TECHNISCHE FAKULTÄT DER
CHRISTIAN-ALBRECHTS-UNIVERSITÄT
ZU KIEL

Digital Signal Processing and System Theory
Prof. Dr.-Ing. Gerhard Schmidt

# Using Beamforming Techniques for EEG Source Analysis

Bachelor Thesis in Electrical Engineering and Information Engineering


written by

**Morten Stabenau**

supervised by

Prof. Dr.-Ing. Gerhard Schmidt (first supervisor)
M.Sc. Christin Baasch (second supervisor)

August 16th 2016

# Declaration

To the best of my knowledge and belief I hereby declare under penalty of perjury that this thesis, entitled

**Using Beamforming Techniques for EEG Source Analysis**

was prepared without aid from any other sources except where indicated. Any reference to material previously published by any other person has been duly acknowledged. This work contains no material which has been submitted or accepted for the award of any other degree in any institution.

I am aware that my thesis, developed under guidance, is part of the examination and may not be commercially used or transferred to a third party without written permission from my supervisory professor.

Kiel, 17.06.2016

_____

(Morten Stabenau)

# Contents

# Notation

Throughout this thesis the notation will follow some basic rules. These rules, all used signals and parameters, as well as the most relevant abbreviations and acronyms will be described in this chapter.

## Basic Rules

In order to make the reading of this thesis as comfortable as possible, the mathematical notation follows same basic rules that are listed here:

- Scalar quantities such as time domain signal are written in lower-case, non-bold letters such as $s(n)$ for a signal or $h_i(n)$ for an impulse response (the $i$-th coefficient of the impulse response at time-index $n$).
- Spectra are written in upper-case, non-bold letters such as $Y(e^{j\Omega})$.
- Vector quantities are written in bold, lower-case letters such as $\boldsymbol{r}$ for a position or $\boldsymbol{s}(n)$ for a multi-channel signal.
- Matrices are written in bold, upper-case letters such as $\boldsymbol{L}$ for the leadfield matrix.
- Number sets are written in blackbord bold, upper-case letters such as $\mathbb{N}$ for the set of natural number or $\mathbb{R}$ for the set of real numbers.

## Symbols

| Symbol | Meaning |
|---|---|
| $E\{\dots\}$ | Expected value |
| $\|\dots\|$ | Round to the closest integer |
| $N \in \mathbb{N}$ | Number of time/frequency samples in the current chunk |
| $M \in \mathbb{N}$ | Number of channels |
| $N_m \in \mathbb{N}$ | Number of chunks stored in the beamformer's memory |
| $P \in \mathbb{R}$ | Power |
| $\sigma \in \mathbb{R}$ | Conducitivity |
| $f_s \in \mathbb{R}$ | Sampling frequency |
| $\boldsymbol{d} \in \mathbb{R}^{3\times1}$ | Dipole vector |
| $\boldsymbol{r} \in \mathbb{R}^3$ | Position vector |
| $\boldsymbol{Y} \in \mathbb{C}^{1\times N}$ | Fourier transformed output signal |
| $\boldsymbol{S} \in \mathbb{C}^{N\times M}$ | Fourier transformed input signal |

| Symbol | Meaning |
|---|---|
| $\boldsymbol{w} \in \mathbb{C}^{M \times 1}$ | Beamformer weight vector |
| $\boldsymbol{c} \in \mathbb{R}^{M \times 1}$ | Predicted potential at each electrode for a dipole at the point of interest |
| $\boldsymbol{R_{ss}} \in \mathbb{C}^{M \times M}$ | Covariance matrix of the signal |
| $\lambda \in \mathbb{R}$ | Lagrange multiplier |
| $\boldsymbol{V} \in \mathbb{C}^{M \times N_m}$ | Beamformer memory |
| $\alpha \in \mathbb{R}$ | Beamformer stability parameter |
| $\boldsymbol{L} \in \mathbb{R}^{M \times 3}$ | Lead field matrix |
| $P_n(a) \in \mathbb{R}$ | n-th degree Legendre polynomial |
| $P_n^m(a) \in \mathbb{R}$ | n-th degree, m-th order associated Legendre polynomial |

Table 0.1: Symbols

## Abbreviations

| Abbreviation | Meaning |
|---|---|
| EEG | Electroencephalography |
| MEG | Magnetoencephalography |
| fMRI | Functional magnetic resonance imaging |
| EMG | Electromyogram |
| ECG | Electrocardiogram |
| DFT | Discrete Fourier transform |
| FFT | Fast Fourier Transform |
| SNR | Signal to noise ratio |
| PSD | Power spectral density |
| ATP | Adenosine triphosphate |
| GUI | Graphical user interface |
| CSF | Cerebrospinal fluid |

Table 0.2: Abbreviations

# 1 Introduction

Nowadays, one of the most important tools for neurological research is the surface electroencephalography (EEG). It is still the easiest and cheapest way to measure brain activity compared to magnetoencephalography (MEG) and functional magnetic resonance imaging (fMRI). Both of the latter systems require a high cost and effort both during the initial setup of the machines and during the measurements. Functional MRI machines also inherently have the disadvantages of a low temporal resolution (on the order of one second versus milliseconds or lower for the EEG) and only being able to sense brain activity indirectly through the neurons oxygen consumption. In contrast, measuring EEG signals can be done with relatively cheap equipment and little setup time for each patient. The main drawback is that it can only provide surface information, whereas the fMRI can also provide depth data.

Figure 1.1: Patients undergoing a MEG (left) and MRI (right)

Source: Oxford Centre for Human Brain Activity[17](left), Universitätsklinikum Heidelberg[18]

Modern signal processing can mitigate this disadvantage by performing source analysis of EEG data either in real-time or offline after the data was recorded. This process tries to reconstruct spatial information from the surface EEG data by estimating the location and strength of the source necessary to produce the observed EEG voltages. The estimation is always ambiguous, because an infinite number of source configuration could result in the same surface EEG. The problem of having to select one of these solutions is called the inverse problem.

In this particular work, the inverse problem is solved by applying a beamforming algorithm in the frequency domain to the data before transforming the data back into the time domain. For the beamformer to work, it is first necessary to ascertain how each point in the brain effects each EEG electrode. This is called the forward problem.

## 1.1 Goal of this work

The goal of this work is to develop an algorithm for extracting 3D information from a given set of EEG data. This algorithm could then be used for research or treatment. For example, it could be useful as an additional method for localizing epilepsy foci or for researching short term epilepsy seizures that happen too fast to be observed with other methods.

In this work, the algorithm will be developed and tested with EEG data recorded during finger tapping experiments (see 1.3). It will be evaluated by trying to detect the signs of finger tapping in the brain as they are described in literature ([7] and [8]).

## 1.2 Basics of electroencephalography

An electroencephalogram is created by placing electrodes on the patient's scalp and recording the voltage between the electrodes and a reference electrode. The electrodes are usually covered in a conductive fluid or gel to improve the conductivity between the scalp and the electrode. It is necessary to use either a physical reference electrode (one possibility is to attach them to the earlobes or a distant ground potential) or a virtual one (for example using the average potential of all electrodes) to measure the voltages.

Using an earlobe as reference is advantageous, because it is also affected by noise sources from within the body (for example the heartbeat or muscle movements) and outside the body (like a mains hum from a nearby device). These influences are filtered out by measuring the voltage between the earlobe and the electrodes, because any potential generated by noise sources will be equally present at the earlobe and the EEG electrode. Since the measured voltage is different between both electrodes, the noise voltage is cancelled out. However, using the earlobe as a reference has the disadvantage that it is also affected by the wanted brain signals. The hemisphere closer to the earlobe influences it more, leading to an imbalance in signal detection between both hemispheres.

Another method would be using the average of all electrodes as a reference. This also filters out signals that equally affect all electrodes, but does not require a physical electrode and generates no imbalance between the two hemispheres.

After disturbances and artefacts have been cleaned away, the EEG electrodes measure potentials created by groups of neurons in the brain. The measurements are dampened and distorted by the tissues between the neurons and the electrodes, for example the skin, the skull and the cerebrospinal fluid. My models will have to account for these tissues later (see 3).

The following section describes the process of nerve conduction and its effect on the EEG electrodes.

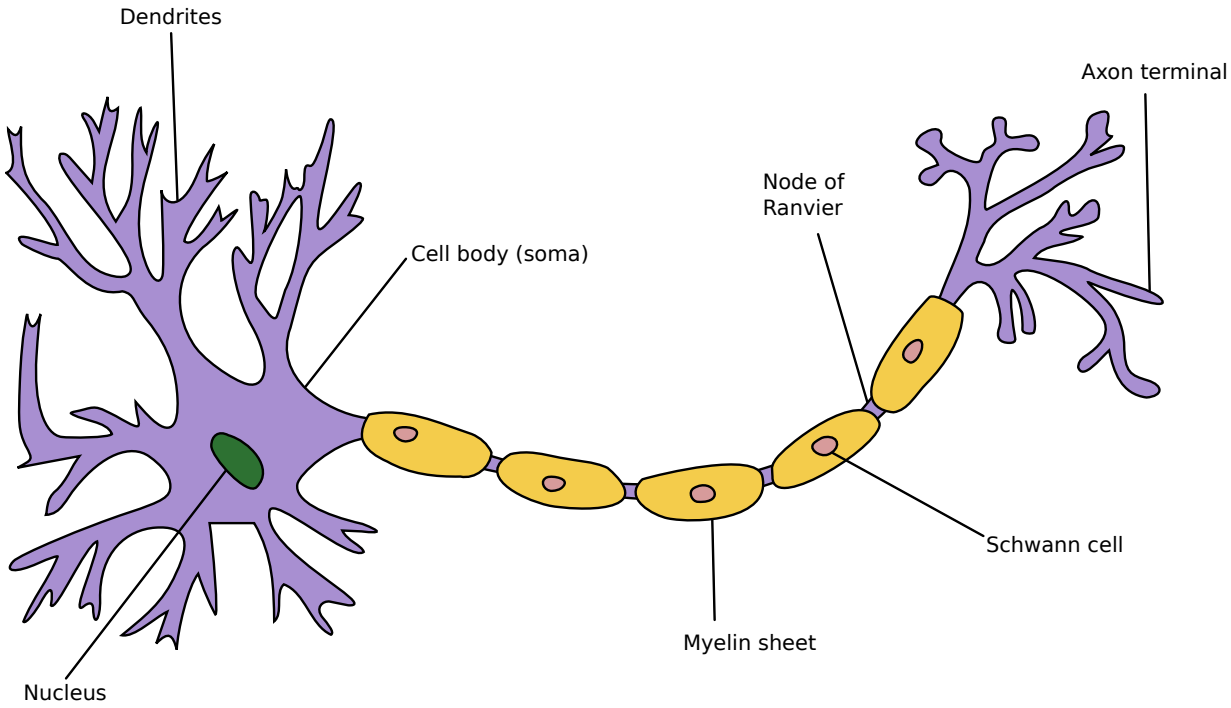


Figure 1.2: Schematic structure of a neuron

Source: Wikimedia Commons[11], edited

Nerve cells (neurons) are made of three main parts:

- Cell body (soma): This is where the majority of the biological processes happen. It contains for example the nucleus, which holds the genetic information, and the mitochondria, which generate ATP from sugar. ATP (adenosine triphosphate) is used as an energy source for various cellular processes, including the active ion channels necessary for the nerve conduction.
- Axon: carries signals to muscles or to other nerve cells. The axon might split up near its target to reach multiple other neurons or muscles. In vertebrates (including humans) the axon is surrounded by a thick myelin sheet, which insulates it from the other tissues. The myelin sheet is interrupted by short necks (about 1 µm every 1 mm) called Nodes of Ranvier.
- Dendrites: connect to other nerve cells and act as inputs to the nerve cell. Other nerve cells' axons connect here via synapses, which release neurotransmitters to affect the neuron's potential.

A neuron and its axon permanently keep a potential difference about $-70$ mV (measured from the inside to the outside) called the rest potential. This potential has to be actively maintained by the cell by pumping negative ions in or positive ones out. Electrochemical inputs at the dendrites can change this voltage. If the voltage rises above a certain voltage (called the threshold potential, $-60$ mV in the graphic below), voltage-sensitive ion channels open. These carry positive ions like $Na^+$ into the cell, which depolarises it, putting its voltage above 0 mV.

After 1-2 ms, the sodium channels ion channels close and can not be reopened until the cell

is charged negatively again. Another set of ion channels transport $K^+$ ions out of the cell, thereby returning the potential to a level a little lower than the rest potential. Subsequently, the balance of ions is restored during the refractory period. While this period lasts, the cell can not be activated again. This whole process produces the characteristic spike called the action potential.

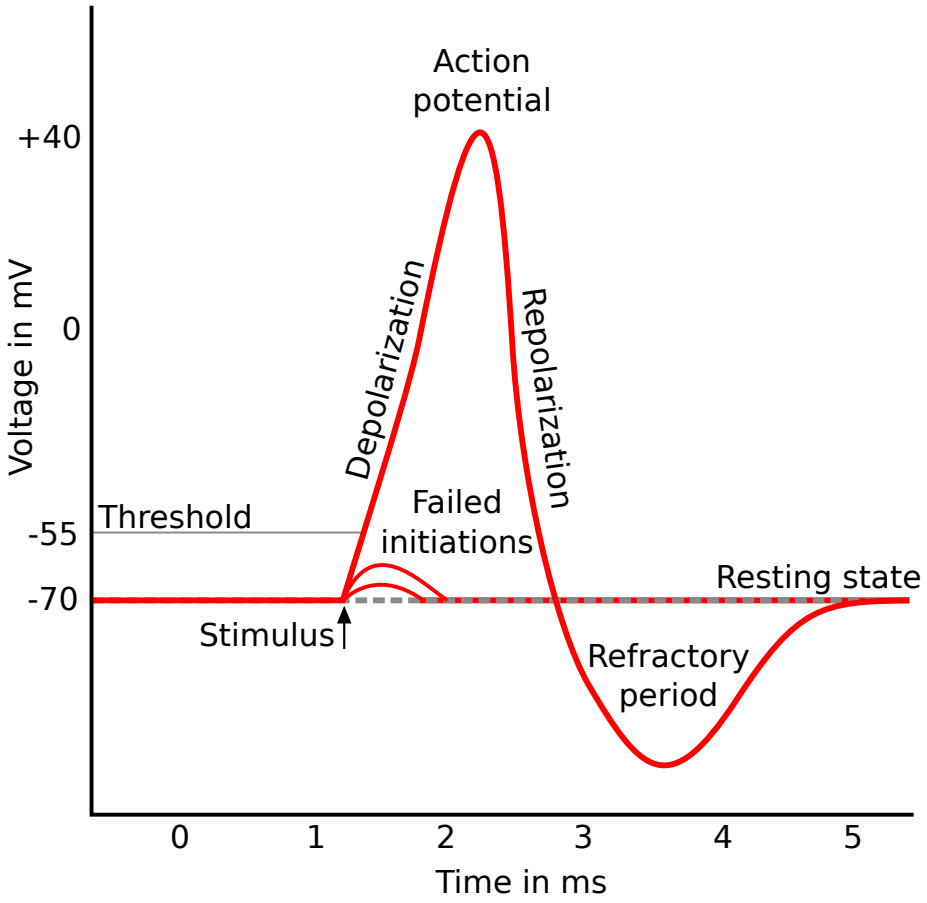


Figure 1.3: Voltage progression during an action potential

Source: Wikimedia Commons[15]

The change in charge due to the initial influx of $Na^+$ influences the charge at the first Node of Ranvier. There, the voltage depolarises from the resting potential above the threshold potential, which in turn opens the ion channels at the node. At each node the ion channels are a lot more concentrated than on the rest of the axon. The influx of ions in turn influences the next node, which continues the conduction of the signal. In effect, the signal seems to jump from one Node of Ranvier to the next, which is why this process is called saltatory conduction (from the Latin word saltare meaning jump).

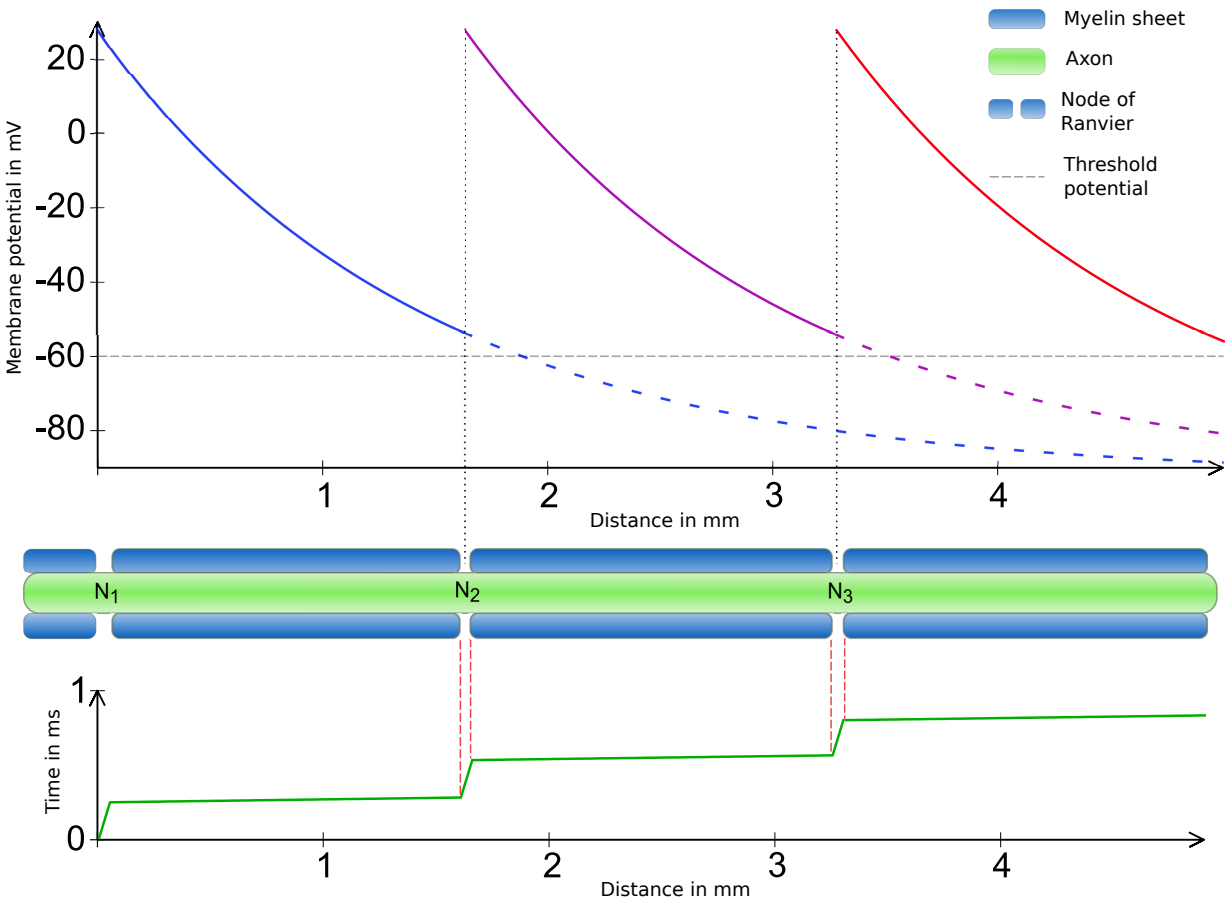


Figure 1.4: Saltatory conduction

Source: Wikimedia Commons[13], translated

With this process, myelin sheets enable vertebrates to have superior cognition speeds compared to non-vertebrates. Without the myelin sheets and saltatory conduction, the signal transfer speed is mainly dependent on the diameter of the axon. To enable similarly high cognition, one would require impossibly thick axons. Humans can contract multiple neurological diseases like multiple sclerosis and Guillain–Barré syndrome that destroy the myelin sheet.

The influx of ions can not be measured on the outside as it is cylindrically symmetrical and the ions on the outside shield the charges on the inside. However, there is also a current in the direction of the axon, because of the indirect influence of a depolarized node on a resting node. This current produces an electric field that is theoretically measurable outside the neuron.

Normally, the field of the ionic current within a single axon is too weak to detect. Additionally, many millions of neurons might affect a single electrode on the scalp, leading to the voltages cancelling out at the electrode. Only if large groups of neurons fire at the same time in the same direction, voltages can be measured on the scalp. Nevertheless, the voltages are tiny, typically in the range of a few 10 µV. They require sophisticated amplifiers to make them measurable.

One usually detects voltages when the area near the electrode is in a relative resting state. When the area is processing a lot of information the neurons tend to fire into different directions at seemingly random times. While a group of neurons is resting they tend to fall into regular

rhythms called neural oscillations, which can be measured by using an EEG device.

Different rhythms are often grouped into wave bands. Although there is no agreed-upon definition of these wave bands, this work will use the following definitions [14]:

- Alpha: 8-13 Hz
- Beta: 16-31 Hz
- Theta: 4-7 Hz
- Delta: 1-3 Hz

Below, one can see an example of an alpha oscillation. It was recorded at the O2 electrode, which is placed on the lower right back of the head (for more information on the naming conventions of electrodes, see 1.3).

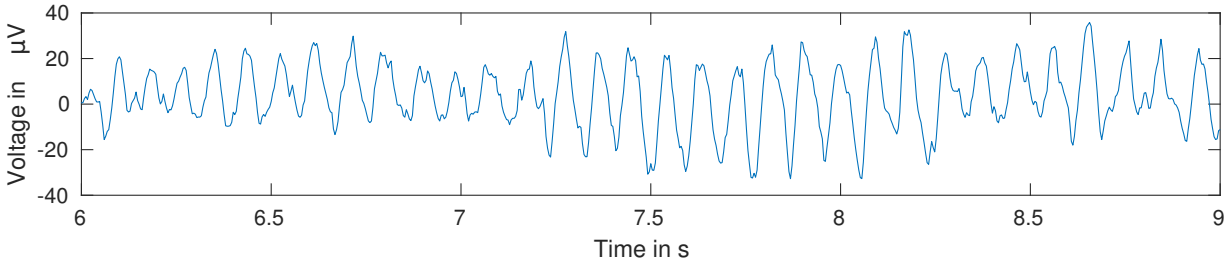


Figure 1.5: Example of an alpha wave, voltage recorded at the O2 electrode

## 1.3 Provided data

I was provided with previously recorded data for this work by Professor Gerhard Schmidt. The EEG data was recorded during an fMRI (which can be used for comparison) and multiple additional electrodes where placed on the patient's body. Those electrodes included a electrocardiogram (ECG) and multiple electromyograms at both hands and the chin.

The data I received was recorded in nine sessions of about ten minutes each with different patients over multiple months in 2008. The patients were told to periodically tap their thumb against their forefinger. They alternated between tapping on both hands, tapping on one hand and not tapping at all.

The EEG that was used has 30 electrodes integrated on a rubber cap, which places the electrodes approximately in the right position relative to one another on the patient's head. The cap has to be carefully adjusted, since the electrodes must be at the same position for each patient as defined by the international 10-10-system. Each electrode has a short alphanumeric label signifying where it is supposed to be located. For example, the CP5 electrode has to be placed centrally above the parietal brain lobe.

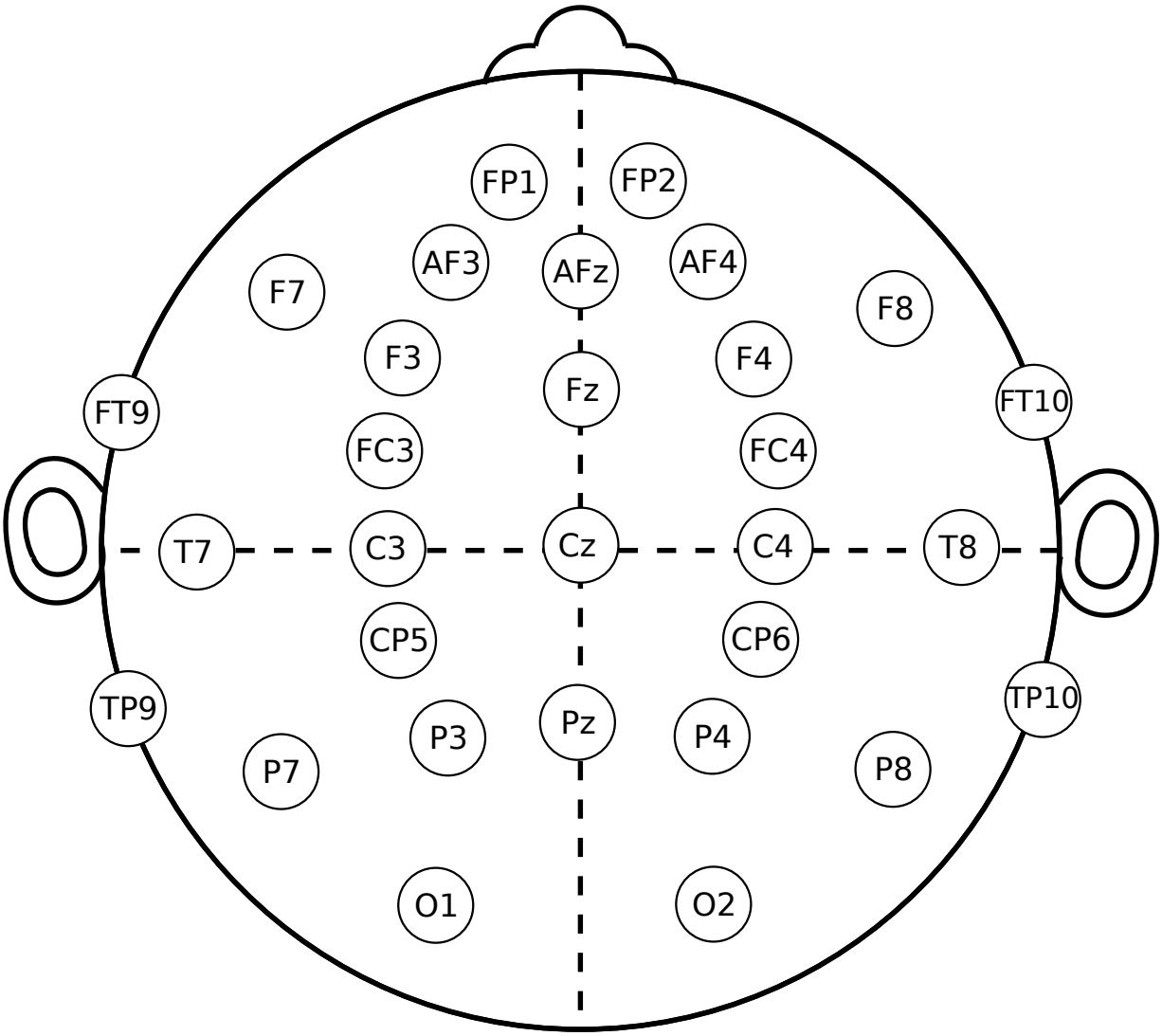


Figure 1.6: Electrode placement according to the international 10-10-system

Source: Wikimedia Commons[16], edited and extended

The data itself consists of a header file, a data file and multiple files unused in this work (including a marker file) for each session. The header file contains meta information about the recorded data, including:

- the location of the data and marker file,
- the data orientation (whether one block in the data file contains the whole time for one sensor or one time point for each sensor),
- the number of channels and data points,
- the sampling interval in microseconds (this can be converted to the sampling rate $f_s$),
- the label and measurement unit for each channel,

- the position of the EEG electrodes (just nominal positions, not depending on the patient).

The data file contains the binary voltage data, which has been preprocessed to remove the artefacts caused by the MRI machine. Since the MRI scanner works with extremely powerful, rapidly alternating magnetic fields, it induces strong voltages in the EEG electrodes. Those have a higher amplitude than the EEG voltages by multiple magnitudes. The following paragraphs outline the artefact correction algorithm, which was not part of this work.

The artefact correction algorithm leverages the fact that the MRI pulses are highly predictable and that the EEG data is added on top of these pulses. After the beginning of each pulse has been exactly marked by a detection algorithm, multiple pulses are averaged with one another. This way all EEG data averages out, since it has a mean of zero. The resulting average pulse is used as a template and subtracted from the original data at each marker point. The remaining data points are the EEG voltages.

Even with this algorithm, one needs special equipment to record an EEG in parallel with an fMRI. For instance, the amplifier has to have a high range to avoid going into saturation during the pulses, but still a high sensitivity to detect the desired EEG voltages. During these measurements, the digitizer was also externally triggered from an output of the MRI machine to keep the sampling points in synch with the MRI pulses and minimize timing jitter. The EEG had to be recorded at a high sampling frequency to accurately display the high gradient of the MRI pulses. After the artefact correction, the data was sampled down after the preprocessing to $f_s = 250$ Hz.

# 2 Overview over the algorithm

This chapter gives an overview over the algorithm and the methods used to implement it. Additionally, the methods for importing the data are explained.

The algorithm was implemented in MATLAB, which is a commercial numerical mathematics software developed by MathWorks. MATLAB is heavily used in research, engineering and economics. The software is used to develop and execute programs written in the programming language of the same name.

My algorithm depends on two MATLAB toolboxes apart from the MATLAB base program, the *Signal Processing Toolbox* and the *Statistics and Machine Learning Toolbox*. Those need to be bought and installed in addition to the MATLAB base installation. The program was developed in MATLAB 2015a and also tested in MATLAB 2016a on Microsoft Windows, Ubuntu and Arch Linux.

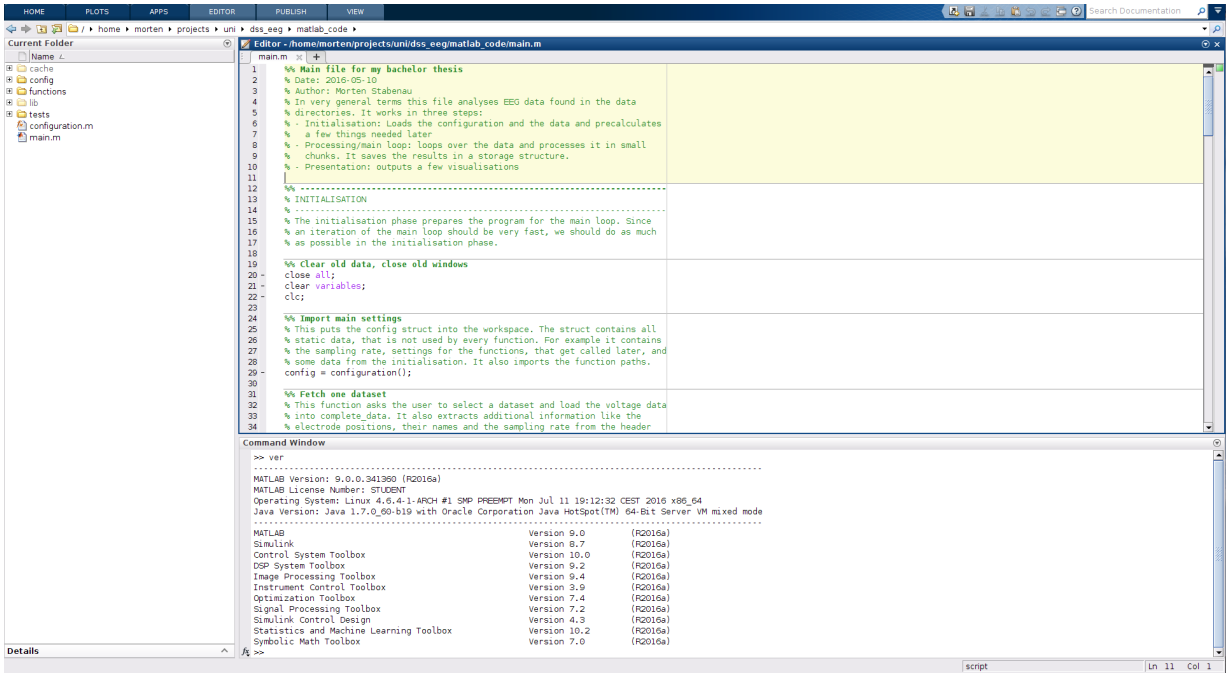


Figure 2.1: Example Screenshot of MATLAB 2016a

In general, the algorithm can be divided into three steps:

- Initialisation: loads the configuration, the data and calculate the forward model. Also loads the cache (see 2.3) if the user chooses to.
- Processing: the data is split into short overlapping chunks and then transformed into the frequency domain using the filter bank (see 4.1). Thereupon, the MVDR beamformer (see 5) is applied before transforming the data back into the time domain using the inverse filter bank (see 4.2).
- Presentation: the data is displayed using five different visualisations (see 6).

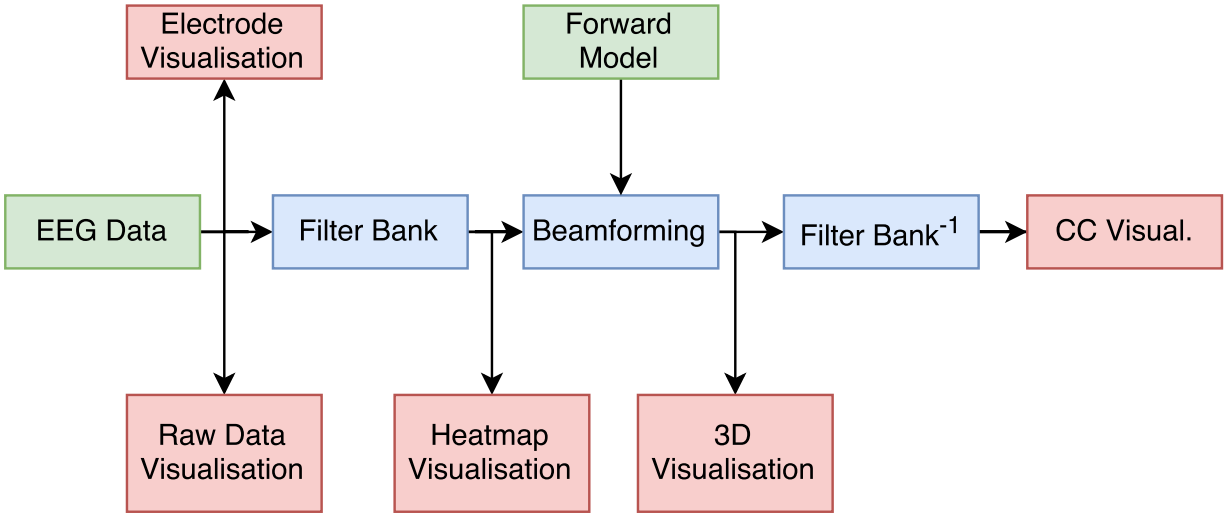


Figure 2.2: Data flow through the software

Above, we can see the data flow through the algorithm. The initialisation is highlighted in green, the processing step in blue and the presentation step in red. Please note that this is not the right temporal sequence. The graph does not reflect some recent changes of the raw data visualisation (see 6.2), which would have made the graph too confusing.

In the processing step, the data is split into small chunks (about a second long), which are then put through the processing pipeline individually. This is done in order to simulate the same requirements on the algorithm as if it had to be implemented for a real time application. Algorithms implemented in this pseudo-real-time fashion are usually more efficient both in terms of memory and of processing time. It is also easier to port them to a real time environment if this is ever desired (see 8).

The initialisation and presentation steps are not done in pseudo-real-time. For the initialisation, this would not have been possible and I decided not to do this for the presentation step. Some visualisations in the presentation step require access to the full data to properly display anything. For example, the heat map visualisation (see 6.3) requires the maximum power spectral density of the signal in each band beforehand to correctly scale the displayed colours.

## 2.1 Configuration

The program contains a comprehensive configuration in the form of multiple MATLAB files. Those can be used to change the behaviour of the software, to fine tune parameters and to disable certain features. The configuration is split up into numerous sections, each one corresponding to a component of the program:

- configuration: main configuration file includes all sub files and configures basic options like the function paths, the profiler (a MATLAB utility for performance measurements) and switches to disable single visualisations

- config_load_data: contains a switch to enable or disable the file selection GUI (see 2.2) and which file to load if the GUI is disabled.
- config_cache: contains a switch to disable the cache (see 2.3) and the folder in which to save the cache files
- config_forward: contains the configuration of the forward model (see 3), including the conductivities and radii of the brain model and some numerical parameters
- config_processing: contains the resolution of the beamformer targets, the length and overlap of the chunks, the low-pass frequency of the filter bank and some additional settings for the beamformer
- config_brain_model: contains the configuration for the 3D brain model used in multiple visualisations (see 6.4 and 6.1). Here, one can change the colour of the different lobes, the transparencies and the scale and the positioning of the model
- config_analysis: contains the wave bands as defined in 1.2
- config_test: contains debug options and a setting to limit the amount of data processed. This is also useful for testing purposes since the analysis takes multiple minutes to complete a full data set.
- a configuration for each visualisation

The configuration is stored in a structure, which is passed to most functions. Some methods (e.g. the data loading) also add new options to the configuration.

## 2.2 Loading the data

In the first step, the user is asked to select a data set for analysis using a graphical user interface (GUI). If the GUI has been disabled in the configuration, the program selects a default data set from the configuration.

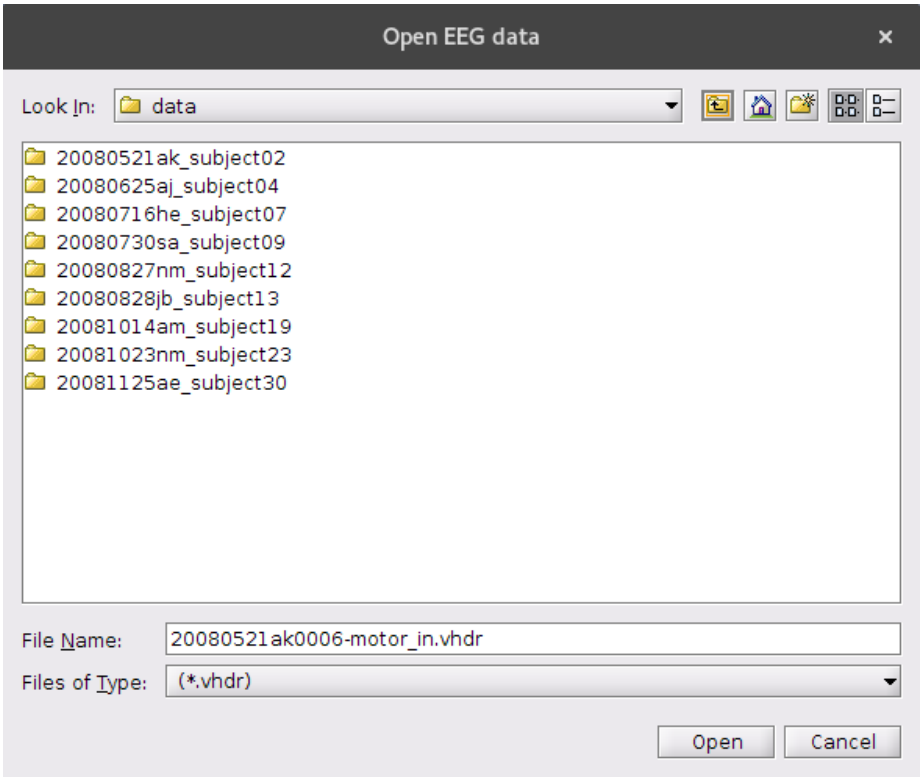


Figure 2.3: GUI for selecting the data set

Then the header file is opened and parsed completely into a structure. From this structure, the electrode information like labels and positions is extracted. The electrode positions are stored in spherical coordinates and have to be converted to Cartesian coordinates first. Interestingly, the EEG-device that stores these header files uses a different definition for spherical coordinates compared to MATLAB:

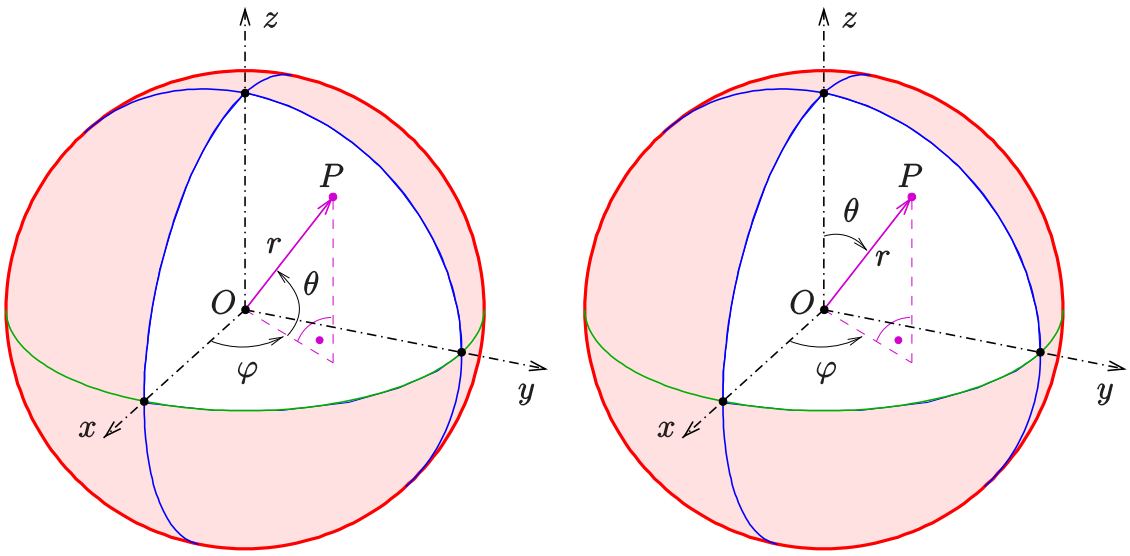


Figure 2.4: Different definitions of spherical coordinates: MATLAB (left) and EEG (right)

Source: Wikimedia Commons, left one was edited

Note that the angle $\theta$ is defined vice versa when comparing both definitions. The transformation into spherical coordinates using the definition form the EEG header file works as follows:

$$\begin{aligned} x &= r \sin\theta \cos\phi \\ y &= r \sin\theta \sin\phi \\ z &= r \cos\theta \end{aligned} \tag{2.1}$$

For the EEG electrodes $r \equiv 1$, meaning that they are supposedly located on a perfect sphere with a radius of one. On the other hand, the EMG electrodes have no position, but a $r$, $\theta$ and $\phi$ are always zero. This is used to distinguish EEG from EMG electrodes. It would have been possible to just select the first 30 channels as EEG, but since an EEG with a different number of channels might be used in the future, the former method was implemented. The EEG electrodes' positions are scaled by the head models outer radius($r_4$, see 3.2) to convert them into centimetres.

After the header information has been read and stored, the data file is read using a small function written by Helmut Laufs. This function returns the electrode voltage data measured in µV as a MATLAB array.

Consequently, the EEG and EMG data are split into two separate arrays so that only EEG data can be put through the processing pipeline later. The electrode meta information (positions, labels and whether it is an EEG or an MEG electrode) is saved in the configuration structure. The sampling interval is extracted from the header structure and converted into the sampling rate $f_s$.

With this information, the chunk length can be calculated and stored in the configuration structure. The chunk length $T$ in the configuration is measured in seconds, which have to be converted into a number of data points using the sampling rate. The chunk length $N$ then is rounded to the closest power of two to make the Fast Fourier transformation (see 4.1) more efficient:

$$N = 2^{\| \log_2 T f_s \|} \tag{2.2}$$

Additionally, the function stores the chunk overlap (see 4), the data file name and the Hann-window function with a length of $N$ (see 4.1) in the configuration structure.

Because the processing pipeline has a delay (see 5.2) and can only process full chunks of data, the signal (both EEG and EMG) is padded with a number of zeros. This assures that the full data set is used.

## 2.3 Caching

Since running the whole program takes a long time (~30 minutes on my machine for 10 minutes of data), it is necessary to store the result data on disk. This way the data is kept intact even when MATLAB is closed or the computer is restarted.

When the program finishes its processing step the data is saved in a .mat file in the cache folder. This data includes everything in the storage structure (the spectral data after the filter banks, the data after the beamforming and the final time domain data) and some configuration options:

- the electrode information,
- the generated grid of beamformer targets (see 3),
- the sampling rate,
- the start and end position of each chunk,
- how many zeros were added to the end of the data (see 5.2).

These configuration options need to be stored so that the visualisations can be safely executed at the end of the program, as the options in the configuration file might have changed since the cache file was generated. The name of the cache .mat file is the name of the header file plus the different extension.

Now every time the program is rerun it checks if a cache file exists after a data set has been selected. If a cache file has been found, the user is asked whether to use it:

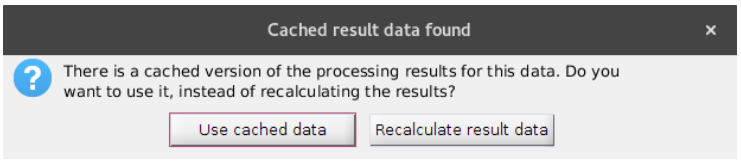


Figure 2.5: Question to the user concerning the cache

If the user chooses to load the file, part of the initialisation phase and the whole processing phase is skipped. The program still takes a noticeable time to load the data, since the files are about 700 MB in size for 10 minutes of data. The cache files can be deleted without any repercussions.

# 3 Forward Modelling

To accurately predict the location of source in the brain, one has to model how said source affects the EEG electrodes. This is called the forward model of the brain. A common method to do this is to assume that the sources are ideal current dipoles with no spatial extent[4]. The sources then have two parameters: their position $\boldsymbol{r}$ and their dipole vector $\boldsymbol{d}$, which contains their orientation and strength.

The ideal dipole field of the source is dampened and distorted by the surrounding tissues, including the brain matter, the skull and the scalp. One way to model these tissues is to assume that the head and the brain are approximately spherical. One can now imagine multiple concentric spheres with uniform electric properties to represent different tissues. This way the voltages on the scalp can be analytically computed using field theory.

Evidently, this approach ignores the complex geometry and distribution of different tissues in the brain. Unfortunately, it is very hard to accurately model this, as it would require a 3D model of different tissues of each patient or if that is impossible, a sort of average brain for all the patients. Then one could compute the surface potentials using numeric methods like finite element modelling. This method is out of the scope of this work due to its complexity.

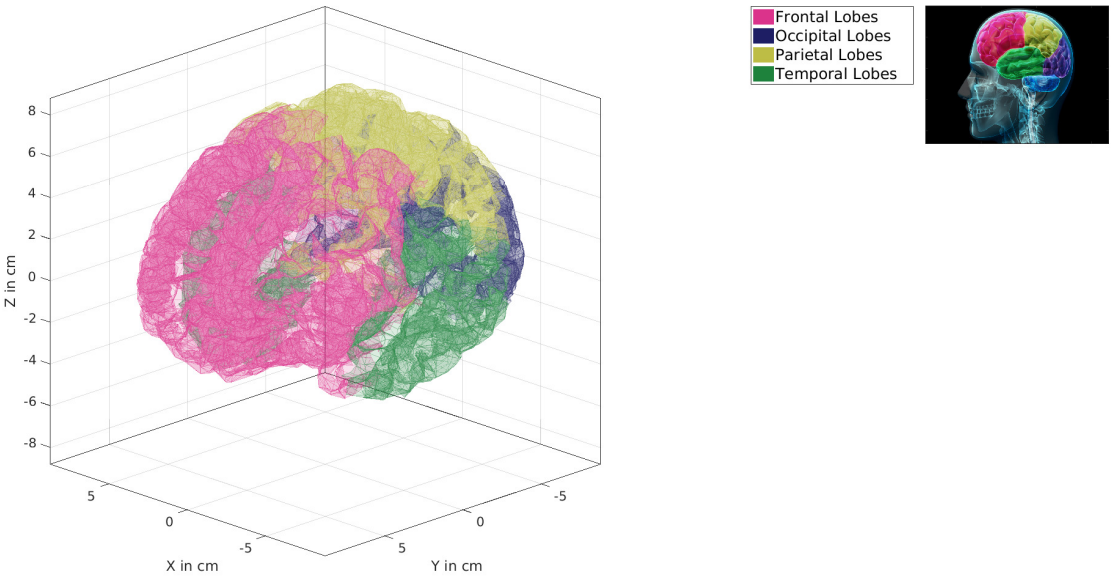


Figure 3.1: A 3D model of the brain, displaying the geometric complexity of the brain

Regardless, of the method the resulting voltages of the forward model are usually arranged in a specific form called a *lead field matrix*. Each column of the matrix contains the potentials at the electrode positions for a unit dipole in one of the three main directions (X, Y and Z). For example, a resulting lead field matrix might look like this:

$$\boldsymbol{L} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{M \times 3} \tag{3.1}$$

This matrix contains the voltages for five electrodes. The first and fourth are not affected by dipoles at the current location. The second and fifth are only affected by the Y and X directions respectively. The third electrode is equally, yet weakly affected by either direction. The advantage of that arrangement is that one can easily calculate the resulting voltages by multiplying the matrix with the dipole vector $\boldsymbol{d}$:

$$\boldsymbol{c} = \boldsymbol{L}\boldsymbol{d} \tag{3.2}$$

Here, $\boldsymbol{c}$ refers to the predicted potential from the forward model.

The lead field matrix depends on the location of the selected point and might additionally depend on the frequency of the source if one chooses to model the dispersion of the different tissues. It is generally not useful to make the lead field matrix complex for EEG applications, because the electric signals in the brain propagate close to the speed of light. This makes the phase differences between electrodes negligible. Furthermore, the interesting frequencies in the brain are quite low (usually lower than 50 Hz), which results in very long wave lengths further diminishing the phase differences.

To receive voltages from these potentials, one has to know which kind of reference (see 1.2) was used to measure the EEG. If, for example, a distant reference was used, the potentials could be directly used as voltages, as the distant reference electrodes potential can be assumed to be permanently zero. For other methods, additional considerations have to be made (see 3.2).

It is also necessary to select a set of points within the brain that serve as beamformer targets. For these targets, the lead field matrix will be determined. For this work, a simple approach of a regular 3D grid truncated by the innermost sphere was chosen. The distance on one axis between two targets was set to $d = 3$ cm. The results can be seen in figure 6.13.

## 3.1 Three Spheres Model

Since numerical models are too complex for this work, I chose to implement the simplest model that could produce somewhat realistic results: a model with three concentric spheres as described by Salu et al.[3]. The three spheres represent the brain, the skull and the scalp:

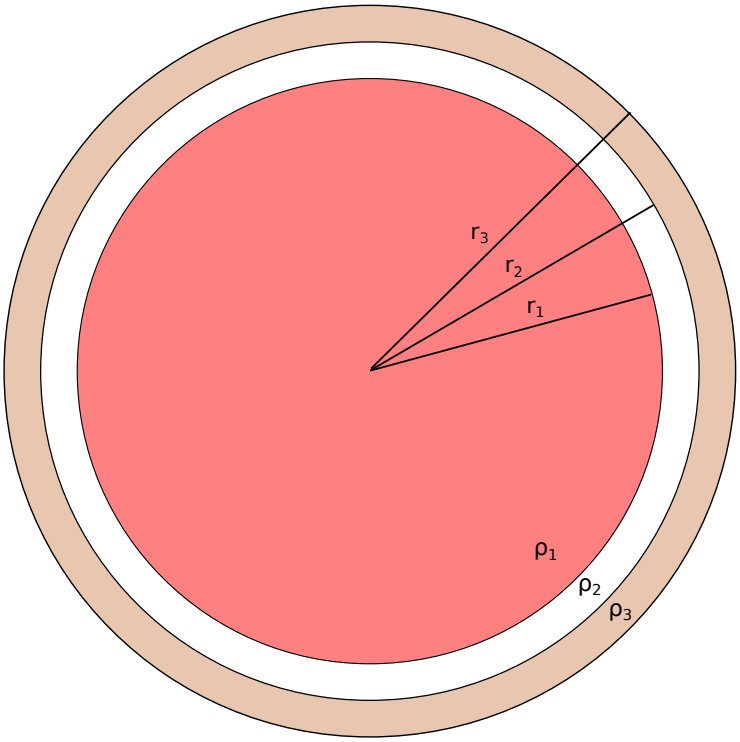


Figure 3.2: An outline of the three spheres model (not to scale)

The innermost region represents the brain, middle region the skull and outer region the scalp.

The model contained a total of five parameters: the three radii of the spheres $r_1, r_2, r_3$ and the conductivity for the soft tissue $\sigma_1 = \sigma_3$ and for the skull $\sigma_2$. The brain and the scalp are assumed to have the same conductivity [3] [2].

The potentials on the scalp at $\boldsymbol{r}_s$ are generated by a unit dipole at $\boldsymbol{r}_d$ and can be calculated by:

$$v_a = \frac{1}{4\pi\sigma_1 r_3^2} \sum_{n=1}^{\infty} \left[ \frac{\sigma_2(2n+1)}{\sigma_1 d_n (n+1) n} \left( \frac{|\boldsymbol{r}_d|}{r_3} \right)^{n-1} \left( n \frac{\boldsymbol{r}_d \boldsymbol{e}_a}{|\boldsymbol{r}_d|} P_n(\cos(\alpha)) + T_a P_n^1(\cos(\alpha)) \right) \right] \tag{3.3}$$

In this equation, $a \in \{1, 2, 3\}$ and represents the direction the dipole is in (X, Y, or Z), $P_n$ and $P_n^1$ are the Legendre polynomial and associated Legendre polynomial of the first order respectively. $e_a$ is the unit vector in the direction $a$, $T_a$ is given by the following equation:

$$T_a = \frac{\boldsymbol{t} e_a}{|\boldsymbol{t}|} \quad \text{with} \quad \boldsymbol{t} = \boldsymbol{r}_s |\boldsymbol{r}_d|^2 - \boldsymbol{r}_d (\boldsymbol{r}_d \boldsymbol{r}_s) \tag{3.4}$$

$\cos(\alpha)$ is the angle between the position of the electrode and the position of the dipole. It can be calculated by:

$$\cos(\alpha) = \frac{\boldsymbol{r}_d \boldsymbol{r}_s}{|\boldsymbol{r}_s||\boldsymbol{r}_d|} \tag{3.5}$$

Finally, $d_n$ can be obtained with this equation:

$$d_n = [(n+1)X+n]\left[\frac{nX}{n+1}+1\right] + (1-X)[(n+1)X+n](f_1^{2n+1} - f_2^{2n+1}) - n(1-X)^2\left(\frac{f_1}{f_2}\right)^{2n+1}$$
$$\text{with} \quad f_1 = \frac{r_1}{r_3}, \quad f_2 = \frac{r_2}{r_3} \quad \text{and} \quad X = \frac{\sigma_2}{\sigma_1} \tag{3.6}$$

Unfortunately, this model has some drawbacks. First of all, in equation 3.3 one can see in one of the middle terms $|\boldsymbol{r}_d|$ gets divided by $r_3$. If $|\boldsymbol{r}_d| = 0$, meaning the dipole is located in the middle of the head, the whole equation turns zero.

Moreover, multiple tests (see 3.5) proved that this model is not realistic. This might be due to an implementation error on my part or an error in the paper.

## 3.2 Four Spheres Model

Since the first model did not work sufficiently, I looked for different approaches regarding forward model simulations. I looked at the source code of FieldTrip[9], an open-source MATLAB toolbox for EEG and MEG analysis developed by the Donders Institute for Brain, Cognition and Behaviour. The toolbox implements the three spheres model by using a four shell model and setting one of the radii to zero. The four shell model in FieldTrip is based on the habilitation work by B. Lütkenhöner from 1992, to which I unfortunately do not have access.

However, FieldTrip still had some unused source code for the implementation of the same model based on a different paper by B. Neil Cuffin et al. from 1979 [2]. I installed the toolbox and tested if both implementations would yield identical results. This was done by re-enabling the source code based on Cuffins implementation and testing for:

$$\frac{E\{\boldsymbol{L}_1 - \boldsymbol{L}_2\}}{E\{|\frac{\boldsymbol{L}_1+\boldsymbol{L}_2}{2}|\}} < 10^{-3} \tag{3.7}$$

Here, $\boldsymbol{L}_1$ is the matrix generated by the Cuffin et al. implementation and $\boldsymbol{L}_2$ is generated by the Lütkenhöner implementation. This test was done for each beamformer target and proved that in fact both implementations result in the same matrices for my parameters. I presume that the FieldTrip developers chose the newer implementation for performance reasons.

Hence, I based my forward model on the four shell model as proposed by Cuffin et al. It would have been possible to use the equations to provide a three sphere model, but other papers[5] suggest that modeling the cerebrospinal fluid (CSF) as a fourth sphere between the brain and the skull provides massive benefits in terms of accuracy of the forward model. This is why I settled for a four sphere model:

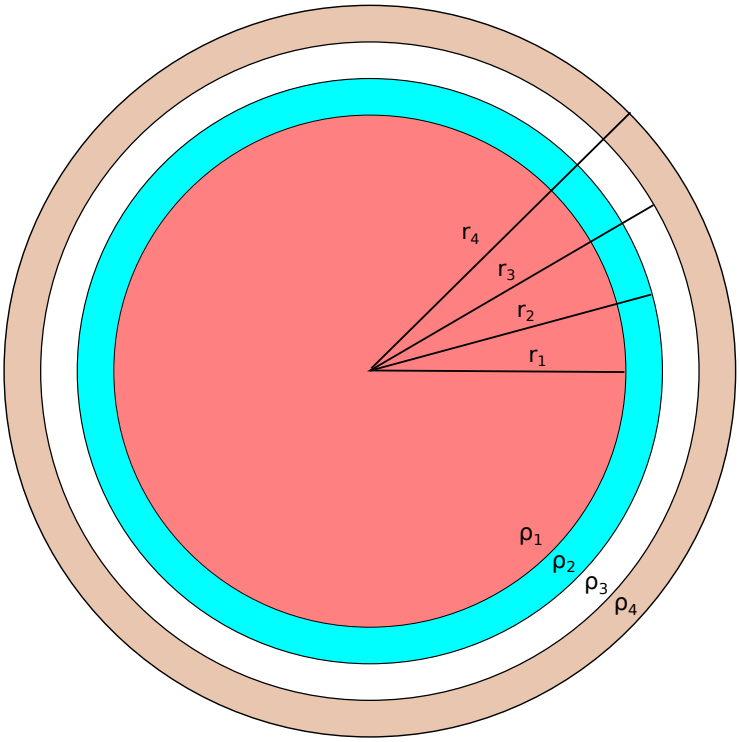


Figure 3.3: An outline of the four spheres model, not to scale

The innermost region represents the brain, the next the CSF, the third one the skull and the outermost the scalp. Each has a different radius $r_1, r_2, r_3, r_4$ and conductivity $\sigma_1, \sigma_2, \sigma_3, \sigma_4$. Nonetheless, the scalp is assumed to have the same conductivity as the brain $\sigma_1 = \sigma_4$.

The paper states that a unit dipole on the z-axis pointing in the x-direction generates the following potential:

$$v_x = \frac{\cos(\phi)}{4\pi\sigma_4 r_4} \sum_{n=1}^{\infty} \frac{(2n+1)^4 f^{n-1} (cd)^{2n+1} P_n^1(\cos(\theta))}{n\Gamma(n)} \tag{3.8}$$

The voltage for a unit dipole in y-direction is calculated by:

$$v_y = \frac{\sin(\phi)}{4\pi\sigma_4 r_4} \sum_{n=1}^{\infty} \frac{(2n+1)^4 f^{n-1} (cd)^{2n+1} P_n^1(\cos(\theta))}{n\Gamma(n)} \tag{3.9}$$

And finally for a dipole in z-direction:

$$v_z = \frac{1}{4\pi\sigma_4 r_4} \sum_{n=1}^{\infty} \frac{(2n+1)^4 f^{n-1} (cd)^{2n+1} P_n(\cos(\theta))}{n\Gamma(n)} \tag{3.10}$$

$\theta$ and $\phi$ are the angles of the spherical coordinates of $\boldsymbol{r}_s$, the position of the electrode. They can be calculated by:

$$\phi = \arctan\left(\frac{r_{s,y}}{r_{s,x}}\right) \tag{3.11}$$

$$\theta = \arctan\left(\frac{r_{s,z}}{\sqrt{r_{s,x}^2 + r_{s,y}^2}}\right) \tag{3.12}$$

$$\tag{3.13}$$

The function $\Gamma(n)$ is defined by:

$$\begin{aligned}\Gamma(n) =& d^{2n+1}\{b^{2n+1}n(k_1-1)(k_2-1)(n+1) + c^{2n+1}(k_1 n + n + 1)(k_2 n + n + 1)\} \\ & \{(k_3 n + n + 1) + (n+1)(k_3 - 1)d^{2n+1}\} + (n+1)c^{2n+1} \\ & \{b^{2n+1}(k_1-1)(k_2 n + k_2 + n) + c^{2n+1}(k_1 n + n + 1)(k_2 - 1)\} \\ & \{n(k_3 - 1) + (k_3 n + k_3 + n)d^{2n+1}\}\end{aligned} \tag{3.14}$$

Its parameters are:

$$b = \frac{r_1}{r_4}, \quad c = \frac{r_2}{r_4}, \quad d = \frac{r_3}{r_4} \tag{3.15}$$

$$k_1 = \frac{\sigma_1}{\sigma_2}, \quad k_2 = \frac{\sigma_2}{\sigma_3}, \quad k_3 = \frac{\sigma_3}{\sigma_4} \tag{3.16}$$

$$f = \frac{|\boldsymbol{r}_d|}{r_4} \tag{3.17}$$

As one can see, the original equation 3.8, 3.9 and 3.10 are similar. By rearranging the equations one receives:

$$\begin{aligned} v_x =& \frac{\cos(\phi)}{4\pi\sigma_4 r_4}\sum_{n=1}^{\infty}\frac{(2n+1)^4 f^{n-1}(cd)^{2n+1}P_n^1(\cos(\theta))}{n\Gamma(n)} \\ =& \sum_{n=1}^{\infty}\frac{(2n+1)^4 f^{n-1}(cd)^{2n+1}\cos(\phi)P_n^1(\cos(\theta))}{n\Gamma(n)4\pi\sigma_4 r_4}\end{aligned} \tag{3.18}$$

Now one can substitute the parts that are identical in every equation for $f(n)$:

$$f(n) = \frac{(2n+1)^4 f^{n-1}(cd)^{2n+1})}{\Gamma(n) 4\pi\sigma_4 r_4} \tag{3.19}$$

$$v_x = -\sum_{n=1}^{\infty} \frac{\cos(\phi) P_n^1(\cos(\theta)) f(n)}{n} \tag{3.20}$$

$$v_y = -\sum_{n=1}^{\infty} \frac{\sin(\phi) P_n^1(\cos(\theta)) f(n)}{n} \tag{3.21}$$

$$v_z = \sum_{n=1}^{\infty} P_n(\cos(\theta)) f(n) \tag{3.22}$$

$$\boldsymbol{v}(\boldsymbol{r}_d, \boldsymbol{r}_s) = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{3.23}$$

## 3.3 Implementation

To implement the aforementioned algorithm, there are three problems left to solve. Firstly, one has to compute the infinite sum in the equations 3.20, 3.21 and 3.22. Then a solution for dipoles outside the z-axis has to be found. Finally, the EEG reference has to be considered and built into the model.

### 3.3.1 Infinite sum

The infinite sum is computed iteratively:

$$\boldsymbol{v}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.24}$$

$$\boldsymbol{v}(n+1) = \Delta\boldsymbol{v}(n+1) + \boldsymbol{v}(n) \tag{3.25}$$

$$\Delta\boldsymbol{V}(n) = \begin{bmatrix} -\frac{\cos(\phi) P_n^1(\cos(\theta)) f(n)}{n} \\ -\frac{\sin(\phi) P_n^1(\cos(\theta)) f(n)}{n} \\ P_n(\cos(\theta)) f(n) \end{bmatrix} \tag{3.26}$$

Since this calculation would take forever, an exit condition is defined. As soon as the contribution of $\Delta\boldsymbol{v}$ becomes too small, the computation is declared finished:

$$|\Delta\boldsymbol{v}(n)| < s|\boldsymbol{v}(n)| \tag{3.27}$$

$s$ is called the sensitivity parameter and is set to $10^{-3}$.

### 3.3.2 Dipoles outside the z-axis

The current equations are only defined for dipoles on the z-axis, but since the problem is rotationally symmetric (those are spheres, after all), one can rotate the dipole and the electrode so that the dipole will land on the z-axis. At first, the necessary rotation matrix is defined:

$$\boldsymbol{R}(\boldsymbol{r}_d) = \begin{bmatrix} \frac{r_{d,x}r_{d,z}}{|\boldsymbol{r}_d|l_{xy}} & -\frac{r_{x,y}}{l_{xy}} & \frac{r_{d,x}}{|\boldsymbol{r}_d|} \\ \frac{r_{x,y}r_{d,z}}{l_{xy}|\boldsymbol{r}_d|} & \frac{r_{d,x}}{l_{xy}} & \frac{r_{d,y}}{|\boldsymbol{r}_d|} \\ -\frac{l_{xy}}{|\boldsymbol{r}_d|} & 0 & \frac{r_{d,z}}{|\boldsymbol{r}_d|} \end{bmatrix} \quad \text{with} \quad l_{xy} = \sqrt{r_{d,x}^2 + r_{d,y}^2} \tag{3.28}$$

This transformation was also taken from FieldTrip[9] and adjusted for my needs. Now the voltages can be calculated using the rotated electrode and dipole:

$$\boldsymbol{r}'_d = \boldsymbol{r}_d\boldsymbol{R}(\boldsymbol{r}_d) \tag{3.29}$$
$$\boldsymbol{r}'_s = \boldsymbol{r}_d\boldsymbol{R}(\boldsymbol{r}_d) \tag{3.30}$$
$$\boldsymbol{v}' = \boldsymbol{v}(n, \boldsymbol{r}'_d, \boldsymbol{r}'_s) \tag{3.31}$$

And finally, the voltages are rotated back to assure that the axes point in the right directions. The rotational matrix $\boldsymbol{R}$ is orthogonal, so transposing it has the same effect as inverting it. Since transposing is computationally more efficient, we will use the transposition:

$$\boldsymbol{v} = \boldsymbol{v}'\boldsymbol{R}^T \tag{3.32}$$

### 3.3.3 Reference considerations

During the measurement of the data used in this work, the average of all channels was used as a reference. This has to be compensated during the lead field matrix calculation. The approach that I selected was to subtract the average of each column of the lead field matrix from said column:

$$\boldsymbol{L}'(\boldsymbol{r}_d) = \begin{bmatrix} \boldsymbol{v}(\boldsymbol{r}_d, \boldsymbol{r}_{s,1}) \\ \boldsymbol{v}(\boldsymbol{r}_d, \boldsymbol{r}_{s,2}) \\ \vdots \\ \boldsymbol{v}(\boldsymbol{r}_d, \boldsymbol{r}_{s,M}) \end{bmatrix} \tag{3.33}$$

$$\boldsymbol{L}(\boldsymbol{r}_d) = \boldsymbol{L}'(\boldsymbol{r}_d) - \begin{bmatrix} E\{\boldsymbol{L}_1\} & E\{\boldsymbol{L}_2\} & E\{\boldsymbol{L}_3\} \\ \vdots & \vdots & \vdots \\ E\{\boldsymbol{L}_1\} & E\{\boldsymbol{L}_2\} & E\{\boldsymbol{L}_3\} \end{bmatrix} \tag{3.34}$$

Here, $\boldsymbol{L}_1$ refers to the first column of the lead field matrix. It seems counter-intuitive to subtract something from a transfer matrix, but one has to keep in mind that the average, which is used to offset the potentials, is dependent on the dipoles:

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{c}' - E\{\boldsymbol{c}'\} \\ &= \boldsymbol{L}'\boldsymbol{d} - E\{\boldsymbol{L}'\boldsymbol{d}\} \\ &= \boldsymbol{L}'\boldsymbol{d} - E\{\boldsymbol{L}'\}\boldsymbol{d} \\ &= (\boldsymbol{L}' - E\{\boldsymbol{L}'\})\boldsymbol{d} \\ &= \boldsymbol{L}\boldsymbol{d} \end{aligned} \tag{3.35}$$

After the reference correction is done, the calculation of the lead field matrix is complete. This calculation must be repeated for every beamformer target in the initialisation step.

## 3.4 Parametrisation

The parameters for the forward model were taken from the two cited papers[2][3] and [10]:

| Symbol | Value | Source/Comment |
|---|---|---|
| $r_1$ | 7.9 cm | [2] |
| $r_2$ | 8.1 cm | [2] |
| $r_3$ | 8.5 cm | [2] |
| $r_4$ | 8.8 cm | [2] |
| $\sigma_1$ | 0.2770 S/m | [10], average between grey matter (mixed) and white matter (mixed) |
| $\sigma_2$ | 1.79 S/m | [10] |
| $\sigma_3$ | 0.0347 S/m | 1.25% of $\sigma_1$ according to [3] |
| $\sigma_4$ | 0.2770 S/m | same as $\sigma_1$ according to [2][3] |

Table 3.2: Parameters for the forward model

## 3.5 Evaluation

First of all, the model was evaluated by comparing its results to the FieldTrip implementation of the same forward model. It was found that the results are identical (in the sense of equation 3.7) to the FieldTrip implementation if one ignores the influences of the reference corrections (see 3.3.3).

Moreover, the voltages for multiple chosen dipoles were calculated and checked for reasonableness. For the first test, a dipole was placed at $\boldsymbol{r}_d = [0 \quad -7 \quad 0]^T$ close to and directly between the O1 and O2 electrode. The dipole was pointing in the positive y-direction and its amplitude was adjusted so that the voltage values were in the magnitude of real EEG signals ($|\boldsymbol{d}| = 0.001$ Cm). One would expect that O1 and O2 have identical voltages and that all other electrodes have lower voltages:

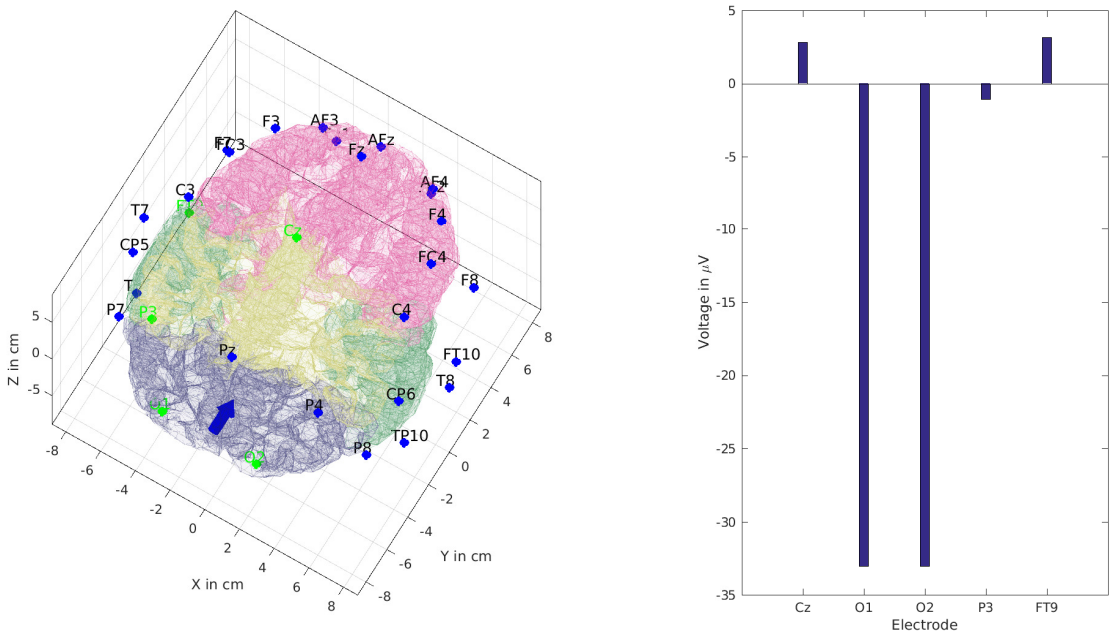


Figure 3.4: Dipole placed between O1 and O2

As one sees in the graph above, the expectations are met. The other electrodes do still receive some voltage, but it is a lot lower than at O1 and O2. After this, the dipole source was moved slightly to the right to $\boldsymbol{r}_d = [0.5 \quad -7 \quad 0]$:

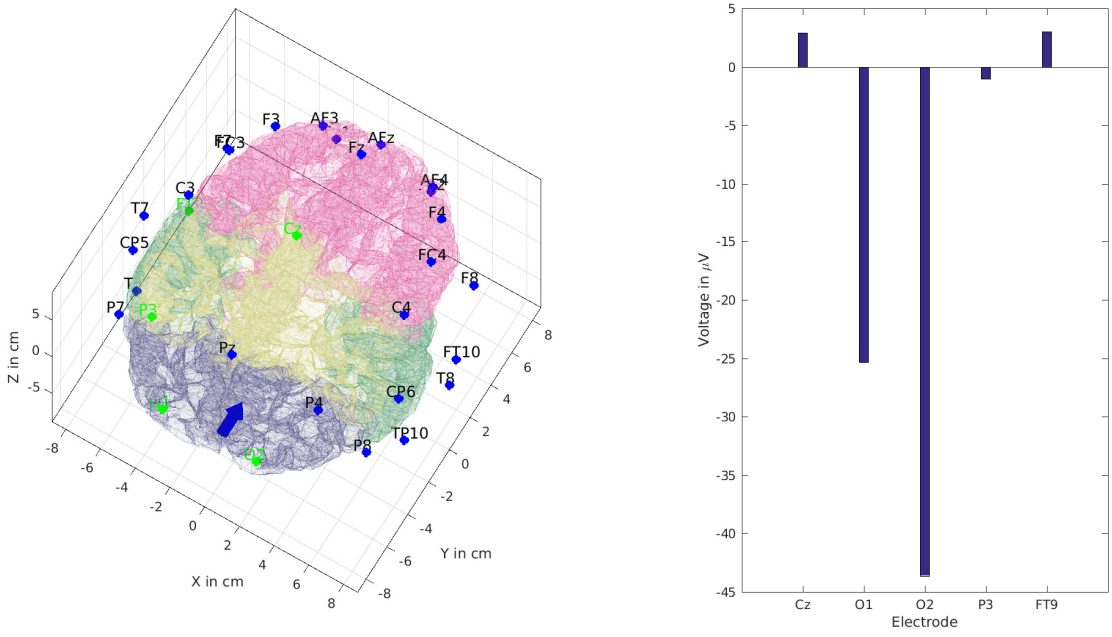


Figure 3.5: Dipole placed between O1 and O2, closer to O2

Now the magnitude of the voltage at O2 is a lot higher than at O1, which also meets the

expectations, since O2 is closer to the source.

The next test places the dipole into the middle of the head ($\boldsymbol{r}_d = [0 \quad 0 \quad 0]$). One would expect all electrodes to have a similar magnitude, which is lower than the magnitudes in previous tests, since those were all close to the surface.

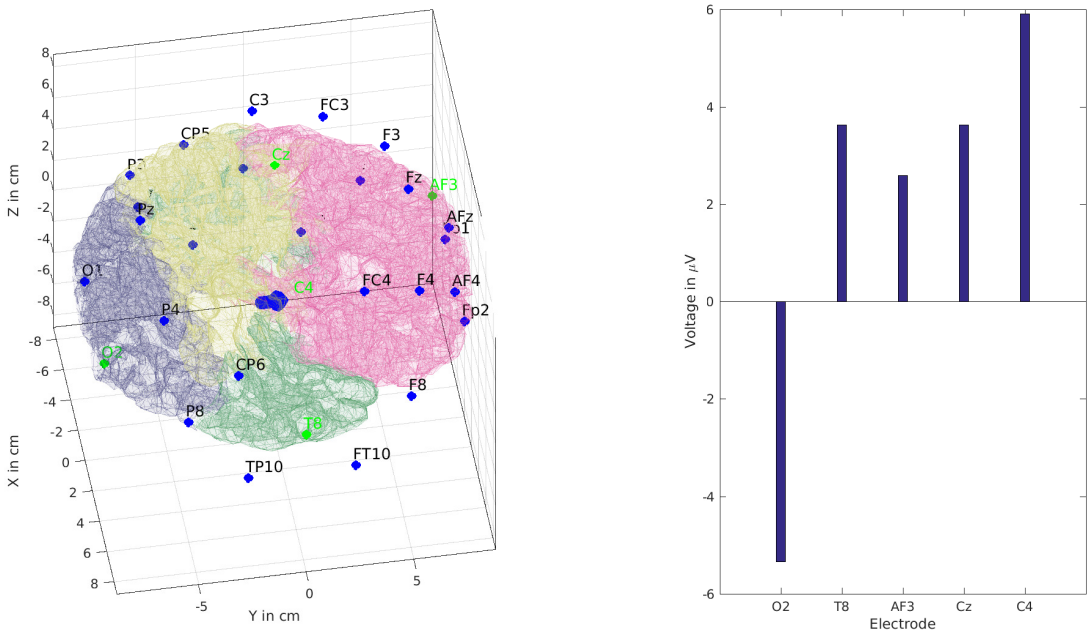


Figure 3.6: Dipole placed in the middle of the head

These tests show that the forward model agrees with physical intuition. The problem that no voltage could be detected at any electrode, when the dipole was placed in the middle of the head, is solved with this new model. Furthermore, the fact that FieldTrip uses an equivalent forward model gives a strong indication that this is correct, since FieldTrip is a widely used and respected toolbox.

# 4 Filter bank and inverse filter bank

At the beginning of the processing step, the data is split into chunks which are about 1 second long ($\frac{256}{250\text{Hz}} = 1.024s$ to be exact). Those chunks overlap one another by $\frac{3}{4}$, meaning that a new chunk starts every 0.256 seconds. The overlapping increases the temporal resolution and avoids that some data points affect the result more than others due to the window function (see next section).

After this, the filter bank transforms the signal into the frequency domain and removes unnecessary frequencies.

## 4.1 Filter bank

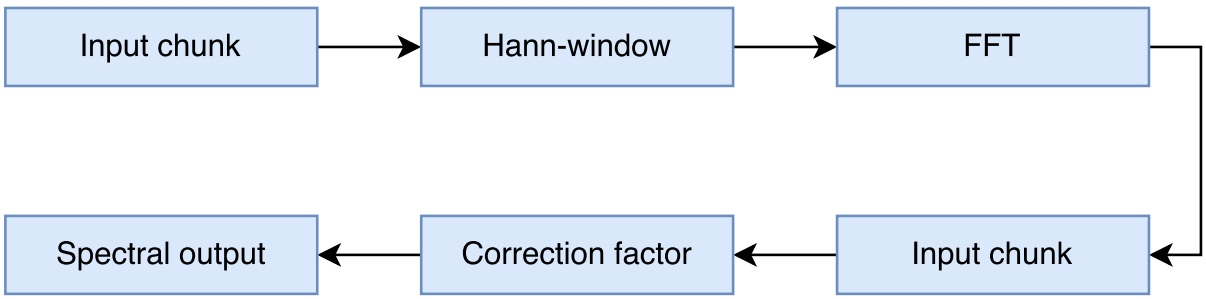


Figure 4.1: Outline of the filter bank algorithm

Firstly, the filter bank applies a Hann-window to the chunk of time domain input data. This is done by element-wise multiplication of the input chunk and a sampling of the Hann-function with the same size. The Hann-window function is defined as:

$$h(n) = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{N-1}\right) \tag{4.1}$$

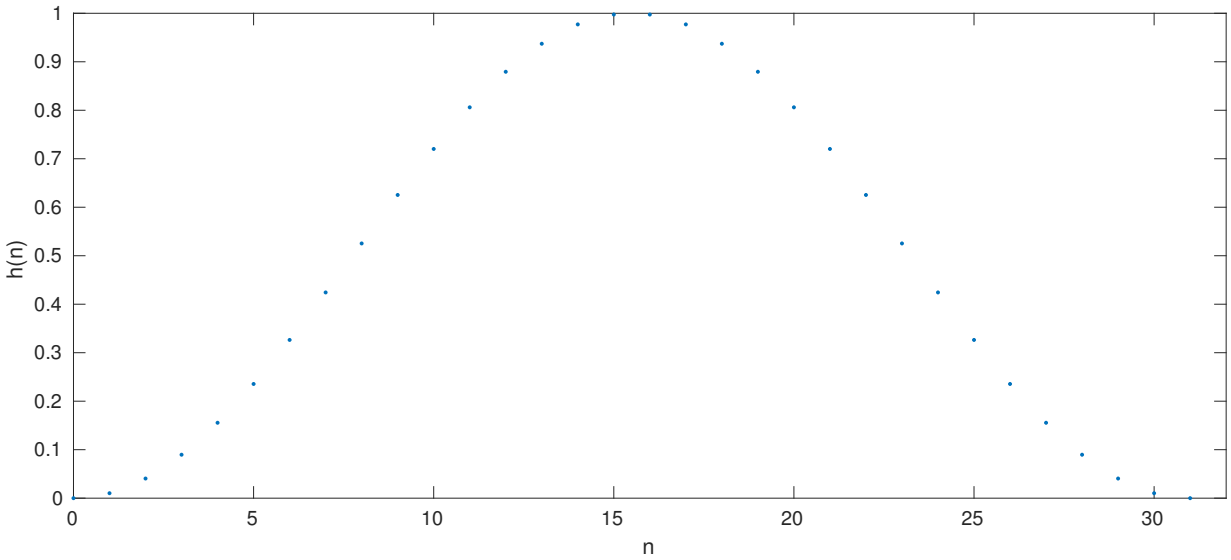


Figure 4.2: A Hann window for $N = 16$

The window function is used to suppress frequency leakage effect and avoid artificially generating high frequency components. This could happen, because the DFT assumes that the signal is periodic, which our data chunk obviously isn't. The possibly high jump going back from the end to the start might induce unwanted frequencies, which can only be avoided if the window function starts and begins with a zero.

After the window has been applied, the signal gets transformed into the frequency domain using a FFT with a length of $N = 256$. Earlier, $N$ was rounded to the next power of two earlier (see 2.2) so that the FFT is at its maximum efficiency.

Subsequently, all the frequency bins above and including $f_{LP} = 50$ Hz are discarded. This is done for two reasons: reducing noise and computation time. Since all interesting processes in the brain happen at 49 Hz or lower, the higher frequencies can be considered as noise. And since every frequency bin will later be processed by beamformer, reducing the number of processed frequencies also results in a substantial performance benefit. Frequencies above $\frac{f_s}{2}$ are symmetrical to the lower frequencies and can be restored later. This is due to the real nature of the signal.

Since a number of frequencies have been discarded, the resulting spectrum is shorter. To preserve its original magnitude, one has to multiply a correction factor $a$ to the spectral chunk:

$$a = \frac{2f_{LP}}{N} \tag{4.2}$$

This correction factor is not strictly necessary, since it will be inverted in the inverse filter bank. But it is definitely useful for debugging purposes and in case someone wants to do something non-linear in a future iteration of the beamformer. After this, the output data is $\in \mathbb{C}^{50\times M}$.

## 4.2 Inverse filter bank

After the data has been processed by the beamformer (see 5), it should be transformed back into the time domain. This is done by using the inverse filter bank:

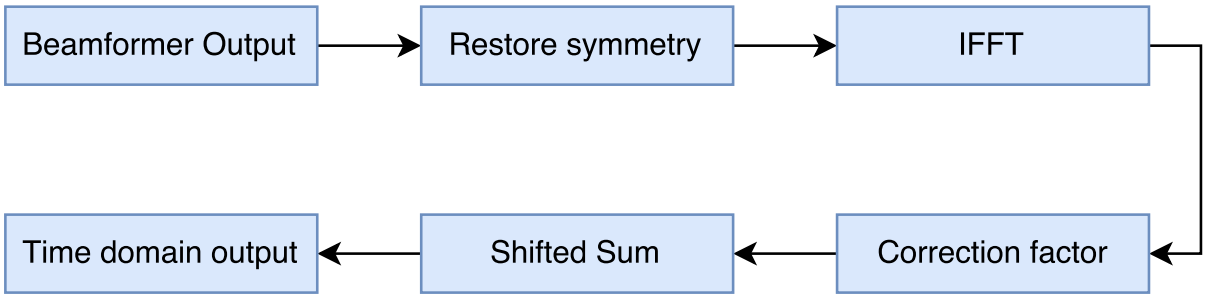


Figure 4.3: Outline of the inverse filter bank algorithm

At first, the original symmetry has to be restored and the spectrum has to be padded with zeros to receive a real output signal with the same sampling rate:

$$\left[\boldsymbol{Y}(e^{j\Omega}) \quad 0 \quad \dots \quad 0 \quad \boldsymbol{Y}^*(e^{j(2\pi-\Omega)})\right] \tag{4.3}$$

Then the chunk is transformed back into the time domain by using the IFFT with a length of $M = 256$. The inverse correction factor from the last section is applied to result in a properly scaled time domain chunk:

$$a^{-1} = \frac{N}{2f_{LP}} \tag{4.4}$$

To add multiple chunks together to a single coherent signal, the first four chunks have to be ignored to compensate the beamformer delay (see 5.2). Then the chunks are aggregated with respect to their original offset. The Hann-window does not need to be inverted, as Hann-windows with specific offsets add to a constant. In this case the offset is $\frac{1}{4}$, so the resulting constant turns out to be 2:

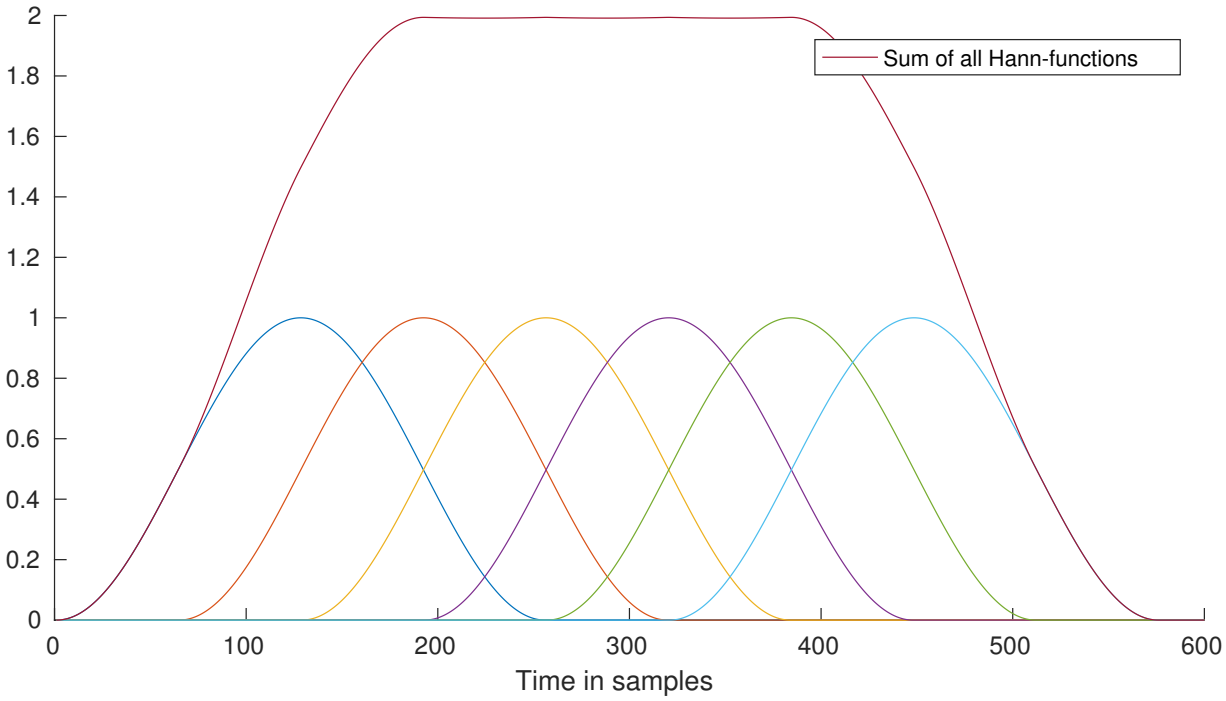


Figure 4.4: Six Hann-window functions and their sum

This factor of 2 has to be corrected by dividing the output signal by 2. There is also a short section at the beginning and at the end without enough Hann-windows to achieve a constant value. This behaviour does not really matter for the whole algorithm, because it is just a short period in a long signal.

## 4.3 Evaluation

To evaluate the filter banks, a test signal is put through the normal processing pipeline while the beamformer is disabled. The resulting output signal is then compared to the original signal and if both match up, the filter banks work properly.

The test signal was chosen to be two sine waves ($f_1 = 1\text{Hz}, f_2 = 13\text{Hz}$) with a low amplitude additive white Gaussian noise and a constant offset of 2.5:

$$s(n) = 2.5 + \sin(2\pi\frac{f_1}{f_s}n) + \sin(2\pi\frac{f_2}{f_s}n) + \frac{N_G(n)}{10} \tag{4.5}$$

$N_G(n)$ is a series of Gauss random values. The signal's length is about 41 seconds (40 chunks $= \frac{256}{250\text{Hz}}40 = 40.96s$). After the filter bank is applied, the signal is restructured to simulate beamforming. Finally, the inverse filter bank is applied. The results are displayed below:

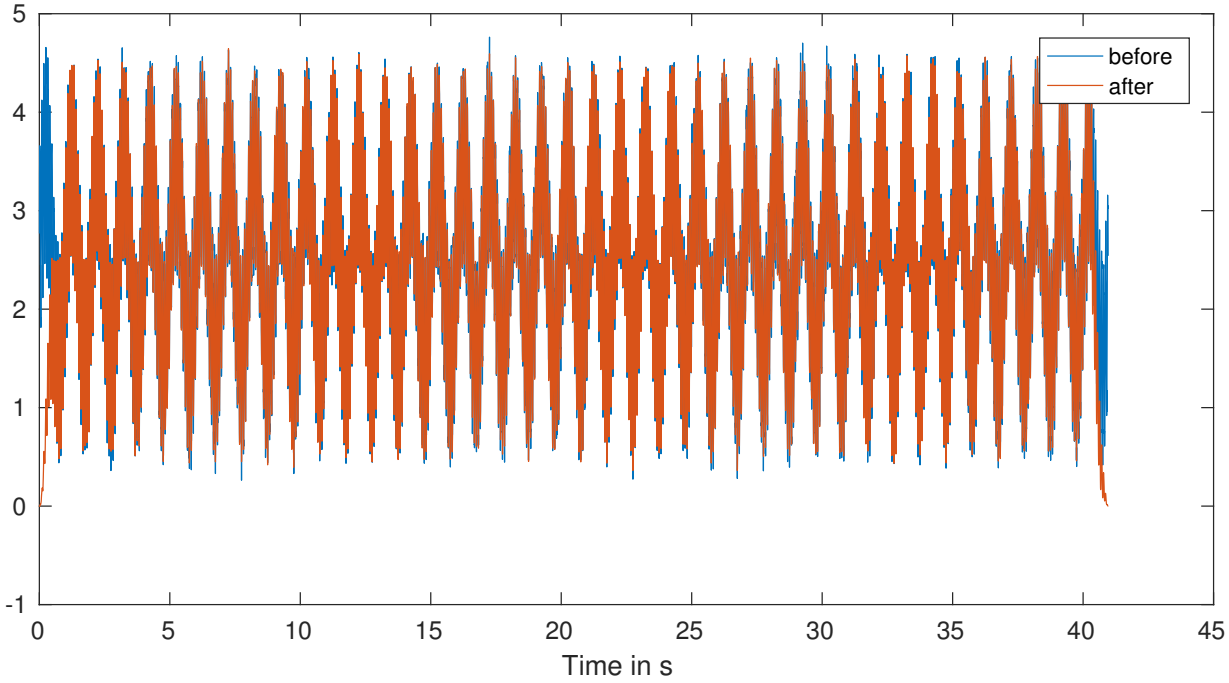


Figure 4.5: Results of the filter bank test

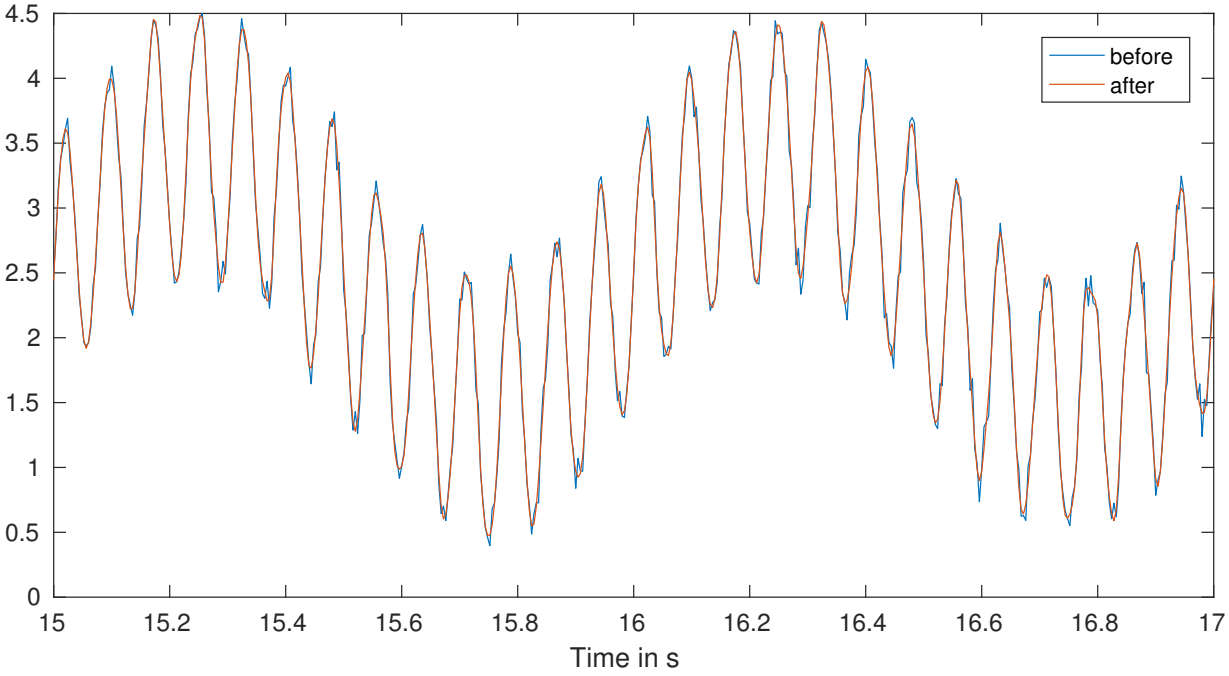


Figure 4.6: Results of the filter bank test (zoomed)

The two figures above clearly display that the filter banks are working as expected. The two signals line up nearly perfectly, except for the beginning and the end, which is due to a fade-in and fade-out. The fade-in and fade-out was discussed earlier and does not have significant influence on the whole program.

Additionally, the average distance between the two signal was measured. It turns out to be in

the $10^{-6}$ range depending on the exact random values. This is due to high frequency noise being cut off by the low-pass filter and that is acceptable for this application.

Moreover, it was tested whether the output signal contains any complex values. If it did, this would be an indication that the symmetry is not correctly restored. The output does not contain any complex values.

# 5 The Beamformer

This chapter contains the theoretical and technical description of the beamformer.

## 5.1 Theoretical Background

Beamforming algorithms are used to extract spatial information from a sensor array or to give an antenna array a spatial directivity. For sensor arrays they weigh the inputs $\boldsymbol{S}(n)$ with a weight vector $\boldsymbol{w}$:

$$y(n) = \boldsymbol{w}^T \boldsymbol{S}(n) \tag{5.1}$$

There are different ways to obtain $\boldsymbol{w}$, depending on which spatial and noise characteristics are desired. Different beamforming approaches also differ in terms of numeric complexity. All of them require a prediction how each point of interest in space will affect each sensor. This prediction is obtained from the forward model (see 3). In the forward model, a lead field matrix is computed for each point of interest. The predicted potential at each electrode $\boldsymbol{c}$ can be either a column of the lead field matrix (for one of the primary directions) or a linear combination of the three columns.

In this chapter, two kinds of beamformers will be discussed, the MVDR beamformer and the quiescent beamformer.

### 5.1.1 The MVDR beamformer

The *minimum variance distortion-less response* (sometimes also called LCMV beamformer for *linear constrained minimum variance*) beamformer is a standard way of defining a beamformer. Distortion-less response means that a signal from the point of interest has to get to the output signal $y(n)$ with a gain of 1, meaning it can not be distorted in any way. This criterion is not sufficient for defining the beamformer, so the second criterion states that the variance (and consequently the energy of the beamformed signal) should be minimal.

In the following section, the beamformer weight formula will be derived from these two requirements.

More formally stated, the requirements mean that the following equation should be optimised:

$$\min_{\boldsymbol{w}} \quad E\{y^2(n)\} \quad \text{s.t.} \quad \boldsymbol{w}^T c = 1 \tag{5.2}$$

Using Parseval's Theorem, the equation is transformed to the frequency domain:

$$\begin{aligned} & \min_{\boldsymbol{w}} \quad E\{|Y(e^{j\Omega})|^2\} && \text{s.t.} \quad \boldsymbol{w}^T\boldsymbol{c} = 1 \\ \Leftrightarrow \quad & \min_{\boldsymbol{w}} \quad E\{|\boldsymbol{w}^T\boldsymbol{S}|^2\} && \text{s.t.} \quad \boldsymbol{w}^T\boldsymbol{c} - 1 = 0 \\ \Leftrightarrow \quad & \min_{\boldsymbol{w}} \quad E\{(\boldsymbol{w}^T\boldsymbol{S})(\boldsymbol{w}^T\boldsymbol{S})^H\} && \text{s.t.} \quad \boldsymbol{w}^T\boldsymbol{c} - 1 = 0 \\ \Leftrightarrow \quad & \min_{\boldsymbol{w}} \quad E\{\boldsymbol{w}^T\boldsymbol{S}\boldsymbol{S}^H\boldsymbol{w}^*\} && \text{s.t.} \quad \boldsymbol{w}^T\boldsymbol{c} - 1 = 0 \end{aligned} \tag{5.3}$$

Since $\boldsymbol{w}$ is not statistical, it can be extracted from the expected value. The remaining construct is very similar to the definition of a covariance matrix (in fact, it is often called a covariance matrix in literature, which is true if the signal has a mean of zero).

$$\begin{aligned} & \min_{\boldsymbol{w}} \quad \boldsymbol{w}^T E\{\boldsymbol{S}\boldsymbol{S}^H\}\boldsymbol{w}^* && \text{s.t.} \quad \boldsymbol{w}^T\boldsymbol{c} - 1 = 0 \\ \Leftrightarrow \quad & \min_{\boldsymbol{w}} \quad \boldsymbol{w}^T\boldsymbol{R}_{\boldsymbol{ss}}\boldsymbol{w}^* && \text{s.t.} \quad \boldsymbol{w}^T\boldsymbol{c} - 1 = 0 \end{aligned} \tag{5.4}$$

At this point, the equation becomes an optimisation problem subject to a linear constraint, which can be solved using the method of Lagrange multipliers. The Lagrange function for this problem is:

$$\mathcal{L} = \boldsymbol{w}^T\boldsymbol{R}_{\boldsymbol{ss}}\boldsymbol{w}^* - \lambda(\boldsymbol{w}^T\boldsymbol{c} - 1) \tag{5.5}$$

The cost function (equation 5.2) becomes optimal when the gradient of the Lagrange function in $\boldsymbol{w}$ becomes zero. The calculation of the gradient can be found in appendix 9.1.

$$\nabla_{\boldsymbol{w}}\mathcal{L} = \frac{1}{4}\boldsymbol{w}^H\boldsymbol{R}_{\boldsymbol{ss}} - \lambda\boldsymbol{c}^T = \boldsymbol{0} \tag{5.6}$$

Now it is possible to solve for $\boldsymbol{w}$:

$$\begin{aligned} & \boldsymbol{w}^H = 4\lambda\boldsymbol{c}^T\boldsymbol{R}_{\boldsymbol{ss}}^{-1} \\ \Leftrightarrow \quad & \boldsymbol{w}^T = (4\lambda\boldsymbol{c}^T\boldsymbol{R}_{\boldsymbol{ss}}^{-1})^* \end{aligned} \tag{5.7}$$

This solution is now inserted into the original constraint to find a solution for $\lambda$:

$$\begin{aligned} & (4\lambda\boldsymbol{c}^T\boldsymbol{R}_{\boldsymbol{ss}}^{-1})^*\boldsymbol{c} = 1 \\ \Leftrightarrow \quad & 4\lambda^*\boldsymbol{c}^H[\boldsymbol{R}_{\boldsymbol{ss}}^{-1}]^*\boldsymbol{c} = 1 \\ \Leftrightarrow \quad & \lambda^* = \frac{1}{4\boldsymbol{c}^H[\boldsymbol{R}_{\boldsymbol{ss}}^{-1}]^*\boldsymbol{c}} \\ \Leftrightarrow \quad & \lambda = \frac{1}{4\boldsymbol{c}^T\boldsymbol{R}_{\boldsymbol{ss}}^{-1}\boldsymbol{c}^*} \end{aligned} \tag{5.8}$$

Since $\boldsymbol{c}$ is real, $\boldsymbol{c}^* = \boldsymbol{c}$. The solution for the Lagrange multiplier $\lambda$ will now be inserted back into equation 5.7 to get the final result:

$$\boldsymbol{w}^T = \frac{\boldsymbol{c}^T \boldsymbol{R}_{ss}^{-1}}{\boldsymbol{c}^T \boldsymbol{R}_{ss}^{-1} \boldsymbol{c}} \tag{5.9}$$

This beamformer will be used for most of the program.

### 5.1.2 Quiescent Beamformer

A simpler beamformer would be the *quiescent beamformer*. It is defined by the equation:

$$\boldsymbol{w} = \frac{\boldsymbol{c}}{\boldsymbol{c}^T \boldsymbol{c}} \tag{5.10}$$

This beamformer is data-independent and optimised to have a minimum gain for additive white noise. It is also more efficient to compute, because the weight factors can be calculated beforehand in contrast to the weights of the MVDR beamformer. Those have to be calculated every time the covariance matrix changes.

The disadvantage is that the quiescent beamformer has an inadequate spatial resolution which will be demonstrated in the evaluation section (see 5.3.4).

## 5.2 Implementation

The beamformer needs a memory to store part of the signal for the estimation of the covariance matrix. This memory is a 3D-matrix $\in \mathbb{C}^{M \times N_m \times N_f}$ and is allocated in the beamformer's initialisation method. $N_m$ is the number of chunks in the beamformer's memory and is set to $N_m = 9$, meaning the beamformer will look approximately one second into the future and the past. The beamformer also needs to be supplied with a lead field matrix from the forward model.

The beamformer's main method is called once for each chunk of data in the main program's loop after the chunk has been transformed into the frequency domain (see 4.1). This function will be discussed in more depth than usual, because it has some interesting optimisations.

The main function requires three nested loops to calculate the output, iterating over frequency, beamformer target and direction. For each new frequency, the covariance matrix is recalculated. Since it is only used in the inverted form, it is inverted directly after its computation to save the effort of having to do this multiple times. The memory was specifically formed so that the calculation of the covariance matrix is most efficient. The calculation is performed by multiplying the part of the memory for the current frequency with the conjugate transpose of itself divided by $N_m$.

Since the inversion of this matrix is numerically unstable because of its symmetry, a small number is added to the main diagonal of the matrix. In effect, the calculation of the inverted covariance results in:

$$\begin{aligned} \boldsymbol{R_{ss}} &= \frac{\boldsymbol{V_f}\boldsymbol{V_f}^H}{N_m} \\ \boldsymbol{R'_{ss}} &= \boldsymbol{R_{ss}} + \boldsymbol{I}E\{diag\{R_{ss}\}\}\alpha \end{aligned} \tag{5.11}$$

Here, $\boldsymbol{V_f} \in \mathbb{C}^{M \times N_m}$ is a part of the memory for the current frequency. $\alpha$ controls the size of the offset. It will be called the beamformer stability parameter and its effects are discussed in section 5.3.2.

After the covariance matrix is updated, the function extracts the chunk located in the middle of the memory. This chunk is later used for the actual beamforming. It would be possible to use the newest chunk, but the estimate of the covariance matrix is better if some future chunks are also included in the estimation process. This has the disadvantage of resulting in a delay of $\frac{N_m-1}{2}$ chunks, which has to be corrected later (see section 4.2).

Then the function extracts the current column of the lead field (depending on point of interest, direction and possibly frequency). Here, it is necessary to check for an edge case, where the predicted potential would be equal to zero for all electrodes. This happened for the old forward model (see 3.1) and for mock forward models for testing purposes and would result in NaN values in the output. It can be fixed by checking if the predicted potential $c$ contains values other than zero.

Afterwards, the configured (either the MVDR or quiescent) beamformer formula is applied.

The innermost loop of this function takes about 80% of the program's computing time, which is why it was heavily optimised. This is mostly due to the number of times it is called. For ten minutes of data, the innermost loop runs about 31 million times. The optimisations include:

- The configuration option for the beamformer type is initially given as a string. It is transformed into a numeric value (1 for MVDR beamformer and 2 for quiescent beamformer) in the initialisation function to save time by avoiding having to call the strcmp function in the innermost loop.
- In the beamformer's main function, it is necessary to have three nested loops (looping over frequency, point of interest and dipole direction). The frequency loop was chosen to be the outermost loop in order to avoid having to recalculate the inverted covariance matrix. This is possible, because the covariance matrix is only dependent on the frequency (and not on the target point or direction).
- The memory array was set up so that a memory block for a specific frequency could be accessed by using the last index. This is slightly more efficient in MATLAB compared to using the first or second index (see the test script tests/matrix_access.m). It also avoids having to call the squeeze function to reshape the matrix.
- When checking the edge case in the innermost loop, nnz(c)==0 is called. $c$ is a vector of the predicted potentials and nnz() is a MATLAB function, which checks if an array contains any zeros (the abbreviation stands for number of non-zero elements). This function is responsible for a significant portion of the processing time in that algorithm, which is why the lead field matrix is checked for columns that only contain zeros during the initialisation. If the matrix contains no such columns, the check can be skipped and a call to nnz can be avoided.

## 5.3 Evaluation

The beamformers were evaluated by generating synthetic signals using the forward model and applying the beamformers to these signals.

During the development of these tests, a synthetic lead field matrix was generated. This matrix had the following transmission coefficients:

$$\boldsymbol{L}_i = \frac{1}{d_i^2}[1 \quad 1 \quad 1] \tag{5.12}$$

This lead field matrix is independent of the dipole direction. It was used to make sure that possible mistakes during the implementation of the forward model did not distort the following evaluation. Nevertheless, the following evaluation uses the lead field matrix calculated by the forward model to obtain more realistic results.

### 5.3.1 General functionality

The first step necessary for the beamformers evaluation process is to check if it actually performs at all. For this purpose a sine source with a frequency of $f_{sig} = 10$ Hz was placed at $\boldsymbol{r} = [-4 \quad 4 \quad 0]$ with an orientation of $\boldsymbol{r}_d = [0 \quad 0 \quad 1]$. To the resulting signal an additive white Gaussian noise source was added so that the resulting SNR would be at 30 dB. The whole signal was produced like this:

$$\boldsymbol{s}(n) = \sin\left(2\pi\frac{f_{sig}}{f_s}n\right)\boldsymbol{L}(\boldsymbol{r})\boldsymbol{r}_{\boldsymbol{d}}^T + \sqrt{\frac{P_s}{SNR}}\boldsymbol{N}_{\boldsymbol{G}}^{\boldsymbol{M}}(n) \tag{5.13}$$

$\boldsymbol{N}_{\boldsymbol{G}}^{\boldsymbol{M}}(n)$ is a series of Gauss-distributed random vectors $\in \mathbb{R}^M$ with a mean of zero and a power of one. $P_s$ is the power of the signal after it was scaled by the lead field matrix. The sampling rate was set to $f_s = 250$ Hz like in the original data and about ten seconds of signal were generated. The beamformer was pointed directly at the source and a few centimetres away from the source.

Then the signal was passed through the standard processing pipeline of the filter bank, the MVDR beamformer and the inverse filter bank. Additionally to the final data, the results after the beamformer (in the frequency domain) were stored. This was done under the assumption that the filter bank and inverse filter bank were working as expected (see 4.3). The results can be seen in the following figure:

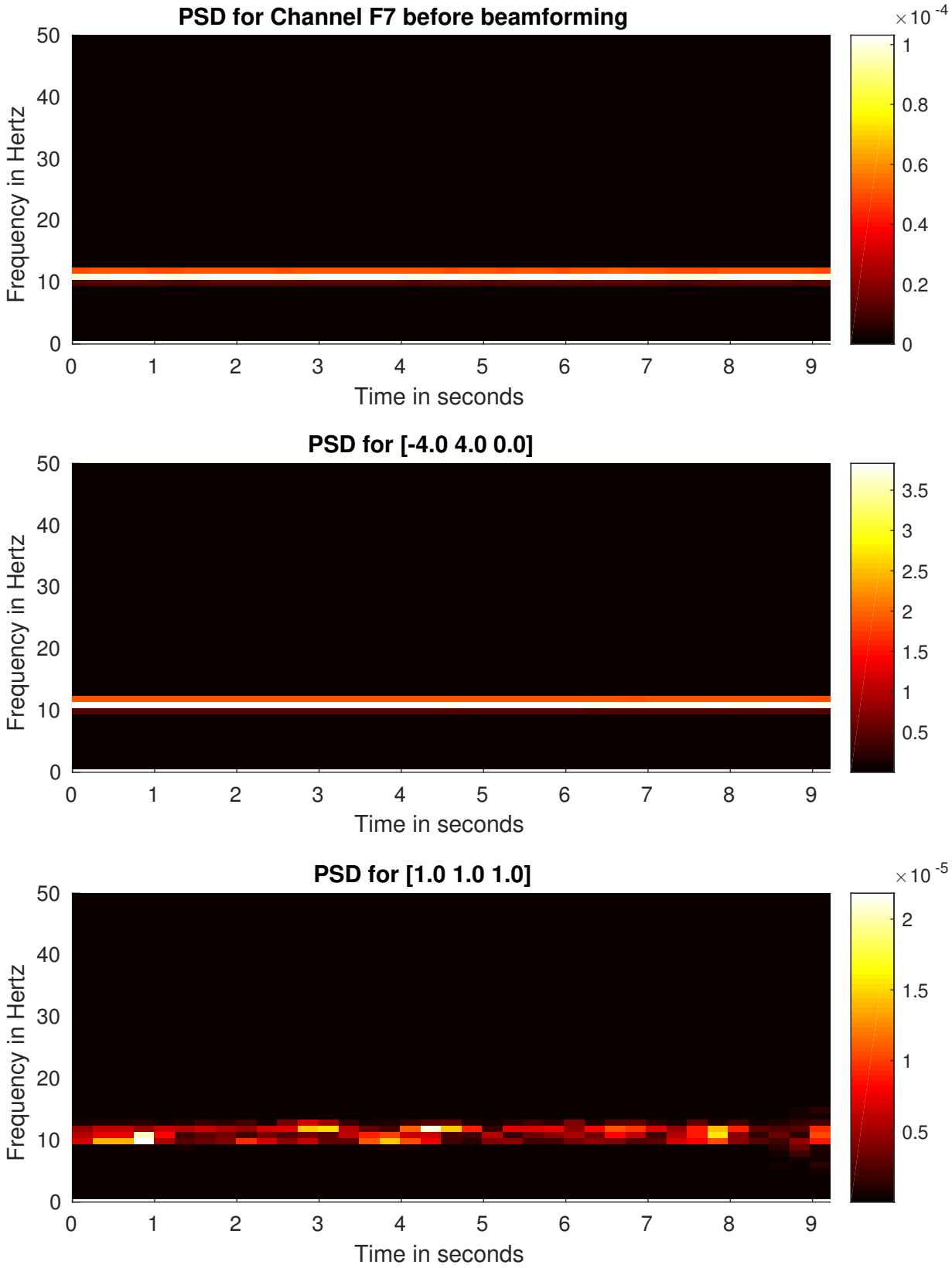


Figure 5.1: Frequency domain results of the first simulation

The first graph shows the power spectral density for the closest electrode (F7) to the source

within the head. The 10 Hz line is clearly visible.

The second graph displays the PSD of the beamformer output pointed directly at the source. Again the 10 Hz line is clearly visible and the amplitude is also in the single digit range.

For comparison the final graph shows the PSD of the beamformer output aimed at a different point in the head ($\boldsymbol{r} = [1 \quad 1 \quad 1]$). The 10 Hz line is still the strongest signal, but it is visibly distorted by the noise. Furthermore the amplitude of the signal is lower by a factor of $10^5$.

The following graph displays the reconstructed time domain data with the beamformer aimed directly at the source:

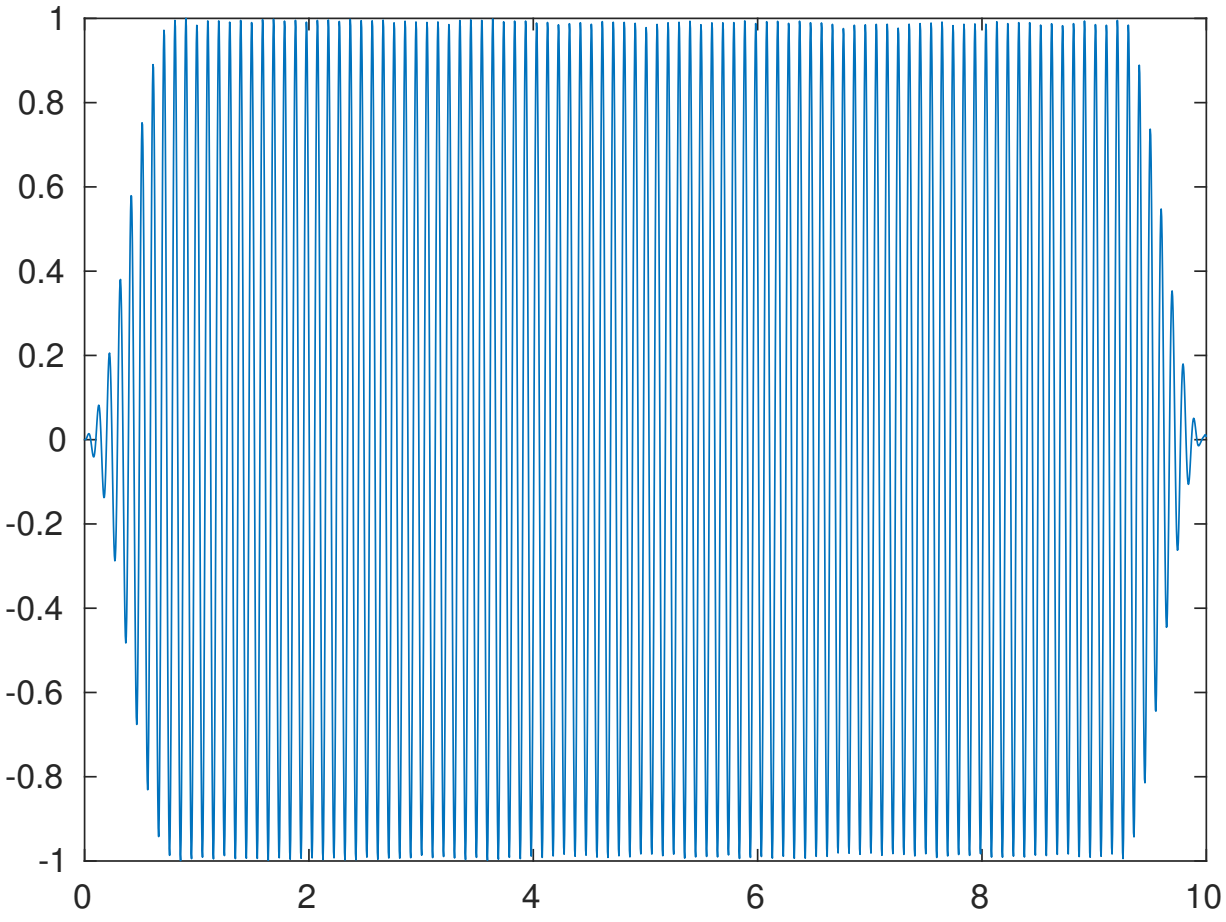


Figure 5.2: Time domain results of the first simulation

Here, it is clearly visible that the beamformer successfully reconstructed the signal. At the beginning and the end of the signal there is a fade-in and fade-out respectively. This is due to the window functions not summing up to one (see more in section 4.2). There also seem to be some fluctuations in the signals amplitude, but those are due to the limited temporal resolution of the signal. Most of the time, there is no data point at the extremes of the signal. This can be seen in the plot below, where the signal is zoomed in to one second:

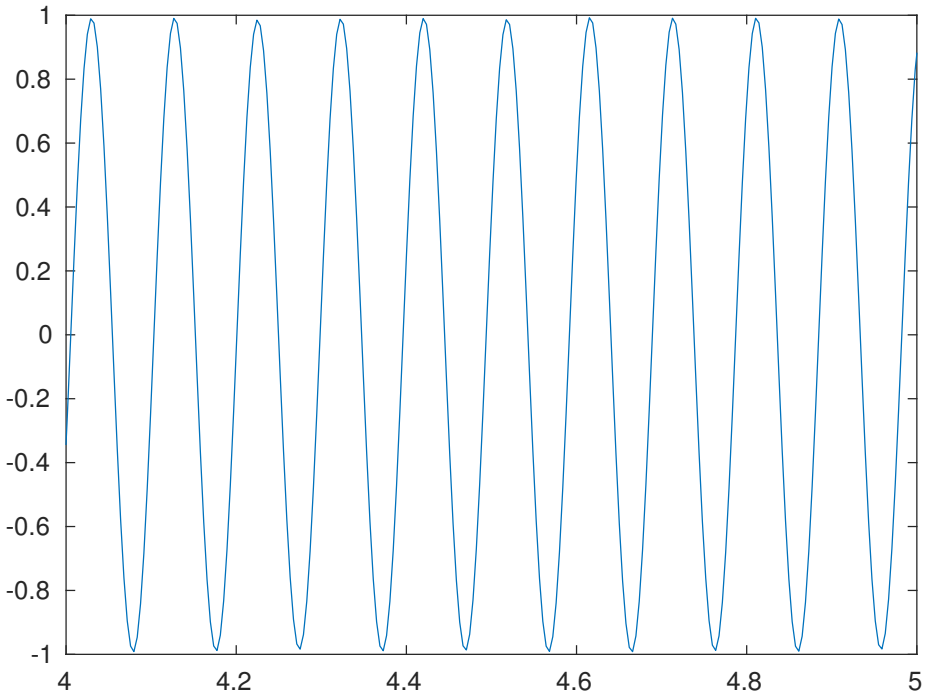


Figure 5.3: Time domain results of the first simulation (zoomed)

This means, that the MVDR beamformer is generally working, because it clearly spatially filters the data. Nevertheless, this test is not realistic due to the SNR of 1000 being a lot higher than one would expect in reality.

### 5.3.2 Stability parameter

After the general functionality of the beamformer has been confirmed, it is necessary to find a value for the stability parameter $\alpha$. For this, a source with a frequency of $f_{sig} = 10$ Hz was placed in the middle of the head model ($\boldsymbol{r} = [0 \quad 0 \quad 0]$) with an orientation of $\boldsymbol{r}_d = [0 \quad 0 \quad 1]$. Additionally, a noise source was placed $\boldsymbol{r}_n = [0 \quad 4 \quad 0]$ with an orientation of $\boldsymbol{r}_{nd} = [0 \quad 0 \quad 1]$ and an SNR of 1000. The whole signal now consists of:

$$\boldsymbol{s}(n) = \sin\left(2\pi\frac{f_{sig}}{f_s}n\right)\boldsymbol{L}(\boldsymbol{r})\boldsymbol{r}_{\boldsymbol{d}}^T + \sqrt{\frac{P_s}{P_n \cdot \mathrm{SNR}}}\boldsymbol{N}_{\boldsymbol{G}}^{\boldsymbol{M}}(n)\boldsymbol{L}(\boldsymbol{r}_{\boldsymbol{n}})\boldsymbol{r}_{\boldsymbol{nd}}^T \tag{5.14}$$

Finally, the beamformer targets were placed on the x-axis in a distance of $\Delta x = 0.02$ cm.

Two characteristics were analysed: the stability of the beamformer and the attenuation of the signal depending on the distance from the source. The stability measure is defined as the coefficient of variation of the power spectral density at the signal frequency when the beamformer is pointed directly at the source ($x = 0$):

$$\mathrm{Stability}(\alpha) = \frac{\sqrt{E\{|Y(e^{\Omega_{sig}})|^4\} - E\{|Y(e^{\Omega_{sig}})|^2\}^2}}{E\{|Y(e^{\Omega_{sig}})|^2\}} \quad \text{with} \quad \Omega_{sig} = 2\pi\frac{f_{sig}}{f_s}, \quad x = 0 \tag{5.15}$$

Moreover, the attenuation measure is defined as the normalized magnitude in dB of the power spectral density of the beamformer output at the source frequency, depending on the distance from the source $x$:

$$\text{Attenuation}(x, \alpha) = 10 \log_{10} \left( \frac{|Y(e^{\Omega_{sig}}, x)|^2}{\max\{|Y(e^{\Omega_{sig}}, x)|^2\}} \right) \quad \text{with} \quad \Omega_{sig} = 2\pi \frac{f_{sig}}{f_s} \tag{5.16}$$

Those two measure were calculated for $\alpha \in \{10^{-6}, 10^{-3}, 10^{-1}, 1, 10\}$. For the stability measure the first three and last three blocks were ignored as they contained a strong jump which warped the measurements. The results can be seen in the figure below:

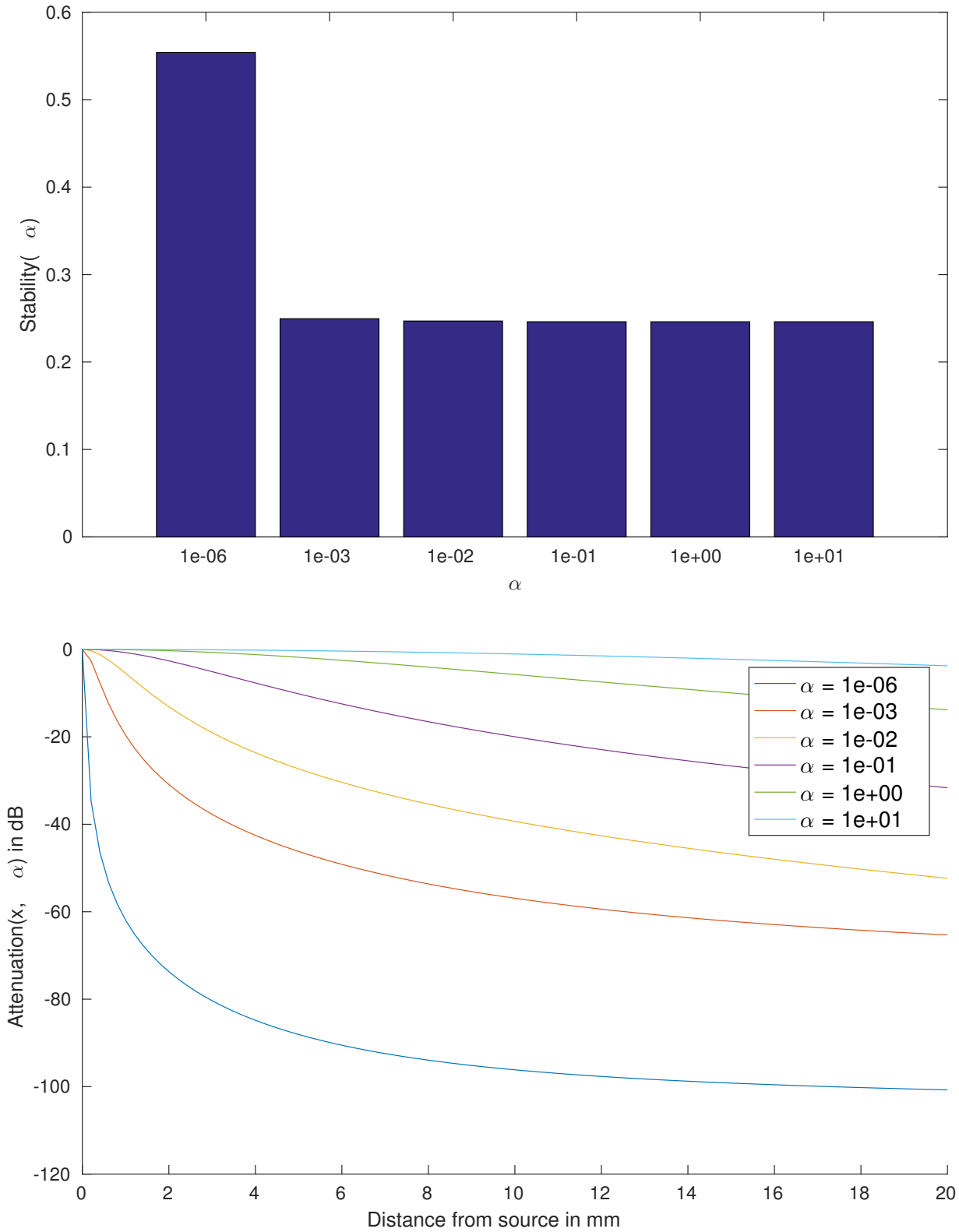


Figure 5.4: Results of the two measures using different stability parameters

These measurements clearly show that a lower stability parameter $\alpha$ results in stronger attenuation of distant signals, but also makes the amplitude of the signal deviate significantly. In simple terms, it can be concluded that it is possible to trade stability for spatial resolution with this

parameter. For the following evaluations the stability parameter was chosen to be $\alpha = 10^{-2}$, which represents a good compromise between stability and spatial resolution.

### 5.3.3 Spatial attenuation (MVDR)

To determine the spatial resolution of the beamformer, a more detailed analysis of the spatial attenuation for the selected stability parameter $\alpha = 10^{-2}$ is performed. The signal and noise source were placed as before, the SNR was set to 10:

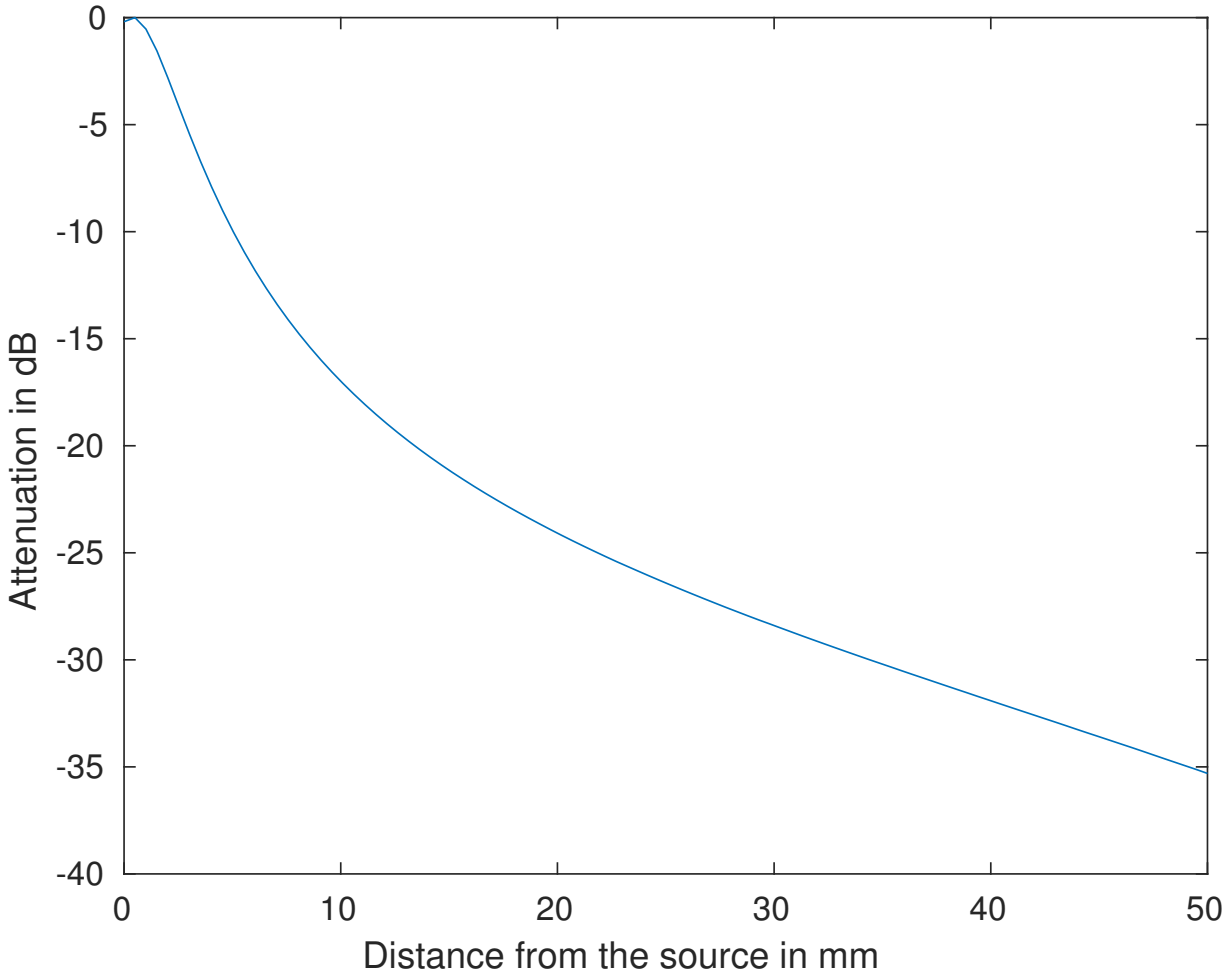


Figure 5.5: Spatial attenuation for $\alpha = 10^{-2}$

Note that this spatial attenuation is only an example, as the exact behaviour can depend on noise level and location of the signal.

If a power attenuation of $-20$ dB is sufficient, the maximum spatial resolution is about 15 mm. But this result is sensitively dependent on the select SNR. If one, for example, assumes that there is no noise, the maximum spatial resolution drops to about 5 mm. Since I do not have a realistic noise estimation, it is impossible to reach a final conclusion on this matter.

### 5.3.4 Spatial attenuation (quiescent)

Now the quiescent is subjected to the same test as defined above. The results can be seen here:

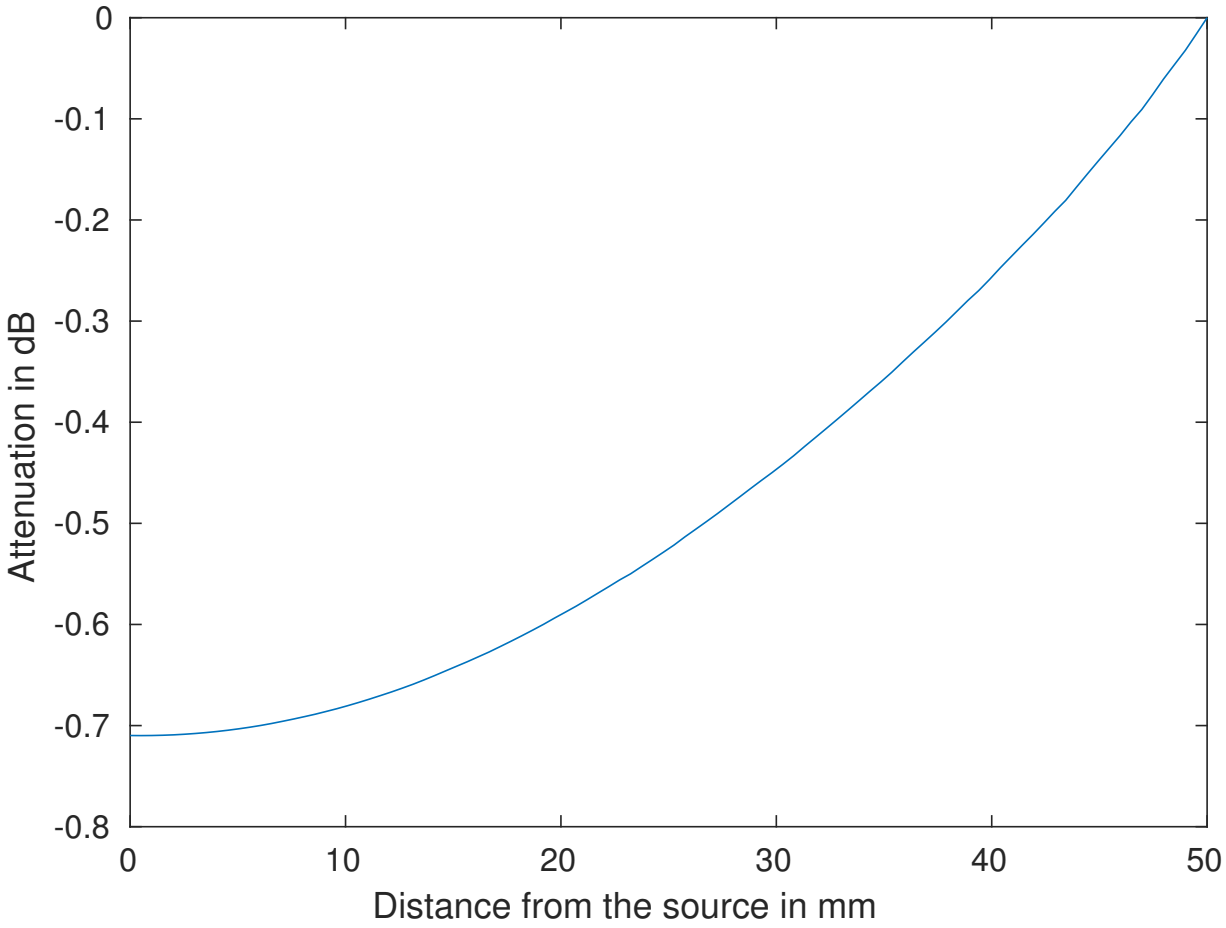


Figure 5.6: Spatial attenuation of the quiescent beamformer

It seems like the quiescent beamformer does not spatially separate the signal at all. On the contrary, the output signal turns out to be slightly higher, although it is multiple centimetres away from the source. This might be due to the specific nature of the lead field matrix or an implementation error on my part. In any case, the quiescent beamformer is not capable of solving the inverse problem.

### 5.3.5 Multiple sources

The last performed test investigates the beamformers behaviour when multiple sources are present. For this purpose, two sine sources are placed at $\boldsymbol{r}_1 = [0 \quad 0 \quad 0]$ and $\boldsymbol{r}_2 = [0 \quad 0 \quad 2]$, both with an orientation of $\boldsymbol{r}_{d,1} = \boldsymbol{r}_{d,2} = [0 \quad 0 \quad 1]$. The sources are set to oscillate at two different frequencies ($f_1 = 10\text{Hz}, f_2 = 15\text{Hz}$) so that the analysis can clearly separate the sources. The test is performed without noise. It measures the attenuation for both sources depending on the position on the z-axis:

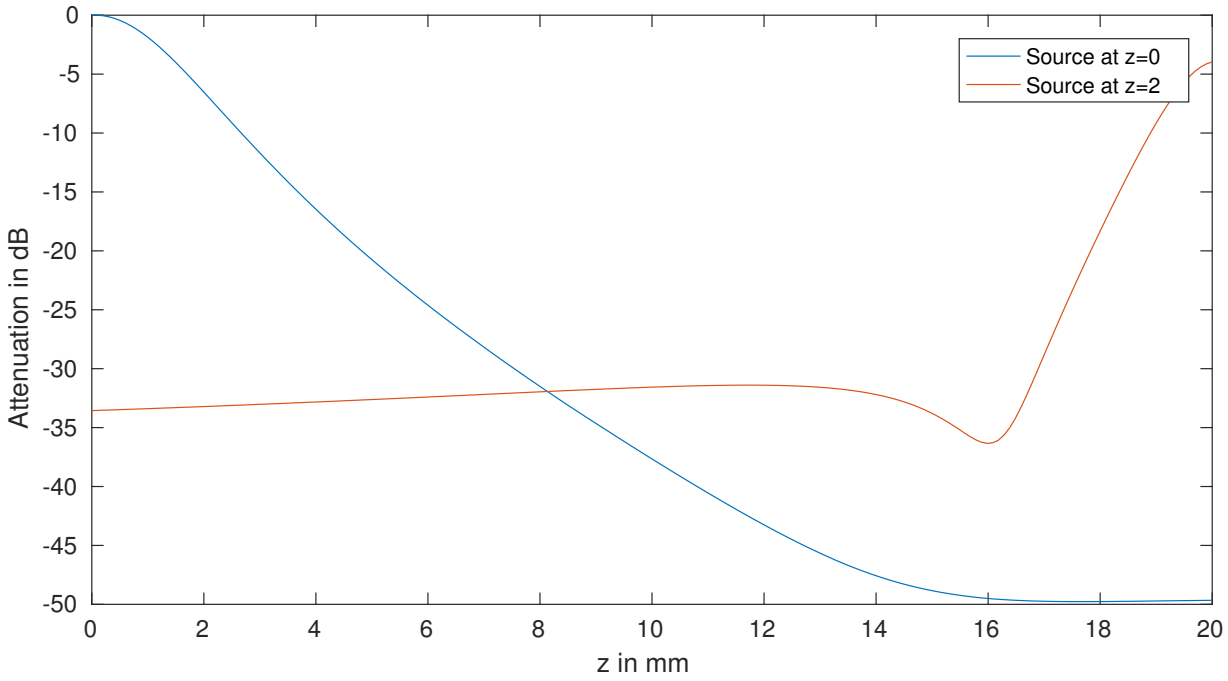


Figure 5.7: Spatial attenuation for two sources

This result illustrates how differently the attenuation curves can look for diverse positions in the brain. As it was planned to be, it displays that different frequencies do not influence one another.

### 5.3.6 Summary

All things considered, the MVDR beamformer works sufficiently well. It can clearly separate sources in the brain, although it is very susceptible to noise. The quiescent beamformer can not be used for that task.

# 6 Visualisation

## 6.1 3D electrode visualisation

The first visualisation displays the position of the EEG electrodes in 3D. Each electrode is displayed as a blue point together with its medical label. For easier orientation, the visualisation also renders a 3D brain model and a small explanatory image that were provided to me by Professor Schmidt.

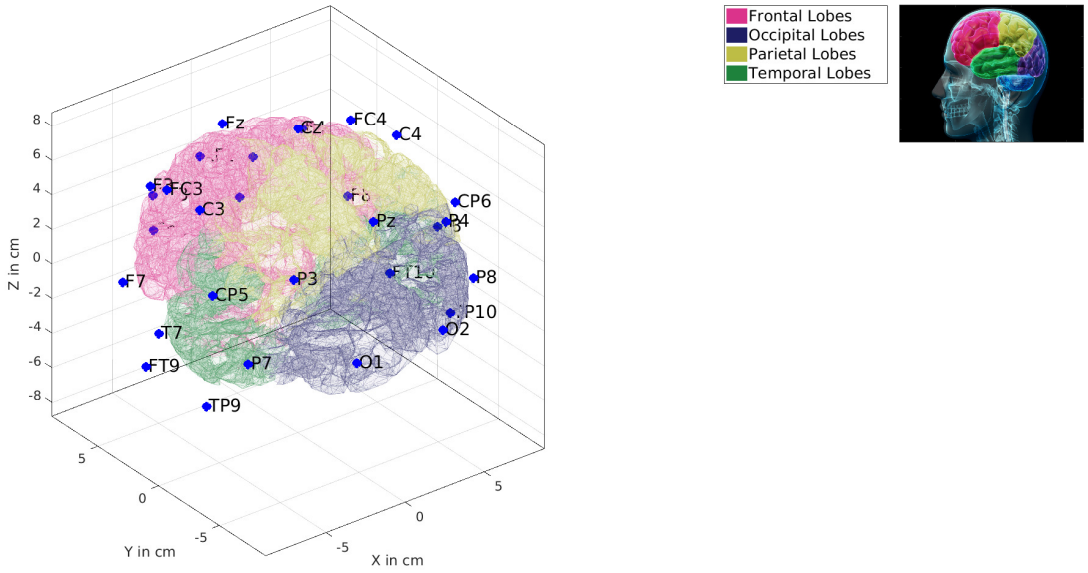


Figure 6.1: 3D Electrode visualisation

The model comes in eight different .mat files, which contain the vertices that the model is composed of. Each file contains a half (right or left hemisphere) of one of the four brain lobes, the frontal, occipital, parietal and temporal lobe. The different brain lobes are coloured according to the explanatory image. Then the whole model is rotated, scaled up and moved so that it fits the positions orientations of the electrodes. Finally, a small legend is rendered to explain the colours.

For demonstration purposes, the visualisation also has the option to render a short video of the model and the electrodes rotating. This way, a user can see more easily where the electrodes are located in 3D space.

## 6.2 Raw data visualisation

The raw data visualisation displays a number of 2D plots, showing data from all places in the program:

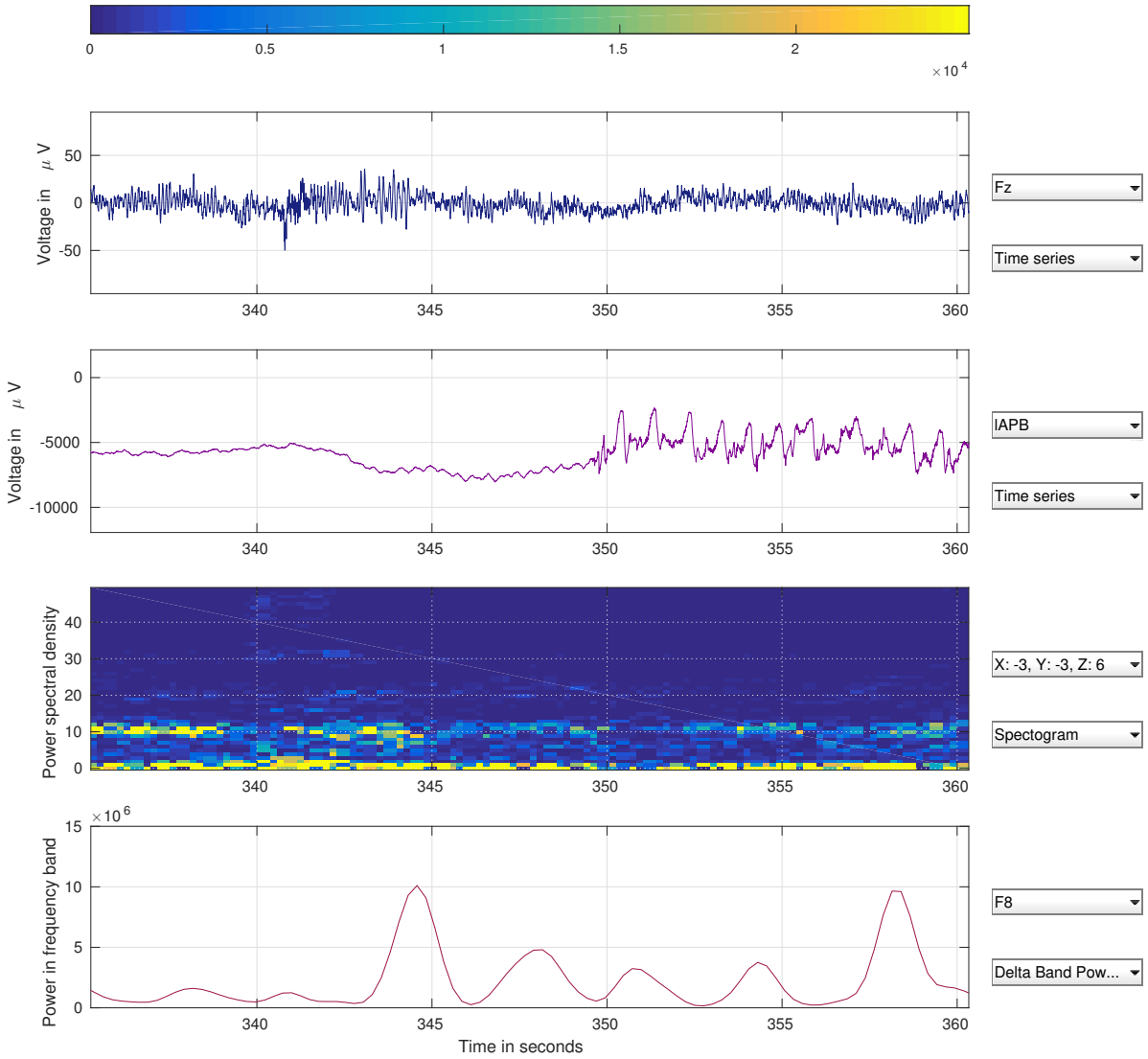


Figure 6.2: Raw data visualisation

All these plots share a single x-axis with the time on it. When the user zooms one plot in the x-axis, the other plots zoom to the same place. This makes it easier to compare different signals at the same point in time.

Next to the plots are two drop-down boxes to select the signal source and the plot type. The possible signal sources are the EEG electrodes, the EMG electrodes and the beamformer targets. The plot can display either the raw time-domain data, a frequency-domain spectrogram or a plot of the power in one of the four frequency bands (alpha, beta, theta and delta). The plot updates automatically when one of the drop-down boxes is changed.

All displayed spectrograms share a single color bar that is displayed above all the plots.

The wave band signals are low-pass filtered, because they fluctuate heavily. The filter is implemented as a convolution with an eight element Hann function.

To scale the y-axis of the time domain plots for EMG electrodes, the first and last ten percent of the data were ignored. Then the minimum and maximum of the remaining data were used as axis limits. Due to this approach, the deviations at the beginning and the end of the data were ignored. The y-axis of the EEG and beamformer target time domain plots were limited symmetrically to the 99 % quantile of the data points. This suppresses strong deviations from the mean of the data and makes it easier to view it.

Since there is no frequency domain data available for the EMG electrodes, the visualisation displays an error message when trying to render a spectrogram or a band power for an EMG electrode:

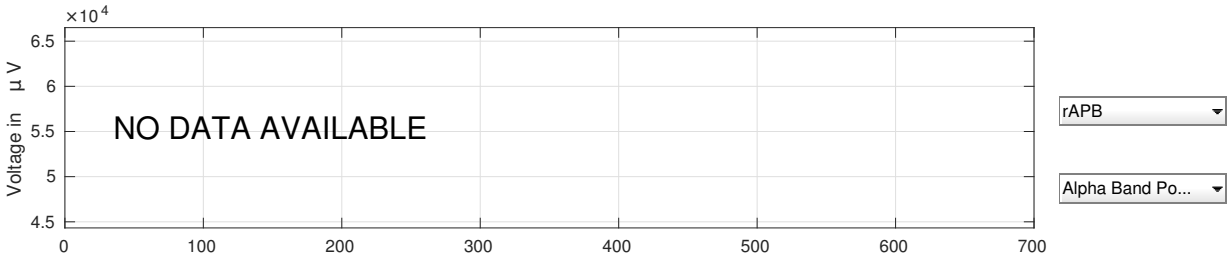


Figure 6.3: Error message for unavailable frequency domain data

The wave band and spectrogram plots were added late in the development, which is why they are not reflected in the figure 2.2. The raw data visualisation additionally needs access to all the data since these changes were implemented, which would have made the graphic very confusing.

## 6.3 Heatmap visualisation

The heatmap visualisation displays different frequency band powers (alpha, beta, theta and delta) as heatmaps. It uses the spectral data before the beamforming.

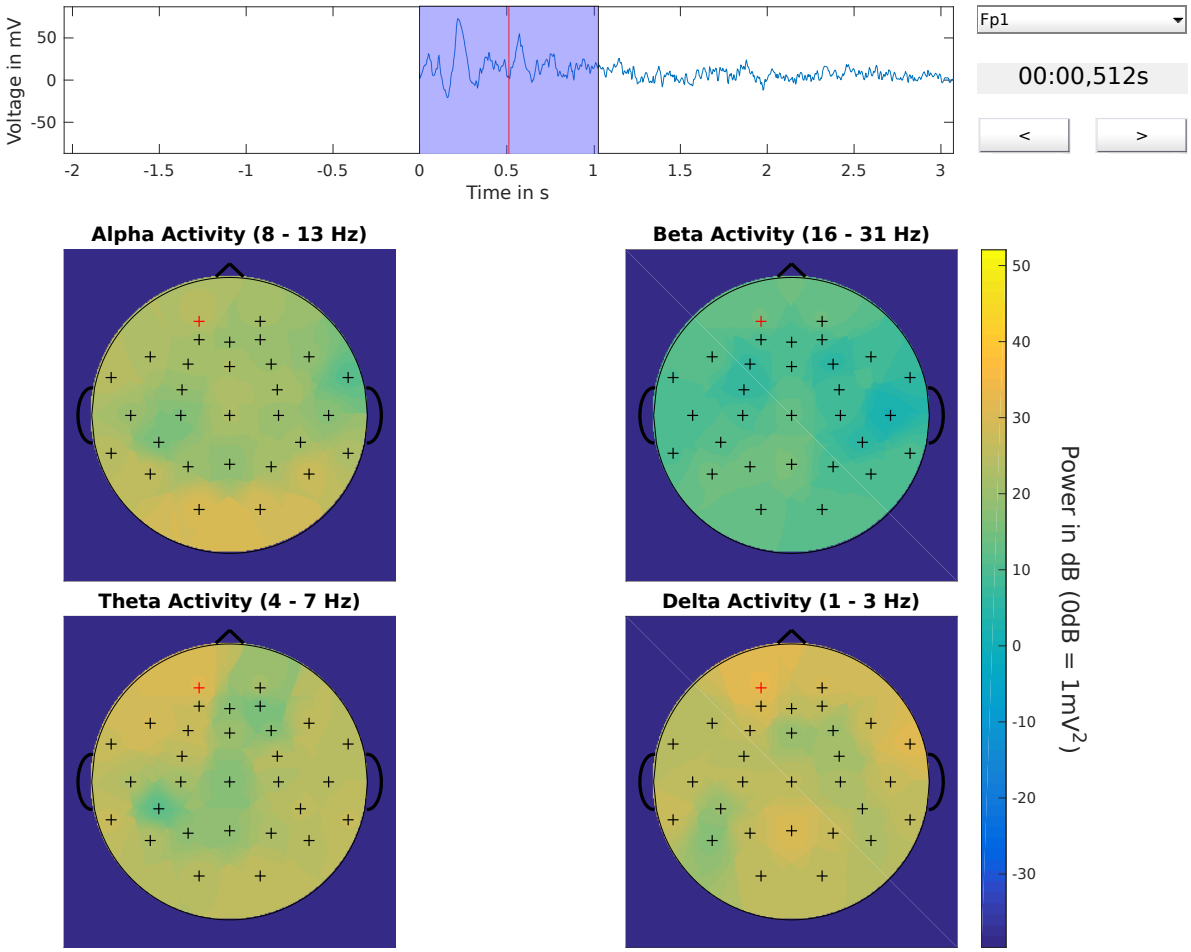


Figure 6.4: Heatmap visualisation

In the upper part of the visualisation, a raw time-domain signal is displayed. The electrode to be displayed can be changed by using the drow-down menu on the right. The blue box on top of the plot represents the current chunk, the red bar is the middle of the chunk.

Below the drop-down, there is a text that displays the current time and two buttons to skip one chunk ahead or backwards. The user can jump to an arbitrary point in time by clicking the time display and entering a time in the dialog box.

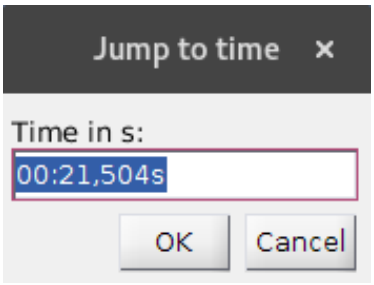


Figure 6.5: Dialog box to select any point in time

The main part of this visualisation's user interface consists of the four stylised heads with the

heatmaps on top. They display the power in decibels. The small crosses are the positions of the electrodes, the red one highlights the electrode currently selected by the drop-down box above. There is a small triangle on top of each head representing the nose and two circle sections representing the ears.

To draw the 2D headmap, two problems need to be solved: how to project the electrode positions from 3D into 2D and how to interpolate between multiple electrodes.

### 6.3.1 Stereographic projection

To display the electrode positions in 2D, a self-written stereographic projection is used. To apply it, the electrodes are converted into spherical coordinates (using MATLAB's definition, compare with figure 2.4). The projection function then uses the polar coordinate system to project the 3D coordinates. The (2D) radius is calculated by normalising the (3D spherical) polar angle of the electrode and the polar angle (2D) is determined by using the (3D) inclination angle:

$$r' = \frac{\pi}{2} - \arcsin(z) \tag{6.1}$$

$$\phi = \frac{\pi}{2} + \arctan\left(\frac{x}{y}\right) \tag{6.2}$$

Moreover, the radius is normalised:

$$r = \frac{0.9r'}{max\{r'\}} \tag{6.3}$$

This results in a maximum radius of 0.9, which is a good distance to the circle (representing the skull) at a radius of 1. Finally, the 2D polar coordinates are converted into Cartesian coordinates:

$$\begin{aligned} x &= -r\cos(\phi) \\ y &= r\sin(\phi) \end{aligned} \tag{6.4}$$

This time $x$ and $y$ mean the final 2D coordinates.

### 6.3.2 Interpolation

To display any data between the electrodes, the power has to be interpolated between the electrodes. Normally, one would do this with a polynomial interpolation (mostly linear or quadratic) or by spline interpolation. Unfortunately, both of these methods require a regular grid of sensor points. Since the electrodes are distributed unevenly across the head, this is not possible. Instead, I based the interpolation on the distance to the closest $g$ electrodes. For each selected electrode the interpolation weight $a_i$ is:

$$a'_i = \frac{1}{d_i^h} \tag{6.5}$$

$$a_i = \frac{a'_i}{\sum_{j=1}^{M} a_j} \tag{6.6}$$

$$\tag{6.7}$$

$d_i$ is the distance between the current position and the electrode $i$. The exponent $h$ is a parameter that has to be determined later. The higher $h$ is, the more distinct the edges become (see below). A higher value for $g$ means that more electrodes are considered for interpolation, which results in an overall smoother image. The final power $P_{dB}$ for each 2D point $\boldsymbol{r}_p$ is:

$$P_{dB}(\boldsymbol{r}_p) = \boldsymbol{a}\boldsymbol{P}_{dB} \quad \text{with} \quad \boldsymbol{a} = [a_1, a_2, \dots a_M] \tag{6.8}$$

$\boldsymbol{P}_dB$ is the vector of powers at each electrode measured in decibels. 0 dB represent a power of 1 $\text{mV}^2$. To find proper values for $g$ and $h$, multiple different values are compared below.

### g = 1, h = 1

With both $g = 1$, $h$ has no effect. Each point receives the same value as its closest electrode:

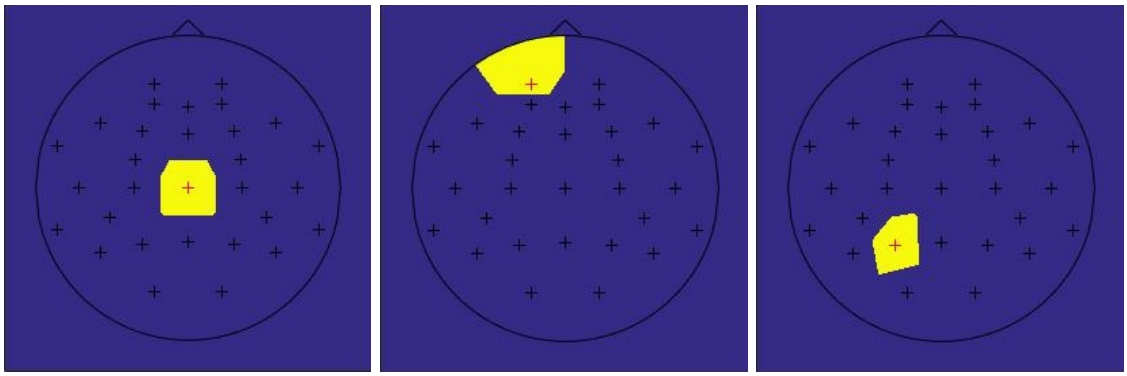

Figure 6.6: $a_i$ for different electrodes with $g = 1, h = 1$

These graphics display how much influence the currently selected sensor (red cross) has on each pixel of the graphic. This is done by displaying $a_i$ for the currently selected sensor on each graphic. All images share the same color scale:

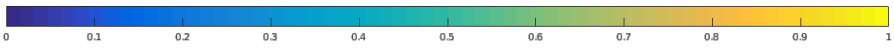


Figure 6.7: Color scale for $a_i$

Below, there is is an example of how the interpolation looks with these parameters:

This method of interpolation produces too hard borders between different areas.

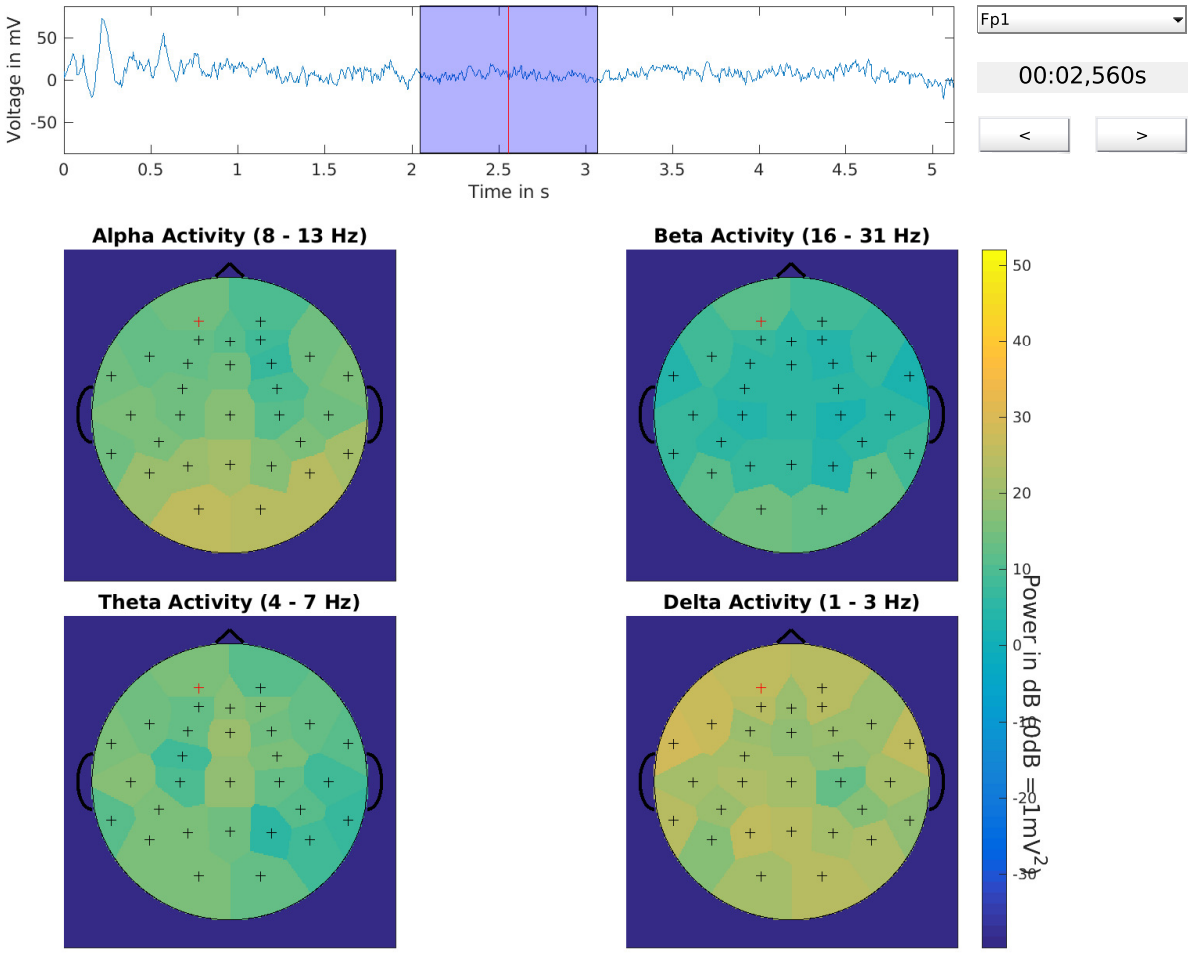


Figure 6.8: Realistic example for $g = 1, h = 1$

### g = 2, h = 1

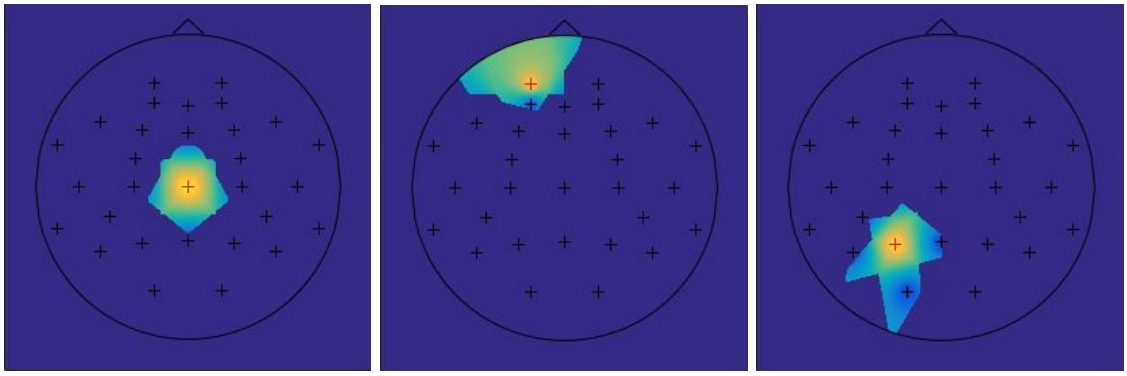

Figure 6.9: $a_i$ for different electrodes, for $g = 2, h = 1$

This method has smoother edges, but unfortunately it produces strong artefacts, which is why it is not sufficient for our purposes.

### g = 2, h = 2

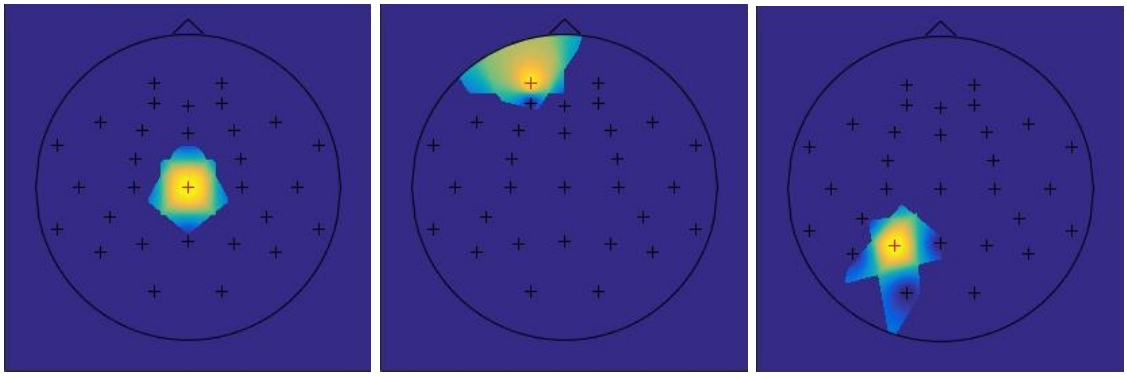

Figure 6.10: $a_i$ for different electrodes, for $g = 2, h = 2$

Compared to the last parameter set, this method has a larger central part with $a_i$ close to 1. But the artefacts still have too high values.

### g = 2, h = 3

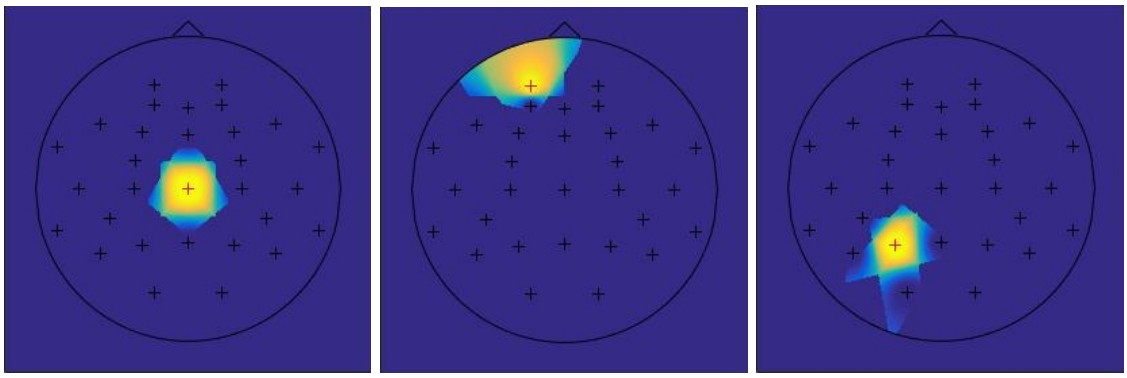

Figure 6.11: $a_i$ for different electrodes, for $g = 2, h = 3$

This parameter set also produces a large central section, but the artifacts are weaker than for the last method.

### g = 3, h = 2

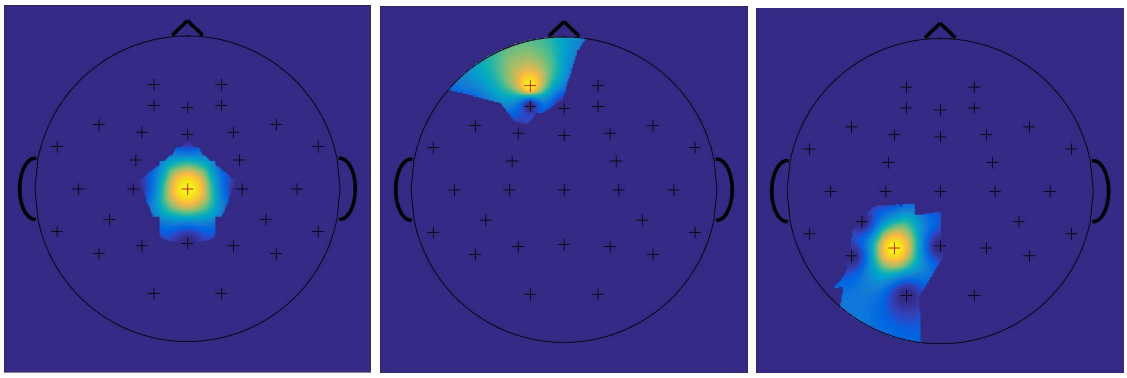

Figure 6.12: $a_i$ for different electrodes, for $g = 3, h = 2$

This configuration allows three electrodes to affect a single point. This seems to be too much, because a single electrode is able to affect a huge area, as can be seen in the third image.

### Conclusion

I selected $g = 2, h = 3$, because it produces the best compromise between smoothness, lack of artefacts and small area of effect.

## 6.4 3D arrow visualisation

The 3D arrow visualisation displays the dipole vectors generated by the beamformer in a specific frequency band. It renders them as arrows in 3D on top of the brain model:

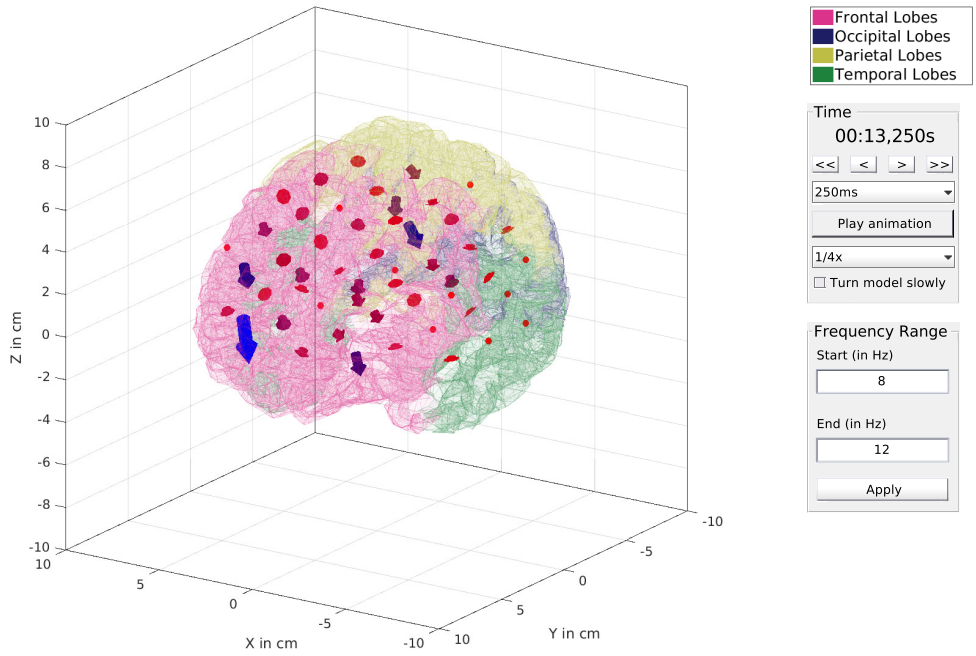


Figure 6.13: 3D arrow visualisation

For each electrode, the spectrum is averaged over the selected frequency range (in the case of the image, alpha band with 8-12 Hz). Then the absolute of this complex vector can be displayed as an arrow. The user can select a time in the right panel or activate the animation, which periodically advances the time by a fixed step size. It is also possible to rotate the model slowly by clicking on the checkbox in the right panel. This way, the user can see the evolution of the data. Additionally, the user can change the selected frequency range in the bottom right panel. The visualisation then takes a few seconds to recalculate the dipole vectors.

The 3D arrows are drawn using a function written published on the MATLAB file exchange[19], which I adapted for my needs.

To properly scale the arrows, each vector is divided by the maximum length of all vectors. If the arrow is shorter than 5 % of the maximum length, the arrow does not get rendered.

The colour $\boldsymbol{c}$ of each arrow is dependent on its length relative to the maximum length $l \in \{0 \dots 1\}$ and defined by the following saturation functions:

$$
\begin{aligned}
c_r &= \begin{cases} 1 & \text{for} \quad l < 0 \\ -2l + 1 & \text{for} \quad 0 \leq l \leq 0.5 \\ 0 & \text{for} \quad l > 0.5 \end{cases} \\
c_g &= 0 \\
c_b &= \begin{cases} 0 & \text{for} \quad l < 0 \\ 1.18l & \text{for} \quad 0 \leq l \leq 0.85 \\ 1 & \text{for} \quad l > 0.85 \end{cases}
\end{aligned} \tag{6.9}
$$

$c_r$ is the red component, $c_g$ the green and $c_b$ the blue component. A very long arrow is blue, a short one is red. These equations create a smooth colour transition when an arrow is growing longer or shorter.

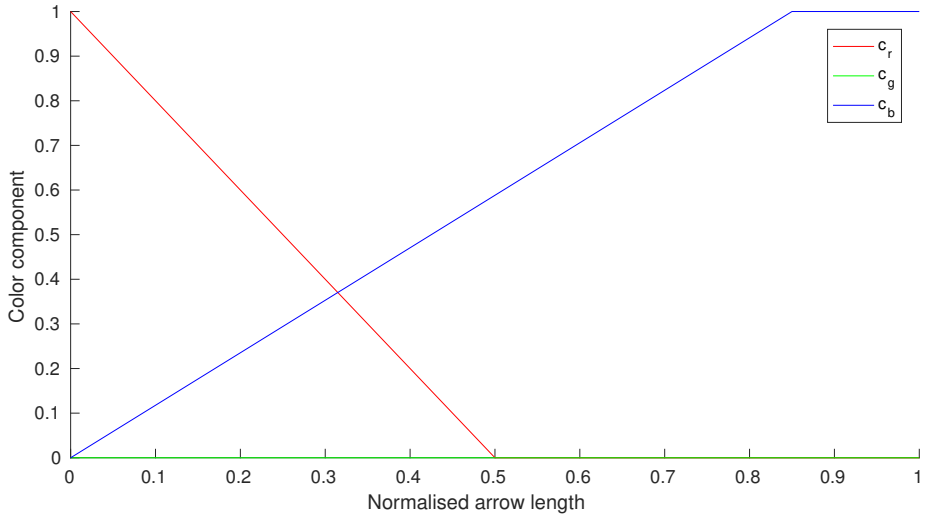


Figure 6.14: Colour of the arrows depending on their length

To provide a smooth transition between two chunks, the dipole arrows are linearly interpolated between two chunks. For a point in time $t$ between two chunks at $t_1$ and $t_2$ the dipole vector $\boldsymbol{d}(t)$ is defined as:

$$
\boldsymbol{d}(t) = a\boldsymbol{d}(t_1) + (1 - a)\boldsymbol{d}(t_2) \quad \text{with} \quad a = 1 - \frac{t - t_1}{t_2 + t_1} \tag{6.10}
$$

The interpolation and colour functions provide a smooth look for the animation, which can also be recorded as a video.

## 6.5 Cross-correlation and coherence visualisation

The final visualisation displays the cross-correlation and coherence between two beamformer targets:

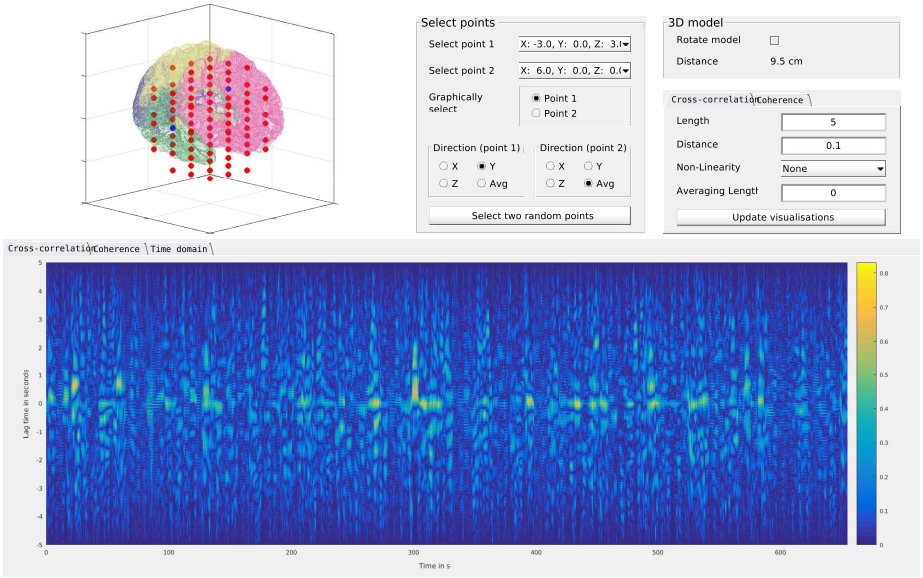


Figure 6.15: Cross-correlation and coherence visualisation

To select the points to cross-correlate, the user can either pick them from the select lists in the middle panel or use the data cursor tool and click on them in the 3D model. The point that should be selected next through the use of the 3D model, can be selected in the middle panel as well. It changes every time the user selects a point in the 3D model. When clicking on a point in the 3D model, a message displaying the coordinates of the point is displayed for a short time. After that, the point that should be selected next changes.

When the user clicks on anything else in the 3D model a short error message is displayed:

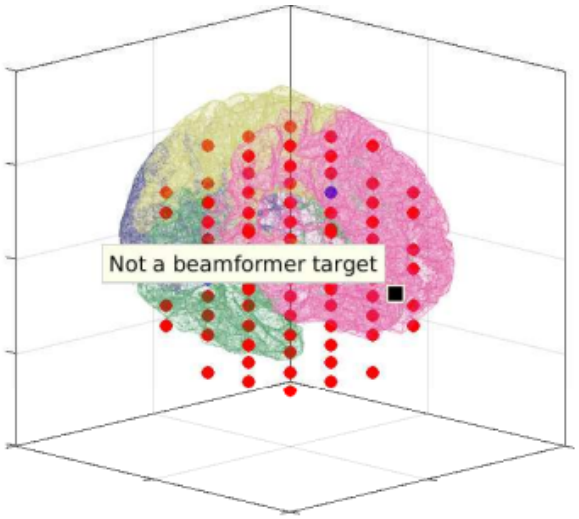


Figure 6.16: Point selection error message

Since each point has three channels, one for each direction, the user can also select which directions to correlate. This is done by using the checkboxes in the bottom of the middle panel. There is also an option to correlate the average of the three directions.

Furthermore, there is a button to select two random points in the bottom of the middle panel.

In the 3D model panel, the distance between the two selected points is displayed. The panel also contains a checkbox to let the model slowly rotate. It is useful for user to visualise the selected beamformer target positions.

The bottom right panel is split up into two tabs, one contains the options for the cross-correlation, the second contains those for the coherence. Both accommodate a button to recalculate the results displayed in the bottom half of the visualisation.

The cross-correlation has four options:

- The length of the cross-correlation determines the maximum shift between the signals that should be calculated. It regulates the length of the y-axis.
- The distance between two cross-correlations defines how often a new calculation should be started, thereby setting the resolution for the x-axis.
- The non-linearity can optionally be applied before the two signals are cross-correlated. This can be useful if one wants to correlate the power of the signals. The available options are square, absolute, logarithm or no non-linearity.
- The averaging length is used to apply a moving-average filter to the signals. If the user wants to correlate the power of the signals, this can be used after the square non-linearity to smooth out the high frequency oscillations of the power. The averaging length is measured in seconds and determines how many samples should be considered for averaging. It is set to zero per default.

The whole process of the cross-correlation looks like this:

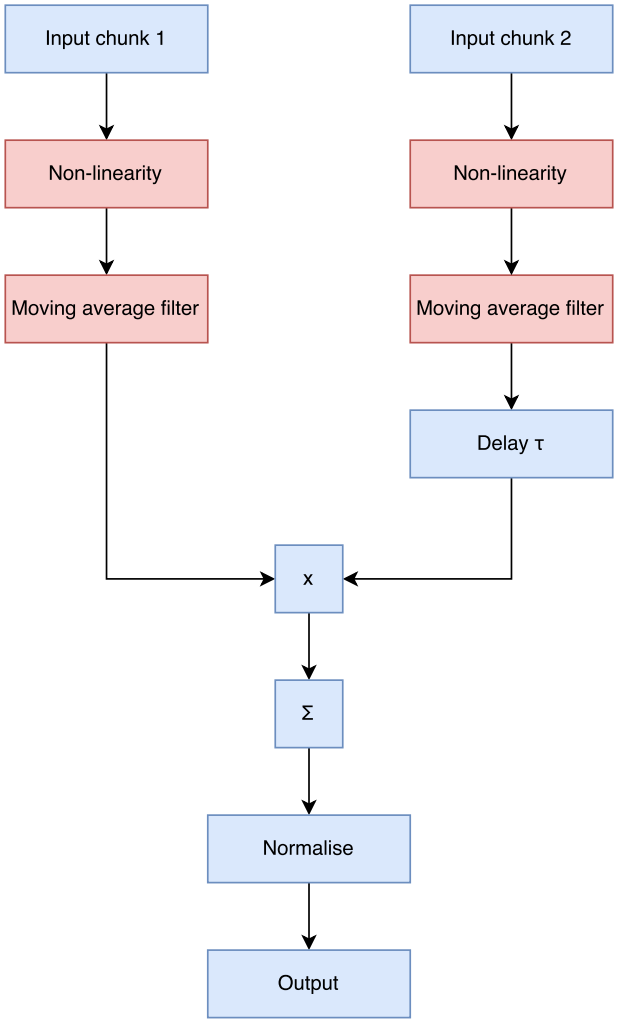


Figure 6.17: Outline of the cross-correlation algorithm

The red boxes (non-linearity and moving average filter) are optional. The normalised cross-correlation $s_{v_1,v_2}$ between the input signals $v_1$ and $v_2$ is defined as:

$$s_{v_1,v_2}(\kappa) = \sum_n \frac{(v_1(n) - E\{v_1(n)\})(v_2(n+\kappa) - E\{v_2(n)\})}{\sigma_{v_1}\sigma_{v_2}} \tag{6.11}$$

The options for the coherence are in the second tab of the bottom right panel. The user can change the averaging method and length there. The coherence is a similar measure like the cross-correlation but in the frequency domain. It needs to average over several chunks of frequency domain data to obtain meaningful results.

The available averaging methods are the unweighted mean, which is equal to the normal average, and the Hann-weighted mean. The Hann-weighted mean is calculated by element-wise multiplication of the input signals with a Hann-function.

The coherence $C_{U,V}$ between two spectra $U$ and $V$ is defined as:

$$C_{U,V} = \frac{|E\{U^*V\}|^2}{E\{|U|^2\}E\{|V|^2\}} \tag{6.12}$$

The bottom half of the visualisation is filled by the data displays. There are three tabs selecting which data should be displayed. All tabs share a linked time x-axis, just like in the raw data visualisation (see 6.2). The first tab contains the cross-correlation plot, which displays the value of the cross-correlation as colour and the lag-time as y-axis. The second tab contains the coherence plot with the frequency as a y-axis and the value of the coherence displayed as colour. The final tab displays the two time domain signals at the beamformer targets. Its y-axis is the dipole moment.

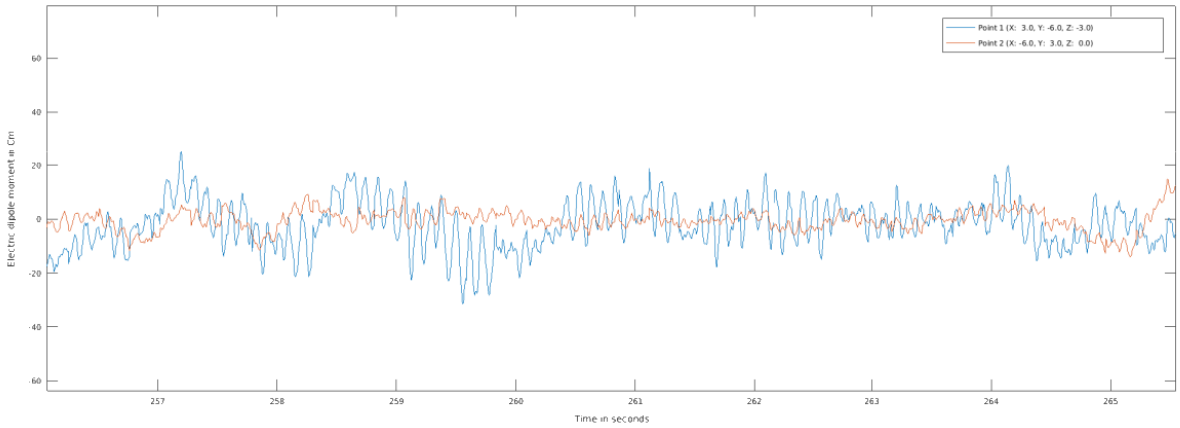


Figure 6.18: Time domain data display

# 7 Evaluation

To evaluate the software, it is tested whether it can detect the finger tapping, which was performed during the measurements (see 1.3). According to [7] and [8], one should be able to see a drop in alpha band power in the upper motor cortex as soon as finger tapping is starting. This should be detectable at the electrodes C3 and C4.

## 7.1 Manual evaluation

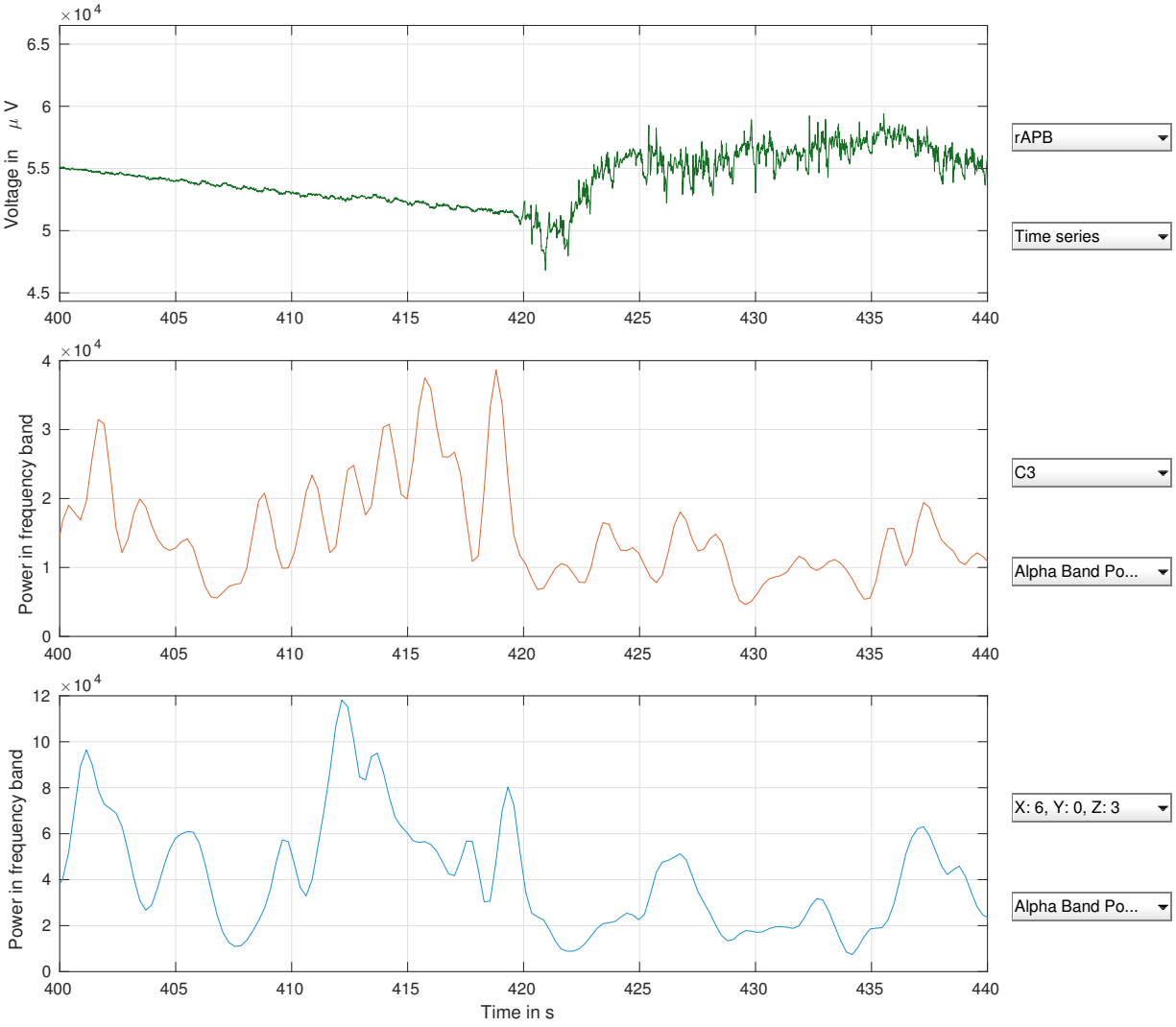


Figure 7.1: Example of alpha band activity in the motor cortex

In the top graphic above above, an electromyogram is displayed. The label rAPB means right abductor pollicis brevis, which is the Latin word for the big muscle in the hand that pulls on the thumb. At about 420 seconds one can see a sudden change in the EMG. This is when the patient starts tapping his or her finger. Each subsequent minimum is a single tap.

About one second before the tapping begins, a sudden drop in alpha activity is detected at the

C3 electrode (middle graph). This electrode is located above the left motor cortex and since the right finger is connected to the left hemisphere, this is where one would expect a drop in activity. A similar pattern can be seen in the bottom graphic that displays the alpha band power at a nearby beamformer target ($\boldsymbol{r} = [6 \quad 0 \quad 3]$).

Unfortunately, this behaviour can not be detected every time the patient starts tapping. Instead, the alpha band power permanently fluctuates wildly, as can be seen in the graph below.

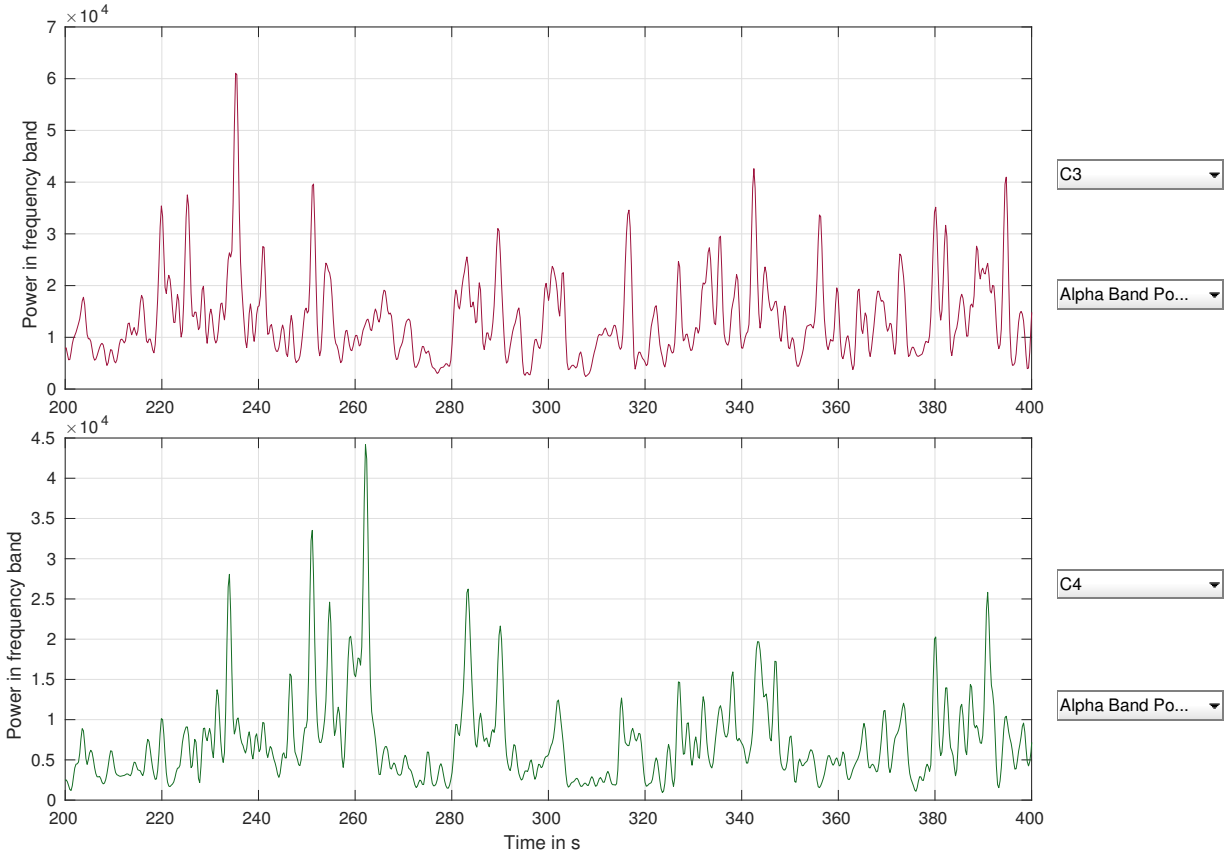


Figure 7.2: Alpha band activity over a wider time range

A subjective test was performed on one data set to receive some statistical data about the phenomenon. The first data set 20080521ak_subject02 was analysed by comparing the EMG data (rAPB and lAPB) to the alpha band power of the EEG data (C3 and C4). When the tapping at one finger starts, the corresponding electrode is checked for a drop in alpha activity.

The result is rated as an integer between zero and three:

- 0: no drop in alpha activity
- 1: ambiguous, more likely no drop in alpha activity
- 2: ambiguous, more likely a drop in alpha activity
- 3: alpha activity dropped

| **Left** | | **Right** | |
|---|---|---|---|
| Time frame | Result | Time frame | Result |
| 97 – 135 s | 3 | 33 – 60 s | 1 |
| 167 – 227 s | 3 | 97 – 198 s | 2 |
| 295 – 323 s | 3 | 267 – 294 s | 1 |
| 349 – 386 s | 2 | 323 – 349 s | 1 |
| 421 – 448 s | 3 | 421 – 485 s | 3 |
| 485 – 523 s | 2 | 581 – 608 s | 0 |
| 553 – 580 s | 3 | 640 – 650 s | 3 |
| Average | 2.71 | Average | 1.57 |

Table 7.1: Results of the subjective test

Interestingly, the results for left and right hand differ significantly. On the left side, one can see a clear correlation between alpha power drop and finger tapping (71% positive results, 29% ambiguous and 0% negative results), whereas the results on the right side are inconclusive (29% positive results, 58% ambiguous and 14% negative results). The disparity between the right and left hand could either be a coincidence or an effect of right- and left-handedness.

To receive more conclusive results, one would have to analyse more data manually or average over multiple trials automatically.

## 7.2 Averaging over multiple trials

Since analysing the raw data did not conclusively detect the expected patterns, I hope averaging the results across multiple trials within a data set will improve the results. For this purpose, one marks the beginning of each tapping trial, which is then used to extract multiple sections from the data around the markers. These sections can then be averaged, hopefully removing all brain activity that is unrelated to the finger tapping.

### 7.2.1 Positioning the markers

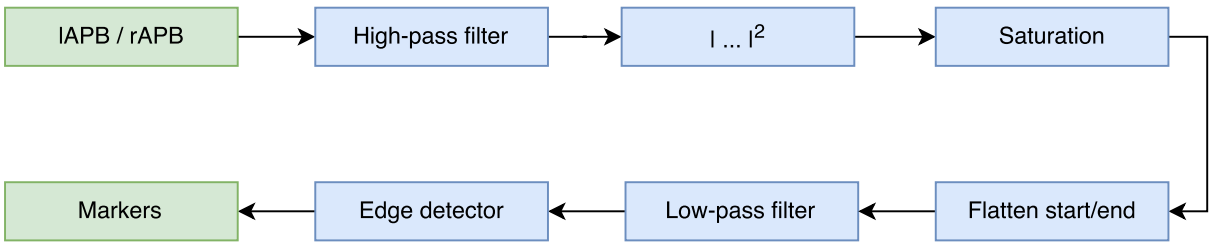


Figure 7.3: Outline of the marker placement algorithm

To detect the beginning of each tapping trial, the electromyogram data of the thumb (rAPB and lAPB) is analysed. Initially, the data is high-pass filtered to remove the sensor drift (see top graphic in figure 7.1). The high-pass filter is designed with the Butterworth method at a cutoff frequency of $f_{HP} = 2$ Hz and has five coefficients. It is applied using a zero-phase method so

that only the magnitude of the signal is changed. This is done to preserve the exact point in time when the tapping starts.

Then the power of the data is calculated by squaring the absolute of every data point. The magnitude of all data points is limited to the 90 % quantile of the data points, to remove high amplitude flukes.

Since there are strong distortions at the beginning and at the end of the data, the first and last 3 % of the data are flattened by setting them to a constant value.

Subsequently, the data is low-pass filtered by applying a moving average filter with four seconds length to it. This transforms the fast oscillations of the finger tapping into a continuous high value. The low-pass filter, like the high-pass filter before, is applied using a zero-phase filtering method.

Finally, the algorithm detects when the resulting function $s(n)$ is rising above a limit of half its average. This is done by subtracting the limit from the data and then detecting positive sign changes:

$$a(n+1) - a(n) > 0 \quad \text{with} \quad a(n) = \operatorname{sgn}\left(s(n) - \frac{E\{s(n)\}}{2}\right) \tag{7.1}$$

sgn is the sign function, which is defined as:

$$sgn(x) = \begin{cases} -1 & \text{for} \quad x < 0 \\ 0 & \text{for} \quad x = 0 \\ 1 & \text{for} \quad x > 0 \end{cases} \tag{7.2}$$

A marker is only assumed valid if the next sign change is farther away then ten seconds. This prevents accidental movements from being counted as the beginning of a trial.

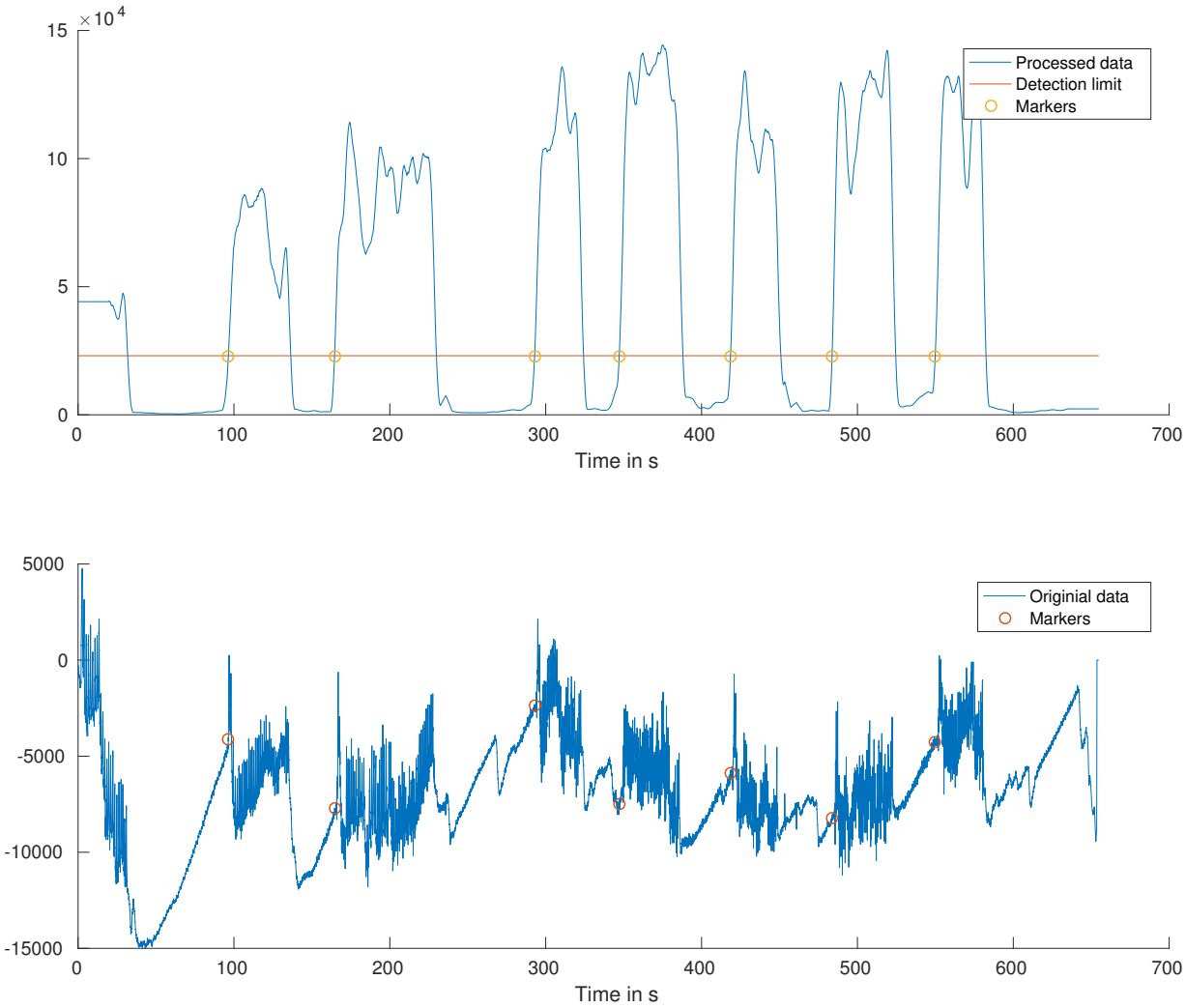


Figure 7.4: Example of marker placement results

In the top graphic, the processed EMG data is displayed. The red line represents the detection limit of $0.5 \cdot E\{s(n)\}$. The bottom graphic shows the unaltered EMG data and the resulting markers.

To evaluate the detection algorithm, its results are compared to the manual analysis:

| **Left** | | **Right** | |
|---|---|---|---|
| Manual | Automatic | Manual | Automatic |
| 97 s | 96.4 s | 33 s | 29.5 s |
| 167 s | 164.5 s | 97 s | 95.0 s |
| 295 s | 292.9 s | 267 s | 264.5 s |
| 349 s | 347.2 s | 323 s | 320.8 s |
| 421 s | 418.6 s | 421 s | 418.2 s |
| 485 s | 483.7 s | 581 s | 578.7 s |
| 553 s | 549.9 s | 640 s | Not detected |
| Average distance | 1.97 s | Average distance | 2.55 s |
| Standard deviation | 0.82 s | Standard deviation | 0.54 s |

Table 7.2: Results of the subjective test

The discrepancies between both methods are due to an initial spike in the EMG data, that occurs before each trial. This spike is counted by the algorithm as the beginning of the trial, opposed to the manual analysis, where it was ignored. But since the difference has a low variation, it should not influence the end result significantly.

The last trial for the right hand could not be detected by the algorithm, because it is part of the last 3 % of the data, which are excluded from the detection.

### 7.2.2 Averaging

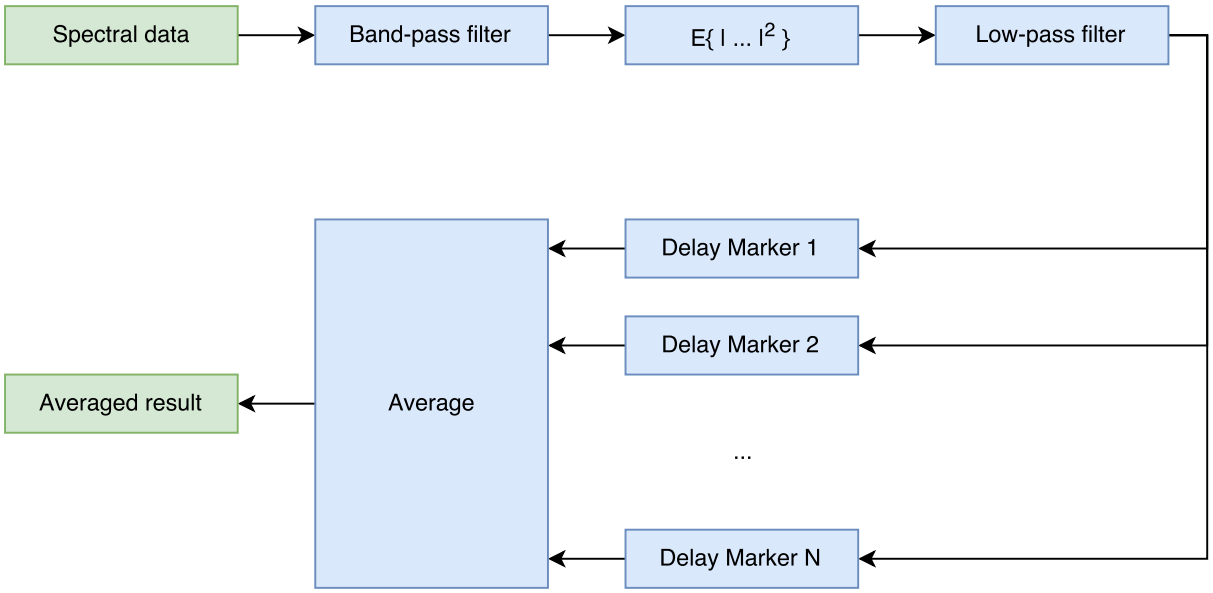


Figure 7.5: Outline of the averaging algorithm

To remove all brain activity that is independent of the finger tapping, the alpha band power is averaged at the markers. This is achieved by calculating the alpha band power the same way as in figure 7.1. The spectral data at the electrodes C3 and C4 is band-pass filtered to remove any non-alpha activity. Then its wave band power is calculated by absolute squaring it and averaging

it over the alpha band range. To remove any high-frequency, noise the power is low-pass filtered using a 20 element moving-average filter.

The resulting alpha band power is then split into sections around the markers. For this purpose, the closest spectral chunk to each marker position is found. This chunk is grouped with a number of chunks directly adjacent to it to form a short section of the total signal, which can be modeled as delaying the signal by the negative time of the marker. The length of the sections is based on the length of the shortest trial to assure that no data outside the trials is fed into the averaging process at the end.

Finally, the resulting signals are averaged, which removes the unwanted brain activity.

The same algorithm can be applied to beamformed signals instead of the EEG sensor data. For this purpose, the beamformed signals have to be averaged across their dipole directions.

### 7.2.3 Results and conclusion

The resulting average of the alpha band powers at the electrodes C3 (for the right hand) and C4 (for the left hand) is displayed below. At $t = 0$ the tapping starts.

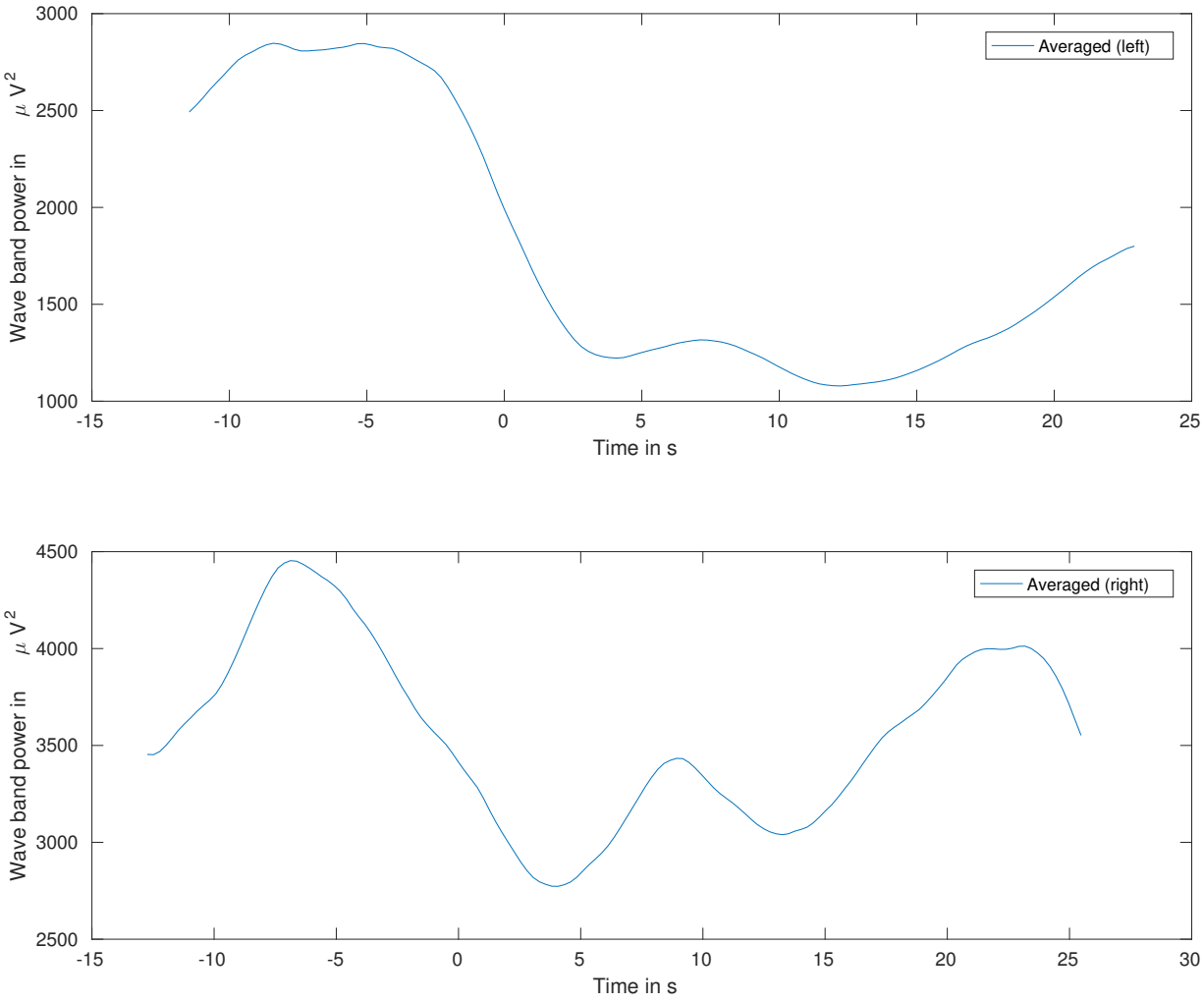


Figure 7.6: Averaged alpha band powers

In the graphic above one can see the drop in alpha activity that was predicted by the literature. It was inconclusive for a single trial, because unrelated brain activity was influencing the results.

To assure that the processing is working as expected, the resulting data after the beamforming is analysed. If everything works as desired, the same pattern should be visible in this data. The beamformer targets closest to the electrodes C3 ($\boldsymbol{r} = [-6 \quad 0 \quad 3]$) and C4 ($\boldsymbol{r} = [3 \quad 0 \quad 6]$) are selected.

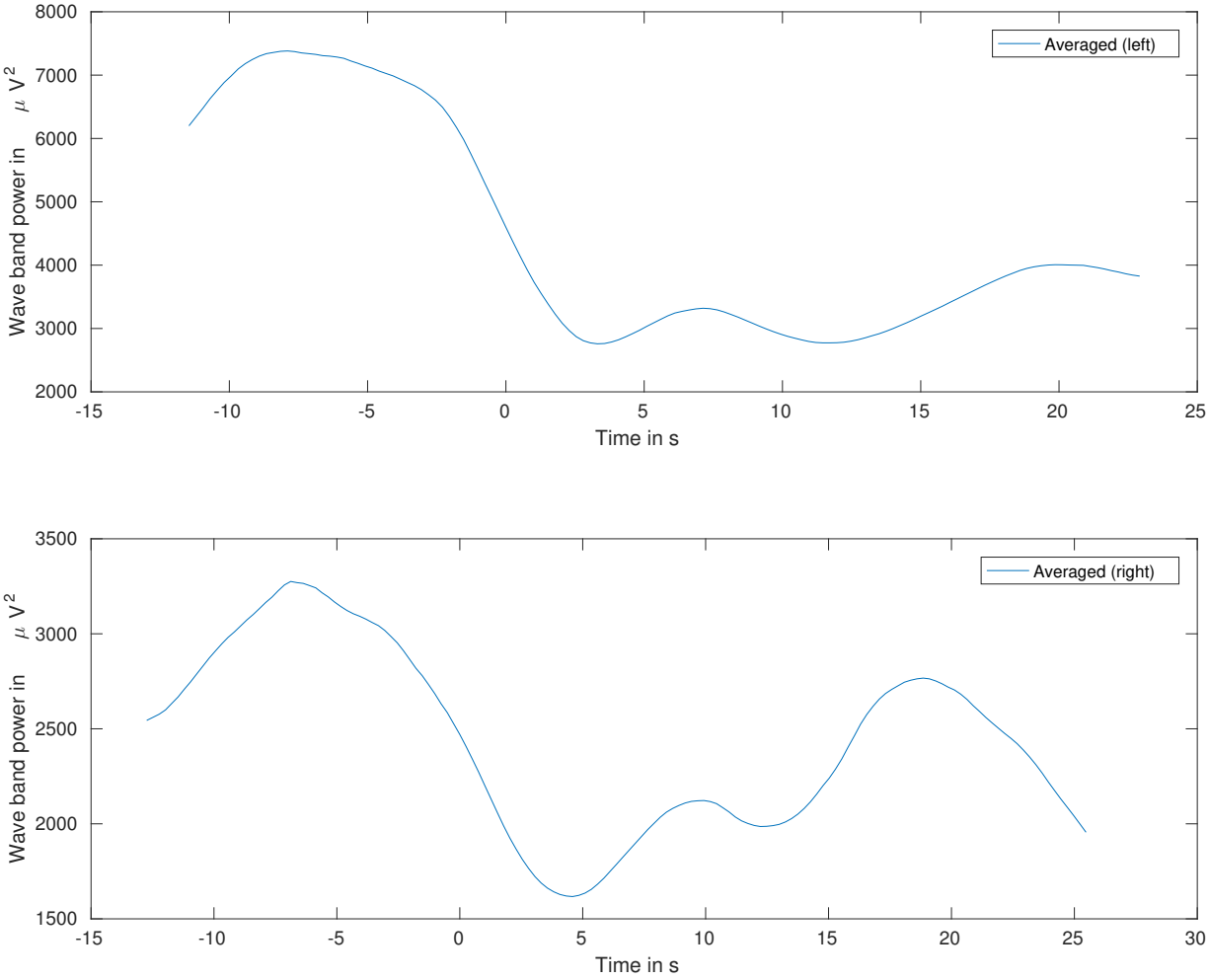


Figure 7.7: Averaged alpha band powers for the beamformed data

The two figures above show that the same pattern can be observed in the beamformed data. This is a clear indication that the processing works and that it can be used for further research.

# 8 Outlook

There are some things that could be done to improve the results of this software:

- Due to the crude nature of approximating the brain as four concentric spheres, the forward model has room for improvements. The paper by Johannes Vorwerk et al. [5] suggests that distinguishing gray and white matter improves the forward model noticeably, which would require an actual three-dimensional map of the brain. A finite element method might work for this.

- The averaging process (see 7.2) is currently not integrated into the main program. Also, the selected electrodes (C3 and C4) are hardcoded and can not be changed without changing the source code. It might be useful to integrate this process as a visualisation into the main program.

- The performance of the program could be hugely improved by using parallelisation in the beamformer. The beamformer has to run once for every frequency, beamformer target and direction for each chunk of data. In theory, these calculations are independent from one another, yet currently they run on a single CPU core in sequence. By running the program in multiple threads, it would be possible to balance the workload over multiple cores, resulting in a massive performance gain. An additional improvement might be obtained by running the program on a graphics card, which would allow for an even higher parallelisation. MATLAB provides a toolbox for these purposes called the *Parallel Computing Toolbox*. Unfortunately, I did not have access to said toolbox during the development of the software.

- Currently, the program assumes that the frequency bins are 1 Hz away from one another. This is only the case when the FFT window length is equal to one second.

- At the time of writing this work it is still debated whether EEG data can make accurate statements about the processes in the inner brain. It is possible that the data received from beyond the cortex (the surface of the brain) is not meaningful in any way. Therefore, it might be prudent to restrict the beamformer targets only to the cortex, which would require a more sophisticated algorithm than just placing them in a regular grid.

- If required, the algorithm could be implemented in a real-time environment like KiRAT (**Ki**el **R**eal-time **A**pplication **T**oolkit). MATLAB is not equipped for real-time analysis, but since the processing step is already implemented in pseudo-real-time, it will be easy to translate the software to C++ or something similar.

- As stated in [4], it might be useful to implement a different beamformer than the MVDR algorithm.

# 9 Appendix

## 9.1 Lagrange function gradient

The gradient of the following Lagrange function should be determined:

$$\mathcal{L} = \boldsymbol{w}^T \boldsymbol{R_{ss}} \boldsymbol{w}^* - \lambda(\boldsymbol{w}^T \boldsymbol{c} - 1) \tag{9.1}$$

At first, the gradient is split into its vector components:

$$\begin{aligned} \underset{\boldsymbol{w}}{\nabla}\mathcal{L} = & \sum_{i=1}^{M} \left[ \frac{\partial \mathcal{L}}{\partial w_i} \boldsymbol{e}_i = \sum_{i=1}^{M} \frac{\partial}{\partial w_i} \left( \boldsymbol{w}^T \boldsymbol{R}_{ss} w_i^* \right) - \frac{\partial}{\partial w_i} \lambda(\boldsymbol{w}^T \boldsymbol{c} - 1) \boldsymbol{e}_i \right] \\ = & \sum_{i=1}^{M} \frac{\partial}{\partial w_i} \begin{bmatrix} w_1 w_i^* \\ \vdots \\ w_i w_i^* \\ \vdots \\ w_M w_i^* \end{bmatrix}^T \boldsymbol{R}_{ss} - \lambda \frac{\partial}{\partial w_i} \begin{bmatrix} 0 \\ \vdots \\ w_1 c_1 + \cdots + w_i c_i + \cdots - 1 \\ \vdots \\ 0 \end{bmatrix}^T \end{aligned} \tag{9.2}$$

Now the solution from appendix 9.2 are inserted into the equation:

$$\underset{\boldsymbol{w}}{\nabla}\mathcal{L} = \sum_{i=1}^{M} \begin{bmatrix} 0 \\ \vdots \\ \frac{w_i^*}{4} \\ \vdots \\ 0 \end{bmatrix}^T \boldsymbol{R}_{ss} - \lambda \begin{bmatrix} 0 \\ \vdots \\ c_i \\ \vdots \\ 0 \end{bmatrix}^T = \frac{1}{4} \boldsymbol{w}^H \boldsymbol{R}_{ss} - \lambda \boldsymbol{c}^T \tag{9.3}$$

## 9.2 Complex derivatives

The complex derivatives of three functions have to be calculated:

$$\frac{dw_i^*}{dw_i} \tag{9.4}$$

$$\frac{d\left(w_i w_i^*\right)}{dw_i} \tag{9.5}$$

$$\frac{dw_i}{dw_i} \tag{9.6}$$

The complex derivative is defined as a Wirtinger derivative for a complex variable $z = a + jb$:

$$\frac{d}{dz} = \frac{1}{2}\left(\frac{\partial}{\partial a} - j\frac{\partial}{\partial b}\right) \tag{9.7}$$

So the first derivative (9.4) can be solved as:

$$\frac{dz^*}{dz} = \frac{1}{2}\left[\frac{\partial(a+jb)}{\partial a} - j\frac{\partial(a-jb)}{\partial b}\right] = \frac{1}{2}\left[1+jj\right] = 0 \tag{9.8}$$

The second derivative (9.5) equates to:

$$\frac{d\left(zz^*\right)}{dz} = \frac{d}{dz}\left[a^2+b^2\right] = \frac{1}{2}\left[\frac{\partial\left(a^2+b^2\right)}{\partial a} - j\frac{\partial\left(a^2+b^2\right)}{\partial b}\right] = \frac{1}{2}\left[\frac{a}{2} - j\frac{b}{2}\right] = \frac{1}{4}z^* \tag{9.9}$$

Finally, the third derivative is equal to:

$$\frac{dz}{dz} = \frac{1}{2}\left[\frac{\partial(a+jb)}{\partial a} - j\frac{\partial(a+jb)}{\partial b}\right] = \frac{1}{2}\left[1-jj\right] = 1 \tag{9.10}$$

# List of Figures

# List of Tables

# Bibliography


[1] Rainer Klinke, Hans-Christian Pape, Armin Kurtz, Stefan Silbernagel: *Physiologie* (6. edition)

[2] B. Neil Cuffin, David Cohen: *Comparison of the Magnetoencephalogram and Electroencephalogram*, Electroencephalography and Clinical Neurophysiology, 1979 3.1, 3.2, 3.4, 3.1

[3] Yehuda Salu, Leonardo G. Cohen et al.: *An Improved Method for Localizing Electric Brain Dipoles*, IEEE Transactions on Biomedical Engineering, Vol. 37, No. 7, July 1990 3.1, 3.1, 3.4, 3.1

[4] G. Van Hoey, R. Van de Walle et al.: *Beamforming Techniques Applied in EEG Source Analysis*, IEEE ProRisc99, 1999 3, 8

[5] Johannes Vorwerk, Jae-Hyun Cho et al.: *A guideline for head volume conductor modeling in EEG and MEG*, NeuroImage 100, 2014 3.2, 8

[6] Gerhard Schmidt: Script for the lecture *Recognition and Audio Effects*

[7] Dennis Js. McFarland, Laurie A. Miner et al: *Mu and Beta Rhythm Topographies During Motor Imagery and Actual Movements*, Brain Topography, Volume 12, Number 3, 2000 1.1, 7

[8] G. Pfurtscheller, A Stancák Jr., Ch. Neuper: *Event-related synchronisation (ERS) in the alpha band - an electrophysiological correlate of cortical idling: A review*, International Journal of Psychophysiology 24, 1996 1.1, 7

[9] Donders Institute for Brain, Cognition and Behaviour: *FieldTrip*, https://github.com/fieldtrip/fieldtrip 3.2, 3.3.2

[10] Foundation for Research on Information Technologies in Society - *Tissue Properties - Low Frequency Conductivity* http://www.itis.ethz.ch/virtual-population/tissue-properties/database/low-frequency-conductivity/ 3.4, 3.1

[11] Wikipedia: *Neuron*, https://en.wikipedia.org/wiki/Neuron (on 2016-07-27) 1.2

[12] Wikipedia: *Saltatory conduction*, https://en.wikipedia.org/wiki/Saltatory_conduction (on 2016-07-27)

[13] Wikipedia: *Erregungsleitung*, https://de.wikipedia.org/wiki/Erregungsleitung#Saltatorische_Erregungsleitung (on 2016-08-11) 1.4

[14] Wikipedia: *Alpha Waves*, https://en.wikipedia.org/wiki/Alpha_wave (on 2016-07-27) 1.2

[15] Wikipedia: *Action Potential*, https://en.wikipedia.org/wiki/Action_potential (on 2016-08-11) 1.3

[16] Wikipedia: *10-20 Syste (EEG)*, https://en.wikipedia.org/wiki/10-20_system_(EEG) (on 2016-08-11) 1.6

[17] Oxford Centre for Human Brain Activity: *Magnetoencephalography (MEG)*, https://www.ohba.ox.ac.uk/groups/magnetoencephalography-meg/ (on 2016-08-11) 1.1

[18] Universitätsklinikum Heidelberg: *Wie fühlt das Gehirn?*, https://www.klinikum.uni-heidelberg.de/ShowSingleNews.176.0.html?tx_ttnews%5Btt_news%5D=5705 (on 2016-08-11) 1.1

[19] Moshe Lindner: *3D arrow plot*, http://www.mathworks.com/matlabcentral/fileexchange/28324-3d-arrow-plot 6.4