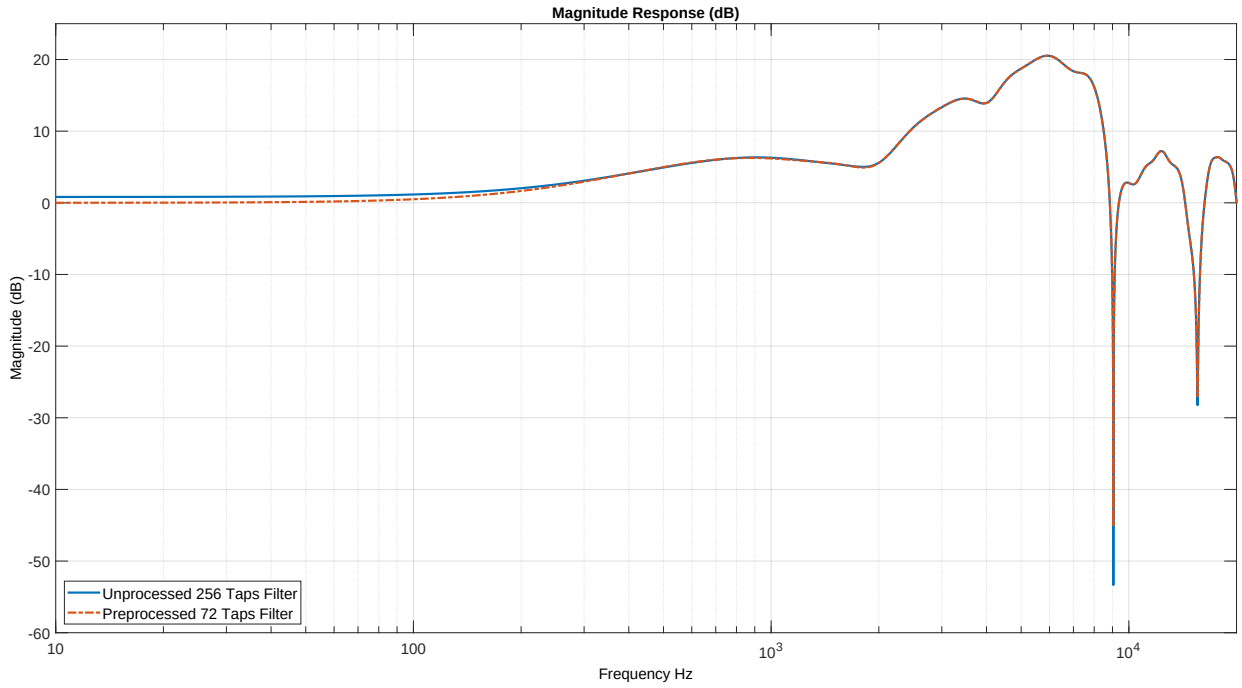


Figure I.8: Frequency response for the comparison between the original HRTF for 90° azimuth and 0° elevation and the fully preprocessed HRTF for the same location.

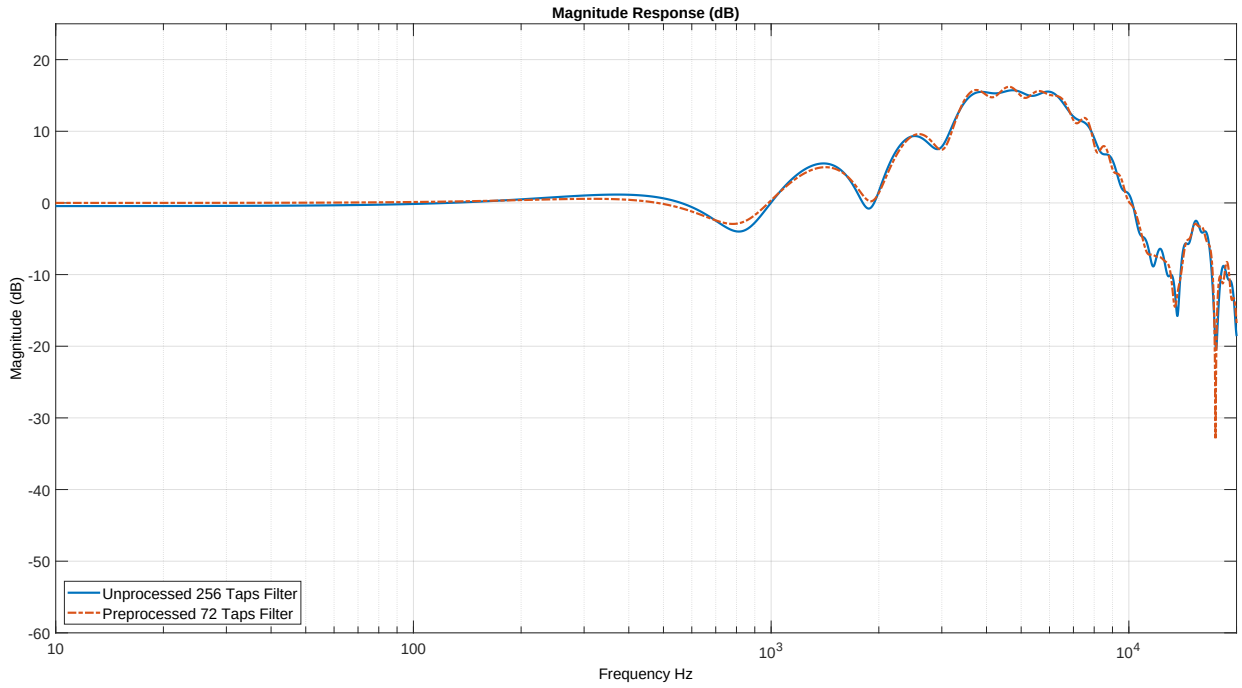


Figure I.9: Frequency response for the comparison between the original HRTF for 0° azimuth and 45° elevation and the fully preprocessed HRTF for the same location.

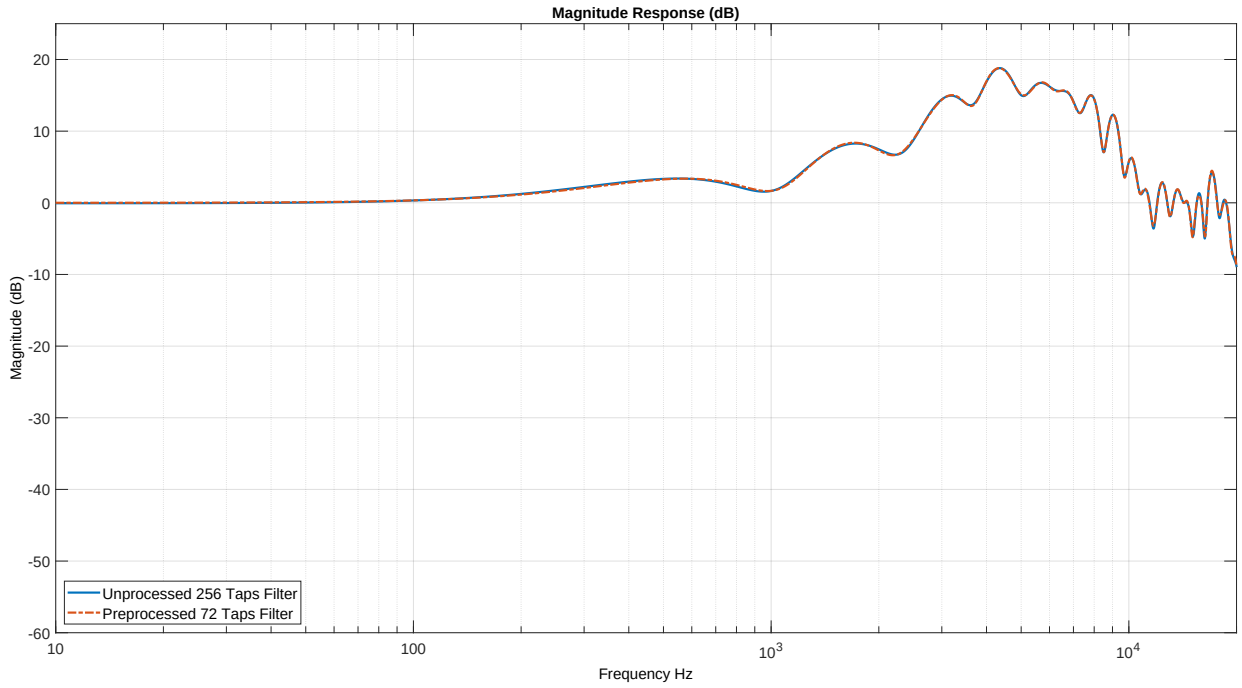


Figure I.10: Frequency response for the comparison between the original HRTF for 90° azimuth and 45° elevation and the fully preprocessed HRTF for the same location.

Appendix J

Test Journal: Measurement of the Audio Codec on the DSP

J.1 Introduction

Similar to the headphone frequency response test in appendix H, the audio codec on the DSP is going to be tested to see if its frequency response will have any effect on the HRTF filtering. To measure the frequency response, the DSP is set to directly send the input at the ADC to the output of the DAC.

J.2 Test frame

J.2.1 Test Setup and Procedure

To measure the frequency response the Analog Discovery 2 is used in combination with the BNC breakout board. Here, the wavegen channel 1 is connected to both input channels of the ADC on the DSP board. Scope channel 1 is connected to the right channel output of the DAC and scope channel 2 is connected to the left channel on the DSP board. In the Waveforms software the Network Analyzer application is used with the following settings:

- Start frequency 20 Hz and stop frequency 20 kHz
- Samples: 1001
- Wavegen setup: Offset is 0 V and amplitude is 300 mV

The measurement setup can be seen on figure J.1

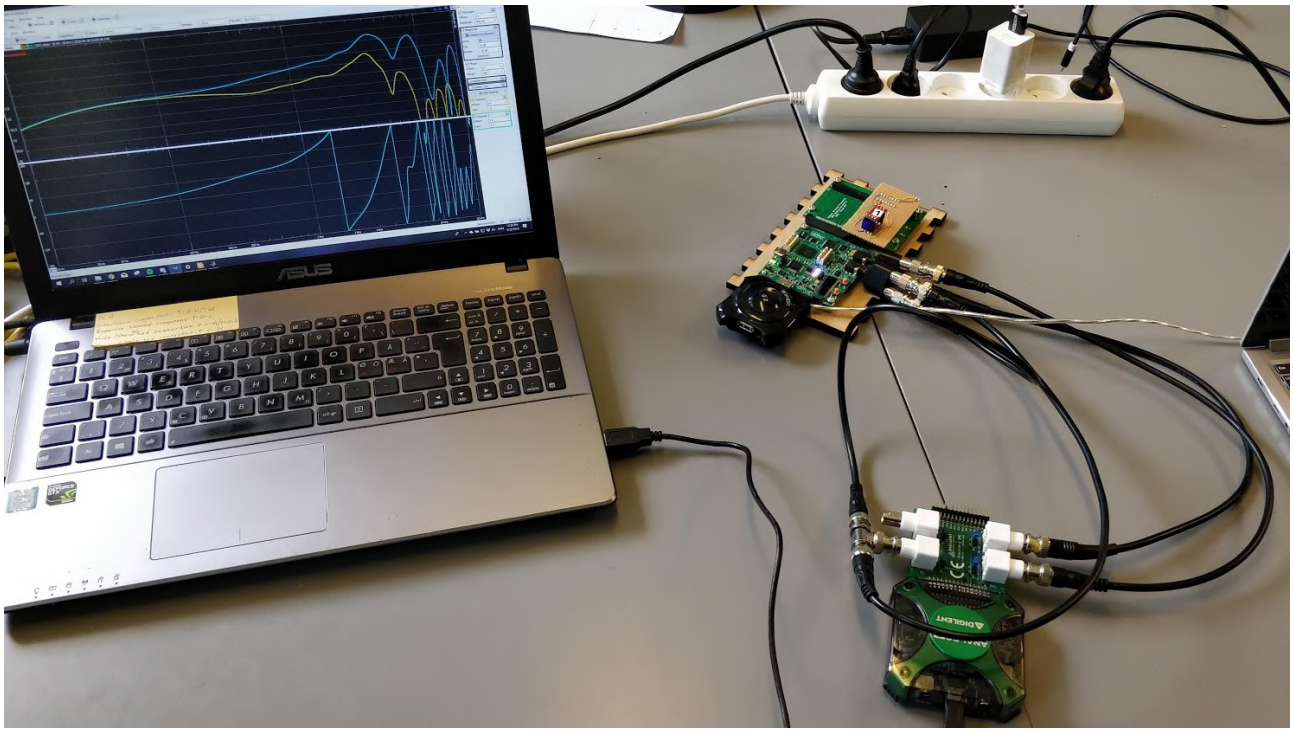


Figure J.1: Picture of the measurement setup used to measure the frequency response of the DSP with the Analog Discovery 2 and a computer.

J.2.2 Measurement Equipment

Test Equipment		
Name / Description	*Type*	*AAU-ID*
eZdsp Board	DSP	-
Digilent Analog Discovery 2	Digital Oscilloscope	2179-08
Digilent Discovery BNC Board	Oscilloscope Breakout board	-
BNC Cables	-	-
BNC to RCA Converter	-	-
RCA to stereo jack Converter	- 3.5 mm jack	-
Mini USB to USB A cable	-	-

J.3 Test Results and Data Processing

The test results can be seen on figure J.2. On the figure it can be seen that the right channel has a very flat frequency response as is expected. However, the left channel has a lot of distortion above 2 kHz. During preliminary tests, it was clear that there was some aliasing on the left channel which supports the results shown on the frequency response. The preliminary test consisted of the ADC sampling a sine wave created by the digital oscilloscope and the DAC's output was measured on the oscilloscope.

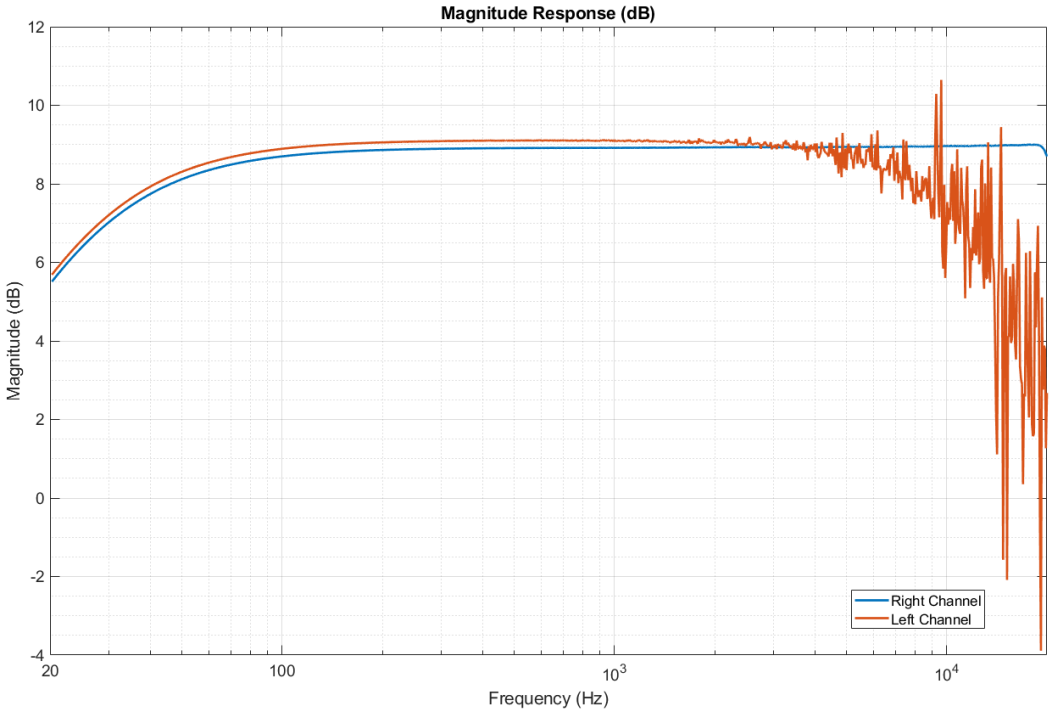


Figure J.2: Frequency response of both codec channels on the DSP.

Appendix K

Test Journal: Verifying the HRTF Filter Selection on the DSP

K.1 Introduction

The purpose of this test is to ensure that the filter selection code works as intended, since selecting the wrong filter ruins the purpose of the entire system. The test is also to check that the filter coefficients are saved correctly onto the flash chip on the DSP, as well as that the coefficients are exported correctly from Matlab.

K.2 Test frame

K.2.1 Test Setup and Procedure

This test is completed in combination with appendix M where several HRTFs are measured. When several HRTFs are being measured on the DSP they can be compared with the corresponding preprocessed HRTFs to see if they are correct. If the measured and preprocessed HRTFs match, that must also mean that the filter selection match. In addition to measuring the HRTFs like they are done in appendix M, the exact filter coefficients can also be read in the DSP's memory, and these numbers can be compared with those of the preprocessed HRTFs.

K.2.2 Measurement Equipment

Test Equipment		
Name / Description	*Type*	*AAU-ID*
eZdsp Board	DSP	-
Digilent Analog Discovery 2	Digital Oscilloscope	2179-08
Digilent Discovery BNC Board	Oscilloscope Breakout board	-
BNC Cables	-	-
BNC to RCA Converter	-	-
RCA to stereo jack Converter	- 3.5 mm jack	-
Mini USB to USB A cable	-	-
Female USB A to male USB A cable	-	-
Code Composer Studio v8	Debugging Software	-

K.3 Test Results and Data Processing

For the tested HRTFs locations, all HRTFs in 0° elevation were successfully indexed and properly filtered on the DSP. This can be seen for e.g. $-45°$ azimuth where the measured frequency response is compared with the preprocessed HRTF in figure K.1. A slight decrease in overall amplitude is

seen, but is deemed to be as a result of loss in measuring equipment or mismatch in in- and output voltages. Additionally, there is an increasing attenuation at lower frequencies, which is due to the frequency response of the audio codec on the DSP (see appendix J). Additional comparisons between preprocessed and original HRTFs at 0° elevation can be found in the results of appendix M.

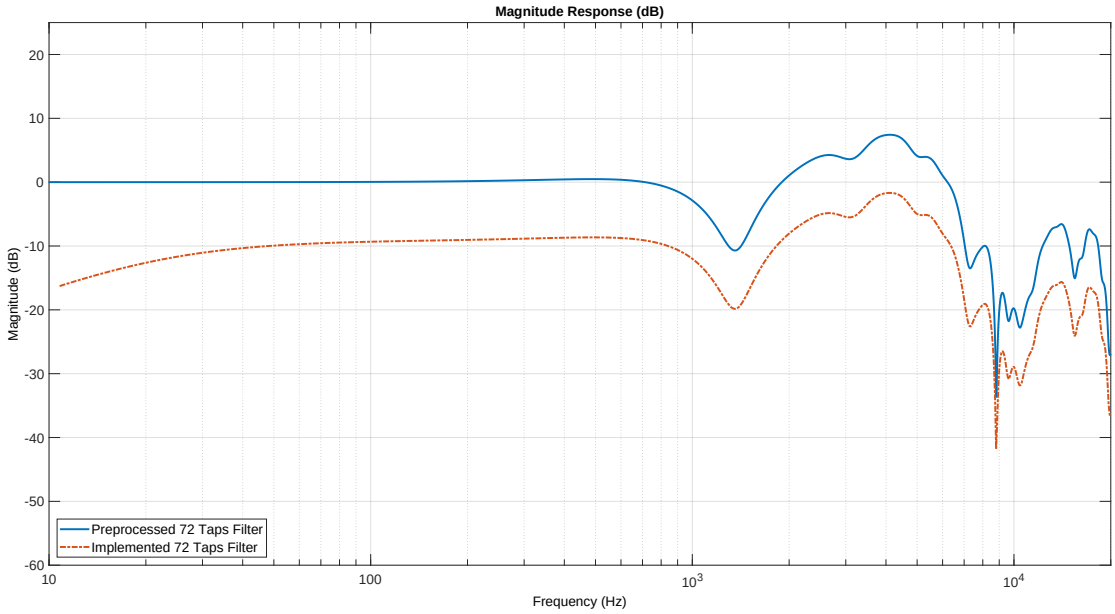


Figure K.1: Frequency response comparison between the measured HRTF and the preprocessed HRTF for −45° azimuth and 0° elevation.

When attempting to filter HRTFs above 0° elevation, i.e. between 1° and 90° elevation, all the filter coefficients returned −1 (or the equivalent of FFFF in 16-bit fixed point). When filters below 0° elevation were measured, e.g. −45° and −45° elevation on figure K.2, the measured response was not the same as the response of the preprocessed HRTF for same location. Therefore, it is concluded that there is likely some error in the way the HRTF database is being indexed on the DSP or how the database is stored in the flash memory on the DSP.

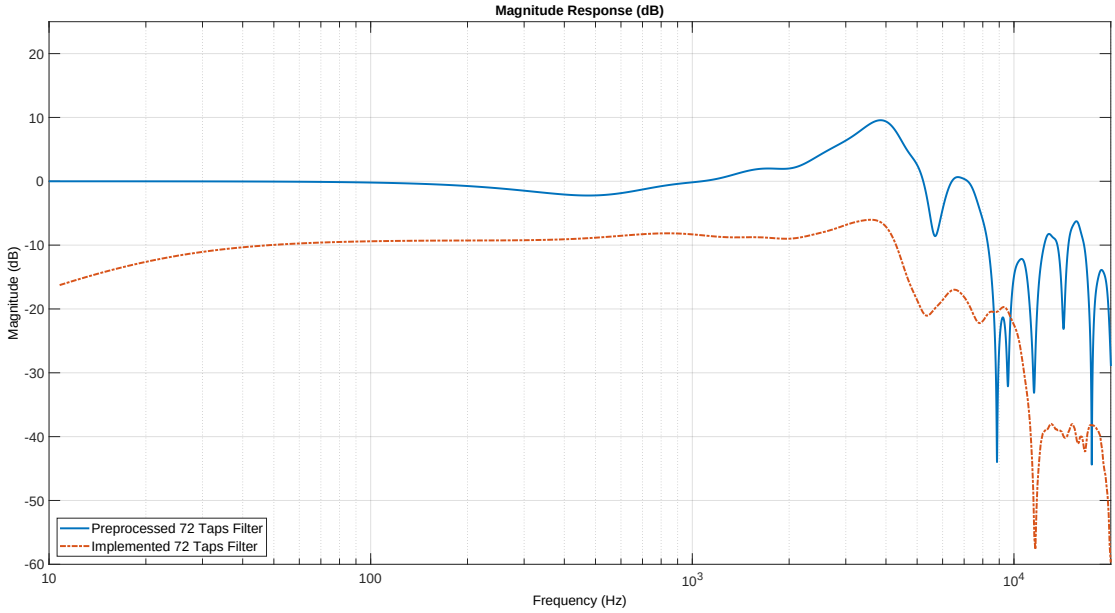


Figure K.2: Frequency response comparison between the measured HRTF and the preprocessed HRTF for $-45°$ azimuth and $-45°$ elevation.

Appendix L

Test Journal: Measurement of the ITD in the HRTF Filtering on the DSP

L.1 Introduction

As stated in section 1.1 the ITD is an important localization cue, therefor it is important to make sure that the ITD is correctly implemented in the HRTF filtering on the DSP. To measure the ITD, an impulse response of the HRTF filtering on the DSP is measured for several HRTF positions on the sphere for both ears. Thereafter, the impulse response can be plotted and compared for both the left and right HRTF filtering and the ITD can be found. Finally, the measured ITD can be compared with the ITD which is saved with the filter coefficients.

L.2 Test frame

L.2.1 Test Setup and Procedure

To measure the impulse response the DSP is programmed to send a Dirac delta function through all the HRTF filtering. Both output channels of the DSP's DAC are then measured on the Analog Discovery's oscilloscope with scope channel 1 and 2. The oscilloscope application was set 1024 samples and to measure the signal 1000 times and then calculate an average signal. The following four HRTF locations were measured:

- 0° azimuth and 0° elevation
- 90° azimuth and 0° elevation
- −45° azimuth and 0° elevation
- −90° azimuth and 0° elevation

L.2.2 Measurement Equipment

Test Equipment		
Name / Description	*Type*	*AAU-ID*
eZdsp Board	DSP	-
Digilent Analog Discovery 2	Digital oscilloscope	2179-08
Digilent Discovery BNC board	Oscilloscope breakout board	-
BNC Cables	-	-
BNC to RCA converter	-	-
RCA to stereo jack converter	- 3.5 mm jack	-
Mini USB to USB A cable	-	-

L.3 Test Results and Data Processing

The measured ITD for the following HRTF locations are:

- 0° azimuth and 0° elevation on figure L.1: 20 µs
- 90° azimuth and 0° elevation on figure L.2: 605 µs
- −45° azimuth and 0° elevation on figure L.3: 401 µs
- −90° azimuth and 0° elevation on figure L.4: 633 µs

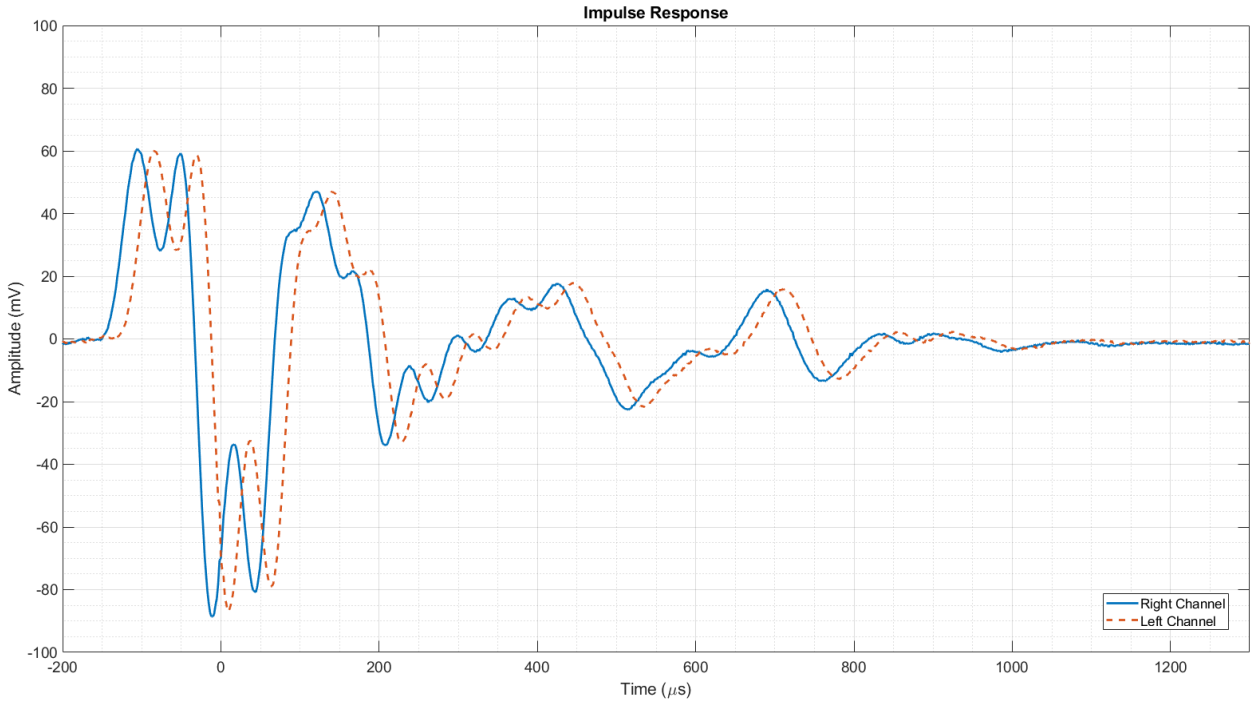


Figure L.1: Impulse response measured for both channels for 0° azimuth and 0° elevation.

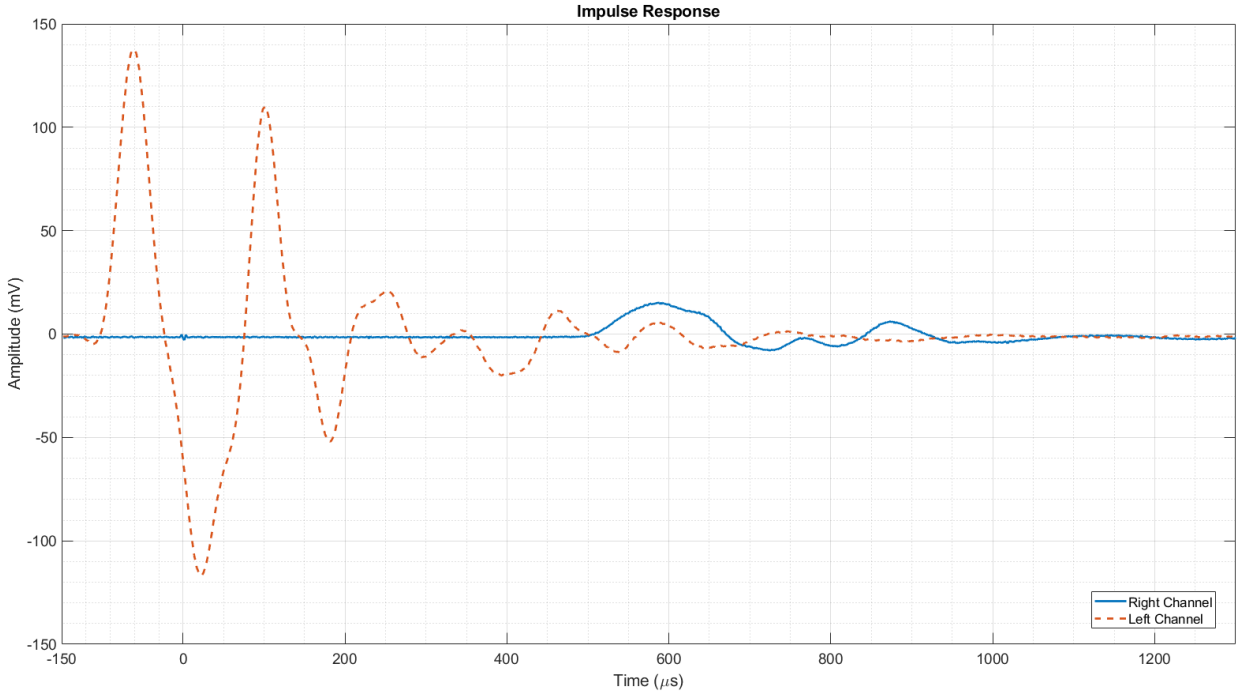


Figure L.2: Impulse response measured for both channels for −90° azimuth and 0° elevation.

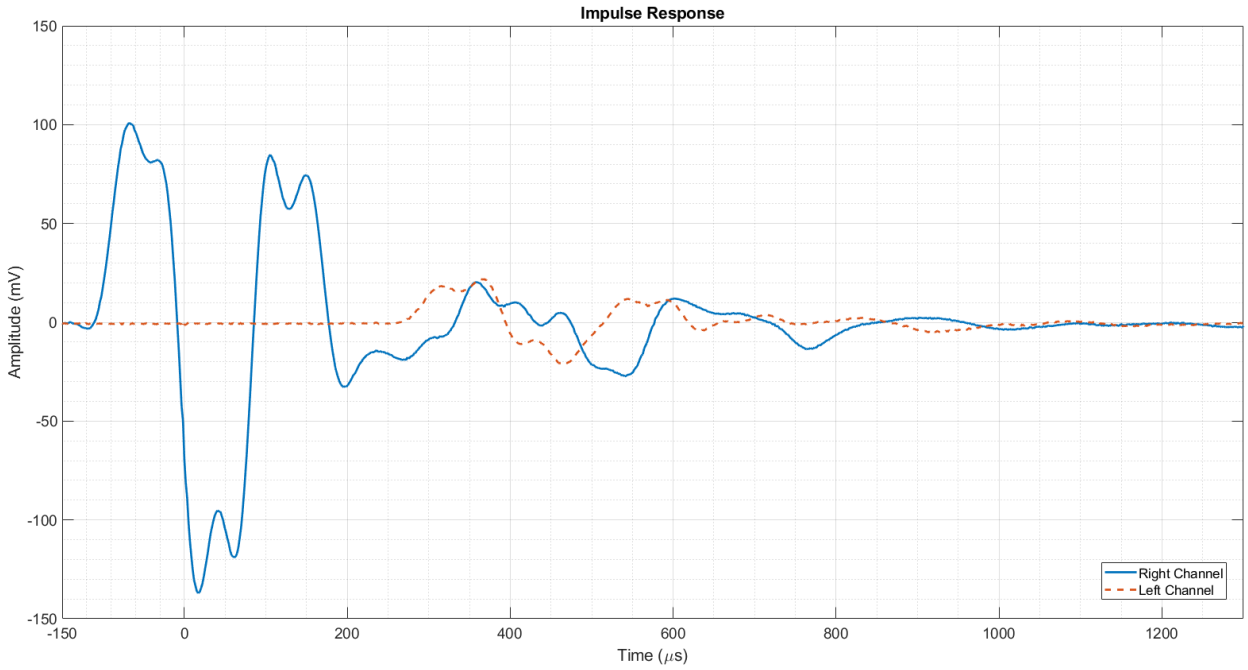


Figure L.3: Impulse response measured for both channels for 45° azimuth and 0° elevation.

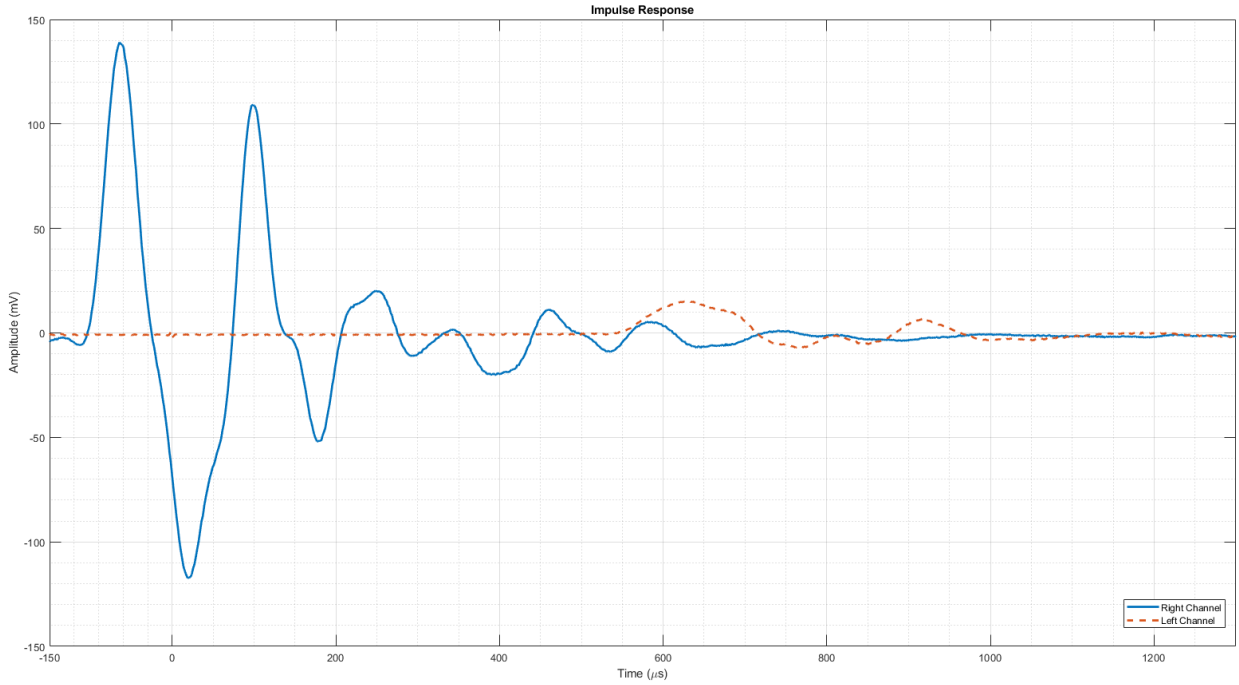


Figure L.4: Impulse response measured for both channels for 90° azimuth and 0° elevation.

L.4 Sources of Error and other uncertainties

Although it is unclear where the error originates, it is clear that the 20 µs ITD (equal to one sample at 48 kHz sample rate) for 0° azimuth and 0° elevation is incorrect as the ITD should be as close as possible to 0 µs. As it can be seen on figure L.1, the left channel is the one which is delayed compared to the right channel. So, it is assumed that this 20 µs error is also apparent in three other measurements which affects the measured ITDs.

Appendix M

Test Journal: Verifying the HRTF Filtering Is Correct

M.1 Introduction

In this test all of the HRTF filtering on the DSP is going to be tested so that the results can be used to compare with the results from the preprocessing test in appendix I. If both test results are comparable, then it can be concluded that the HRTF filtering on the DSP is done correctly. To measure the HRTF implementation, an impulse response of the HRTF filtering on the DSP is measured for several HRTF positions on the sphere for both ears. The frequency response is also measured for several HRTF positions. Thereafter, the impulse and frequency response can be plotted and compared with those in other tests.

M.2 Test frame

M.2.1 Test Setup and Procedure

This test is being completed at the same time as the measurement of the ITD test found in appendix L and the measurement of the codec frequency response in appendix J, this test uses exactly the same test setups and procedures.

M.2.2 Measurement Equipment

Test Equipment		
Name / Description	*Type*	*AAU-ID*
eZdsp Board	DSP	-
Digilent Analog Discovery 2	Digital Oscilloscope	2179-08
Digilent Discovery BNC Board	Oscilloscope Breakout board	-
BNC Cables	-	-
BNC to RCA Converter	-	-
RCA to stereo jack Converter	- 3.5 mm jack	-
Mini USB to USB A cable	-	-

M.3 Test Results and Data Processing

Below are four frequency response plots in figure M.1 through M.5, these show comparison between preprocessed HRTFs and the measured HRTFs on the DSP. Unless otherwise stated, these frequency responses are only for the right channel. In the first three figures it can be seen that despite an offset in magnitude, the response of the measured HRTFs are identical to the preprocessed HRTFs. It is also apparent that the low frequency roll-off, which was measured in appendix J, occur when the DSP is filtering HRTFs.

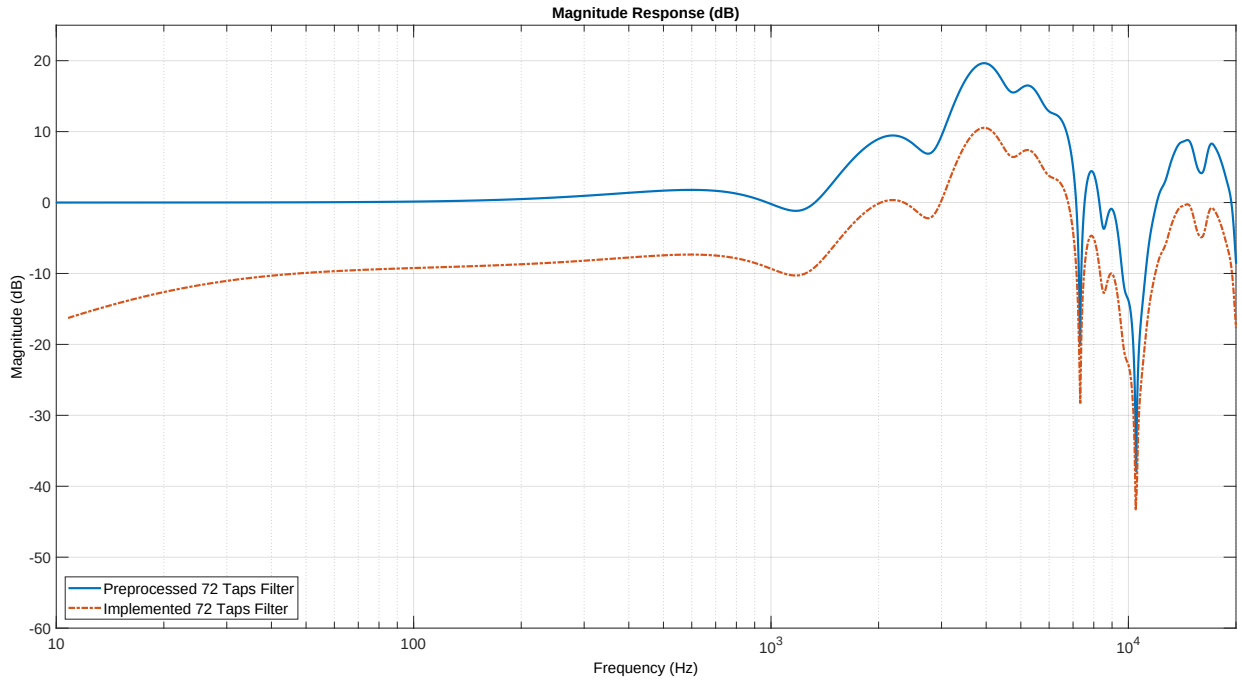


Figure M.1: Frequency response comparison between the measured HRTF on the DSP and the preprocessed for 0° azimuth and 0° elevation.

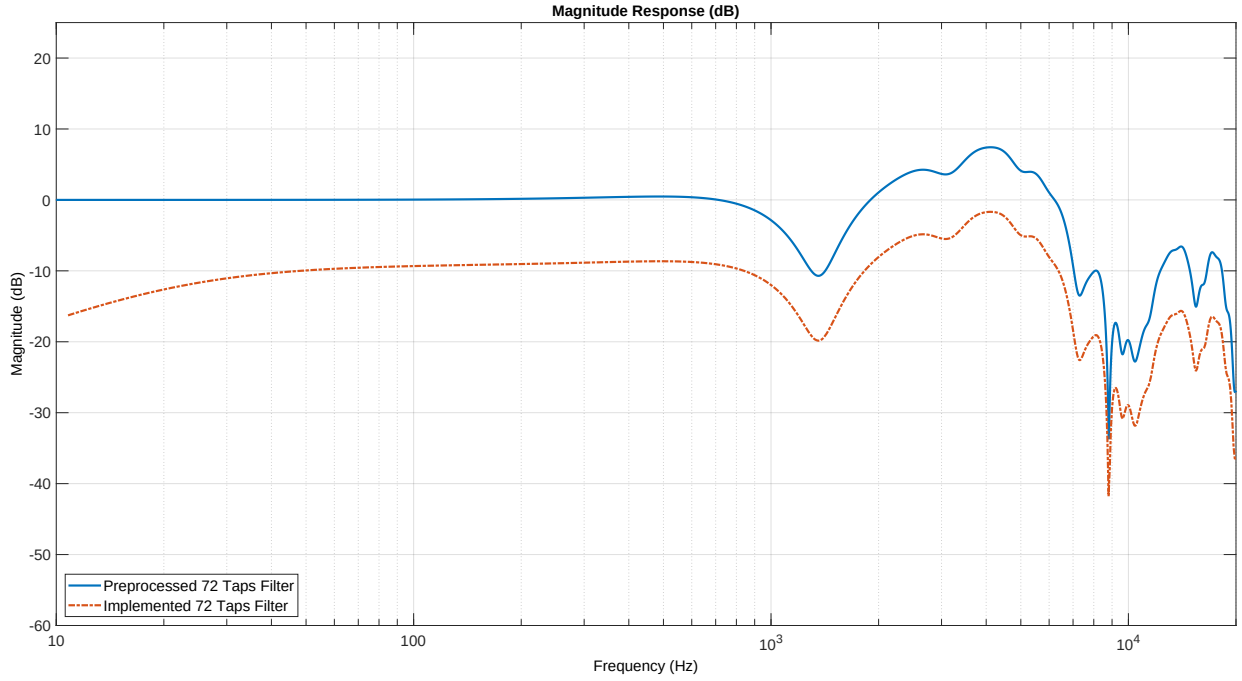


Figure M.2: Frequency response comparison between the measured HRTF on the DSP and the preprocessed for 45° azimuth and 0° elevation for the right ear.

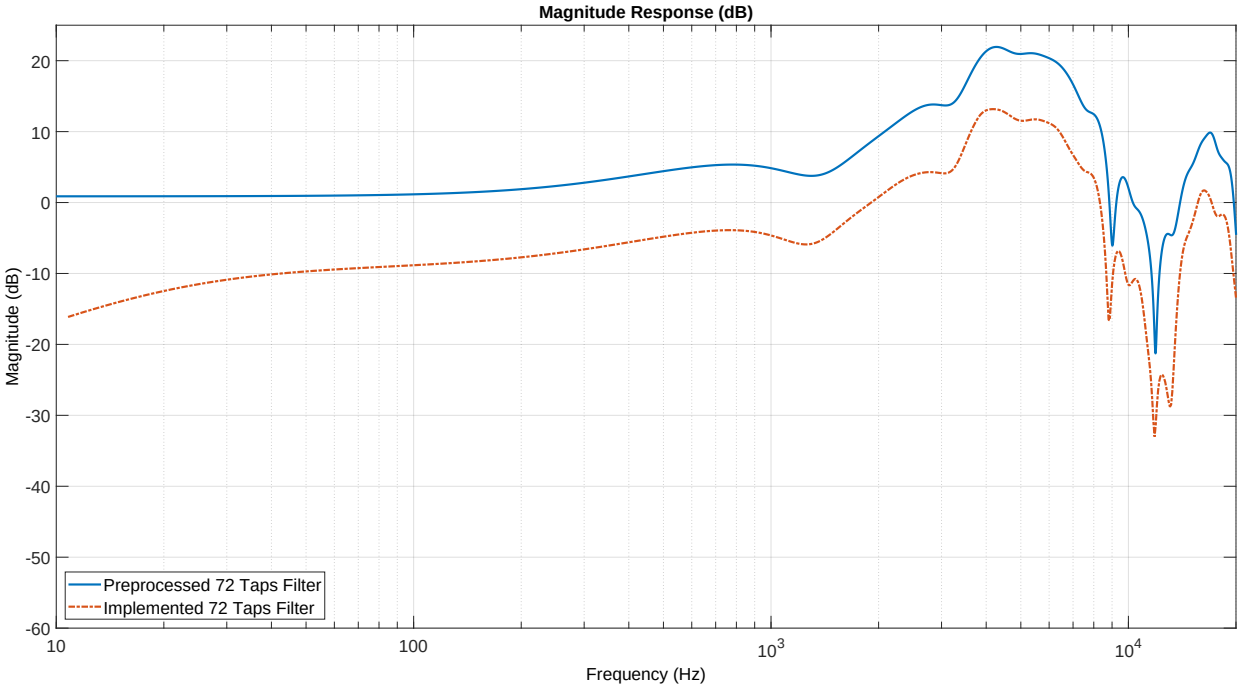


Figure M.3: Frequency response comparison between the measured HRTF on the DSP and the preprocessed for 45° azimuth and 0° elevation for the left ear.

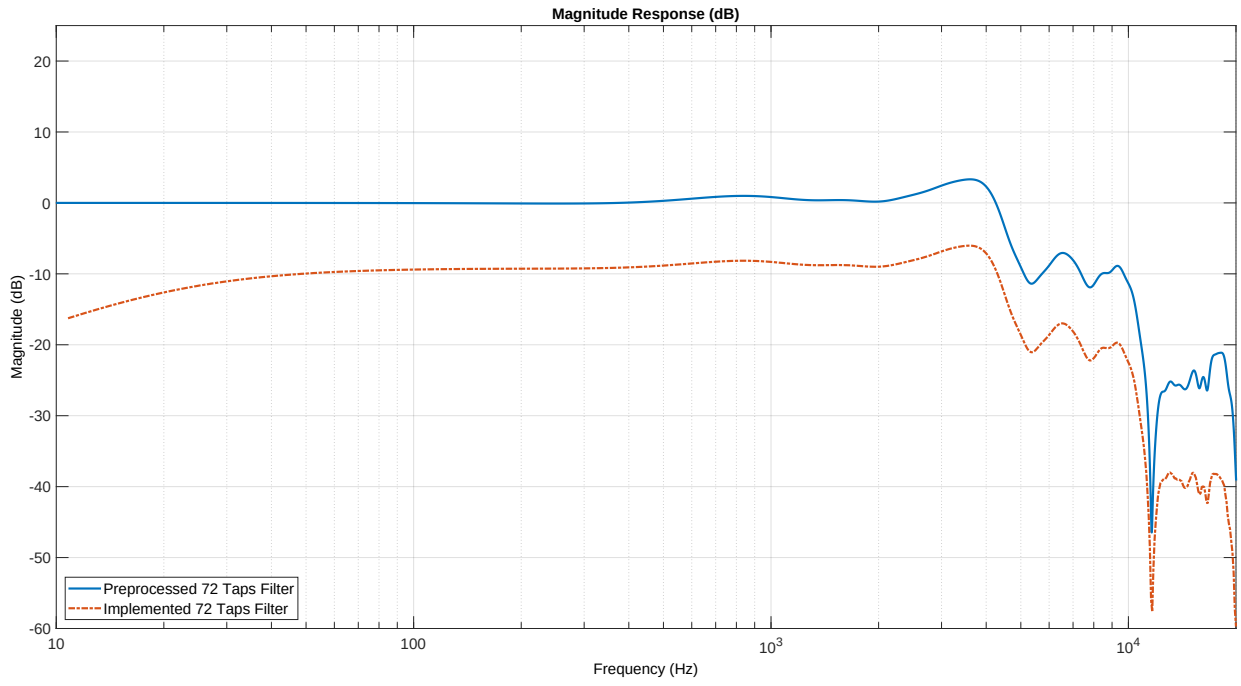


Figure M.4: Frequency response comparison between the measured HRTF on the DSP and the preprocessed for 90° azimuth and 0° elevation.

As it has been noted in appendix K, there is a fault in the implementation of the filter selection tool. Therefor the system is unable to filter frequencies above or below 0° elevation as can be seen on figure

M.5.

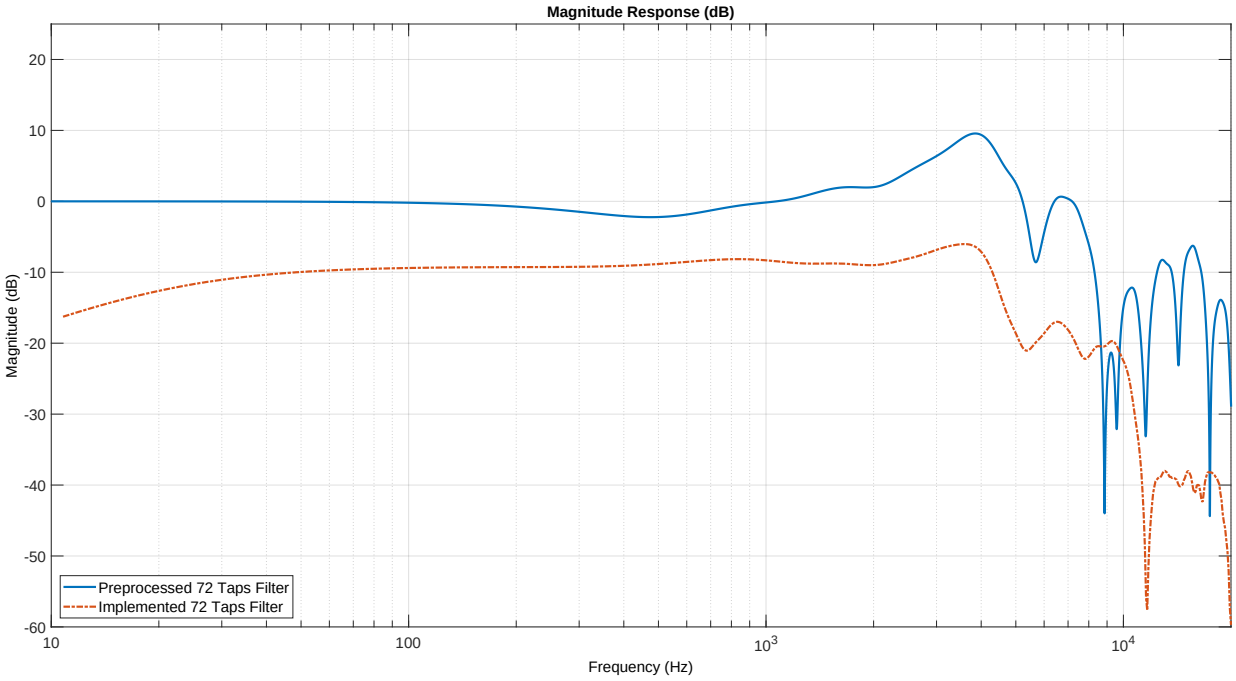


Figure M.5: Frequency response comparison between the measured HRTF on the DSP and the preprocessed for 45° azimuth and −45° elevation.

M.4 Sources of Error and other uncertainties

As it was already commented on, see section K.3 for discussion about the faulty elevation measurement in figure M.5.