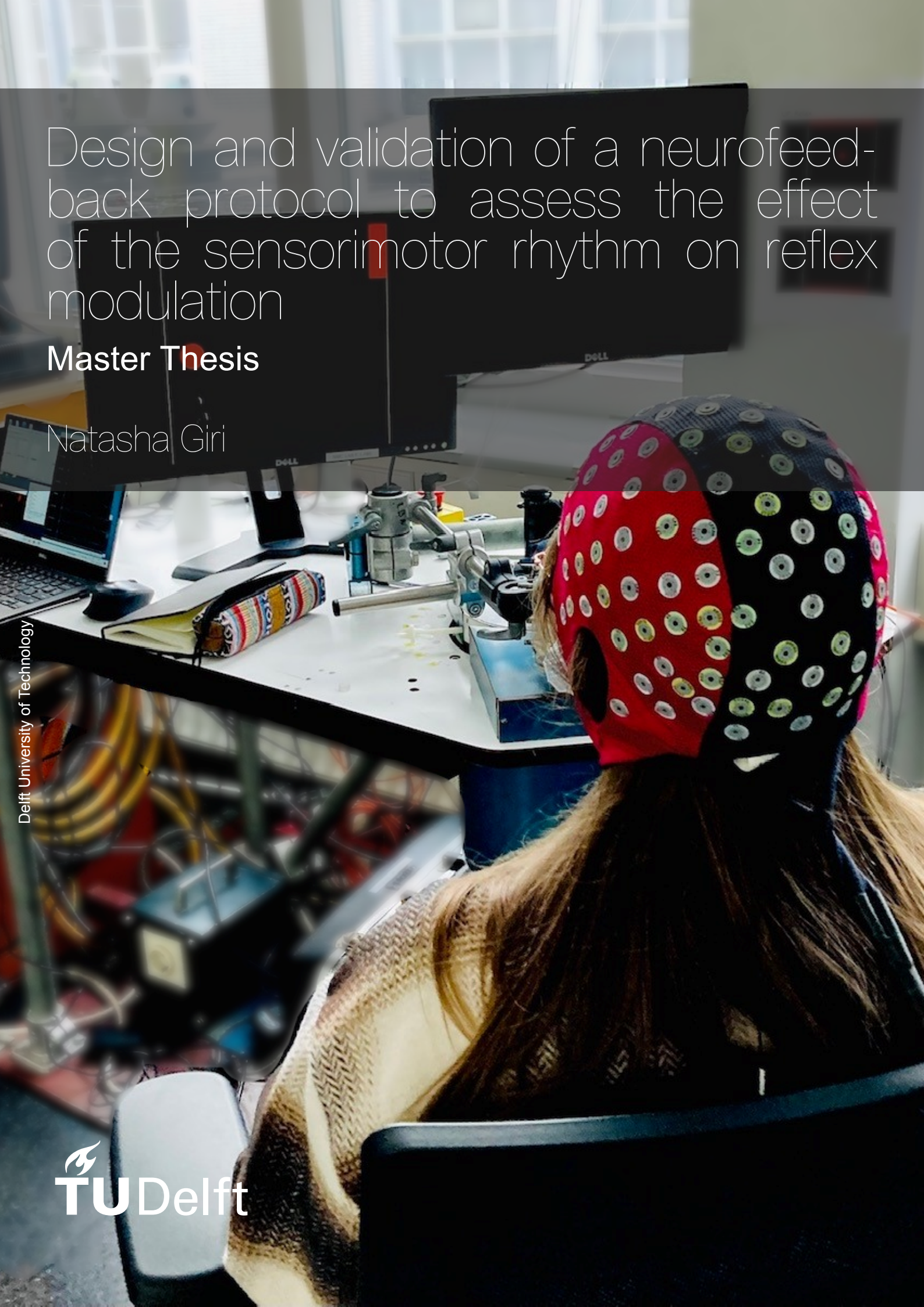
Design and validation of a neurofeedback protocol to assess the effect of the sensorimotor rhythm on reflex modulation

Master Thesis

Natasha Giri



Delft University of Technology

TU Delft

Design and validation of a neurofeedback protocol to assess the effect of the sensorimotor rhythm on reflex modulation

Master Thesis

by

Natasha Giri

to obtain the degree of

Master of Science in Biomedical Engineering

at the Neuro Muscular Control Lab in Delft University of Technology
to be defended publicly on March 31st 2022, at 9:30.

Student number:	4543440
Supervisor:	Dr. ir. Alfred Schouten
Daily supervisor:	Dr. ir. Mark van der Ruit
External thesis committee member:	Dr. ir. Justin Dauwels
Institution:	Delft University of Technology
Place:	Mechanical, Maritime and Materials Engineering, Delft
Project Duration:	May, 2021 - May, 2022

TU Delft

Dedicated to the memory of my beloved grandmother

Acknowledgements

I gratefully acknowledge the guidance of my daily supervisor, Dr. Mark van der Ruit, throughout the project. He provided tremendous help while leaving a lot of creative freedom in the project for me, he was always willing to think along and show me where to look for solutions whenever I faced a problem. He supported my ideas and encouraged me to think critically. He was closely involved in the experiments and data analysis, was always willing to discuss potential changes to the protocol or provide interesting insights into the interpretation of the results. He also showed a lot of understanding for any of my personal reasons that contributed to delaying my graduation. I am very thankful for all of this.

I would also like to thank Dr. Schouten for his patience, counsel and guidance, and for his enthusiasm in this project, which gave me a lot of motivation.

Furthermore, I would like to thank Judith, Marjolein, Eva, Stan, Kim, Zhengping and Sacha for investing their time to help me in this project, for their input, their endless patience, their kindness, and their support. Without them, I would not be able to present the results in this report.

A special thank you to Francisco, who helped me learn the fundamental principles of C++, helped me writing code, provided useful input on the report, and supported me throughout my master's program.

Abstract

People can learn voluntary regulation of certain brain rhythms with help of real-time feedback through a brain-computer interface. Modulation of brain rhythms associated with movement, also known as sensorimotor rhythms (SMR), has shown to alter excitability of the spinal cord. This may indicate a potential role of the SMR in reflex modulation. Until now, no study has assessed the effect of SMR modulation in the mechanically induced stretch reflex. Additionally, inconsistencies in electroencephalography processing for extraction of human intent lead to contradictory results in SMR modulation abilities. In this report, a paradigm is suggested in which people are asked to navigate a virtual cursor to a target with their SMR activity. Once they reach adequate control, mechanical perturbations are applied to their wrist to measure reflex responses. This report summarizes the development and validation of the protocol. Five participants took part in the experiment, four of which successfully acquired control over their SMR. Furthermore, results from one subject suggest that with sufficient modulation of the SMR, changes can be seen in the strength of the stretch reflex. This protocol can be used to further investigate the role of the SMR in reflex modulation, which is useful in the context of rehabilitation of reflex abnormalities.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
BCI	Brain-Computer Interface
ECR	Extensor Carpi Radialis
EEG	Electroencephalography
EMG	Electromyography
FCR	Flexor Capri Radialis
H-reflex	Hoffman reflex
MVC	Maximal Voluntary Contraction
PC	Personal Computer
SCI	Spinal Cord Injury
SMR	Sensorimotor Rhythm
TMSi	Twente Medical Systems International

List of Figures

1 Introduction

Reflexes perform important functions in the motor system. They assist in a variety of motor tasks that are performed every day, contribute to limb stability and help maintain posture [1]. Healthy people can change their reflex strength depending on the environment, task instruction or during different phases of (cyclical) movements [2]. Neurological damage, such as that resulting from stroke or spinal cord injury (SCI), can disturb reflex function. Patients suffering from neurological damage may show exaggerated reflex responses [3] and/or lose the ability to modulate their reflex strength [4, 5]. These reflex abnormalities can negatively affect the ability to walk, cause pain and fatigue, and restrict activities of daily living. Restoration of reflex abnormalities remains a challenge due to the limited understanding of the neuronal pathways and the cortical and spinal contributions involved in the reflex modulation. Investigating the neural mechanisms behind reflex modulation can help finding proper targets for rehabilitation protocols.

Pathways responsible for reflex modulation are known to involve higher cortical structures [6]. Descending signals from the brain can increase or decrease spinal cord excitability, which affects reflex strength. Where the signals are generated and what kind of signals are involved is still unclear. Studies offered evidence connecting certain brain rhythms with excitability of the spinal cord [7, 8]. These rhythms, coming from the sensorimotor cortex, can be detected with electroencephalography (EEG) and are known as the sensorimotor rhythm (SMR). The SMR has frequencies in the alpha (8-12 Hz), beta (13-30 Hz) and occasionally in the gamma (around 40 Hz) ranges [9]. Oscillations over the sensorimotor cortex in the 8-13 Hz frequency range are called ‘mu rhythms’, and this term will be used henceforth to avoid confusion with the more widespread alpha rhythm. The SMR decreases in amplitude due to desynchronization upon certain cues related to movement. These cues include movement preparation, motor imagery which is imagination of movement, and movement itself. In absence of these cues, the SMR synchronises, leading to an increase in amplitude. Interestingly, in absence of movement, people can increase and/or decrease the amplitude of SMR voluntarily [10].

Voluntary modulation of SMR is learned through neurofeedback training. As the name indicates, during this type of training, participants are provided with feedback about their EEG signals for learning purposes. Neurofeedback training is thought to facilitate neuroplasticty [7]. SMR-based neurofeedback protocols and brain-computer interfaces (BCIs) are explored by research groups in the context of rehabilitation after stroke [11], SCI [12] and other neurodegenerative diseases [13]. Several small-scale studies applied this training in stroke patients, sometimes combined with sensory feedback, yielding mixed results [14, 15].

Although it is commonly accepted that learning to regulate the SMR over a limited time span is possible, a significant amount (about 50%) of non-learners and incoherent functional outcomes show that finding the right protocol, signal analysis pipeline and screening procedure is necessary. Inconsistent results can be a consequence of a lack of proper guidelines on neurofeedback, and this raises questions about the validity of neurofeedback protocols [16].

Additionally, despite reflex abnormalities being a common consequence of neurodegeneratve diseases, only two studies investigated the effect of SMR neurofeedback specifically on reflex function [17, 18]. Recently, Thompson et al. [18] showed that people can up- or downregulate their reflexes by modulating their SMR. In the study of Thompson et al., participants practiced over more than 10

30-minute training sessions to increase and decrease SMR amplitude in the mu frequency range over the right arm area of the sensorimotor cortex. Each session was a succession of short four-second trials, with the task to either increase (SMR-up) or decrease (SMR-down) the SMR. In the last session, reflexes were elicited with electrical stimulation in a wrist flexor muscle, during SMR-up and SMR-down trials. Reflex strength was significantly larger during SMR-up trials than in between trials, and significantly smaller during SMR-down trials than in between trials. This finding shows that SMR neurofeedback could be a promising tool for increasing or decreasing reflex strength.

The study of Thompson et al. had one important caveat. The group of Thompson et al. assessed the change in reflex strength using an electrically elicited reflex, also called Hoffman reflex (H-reflex). Electrical stimulation of the proper nerve artificially evokes a reflex that is supposed to mimic the stretch reflex, the latter being a naturally occurring response to sudden muscle stretch. Although it is often referred to as the electrical analogue of the spinal stretch reflex, it should be taken into account that physiological properties of the H-reflex and stretch reflex are not identical. The stretch reflex starts with a signal about muscle stretch coming from the muscle spindle. In the H-reflex, the muscle spindle mechanism is bypassed by electrically triggering the afferent nerve. Additionally, it has been shown that the nerve populations triggered by these two stimuli are different, and so is the pattern of excitation in the nerves [19]. Therefore, the effect of SMR amplitude on the stretch reflex might differ from the one on the H-reflex observed by Thompson et al. Due to these distinct neural mechanisms, the observed effect of SMR on the H-reflex is not directly transferable to the stretch reflex. Since, in contrast to the H-reflex, the mechanically evoked stretch reflex occurs in nature and is present in daily movement tasks, it is meaningful to investigate whether there is a similar effect of SMR on the stretch reflex strength. Such information could be used in rehabilitation of people with reflex disorders. This shows the necessity for a paradigm in which the findings of Thompson et al. could be reproduced in mechanically-evoked stretch reflexes.

In this report, a BCI-based neurofeedback paradigm is proposed where a series of neurofeedback training sessions are followed by stretch reflex assessment. The purpose of this thesis project was to design and test an experimental setup where participants learn to regulate their SMR and where their reflex strength can be assessed at different amplitudes of their SMR. This will increase the understanding of how the level of SMR affects reflex strength.

The report is structured as follows: the development of the neurofeedback protocol is described in chapter 2, followed by the results of the validation experiment in chapter 3. Chapter 4 provides a framework for interpretation of the results, discusses several limitations and strengths of the protocol, provides future considerations and directions for future research. Finally, the main findings are highlighted in the conclusion, in chapter 5.

2

Methods

In this chapter, the reader is taken through the development and validation of the neurofeedback paradigm. This chapter is divided into two sections: section 2.1 describes choices on the software used for providing the neurofeedback and development of the EEG processing pipeline from raw EEG to the feedback provided to participants. Section 2.2 contains the methodology of the validation experiments, which consisted of three parts: (1) screening, (2) neurofeedback training, and (3) reflex assessment.

2.1. Development of the EEG neurofeedback paradigm

Development of the neurofeedback paradigm involved rewriting of an open-source software package to connect the EEG amplifier to the BCI software and choosing an adequate EEG processing pipeline to extract the signal that would eventually be translated to the feedback application. Several important choices have been made during the course of protocol development, which will be explained in this section.

2.1.1. Software

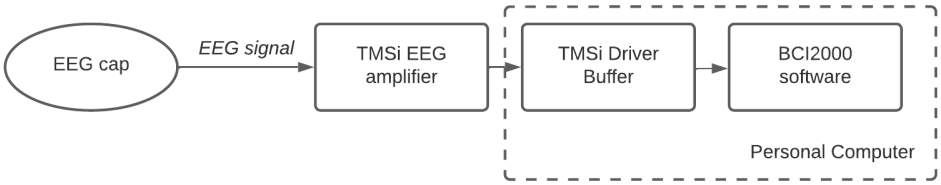
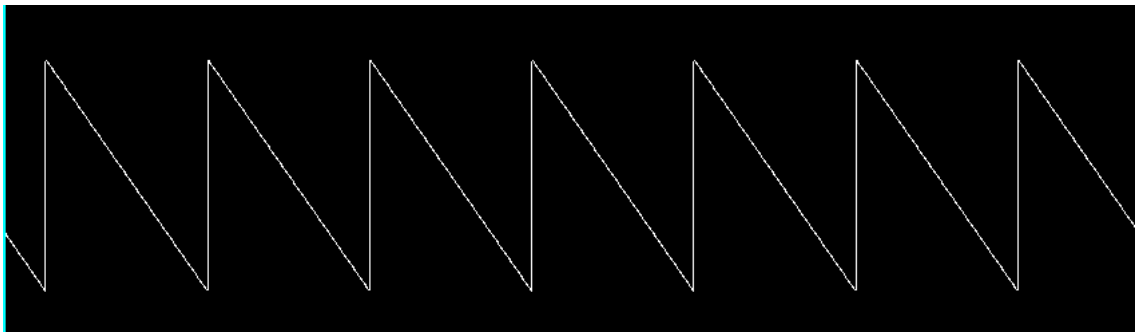


Figure 2.1: A schematic diagram of the path of EEG data before they reach the BCI2000 software on the PC. EEG signals are measured at the EEG cap with electrodes, after which this signal is enhanced with the Twente Medical Systems international (TMSi) EEG amplifier. This amplifier is connected to a personal computer, which contains the TMSi Driver Buffer. The buffer stores the samples until they are 'read' by the BCI2000 software.

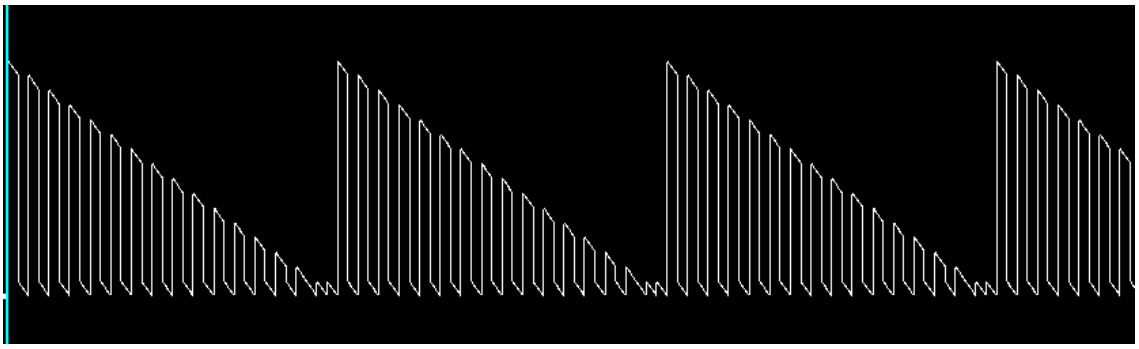
EEG signal processing and feedback presentation were performed through the BCI2000 software platform [20]. System specifications were laid down by two important BCI pioneers, Jonathan Wolpaw and Niels Birbaumer [21]. To date, BCI2000 has been used in numerous BCI studies and had a substantial impact on BCI and related research. This open source software was chosen for several reasons. First, BCI2000 was well suited for the purpose of this project, and it was easy to adopt due to extensive documentation and examples. Its versatility makes it suitable for adaptation in the future, for example, if the experimental focus changes to a different location in the brain. Second, using software that has been used in numerous BCI-experiments allows for comparison of new findings to previous literature, and, likewise, makes new results easily reproducible by others. And finally, choosing a software that has been proven to provide good results in the past gives more information about where to look for improvements if unexpected results are obtained.

Unfortunately, the EEG amplifier from Twente Medical Systems International (TMSi) used in this laboratory in Delft was not supported by the BCI2000 software. Outdated user-contributed C++ code for the BCI2000 software had to be rewritten to connect to the driver of the EEG amplifier and retrieve the signal (see figure 2.1 for a depiction of how the EEG data reach BCI2000). The resulting code is included in appendix B.

After the connection was established, a data loss problem was encountered. This problem was detected through the sawtooth test channel (figure 2.2 (b)). The issue was located in the function that retrieved the EEG samples from the TMSi driver buffer (figure 2.1): the*GetSamples()* function. One of the inputs to this function is the number of samples to be retrieved from the TMSi Driver Buffer, this number will be referred to as N. The documentation on this function advised to set N to the size of the full TMSi Driver Buffer, meaning that whatever is in the Driver Buffer will be retrieved. This was not compatible with the way BCI2000 processes EEG data in real-time. Namely, BCI2000 retrieves and processes data in regularly sized 'blocks' of data, called 'Sample Blocks'. The size of one Sample Block was chosen to be 32 samples. Therefore, the N was set to 32 samples. However, this resulted in an unexpected sawtooth signal, as can be seen in figure 2.2 (b). The reason for this was that each time the function would retrieve the samples, a data loss of exactly 16 samples would occur. This was independent of the size of the Sample Block: changing this variable (and, thus, N) to 48, 50, 64, 96 or other values still resulted in a data loss of 16 samples. The 16 lost samples had to be added to the variable N through hard coding. This solution resulted in the sawtooth signal shown in figure 2.2 (a).



(a)



(b)

Figure 2.2: The normal sawtooth test signal (a) and the sawtooth test signal as a result of data loss (b). Both depicted signals are shown for 7 seconds.

To test the solution, data acquisition through BCI2000 was compared with a known data acquisition software. Twelve minutes of EEG data were collected, six minutes with BCI2000 and six minutes with Asalab (ANT Neuro, Hengelo). This data were compared by comparing the spectra of both data sets. The similarity of these spectra confirmed that the solution worked well and no data loss occurred.

2.1.2. Real-time EEG processing

The goal of the real-time EEG analysis was to extract real-time control signals that would move a virtual cursor up or down in the feedback application. This was done in two steps: first, features were extracted from raw EEG, second, these features were translated into a control signal. Figure 2.3 shows

the processing pipeline that was set up for the real-time EEG analysis. The set of processing steps and the parameters of each of the processing steps were chosen by the researcher. The resulting pipeline is explained here in more detail.

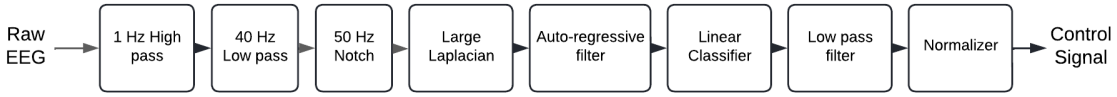


Figure 2.3: A flowchart depicting EEG processing from raw EEG to control signal. The objective of this pipeline is to extract the feature and translate this to a control signal. The resulting control signal controls the cursor movement on the feedback screen in the vertical direction.

Raw EEG data were filtered with a **High-Pass Filter** (1Hz) (a first-order infinite impulse response filter) to remove low-frequency drift, **Low-Pass Filter** (40Hz) (a second order Butterworth infinite impulse response filter) to remove high-frequency noise and **Notch Filter** (50Hz) (two-way third order Cherbychev bandstop filter) to remove power line noise. In the real-time EEG signal processing, the **Large Laplacian Spatial Filter** was applied to channels C3, Cz and C4, shown in figure 2.4.

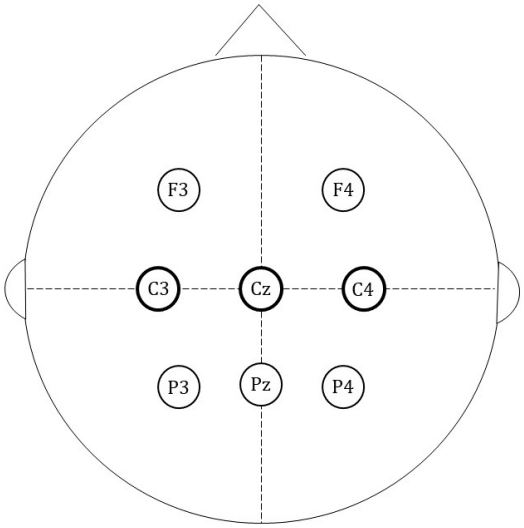


Figure 2.4: Locations of the electrodes used in the validation experiment for both screening and training. The electrodes overlaying the sensorimotor cortex are outlined in black.

In the Laplacian method, the mean activity of a set of neighboring electrodes is subtracted from the activity of each electrode. The activity at C3 was processed as follows:

$$C3' = C3 - \frac{F3 + P3 + Cz + Pz}{4} \tag{2.1}$$

Every 125 ms, the frequency spectrum of the previous 500 ms was estimated for 3 Hz wide bins with an **Auto-Regressive (AR) Filter** with a model order of 32. An AR algorithm computed a model of its input data with the Maximum Entropy Method (also known as Burg method). Its outputs are AR coefficients of a linear filter that transforms white noise into the spectrum of its input data. The estimated power spectrum was obtained by dividing the filter's transfer function by the total power of the signal. To obtain the power for finite-sized frequency bins, the power spectrum needed to be multiplied by total signal power and integrated over the frequency range of a bin by numerical integration at equally spaced points. The square root of the power spectrum output yielded the amplitude of the bin of interest.

The participant-specific frequency bin and target (C3) electrode were specified by the researcher in BCI2000 so-called **Linear Classifier** parameters. The name of this processing block might be misleading, as it did not exactly have the function of a 'classifier': it merely passed on the value that

corresponded to the specified frequency bin and target electrode. The resulting signal was smoothed with a **Low-Pass Filter** with a time constant of 1 s. Then, the signal was sent to the **Normalizer** block, to yield a control signal with zero mean and unit variance. The Normalizer subtracted the mean μ calculated over 30 s and multiplied the signal by the inverse of the variance σ:

$$S_i = \frac{1}{\sigma}(S_{iraw} - \mu) \tag{2.2}$$

The μ was called the Normalizer Offset parameter, and $\frac{1}{\sigma}$ was the Normalizer Gain. For the first two runs of each session, the parameters of the Normalizer were updated after each trial. Updating the parameters happened automatically, but only when the 'Adaptation' parameter in BCI2000 was switched on. When to switch this parameter on and off was decided by the researcher. Generally, the first two runs of each session were used for adaptation of the parameters. Occasionally, when a strong 'bias' was observed i.e. if the cursor seemed to move predominantly in one direction regardless of the target location, the Adaptation would be switched on for one run. The regression formula used to calculate vertical velocity of the cursor for step i was the following:

$$v_i = v_0 \cdot S_i \tag{2.3}$$

where

$$v_0 = screenheight/(2 \cdot FeedbackDuration) \tag{2.4}$$

In these equations, v is the cursor velocity for step i, and S_i is the normalized control signal for step i. The $FeedbackDuration$ parameter was set to 6 seconds, as was the length of one trial. In the ideal case, the control signal will be constant +1 for up trials, and -1 for down trials.

The development of the pipeline required a number of important signal processing choices to be made, the main of which are explained below.

- **Set of electrodes**: A relatively low number of electrodes (8) was chosen for several reasons. First, it minimized EEG preparation time, which gave the participant more time to train before he or she would get fatigued. This also made the session shorter to attract more participants. Second, as the relevant signal is coming from the part of the sensorimotor cortex that is associated with right hand movement, the signal of interest came from one pre-defined electrode (C3). The spatial filtering method used in the protocol required a number of second-nearest neighboring electrodes. Therefore, this choice of electrodes is adequate for covering the region of interest and contains all electrodes for the spatial filter. It was also purposefully decided to exclude electrodes that are prone to containing artifacts. These include frontal electrodes, which are closer to the eyes and therefore may contain blinking and eye movement artifacts. Also the temporal electrodes were excluded, as they have been shown to be affected by horizontal eye movements in the alpha band in literature [22] and indeed gave noisy signal during the pilot experiments.

- **Spatial filter**: The spatial filter was applied to improve the signal-to-noise ratio. The choice of the spatial filter was based on a publication from McFarland et al. [23], where it was shown that a large Laplacian filter is a more adequate choice for an SMR-controlled neurofeedback compared to the small Laplacian filter. A Laplacian filter calculates the second derivative of the instantaneous spatial voltage distribution [23]. This means it emphasizes activity immediately below the electrode, while attenuating activity from distant sources. This is in contrast to a small Laplacian, which attenuates activity from closely neighboring sources.

- **Spectral estimation method**: The choice for the auto-regressive (AR) algorithm was based upon previous findings in literature, stating that in the context of SMR-based BCI, an AR algorithm was preferred to other spectral estimation methods including the Fast Fourier Transform and band-pass filtering [24].

- **AR model order:** The model order is the number of prior predictions used to estimate the next prediction. It has been argued that a considerable amount of BCI studies choose a model order that is too low (6-10) and thereby fail to take into account the complex nature of EEG [24]. In [24], user control seemed to increase with higher model orders and started reaching a plateau at 26. The authors argue that higher model orders (26 and up) are more suitable for mu rhythm BCI

applications. However, a model order that is too high will model high-frequency features, which are likely to include noise, and is therefore undesirable. Based on these results and the results presented in [25], a model order of 32 was chosen.

Development of the software and the EEG processing pipeline was followed by validation experiments, which are explained in the next section.

2.2. Validation of the EEG neurofeedback paradigm

For validation of the protocol, the screening, neurofeedback training and reflex assessment were performed in a small participant group.

Three males and four females aged between 22 and 24 years were asked to participate in experiments to validate the neurofeedback protocol. None of the participants had previous neurofeedback or brain-computer interface experience. None had known neurological disorders. All participants received an information letter about the experiment and gave written informed consent prior to start. All procedures were approved by TU Delft Human Research Ethics Committee. Participants underwent a screening session prior to the training to tune certain parameters of the software. Subsequently, they were asked to come back and perform 40-minute training sessions with a frequency of 2-3 times a week, depending on the availability of the researcher and participants. If the participant showed some amount of success during the training, he or she was asked to come back for a one-time 1.5 hour reflex assessment session.

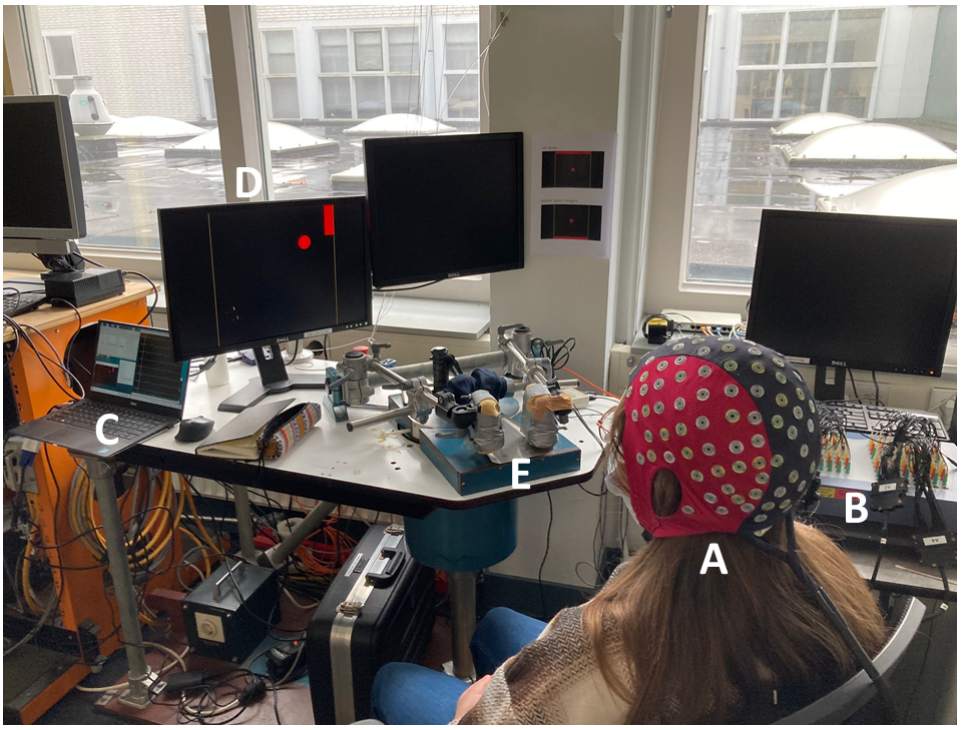


Figure 2.5: A picture of the neurofeedback training setup. Brain signals from the participant (A) are measured with the TMSi Refa EEG system (B). The amplified EEG signal is sent to the TMSi driver on the PC (C). BCI2000 software on the PC reads the EEG signal, extracts the feature and translates this to a control signal, which controls the cursor on the feedback screen (D). In the reflex assessment sessions, participant's right arm is located in the wrist manipulator (E) and EMG electrodes are attached to the flexor and extensor of the wrist (not shown in the image).

2.2.1. Screening

The screening was done in the first session with the purpose to obtain the participant-specific target frequency in the SMR, the amplitude of which would be modulated during the training sessions that followed. The target frequency was found by comparing the EEG signals between two conditions: 'rest' and 'right-hand movement'. The screening comprised a 40-minute session in which the participant gave written consent and was asked to perform a movement task while EEG data were gathered.

EEG Hardware

EEG measurements were performed on a TMSi Refa Amplifier (TMSi, Odenzaal, The Netherlands) at a sample rate of 256 Hz. Eight electrodes (F3, F4, C3, Cz, C4, P3, Pz and P4) on a 128-electrode waveguard cap (Ant Neuro, Hengelo, The Netherlands) were used, the electrode locations are shown in figure 2.4. A ground electrode was fixed to the right mastoid process of the participant. The impedance was tested prior to each session. When a channel kept showing a bad signal and/or a high impedance (> 10 kOhm), this channel was removed, meaning it had a value of 0 throughout the session.

Experimental Task

During the screening task, participants were seated in a chair about 1 meter away from the screen and asked to hold an object (a sponge) in their right hand. Participants were instructed to perform flexion of the wrist for a 6 s interval, alternating with 6 s intervals of rest. The screen showed a red bar on the top of the screen indicating 'rest', or a red bar on the bottom of the screen indicating that flexion should be performed. An A4 piece of paper with these two cues and their meaning was placed on the wall next to the screen, to remind the participants if needed.

Screening Data Analysis

Analysis of the EEG data was performed in BCI2000 software and used components of the pipeline described earlier. Raw EEG data were filtered with high-pass, low-pass and notch filters. Data obtained under the two different conditions (movement and rest) were separated. An AR filter estimated the spectrum for each condition. The coefficient of determination (r^2) was calculated for a range of frequencies between 0 and 70 Hz. This determination coefficient represents the fraction of the total signal variance that is accounted for by the task condition. The determination coefficient is the squared correlation coefficient for a two-dimensional set of data points (x,y):

$$r^2 = \frac{cov(x,y)^2}{var(x)var(y)} \tag{2.5}$$

where x is a vector with the measured SMR amplitudes and y is a vector with labels that indicate the corresponding condition (1 or 2). The r^2 reflects what fraction of a signal can be attributed to the target location (and thus task). A higher value of r^2 for a particular frequency means that the user intent can be more easily inferred from that frequency than from other frequencies. Thus, this becomes the target frequency.

A feature plot (shown in figure 3.1a) was used to confirm that everything went well: high r^2 around C3 was an indication that indeed, the task had been performed successfully and the region responsible for movement of the right arm was engaged. For the choice of the target frequency, the frequency spectrum at C3 together with a plot of r^2 (figure 3.1b) were inspected. The location of the r^2 peak in the 8-13 Hz frequency range determined the initial target frequency. This was subject to change after the first two motor imagery sessions, in which spectra and r^2 plots were analysed again to make sure the peak had not shifted to a different frequency. The feature was updated if the peak in the r^2 plot had shifted.

2.2.2. Neurofeedback training

The neurofeedback training was comprised of five to eight 40-minute sessions, in which participants performed a specific task while receiving feedback over their EEG signal on the screen. The objective of the training was for participants to learn voluntary modulation of the SMR using mental techniques.

Experimental Task

The task was to steer a virtual cursor up or down with brain activity alone. An example of the first trial is shown in figure 2.6. At the beginning of a run, a 'Be prepared' cue appeared on the screen for three seconds. Every trial started with a target appearing on the top or bottom-right of the screen. Two seconds later, a round cursor appeared at the middle left edge of the screen and started moving to the right edge with a constant speed. The participant was given six seconds to navigate the cursor in the vertical dimension to the target so that the cursor hits the target. The result was displayed for one second, and was followed by a one-second inter-trial interval until the next target was shown. The cursor reflected the SMR amplitude of the participant with a 125 ms delay. The following instructions were given prior to start: 'To steer the cursor to the lower target, imagine moving your right hand, as if

you were squeezing an object or clenching your fist. To steer the cursor to the upper target, try to rest and clear your mind.' Participants were also encouraged to try their own strategies and report what worked or not.

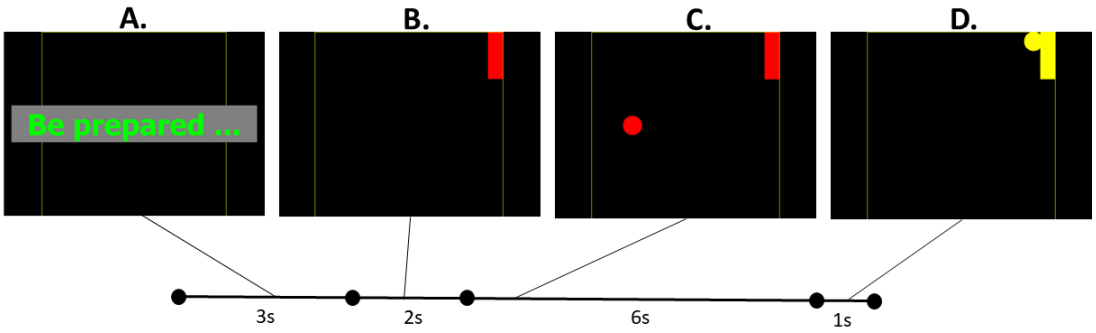


Figure 2.6: Feedback screen. (A) A 'Be prepared' cue was shown for 3s before the beginning of each run. (B) The target appeared 2s before the cursor appeared in the middle on the left edge of the screen (not shown). (C) The cursor moved with a constant speed to the right, so that in 6s it would reach the right edge of the screen. The participant's task was to steer the cursor in the vertical direction according to the target location. To reach the cursor, only 5s were needed, therefore, successful trials were shorter in length. (D) If the cursor hit the target, the cursor and target became yellow. The result was displayed for 1s before the screen turned black and another trial began.

Feedback parameters

Feedback was provided on a 15 inch screen about 1 meter from the participant. Every 125 ms, EEG data from the previous 500 ms determined the control signal. One session consisted of 10 runs of 14 trials each. SMR-up and SMR-down trials were presented in random order and were present in the same proportion in each run. In the first two runs, the Normalizer was actively changing the Offset and Gain parameters (equation 2.2) to adapt to the mean and variance of the current session, accounting for natural day-to-day fluctuations in resting-state SMR amplitude.

2.2.3. Reflex Assessment

The SMR training sessions were followed by a reflex assessment session. Reflexes were assessed with a wrist manipulator, a schematic depiction of which is shown in figure 2.7. The participant was asked to perform the same neurofeedback task as in the training sessions, while this time, the right forearm was kept in the wrist manipulator, where stretch reflexes were evoked.

Changing from neurofeedback training to the reflex assessment setup was a gradual process. One or two sessions before the reflex assessment session, the participant was asked to keep their right arm in the wrist manipulator, to get used to the changed posture. In the reflex assessment session, the first several trials were performed without perturbations or any resisting forces from the handle. Then, a maximal voluntary contraction (MVC) measurement was performed. During this measurement, the lower arm of the participant was constrained in the wrist manipulator. The handle was fixed and the participant was asked to exert maximal force on the handle by flexing the wrist and to repeat this three times in total, releasing in between. For the next several trials, the participant is asked to maintain 5% of the MVC implemented through a force in the handle, without inducing any stretch reflexes. Only then the actual reflex assessment could start.

EMG acquisition and data analysis

EMG was measured from the right forearm. The muscles were first identified by the researcher and marked, then the skin was cleaned with an abrasive gel and alcohol. Electrodes (Delsys Bagnoli system, 10x10mm in size) were placed on the belly of the flexor carpi radialis (FCR) and extensor carpi radialis (ECR) muscles. The ground electrode was placed on the elbow. EMG signals were band-pass filtered at 20-450 Hz prior to sampling.

For background EMG, the rectified pre-stimulus EMG activity, starting at 200 ms and ending at 10y ms before each perturbation, was averaged. The area under the EMG activity curve was computed for the short-latency M1 (20 ms-50 ms) and long-latency M2 (55 ms-100 ms) components. Only successful trials were taken into account. A successful trial was defined as a trial in which the cursor hit the target.

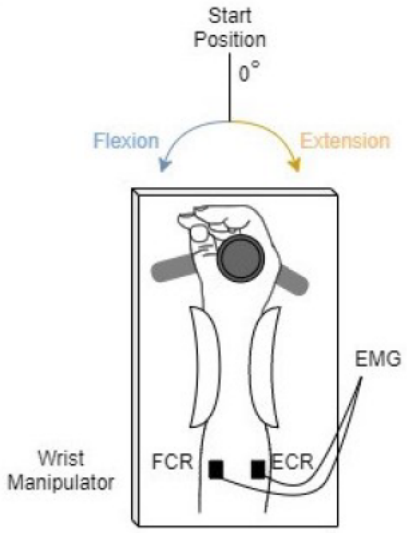


Figure 2.7: Schematic depiction of the wrist manipulator used for reflex assessment. EMG of the FCR and ECR muscles was measured. The wrist manipulator constrains the lower arm, so that only the wrist can be moved. Perturbations are given in the extension direction, indicated by the yellow arrow. Taken from [26]

Experimental Task

In addition to the SMR training task, participants were shown a second screen and had their arm in the wrist manipulator. They were asked to maintain 5% of their maximal voluntary contraction (MVC) by keeping a cursor (reflecting the force on the wrist manipulator handle) at a target position on the second screen.

Wrist perturbations

Ramp and hold perturbations were given through the handle of the wrist manipulator. The perturbation consisted of a 40 ms ramp at an angular velocity of 2 rad/s resulting in a maximal amplitude of 0.08 rad. The perturbation was given in a way that extended the wrist, eliciting a stretch response in the FCR. One perturbation occurred per trial.

Perturbations of the wrist and neurofeedback trials had to be synchronized manually, meaning the neurofeedback application and the MATLAB model that controlled the perturbations were started by the researcher separately. Perturbations occurred every 9.5s, whereas a full trial duration was 9s for successful trials and 10s for unsuccessful trials. The perturbations were started such that the first perturbation occurred near the end of the trial. However, with more successful trials, the perturbation could shift towards the end of a trial, whereas with more unsuccessful trials, the perturbation shifted closer to the beginning of a trial. When the perturbation occurred earlier than at 2/3 of the trial or after the end of the trial, the run was aborted and a new run was performed. Six runs were performed of eight trials each with short breaks in between.

At least 10 reflex measurements are desirable for a reliable measurement of reflex strength. Since only the reflex measurements performed during successful trials (i.e. trials in which the target got hit) were taken into calculations, at least 10 successful trials for each condition were desirable. Six runs of eight trials would contain 24 trials for each condition, meaning that any success rate above 10/24=0.42 would yield enough data for a reliable reflex measurement.

3

Results

Results for five participants are presented for the three phases of the experiment: the screening (section 3.1), the neurofeedback training (section 3.2) and reflex assessment (section 3.3).

3.1. Screening

The objective of the screening session was to obtain the participant-specific target frequency bin by analyzing differences in EEG between two conditions: right-hand movement and 'rest'. Screening data were analyzed by visual inspection of feature plots, determination coefficient (r^2) spectra and topography plots. The r^2 is a metric for how different the EEG signals are between the two conditions. A high r^2 is expected in the channels and frequency ranges that are related to right hand movement.

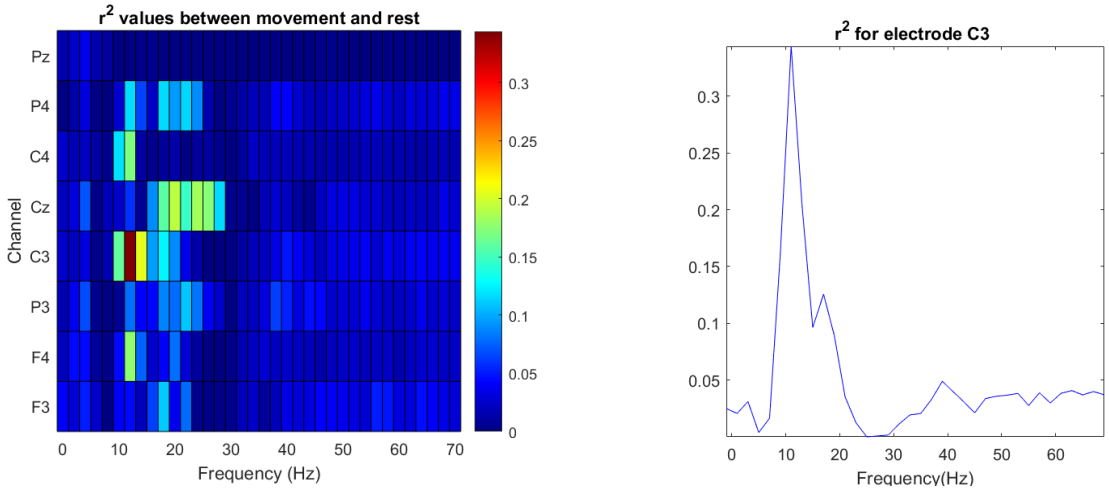


Figure 3.1: An example of a feature plot (a) and an r^2 spectrum (b). In (a), colors indicate the value of r^2 for each frequency and EEG channel. The values corresponding to each color are indicated in the color bar on the right. The highest r^2 for this participant is found at channel C3 between at 11 Hz, which corresponds to the mu frequency range. The feature plot and corresponding r^2 spectrum for participant 5 are shown for illustration purposes and differed between participants.

For every participant, it was confirmed that changes in EEG activity related to the task found place in the expected channels and frequency ranges. C3, Cz and C4 electrodes overlay the sensorimotor cortex, therefore, SMR activity related to movement can be expected to be prominent in the corresponding channels. Most of the response is expected in C3, since C3 is associated with right hand movement. This initial analysis was also performed to confirm that data acquisition and processing and task execution went well, which is important for SMR control in the following sessions. An example feature plot in figure 3.1a shows that higher r^2 (red) values are most prominent in C3 and for low SMR

(10-18 Hz) frequencies, as expected. Some activity is also seen in C4 and Cz channels around mu (10 Hz) and beta (20 Hz) frequencies respectively.

Next, the initial target frequency was determined from the r^2 spectra (figure 3.1b). Initially, the target frequency was selected from the highest peak in the r^2 spectrum for frequencies in the mu (8-13 Hz) or beta (13-26Hz) ranges, as these are all part of the SMR. Gamma frequencies (above 30 Hz) were excluded from consideration, because spectral peaks in the gamma frequency range can be a result from muscle activation artefacts [21, 22] and thus are not reliable. After the pilot experiments, it was decided to restrict the choice of target frequency to the mu rhythm only, disregarding the peaks around beta frequencies. Pilot experiments showed that the amplitude of the mu frequency better conveyed user intent and thus resulted in better control over the cursor. This was supported by offline analysis after motor imagery session (figure 3.2 and Appendix A), which showed that in some participants, the highest r^2 peak was found in beta frequencies during the screening, but became less prominent when motor imagery was performed. The beta peak was not observed in all participants. Meanwhile, the r^2 peak around mu frequencies was more consistent and was present across all participants, and was therefore a better candidate for the target frequency.

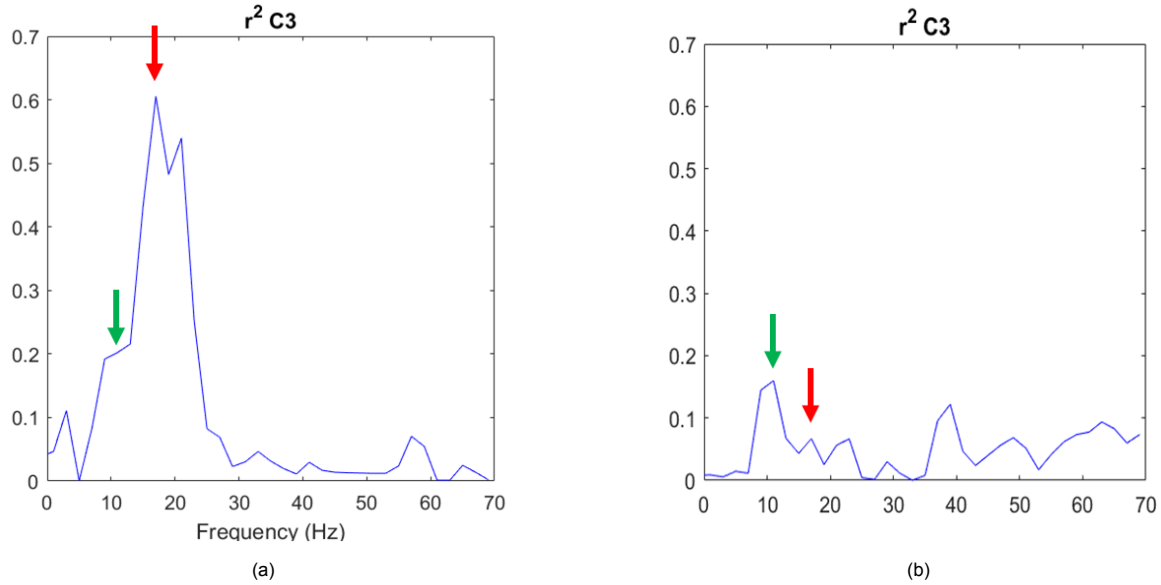


Figure 3.2: Changes in r^2 spectra during movement and motor imagery. r^2 spectra are shown here from the screening session where real movement was performed 3.2a and from the first motor imagery session 3.2b. The r^2 peak at 17 Hz (red arrows) during screening (a) decreased with one order of magnitude during the motor imagery session (b), leaving a peak at 11 Hz (green arrows).

The initially selected target frequency was subject to change: if the r^2 peak shifted to a different frequency during neurofeedback training in the first two motor imagery sessions, the frequency was changed to allow for better SMR control. Since the screening task and neurofeedback task differ, it is possible that the spectral properties of EEG will be different for the two tasks.

The finally selected features for each participant and their corresponding r^2 values can be found in table 3.1. Reported are frequencies that were selected as the center of the 3-Hz frequency bin which is used for real-time analysis. The amplitudes of frequencies in the frequency bin determined the signal that controlled the cursor in the neurofeedback task. Target frequencies ranged from 9 Hz to 13 Hz in the participant group.

Participant	Target frequency	r^2
Participant 3	11 Hz	.056
Participant 4	11 Hz	.347
Participant 5	13 Hz	.297
Participant 6	11 Hz	.533
Participant 7	9 Hz	.341

Table 3.1: Target frequencies and corresponding r^2 for each participant. Participants 1 and 2, who only participated during the pilot experiments, are not included in the table.

3.2. Neurofeedback training

To assess how SMR control changed throughout the training sessions, the percentage of successful trials ('hits') was analyzed in total and for both conditions separately per session. In addition, SMR amplitudes for 'up' and 'down' trials were examined to investigate the course of SMR amplitudes throughout the trial and to analyse how SMR amplitude varied between the two conditions.

As described in chapter 2, participants were instructed to control the height of a virtual cursor to reach a target, located either on the lower (SMR-down) or upper (SMR-up) quadrant of the right edge of the screen. A trial was 'successful' when the cursor hit the target by the end of the trial. Example trajectories of the cursor for 14 trials are shown in figure 3.3. Figure 3.3 shows that the direction of the cursor is influenced by the target location in most of the trials. In this particular case, trajectories of the cursor during SMR-up trials were more spread out than in SMR-down trials. This 'bias' was often present but was not consistent and changed in direction and magnitude between, runs, sessions and participants.

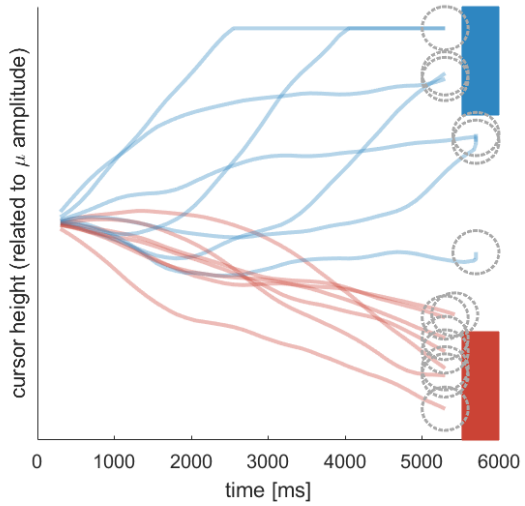


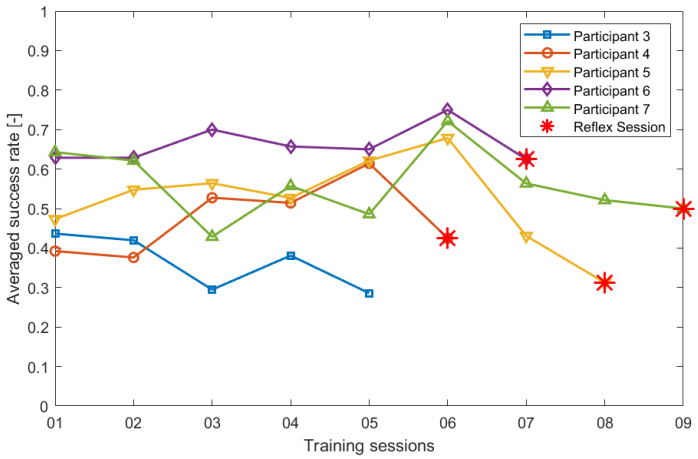
Figure 3.3: Example trajectories of the cursor during a neurofeedback trial, shown for one run. Upper (blue) and lower (red) target locations are indicated by rectangles, and the lines represent trajectories of the center of the cursor during SMR-up (blue) and SMR-down (red) trials. 14 trials are shown.

3.2.1. Success Rates

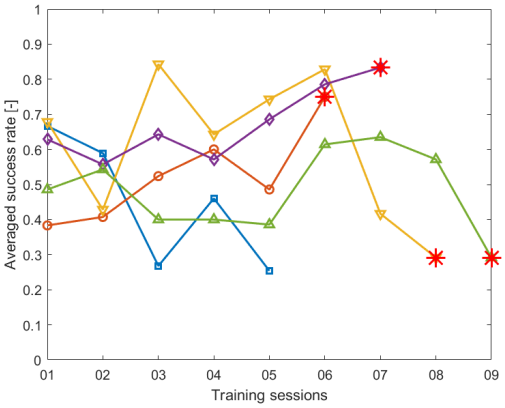
Five participants completed five to eight training sessions of 112-140 trials each. The combined success rates per session and rates for SMR-down and SMR-up trials can be seen in figure 3.4.

Participant 3

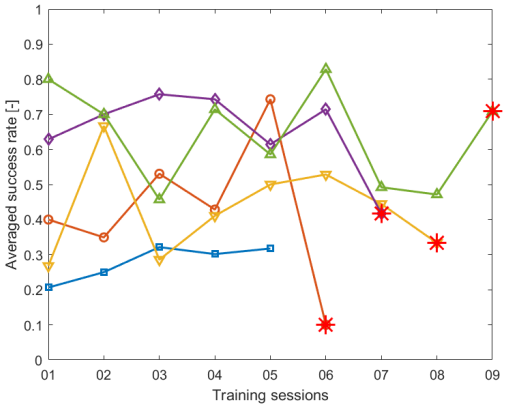
Participant 3 could not acquire voluntary SMR control over five training sessions. This participant had a low r^2, low success rate that did not improve over time, combined with the participant's subjective experience i.e. a lack of 'feeling' of control and seemingly random cursor trajectories i.e. the cursor did not only miss the target but also moved in unexpected directions. Participant 3 trained two to three times a week, but had to stop training for 12 days before session 3. It was decided to discontinue the training after session 5.



(a) Total success rates



(b) Success rates top target



(c) Success rates bottom target

Figure 3.4: Success rates for all participants over the course of the neurofeedback training. Total success rates (a) , as well as success rates per condition (b and c) are shown for each participant. Success rate is defined by the number of hits divided by the number of trials per run. Each data point was averaged over 8-10 runs. Red asterisks indicate the session in which reflex assessment was performed.

Participant 4

Participant 4 started during the pilot phase, meaning that the protocol was not yet final during the first three training sessions for this participant. Participant 4 trained 2 times a week until session 4, after which the participant came back once a week, due to the availability of the participant.

During the first training session, the cursor was controlled by the amplitude of the 16.5-19.5 Hz frequency bin, which was chosen based on the results of the screening session (see the Appendix A) for screening results). This was changed to the 10.5-13.5 Hz bin during the course of session 2, because the motor imagery session left an r^2 peak in the mu frequency range, while the peak at beta frequency range was much lower. This phenomenon is depicted in figure 3.2. Participant 4 may have had the 'disadvantage' of being one of the first to participate in the experiment, as the researcher had less experience and could give less instructions on motor imagery and other possible strategies. Additionally, the application task in the first two sessions differed slightly from the one described in figure 2.6, in that the targets were located on the bottom or on the top of the screen, and the cursor could only move in the vertical direction. This setup gave the participant less insight in how much time was left in the trial, and made it harder to hit the target, which may have had a negative effect on motivation. To compensate for this difficultly, the success rate calculations were adjusted. The success rate for the first two sessions shown in figure 3.4 was still defined as the number of successful trials divided by the total amount of trials, but in this case, a successful trial was defined as a trial in which the cursor occupied the lower or higher quadrant of the screen for SMR-down and SMR-up trials, respectively.

The participant had a feeling of voluntary control over the cursor from the first session, which increased over the training sessions. Figure 3.4 shows an upward trend from session 1 to 5, which is true for both overall success rate as the rates for SMR-up and SMR-down conditions. The success rate decreased rapidly during the reflex session.

Participant 5

Similarly to participant 4, participant 5 had a feeling of voluntary control over the cursor from the first session onward, which increased over the training sessions. This was reflected in the overall success rate as seen in figure 3.4.

During session 2, control over the cursor seemed to be lost. Changing the frequency bin for cursor control from 9.5-12.5 Hz to 11.5-14.5 Hz based on r^2 spectum analysis (Appendix A) restored the control. During session 7, participant 5 was asked to keep their arm in the wrist manipulator to get used to the posture and verify that voluntary SMR control could be reproduced in the new circumstances. Upon this change, the success rate of the participant drastically decreased, and so did the subjective feeling of voluntary control of this participant. This persisted during the reflex assessment.

Participant 6

Participant 6 had a relatively high success rate from the start of the training, and slightly increased the success rate by the end of the training period. The participant trained three times a week. Similarly to participant 5, the participant was asked to keep their arm in the wrist manipulator in session 6. Due to technical problems, the session only lasted for four runs instead of the usual ten, but the result was so good (figure 3.4) that it was decided to perform the reflex assessment in the next session. Participant 6 was the only participant for whom the success rate was unaffected after introducing the wrist manipulator.

Participant 7

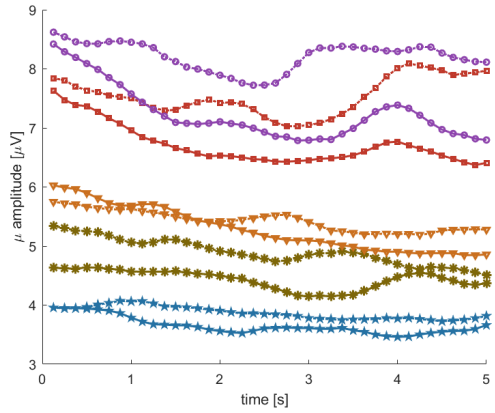
Similarly to participant 6, participant 7 had a relatively high success rate in the first training session. However, the success rate fluctuated over the training period for unknown reasons. This participant trained regularly three times a week. Participant 7 was given more time to get used to the wrist manipulator due to decrease in performance, so an extra training session was performed with the arm in the wrist manipulator. The results of these two training sessions (sessions 7 and 8) are shown in figure 3.4. The success rate stayed low despite the extra training session but was sufficient to continue with reflex assessment.

Summary

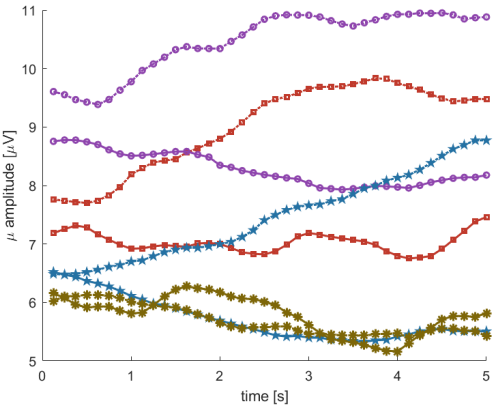
Four out of five participants were able to acquire SMR control within the first three sessions. Three of these four (participants 4,5 and 6) increased their success rate over the training period. Success rates of the other participant, participant 7, oscillated throughout the training sessions. In most participants, the success rate decreased when the arm was placed in the wrist manipulator. This decrease persisted throughout the reflex assessment session.

3.2.2. SMR amplitude during trials

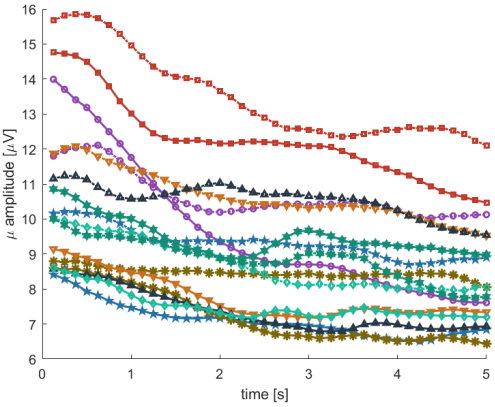
Although directly related, the trajectories of the cursor do not reflect the ‘real-time’ SMR amplitudes but are rather the result of all SMR amplitudes from the start of the trial, as described in chapter 2. To investigate the course of the SMR amplitudes during the trials, the amplitudes for each condition were averaged over all trials of one session. The results are shown in figure 3.5. All participants show a difference in SMR amplitudes between the up (dotted) and down (solid) conditions in most of the sessions, although the amplitude ranges differ across different sessions.



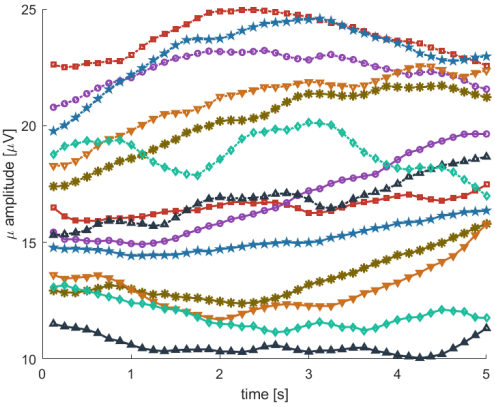
(a) Participant 3



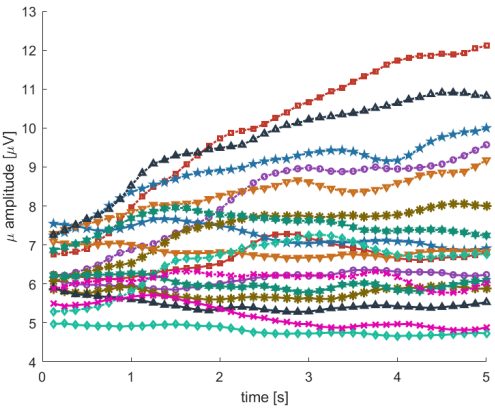
(b) Participant 4



(c) Participant 5



(d) Participant 6



(e) Participant 7

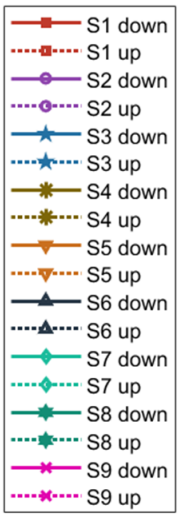


Figure 3.5: SMR amplitude for the first 5 seconds of a neurofeedback trial (minimum length of a trial), averaged over all trials in a session for participants 3 (a), 4(b), 5(c), 6(d) and 7(e). A set of two lines of the same colour and marker represents the amplitudes for up (dotted lines) and down (solid lines) conditions for one session. In the legend, the letter 'S' stands for 'session'

Figure 3.5 shows how diverse the different courses of SMR amplitudes can be. Whereas in participant 4 it is clear that the amplitude is 'steered' in both up and down directions, participant 7 seems to steer the amplitude mainly up. Participants 5 and 6 seem to do the SMR modulation before the start of the trial, as their amplitude at 0 seconds seems to be separated for SMR-up and SMR-down trials.

Presumably this happens in the two seconds before the start of the trial, when the target location is already revealed.

3.3. Reflex Assessment

The neurofeedback training was concluded with a reflex assessment session. Four participants completed the reflex assessment. In the data of participants 4 and 5, reflexes could not be distinguished. Possible reasons are an incorrect placement of EMG electrodes or releasing the wrist manipulator handle short after perturbation. Additionally, these two participants had a very low success rate in the reflex assessment session compared to the training sessions, resulting in much less than 10 successful trials per condition, therefore, the number of reflexes to analyse was too low for a reliable measurement.

SMR control was also diminished in participant 7 (as can be seen from figure 3.4), but was still present and sufficient to obtain a reliable measurement of reflex strength. The success rate of participant 7 decreased by 14% compared to the average success rate during the training for this participant. In addition, the success rate for the SMR-up condition was much lower than the success rate for the SMR-down condition, which was another sign of poor voluntary SMR control. Moreover, as can be seen in figure 3.5, the mean SMR amplitude during SMR-up trials was very close to the mean SMR amplitude during SMR-down trials. This can be seen in figure 3.5 (e): the two lines marked with the pink crosses differ much less than almost any other combination of 'up' and 'down' lines. The difference between mean SMR amplitudes for the two conditions calculated as: [Mean amplitude during SMR-up trials] - [Mean amplitude during SMR-down trials], was reduced by more than half (58%) during the reflex session as compared to the training. A smaller difference in SMR amplitude between two conditions does not only impair the success rate, but can also have consequences for the interpretation of the EMG data, which will be discussed in chapter 4.

The EMG data from participants 6 and 7 are shown in figure 3.6. Size of reflex responses was analysed by computing the areas under the curve of short-latency M1 (20ms-50ms) and long-latency M2 (55-100ms) components. The time points are marked grey in figure 3.6. For participant 6, the effect of the SMR amplitude was limited to the M2 component. M2 area was higher in the SMR up condition with 25.5%. For participant 7, the M1 area during SMR up was slightly (5,8%) bigger than the M1 area during SMR down, whereas M2 area was slightly reduced (5,5%) during SMR up condition.

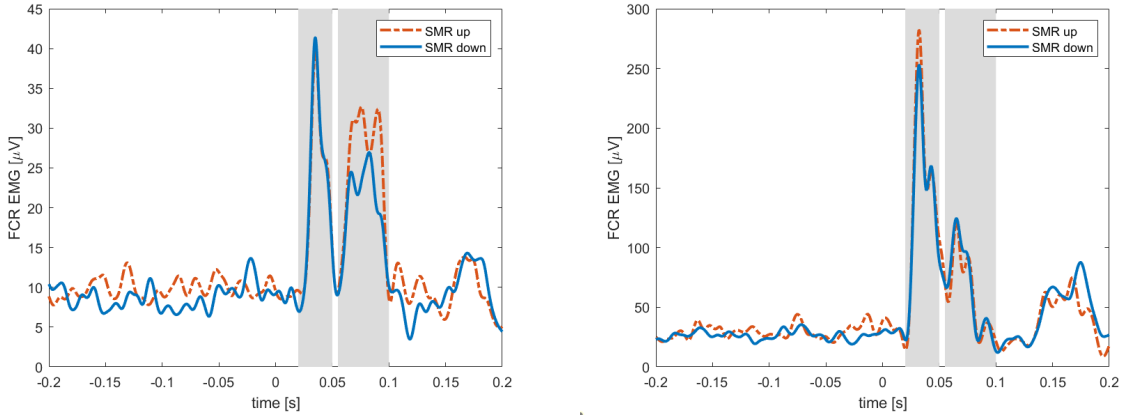


Figure 3.6: EMG from the FCR muscle upon a perturbation of the wrist for participants 6 (a) and 7 (b) for SMR up (red, dotted line) and SMR down (blue, solid line) conditions. M1 and M2 are marked grey. For participant 7, EMG data were averaged over 17 trials for SMR down and 7 trials for SMR up.

4

Discussion

4.1. Discussion of the results

This report presents a neurofeedback protocol to study the effect of SMR amplitude on stretch reflexes. The protocol consists of three parts: (1) screening, (2) neurofeedback training and (3) reflex assessment. During the screening, the optimal center of a 3 Hz-frequency band in the SMR range is found for each participant by comparing the EEG spectra during movement and rest. The amplitude of this frequency band is used to control the height of a virtual cursor during neurofeedback training. Neurofeedback training consists of 5-8 sessions in which participants repeatedly attempt to steer the cursor towards a target by modulating SMR amplitude. Subsequently, participants undergo reflex assessment simultaneously with the neurofeedback task, and the reflex strength during low and high SMR amplitudes is distinguished. Important aspects of the resulting protocol and the results of the validation experiments are discussed below for each of its three parts.

4.1.1. Screening

The frequencies that correlated the most with a change in conditions from rest to right hand movement were found in mu and low beta rhythms and ranged from 9 to 17 Hz. Frequencies that correlated with a change from rest to right hand motor imagery were confined to the mu rhythm and ranged from 9 to 13 Hz. These findings indicate that although both mu and low beta rhythms are a good indicator of movement, mu rhythms are better suited for BCI control in absence of movement. It can be concluded that the EEG signal over the sensorimotor cortex in the mu frequency range is a good candidate for BCI control and reflects the user intent on movement imagery.

In the present study, it was chosen to involve real movement during the screening task, in contrast to motor imagery during the training task. It is known that both movement and motor imagery involve the same cortical regions and are accompanied by desynchronization of the SMR in mu and beta bands [27]. An advantage of using a motor task for the screening instead of motor imagery is that the researcher does not rely on the ability of the participant to perform motor imagery. How well motor imagery is performed is difficult to assess, whereas a motor task is visible for the researcher. Additionally, all participants had no prior experience with motor imagery or neurofeedback. Therefore, using a motor task for screening was a more reliable choice.

Using real movement for selecting the target frequency for BCI control has several downsides. Firstly, the resulting frequency range can be sub-optimal for BCI control. Although the patterns of mu and beta desynchronization with motor imagery are similar to those during movement, they are not identical [27]. The responses of mu and beta frequencies can change between screening and training sessions. This was accounted for by inspecting the r^2 and frequency spectra after the first and second motor imagery sessions, and by restricting the target frequency choice to the mu frequency range which was more suitable for BCI control according to the findings in the present study (Appendix A). Secondly, a motor task cannot be performed by patients that are unable to move their right arm. In that case, the screening would have be performed with motor imagery. When accompanied with appropriate instructions on how to perform motor imagery, this should not be a problem, since inspection of screening and motor imagery spectra obtained in the present study show that motor imagery

performance was executed successfully in most of the participants from their first try.

4.1.2. Neurofeedback training

Results of the neurofeedback training showed that SMR modulation performance based on motor imagery can be obtained in one session and can improve over the course of 5-6 training sessions divided over two to four weeks.

Five participants completed five to eight neurofeedback sessions of about 40 minutes each. Four of the five participants acquired SMR control within the first neurofeedback session. These results are in line with previous studies [17, 28, 29], in which participants learned SMR modulation within one neurofeedback session. Three of the four participants improved their performance over the training sessions. The fourth participant did not show overall improvement but did have good SMR control from the start. Improvement in performance can be attributed to implicit neurofeedback learning, adaptation to the task and environment, development of better strategies and tuning of the EEG processing parameters.

One participant was unable to acquire sufficient control over the SMR amplitude. This is a known phenomenon: a significant proportion of people, up to about 50% in some studies [30, 31], is unable to learn to control their brain signal according to literature [32]. Different reasons have been proposed for this phenomenon. Neurophysiological factors such as differences in SMR characteristics and brain structural properties, but also psychological factors such as mental strategies, concentration, mood, and motivation can be the cause of this. Analysis of the SMR amplitude during up and down trials showed that this participant did on average produce higher SMR amplitudes during up trials than down trials. Apparently this difference was too small to be detected with the processing pipeline at hand and/or was too inconsistent to successfully control the cursor. Possibly, the low amount of hits worked demotivating to this participant, which contributed to his inability to improve performance over the following sessions. Additionally, the training sessions for this participant were highly irregular. Therefore, it can be argued that the term ‘non-learner’ can be deceiving: the same participant might be able to acquire SMR control in a setup with more frequent training sessions, a more motivating feedback application, and/or a protocol with different signal processing.

An interesting observation is that r^2 values that were established after the screening and first training session were good predictors of neurofeedback task performance in the long term. As can be seen in figure 3.4, the highest overall performance was achieved by participant 6, who also had the highest r^2 in the screening session. On the other hand, participant 3 had the lowest performance, and failed to increase this performance. This participant had a r^2 that was an order of magnitude lower than that of other participants. The observation that r^2 is a good predictor of performance on the short term might not be new, since r^2 essentially reflects how well user intent may be inferred from a brain signal. However, the latter does not imply that it predicts whether the participant will improve performance in the following sessions, while the findings presented here do. For example, regardless of the difference in r^2 between participant 3 and the other participants, the success rate in session 1 of participant 3 is similar to that of participants 4 and 5. However, while participants 4 and 5 quickly improved their success rates in the following sessions, the success rate of participant 3 only decreased. Therefore, this remains an interesting finding and can be useful in future studies as a predictor of neurofeedback performance.

Another finding from the present study regarding training task performance is that maintaining the ability to control SMR amplitude during reflex assessment is challenging: several factors could contribute to this complication. Here, three of these factors are distinguished: the change in posture, additional constraint on the arm, and the requirement to maintain muscle contraction. The change in posture and introduction of the constraint alone seemed to considerably worsen performance. It is possible that constraining the arm lead to diminished postural comfort, which is known to affect BCI performance [33]. It is therefore important to put additional emphasis on the level of comfort of the participant while performing reflex assessment in future experiments.

The isometric task during reflex assessment that requires to keep a small force on the handle can change the SMR amplitude characteristics. The Normalizer filter accounts for a changed baseline SMR amplitude and variance as long as enough trials have been performed where the Normalizer filter is allowed to adapt to the new parameters. Yet, it is possible that the isometric task interferes with the ability to modulate SMR. Therefore, it can be helpful to perform several training sessions with the arm of the participant in the wrist manipulator while maintaining a small force, to give the participant enough time to practice SMR control under the changed circumstances.

4.1.3. Reflex Assessment

Stretch reflex data during high and low SMR amplitude conditions was obtained for two participants. In one participant, magnitude of M2 was higher with high SMR amplitude. Results of this participant are in line with the results of Thompson et al. [18], the research group that performed a similar assessment using H-reflexes. They found that the magnitude of the H-reflex was higher during SMR up and lower during SMR down conditions. A similar result is expected in the M1 and/or M2 components of the stretch reflex due to the similarity of the neuronal pathways between the stretch reflex and the H-reflex.

In the other participant, reflex sizes were similar in both conditions, with only slightly higher M1 for SMR up and slightly lower M2 for SMR down conditions. It is possible that diminished SMR control during reflex assessment has played a role. The diminished SMR control was accompanied by a smaller difference in SMR amplitude between the two conditions. It is possible that with small differences in SMR amplitudes, the difference in cortical drive is so small that it is hardly noticeable in the reflex strength.

Increased reflex strength during high SMR amplitude compared to low SMR amplitude can be explained by the role of the SMR in cortical activation. SMR amplitude in the mu and beta range is inversely correlated with cortical activation. High SMR amplitude reflects synchronization of the rhythm, which is associated with cortical inhibition. Low SMR amplitude reflects desynchronization of the rhythm, associated with cortical activation. Therefore, with high SMR amplitude, the cortical drive to spinal motorneurons decreases. Decreased cortical drive to motor neurons is associated with bigger reflex sizes, and increased cortical drive has the opposite effect [18].

4.2. Limitations

The presented neurofeedback protocol design comes with several limitations. One of the them is the uncertainty about the SMR amplitude at the time that the reflex is evoked. Assessment of reflex strength for each condition relied on successful trials: if the perturbation was given during a trial in which the target was subsequently ‘hit’, the resulting EMG data were included in the analysis under the corresponding condition. However, it is debatable whether a successful trial is a good measure of SMR amplitude. Indeed, the cursor height does not directly reflect the instantaneous SMR amplitude, but rather, reflects the course of the SMR amplitude from the beginning of the trial. This means that if a perturbation of the wrist was performed during a successful SMR-up trial, the SMR amplitude might not necessarily be high at the instant of the perturbation. This was an important limitation in this study: even though on average, the SMR amplitude was high near the end of SMR-up trials and low near the end of SMR-down trials (figure 3.5), this cannot be stated for individual trials. This means that it cannot be stated that all EMG data corresponding to the SMR-up condition were indeed measured while the SMR amplitude was high, and the same can be said of the SMR-down condition. A solution would be to only trigger the reflex when the SMR amplitude reaches a certain threshold value, but this would require implementing a new BCI2000 block that would send information to the wrist manipulator.

Another design flaw in this neurofeedback protocol is that the success rate can be a faulty representation of the subjects’ ability to regulate the SMR when the difference in SMR amplitude between SMR-up and SMR-down conditions becomes very small. The amplitude difference needed to reach the upper versus lower target is not standardized between the subjects and the sessions: the ‘threshold’ amplitude that is needed to reach one of the boundaries depends on the Normalizer Gain parameter (chapter 2) which is recalculated at the start of every session. No lower bound exists for this difference: when the variance of the EEG signal is low, the Normalizer Gain will be high, meaning the slightest change in SMR amplitude will move the cursor up or down. This means that even involuntary fluctuations and/or noise become very likely to affect the cursor such that it will hit the target by chance. The consequence of this is that when reflex strength across the two conditions is assessed, the SMR amplitude might not vary significantly, which interferes with the interpretation of the results. Therefore, when reflex strength is assessed for each of the conditions, it is important to examine whether SMR amplitude variance was indeed high and whether the signal was consistent enough to expect the cortical drive to the spinal cord to vary.

Another important limitation is that from the four participants who took part in the reflex assessment, only two had distinguishable reflexes. Unfortunately, after completing the training successfully, the EMG data of the other two subjects could not be used. This is an important limitation: neurofeedback training is a time consuming procedure which spans several weeks, it is therefore important that EMG data can

be used. To overcome this limitation, it is useful to perform a reflex assessment before starting the training program, and make sure both the participant and researcher are well aware of their task. This will eliminate subjects with no measurable reflexes, and as an extra advantage, save instruction time in the final reflex assessment session.

4.3. Recommendations for Future Research

Results presented in this study provide anecdotal evidence that modulation of SMR amplitude can affect the strength of stretch reflexes. Future research should focus on replicating these results in a bigger participant group, then move on to more diverse participant populations. An important milestone would be to replicate the results in people with reflex disorders, who are the target group for rehabilitation protocols. Additionally, it is important to introduce a control group, i.e. a part of the participant group receive no neurofeedback or sham-neurofeedback. Appropriate control conditions are necessary to establish causality, verify non-specific effects and assess the added clinical benefit of the neurofeedback procedure. Furthermore, future research should address the long-term effects of neurofeedback training on reflex size. This includes questions as: How long does the reflex modulation effect last after the end of the neurofeedback trial? And are there any persisting changes in reflex strength one day, one week, or one month after the training? This knowledge is essential in the translation of this protocol to clinical settings.

5

Conclusion

This report presents a protocol in which people can change their SMR amplitude with help of feedback, and their reflex strength is assessed. The presented neurofeedback protocol provides a way to investigate the effect of the SMR on the strength of the mechanically evoked stretch reflex. To the best of the researcher's knowledge, this is the first time that the effect of SMR modulation is examined on mechanically evoked stretch reflexes. The use of this protocol can contribute to the understanding of reflex modulation mechanisms and the function of the SMR, and can potentially help in the design of new rehabilitation protocols for people with neurological disorders. Results of the validation experiment bring new insights on how the stretch reflex is affected by the SMR amplitude. The results indicate that, with sufficient control over the SMR amplitude, it is possible to observe a difference in the strength of the M2 component of the stretch reflex. Based on these findings, the neurofeedback protocol might become valuable in regulating of abnormally high or low reflex activity. However, several important questions regarding the effects of neurofeedback remain to be answered before these findings can be translated to clinical settings.

Bibliography

[1] Jonathan Shemmell, Matthew A Krutky, and Eric J Perreault. Stretch sensitive reflexes as an adaptive mechanism for maintaining limb stability. *Clinical Neurophysiology*, 121(10):1680–1689, 2010.

[2] Pratik K Mutha. Reflex circuits and their modulation in motor control: a historical perspective and current view. *Journal of the Indian Institute of Science*, 97(4):555–565, 2017.

[3] Melanie M Adams and Audrey L Hicks. Spasticity after spinal cord injury. *Spinal cord*, 43(10): 577–586, 2005.

[4] Carel GM Meskers, Alfred C Schouten, Jurriaan H de Groot, Erwin de Vlugt, Bob JJ van Hilten, Frans CT Van der Helm, and Hans JH Arendzen. Muscle weakness and lack of reflex gain adaptation predominate during post-stroke posture control of the wrist. *Journal of neuroengineering and rehabilitation*, 6(1):29, 2009.

[5] Aiko K Thompson, N Mrachacz-Kersting, T Sinkjær, and JB Andersen. Modulation of soleus stretch reflexes during walking in people with chronic incomplete spinal cord injury. *Experimental brain research*, 237(10):2461–2479, 2019.

[6] Jonathan Shemmell, Je Hi An, and Eric J Perreault. The differential role of motor cortex in stretch reflex modulation induced by changes in environmental mechanics and verbal instruction. *Journal of Neuroscience*, 29(42):13255–13263, 2009.

[7] Mitsuaki Takemi, Yoshihisa Masakado, Meigen Liu, and Junichi Ushiba. Event-related desynchronization reflects downregulation of intracortical inhibition in human primary motor cortex. *Journal of neurophysiology*, 110(5):1158–1166, 2013.

[8] M Takemi, Y Masakado, Meigen Liu, and Junichi Ushiba. Sensorimotor event-related desynchronization represents the excitability of human spinal motoneurons. *Neuroscience*, 297:58–67, 2015.

[9] Ernst Niedermeyer et al. The normal eeg of the waking adult. *Electroencephalography: Basic principles, clinical applications, and related fields*, 167:155–164, 2005.

[10] Jonathan R Wolpaw, Dennis J McFarland, Gregory W Neat, and Catherine A Forneris. An eeg-based brain-computer interface for cursor control. *Electroencephalography and clinical neurophysiology*, 78(3):252–259, 1991.

[11] Surjo R Soekadar, Niels Birbaumer, Marc W Slutzky, and Leonardo G Cohen. Brain–machine interfaces in neurorehabilitation of stroke. *Neurobiology of disease*, 83:172–179, 2015.

[12] Alyssa Merante, Yu Zhang, Satyam Kumar, and Chang S Nam. Brain–computer interfaces for spinal cord injury rehabilitation. In *Neuroergonomics*, pages 315–328. Springer, 2020.

[13] Janis J Daly and Jonathan R Wolpaw. Brain–computer interfaces in neurological rehabilitation. *The Lancet Neurology*, 7(11):1032–1043, 2008.

[14] Wenya Nan, Ana Paula Barbosa Dias, and Agostinho Claudio Da Rosa. Neurofeedback training for cognitive and motor function rehabilitation in chronic stroke: two case reports. *Frontiers in Neurology*, 10:800, 2019.

[15] Catharina Zich, Stefan Debener, Clara Schweinitz, Annette Sterr, Joost Meekes, and Cornelia Kranczioch. High-intensity chronic stroke motor imagery neurofeedback training at home: three case reports. *Clinical EEG and neuroscience*, 48(6):403–412, 2017.

[16] Arash Mirifar, Jürgen Beckmann, and Felix Ehrlenspiel. Neurofeedback as supplementary training for optimizing athletes' performance: A systematic review with implications for future research. *Neuroscience & Biobehavioral Reviews*, 75:419–432, 2017.

[17] M Jarjees and A Vučković. The effect of voluntary modulation of the sensory-motor rhythm during different mental tasks on h reflex. *International Journal of Psychophysiology*, 106:65–76, 2016.

[18] Aiko K Thompson, Hannah Carruth, Rachel Haywood, N Jeremy Hill, William A Sarnacki, Lynn M McCane, Jonathan R Wolpaw, and Dennis J McFarland. Effects of sensorimotor rhythm modulation on the human flexor carpi radialis h-reflex. *Frontiers in neuroscience*, 12:505, 2018.

[19] David Burke, Simon C Gandevia, and Brian McKeon. The afferent volleys responsible for spinal proprioceptive reflexes in man. *The Journal of physiology*, 339(1):535–552, 1983.

[20] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

[21] Gerwin Schalk and Jürgen Mellinger. *A practical guide to brain–computer interfacing with BCI2000: General-purpose software for brain-computer interface research, data acquisition, stimulus presentation, and brain monitoring*. Springer Science & Business Media, 2010.

[22] Simon L Kappel, David Looney, Danilo P Mandic, and Preben Kidmose. Physiological artifacts in scalp eeg and ear-eeg. *Biomedical engineering online*, 16(1):1–16, 2017.

[23] Dennis J McFarland, Lynn M McCane, Stephen V David, and Jonathan R Wolpaw. Spatial filter selection for eeg-based communication. *Electroencephalography and clinical Neurophysiology*, 103(3):386–394, 1997.

[24] Dean J Krusienski, Dennis J McFarland, and Jonathan R Wolpaw. An evaluation of autoregressive spectral estimation model order for brain-computer interface applications. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1323–1326. IEEE, 2006.

[25] Dennis J McFarland and Jonathan R Wolpaw. Sensorimotor rhythm-based brain–computer interface (bci): model order selection for autoregressive spectral analysis. *Journal of neural engineering*, 5(2):155, 2008.

[26] Babette Mulder. Design of an experimental protocol to test voluntary reflex modulation in the wrist flexor. 2021.

[27] Dennis J McFarland, Laurie A Miner, Theresa M Vaughan, and Jonathan R Wolpaw. Mu and beta rhythm topographies during motor imagery and actual movements. *Brain topography*, 12(3): 177–186, 2000.

[28] Ana Alves-Pinto, Varvara Turova, Tobias Blumenstein, Conny Hantuschke, and Renée Lampe. Implicit learning of a finger motor sequence by patients with cerebral palsy after neurofeedback. *Applied psychophysiology and biofeedback*, 42(1):27–37, 2017.

[29] Ming-Yang Cheng, Chung-Ju Huang, Yu-Kai Chang, Dirk Koester, Thomas Schack, and Tsung-Min Hung. Sensorimotor rhythm neurofeedback enhances golf putting performance. *Journal of Sport and Exercise Psychology*, 37(6):626–636, 2015.

[30] Simon Hanslmayr, Paul Sauseng, Michael Doppelmayr, Manuel Schabus, and Wolfgang Klimesch. Increasing individual upper alpha power by neurofeedback improves cognitive performance in human subjects. *Applied psychophysiology and biofeedback*, 30(1):1–10, 2005.

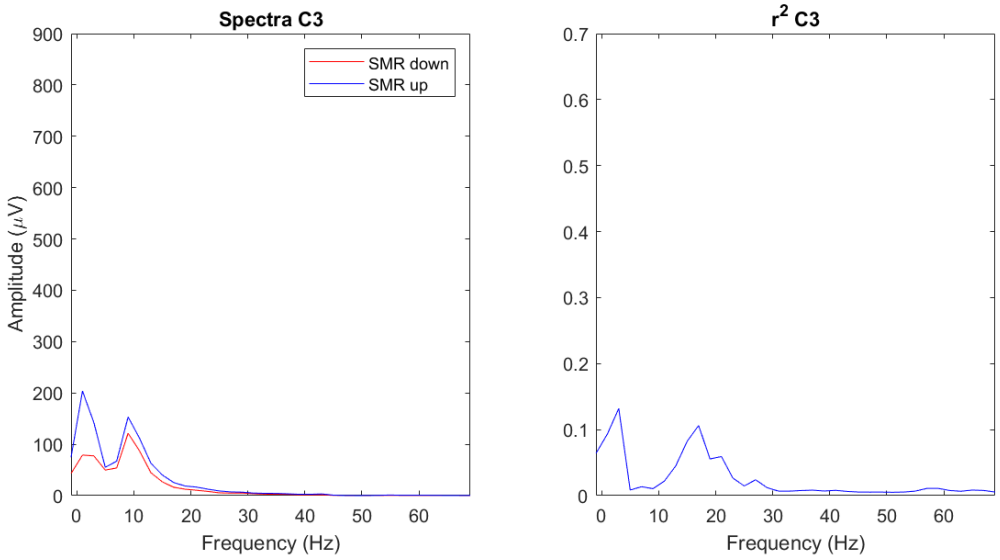
[31] Mirko Doehnert, Daniel Brandeis, Marc Straub, Hans-Christoph Steinhausen, and Renate Drechsler. Slow cortical potential neurofeedback in attention deficit hyperactivity disorder: is there neurophysiological evidence for specific effects? *Journal of Neural Transmission*, 115(10):1445–1456, 2008.

[32] Wenya Nan, Feng Wan, Qi Tang, Chi Man Wong, Boyu Wang, and Agostinho Rosa. Eyes-closed resting eeg predicts the learning of alpha down-regulation in neurofeedback training. *Frontiers in psychology*, page 1607, 2018.

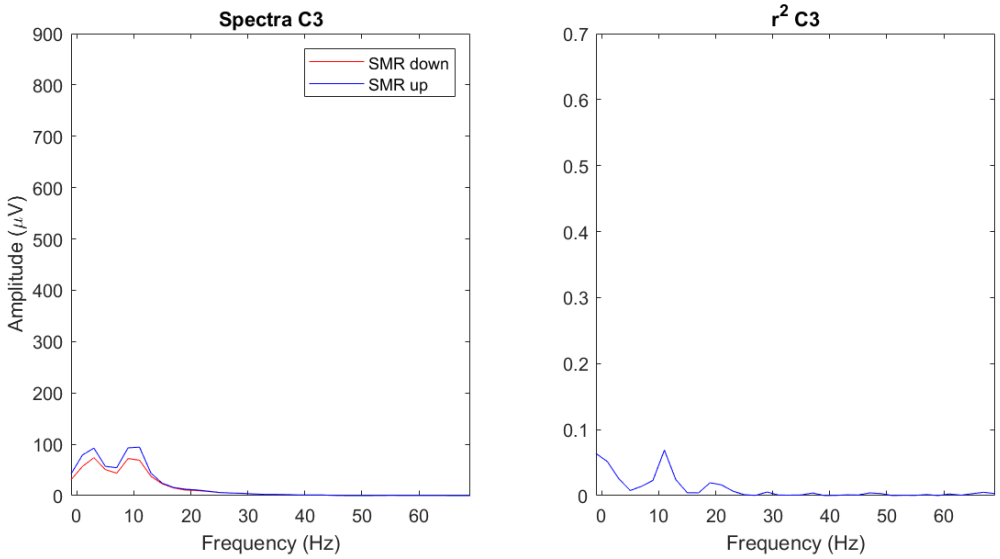
[33] Dojin Heo, Minju Kim, Jongsu Kim, Yun-Joo Choi, and Sung-Phil Kim. Effect of static posture on online performance of p300-based bcis for tv control. *Sensors*, 21(7):2278, 2021.

A

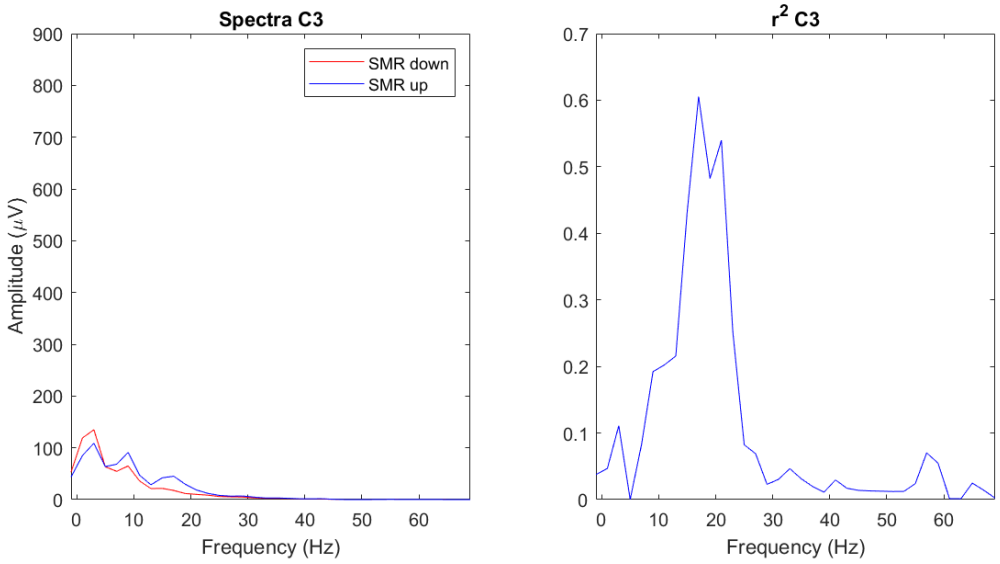
Spectra



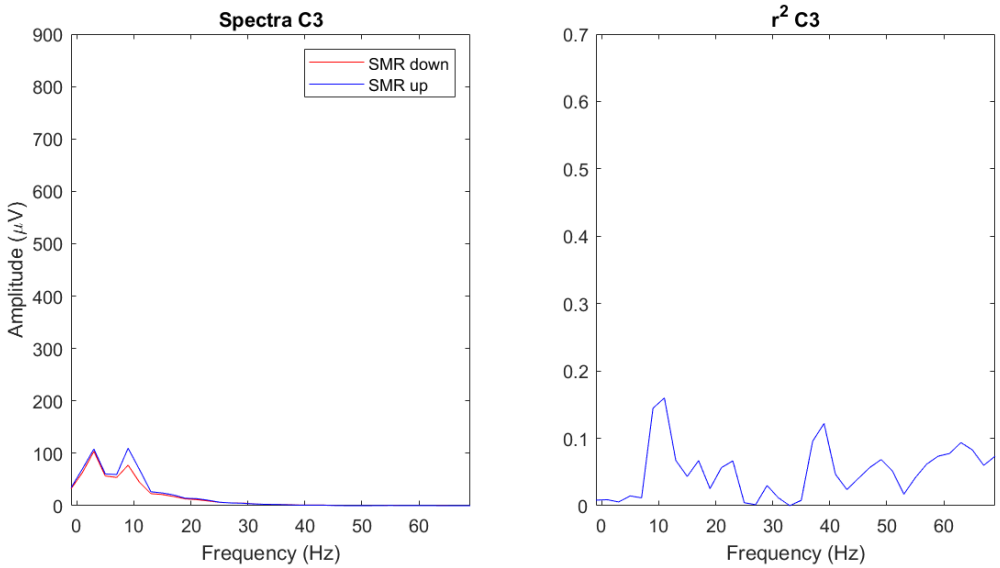
(a) Participant 3 - movement



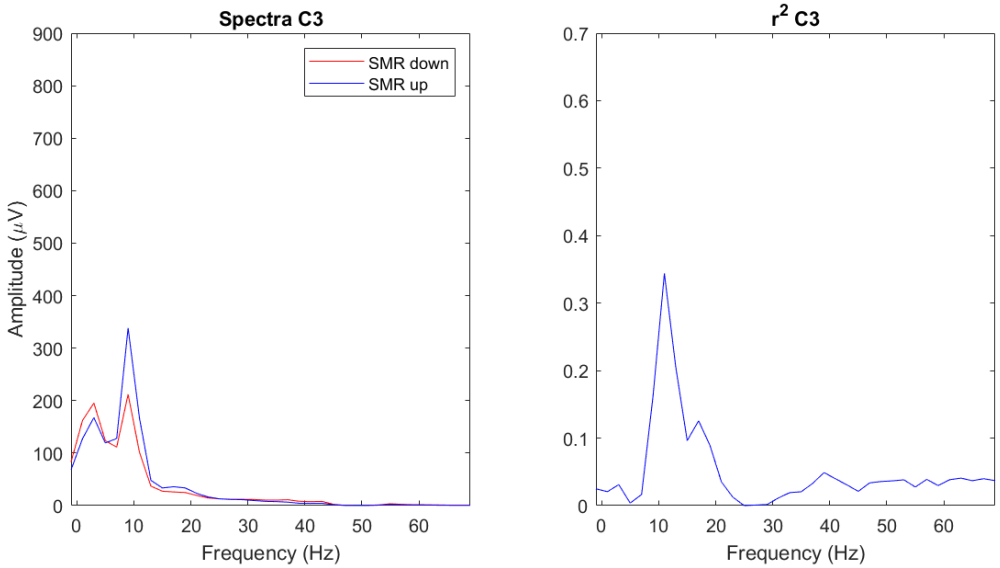
(b) Participant 3 - motor imagery



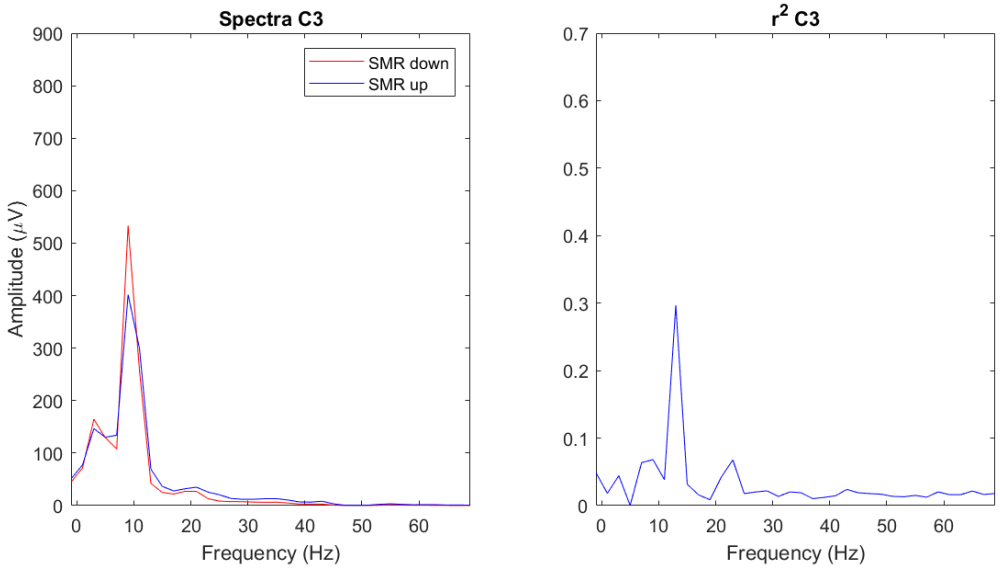
(c) Participant 4 - movement



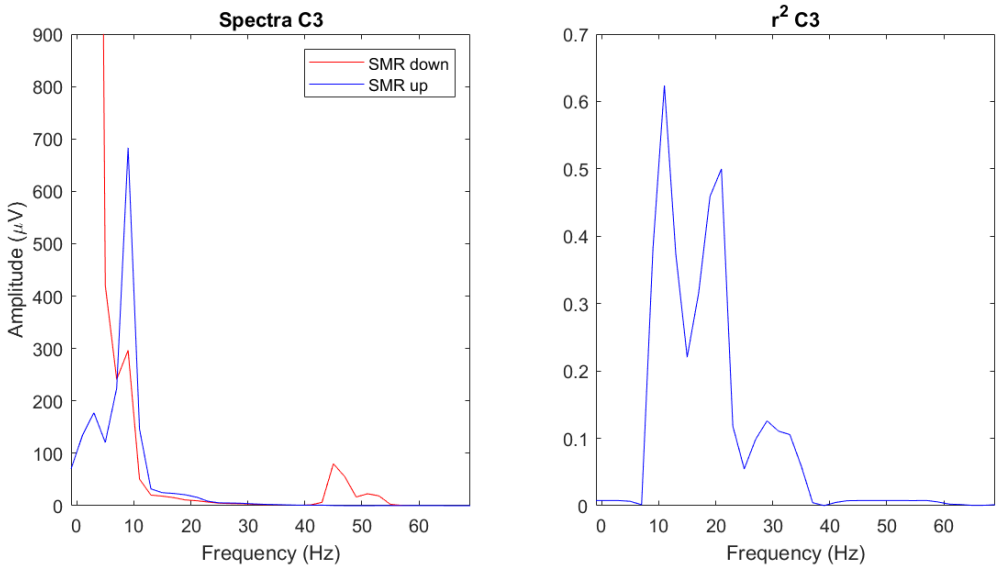
(d) Participant 4 - motor imagery



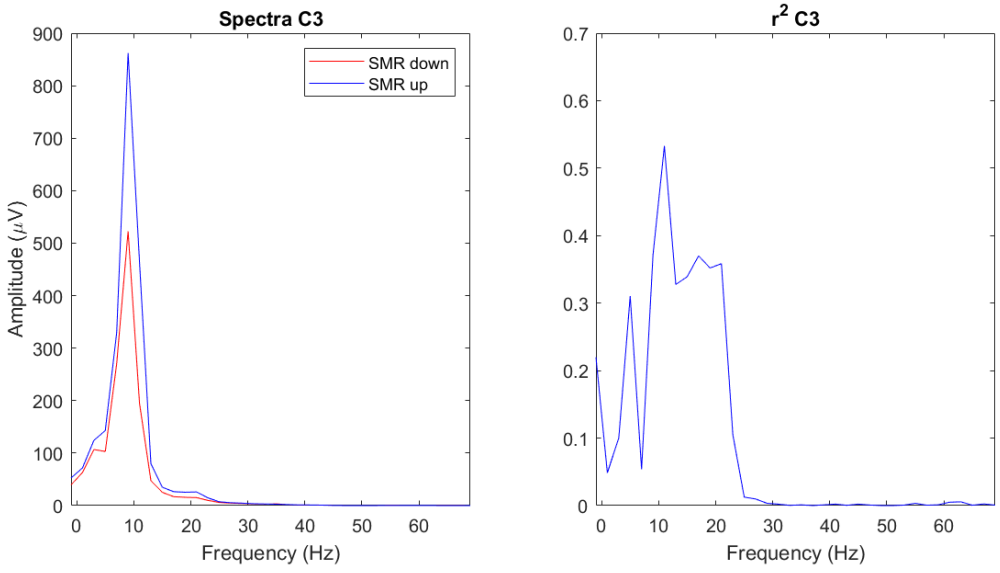
(e) Participant 5 - movement



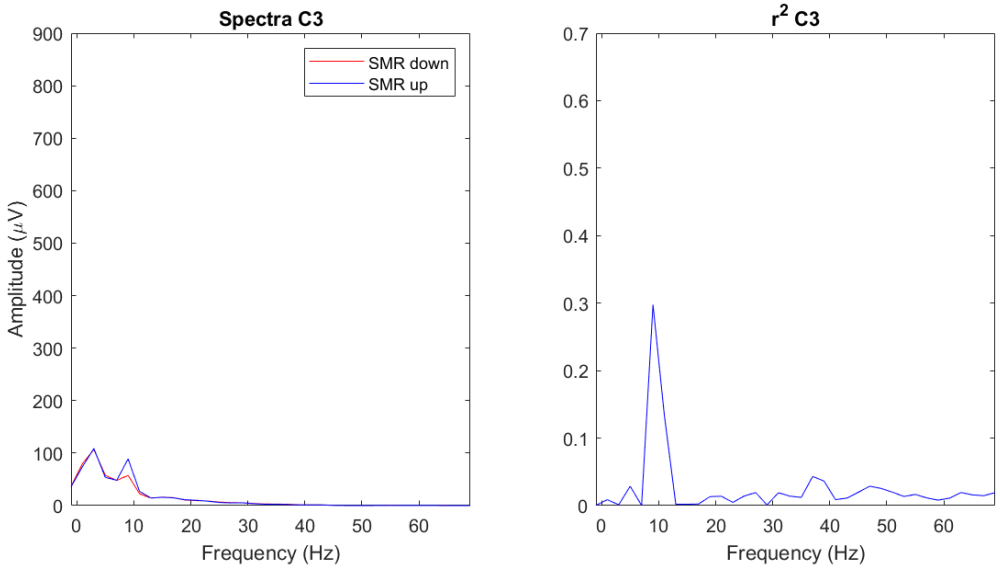
(f) Participant 5 - motor imagery



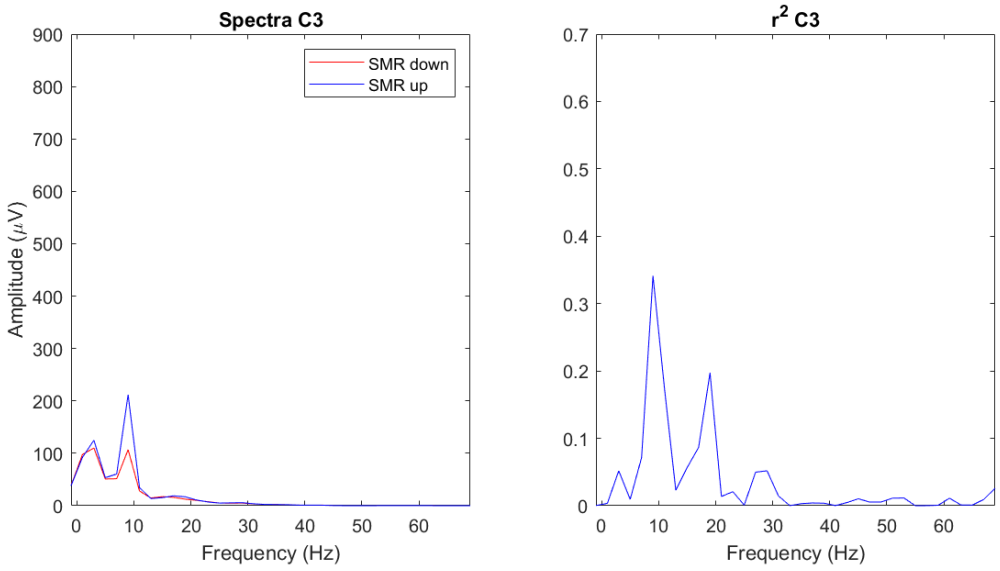
(g) Participant 6 - movement



(h) Participant 6 - motor imagery



(i) Participant 7 - movement



(j) Participant 7 - motor imagery

Figure A.1: Frequency spectra and r^2 spectra for all participants during screening (movement) and the first training (motor imagery) sessions

B

TMSiADC.cpp

This is a script that connects the BCI2000 software to the driver of the TMSi Refa Ext. EEG amplifier.

```
//#define DEBUG

#include "TMSiADC.h"
#include "BCIStream.h"

#include <tchar.h>
#include <iostream>
#include <chrono>
#include <ctime>

static const int cImpedanceRate = 8; // impedance sampling rate in Hz

using namespace std;

RegisterFilter(TMSiADC, 1);


TMSiADC::TMSiADC()
    : mValuesToRead(0),
    mBufferSize(0),
    mSrate(0),
    mBufferMulti(0),
    mHardwareCh(0),
    mSoftwareCh(0),
    mSampleBlockSize(0),
    mSampleRate(0),
    mDigitalChannel(0)
{
    bciout << "" << endl;
    Handle = NULL;
    ErrorCode = 0;

    for (size_t i = 0; i < sizeof(mSignalBuffer) / sizeof(*mSignalBuffer);
     ↪  ++i)
        mSignalBuffer[i] = 0;

    mBufferMulti = OptionalParameter("TMSiBufferSizeInSampleBlocks", 1);
    double x = 100.0 / (double)mBufferMulti;
```

```
    if (x != floor(x)) bcierr << "TMSiBufferSizeInSampleBlocks should be an
     ↪  integer factor of 100" << endl;
    //bciout << "mBufferMulti=" << mBufferMulti << endl;

    int priority = OptionalParameter("TMSiProcessPriority", 0);
    switch (priority)
    {
    case +3: SetPriorityClass(GetCurrentProcess(), HIGH_PRIORITY_CLASS);
     ↪  break;
    case +2: SetPriorityClass(GetCurrentProcess(),
     ↪  REALTIME_PRIORITY_CLASS); break;
    case +1: SetPriorityClass(GetCurrentProcess(),
     ↪  ABOVE_NORMAL_PRIORITY_CLASS); break;
    case  0: SetPriorityClass(GetCurrentProcess(), NORMAL_PRIORITY_CLASS);
     ↪  break;
    case -1: SetPriorityClass(GetCurrentProcess(),
     ↪  BELOW_NORMAL_PRIORITY_CLASS); break;
    case -2: SetPriorityClass(GetCurrentProcess(), IDLE_PRIORITY_CLASS);
     ↪  break;
    default: bcierr << "unsupported value --TMSiProcessPriority=" <<
     ↪  priority << endl;
    }
    mSleepMsec = OptionalParameter("TMSiSleepMsec", 1);

    // impedance measurement
    mMeasureImpedance = bool(int(OptionalParameter("TMSiCheckImpedance",
     ↪  0)));

    ConnectToLibrary();

    // Call LibraryInit
    Handle = fpLibraryInit(TMSiConnectionUSB, &ErrorCode);
    if (ErrorCode != 0)
        bcierr << "Can NOT initialize library, errorcode = %d" << ErrorCode
         ↪  << endl;
    if (Handle == INVALID_HANDLE_VALUE)
        bcierr << "Can NOT initialize library, INVALID_HANDLE_VALUE" <<
         ↪  endl;
    else
        bciout << "LibraryInit returned Handle = " << Handle << endl;

    StartDriver();

}

TMSiADC::~TMSiADC()
{

}


void
TMSiADC::ConnectToLibrary()
{
    TCHAR Path[MAX_PATH];
    TCHAR LibraryName[255] = _T("\\TMSiSDK.dll");
```

```
HINSTANCE LibHandle = NULL;     //Library Handle

GetSystemDirectory(Path, sizeof(Path) / sizeof(TCHAR));
lstrcat(Path, LibraryName);
//bciout << "Create system path for library, system path =" << Path <<
 ↪  endl;

LibHandle = LoadLibrary(Path);
if (LibHandle == NULL)
{
    bciout << "Could not load library" << endl;
}
else
{
    //bciout << "Library loaded succesfully = " << LibHandle << endl;
}

//bciout << "Get pointers to functions in the dll" << endl;

fpOpen = (POPEN)GetProcAddress(LibHandle, "Open");
fpClose = (PCLOSE)GetProcAddress(LibHandle, "Close");
fpStart = (PSTART)GetProcAddress(LibHandle, "Start");
fpStop = (PSTOP)GetProcAddress(LibHandle, "Stop");
fpSetSignalBuffer = (PSETSIGNALBUFFER)GetProcAddress(LibHandle,
 ↪  "SetSignalBuffer");
fpGetSamples = (PGETSAMPLES)GetProcAddress(LibHandle, "GetSamples");
fpGetBufferInfo = (PGETBUFFERINFO)GetProcAddress(LibHandle,
 ↪  "GetBufferInfo");
fpGetSignalFormat = (PGETSIGNALFORMAT)GetProcAddress(LibHandle,
 ↪  "GetSignalFormat");
fpFree = (PFREE)GetProcAddress(LibHandle, "Free");
fpLibraryInit = (PLIBRARYINIT)GetProcAddress(LibHandle, "LibraryInit");
fpLibraryExit = (PLIBRARYEXIT)GetProcAddress(LibHandle, "LibraryExit");
fpGetFrontEndInfo = (PGETFRONTENDINFO)GetProcAddress(LibHandle,
 ↪  "GetFrontEndInfo");
fpSetRefCalculation = (PSETREFCALCULATION)GetProcAddress(LibHandle,
 ↪  "SetRefCalculation");
fpSetRtcTime = (PSETRTCTIME)GetProcAddress(LibHandle, "SetRtcTime");
fpGetRtcTime = (PGETRTCTIME)GetProcAddress(LibHandle, "GetRtcTime");
fpSetRtcAlarmTime = (PSETRTCALARMTIME)GetProcAddress(LibHandle,
 ↪  "SetRtcAlarmTime");
fpGetRtcAlarmTime = (PGETRTCALARMTIME)GetProcAddress(LibHandle,
 ↪  "GetRtcAlarmTime");
fpGetErrorCode = (PGETERRORCODE)GetProcAddress(LibHandle,
 ↪  "GetErrorCode");
fpGetErrorCodeMessage = (PGETERRORCODEMESSAGE)GetProcAddress(LibHandle,
 ↪  "GetErrorCodeMessage");
fpGetDeviceList = (PGETDEVICELIST)GetProcAddress(LibHandle,
 ↪  "GetDeviceList");
fpFreeDeviceList = (PFREEDEVICELIST)GetProcAddress(LibHandle,
 ↪  "FreeDeviceList");


if (fpLibraryInit == NULL)
{
    bciout << "functions in library not found" << endl;
```

```
    }
    else
    {
        bciout << "found functions in library" << endl;
    }

}

void
TMSiADC::StartDriver()
{
    //bciout << "Call StartDriver" << endl;
    DeviceList = NULL;
    NrOfDevices = 0;
    char FrontEndName[MAX_FRONTENDNAME_LENGTH];

    //List all devices
    DeviceList = fpGetDeviceList(Handle, &NrOfDevices);
    if (NrOfDevices == 0)
    {
        ErrorCode = fpGetErrorCode(Handle);
        bcierr << "Frontend list NOT available, errorcode = " << ErrorCode
          ↪  << " " << fpGetErrorCodeMessage(Handle, ErrorCode) << endl;
    }
    else
    {
        bciout << "Number of found connections = " << NrOfDevices << endl;
    }

    Status = 0;
    if (DeviceList != NULL && DeviceList[0] != NULL)
    {
        char* DeviceLocator = DeviceList[0];
        bciout << "Try to connect to frontend using DeviceLocator = " <<
          ↪  DeviceLocator << endl;
        Status = fpOpen(Handle, DeviceLocator);
    }
    if (Status == 0)
    {
        ErrorCode = fpGetErrorCode(Handle);
        bcierr << "Frontend NOT available, errorcode = " << ErrorCode <<
          ↪  endl;
    }
    else
    {
        bciout << "Device Connected" << endl;
    }
    // To turn the reference calculation on, set the parameter to 1
    Status = fpSetRefCalculation(Handle, 1);
    if (Status == 0)
    {
        ErrorCode = fpGetErrorCode(Handle);
        bcierr << "SetRefCalculation NOT set, errorcode = " << ErrorCode <<
          ↪  endl;
    }
    else
```

```
    {
        bciout << "SetRefCalculation set" << endl;
    }
    psf = fpGetSignalFormat(Handle, FrontEndName);
    if (psf != NULL)
    {
        bciout << "FrontEndName = " << FrontEndName << endl;
        bciout << "NumberOfChannels = " << psf->Elements << endl;
        //fpFree(psf); // present in old TMSiADC.cpp but not in examplecode
    }
    else
    {
        ErrorCode = fpGetErrorCode(Handle);
        bcierr << "Can not get SignalFormat, errorcode = " << ErrorCode <<
          ↪  "," << fpGetErrorCodeMessage(Handle, ErrorCode) << endl;
    }
    mHardwareCh = psf->Elements;// 'NumberOfChannels' in examplecode =
      ↪  'mHardwareCh' in old TMSiADC.cpp
    Gain = std::vector<double>(mHardwareCh, 0);
    Offset = std::vector<double>(mHardwareCh, 0);

    for (unsigned int i = 0; i < mHardwareCh; i++)
    {
        int Exponent = psf[i].UnitExponent;
        if (mMeasureImpedance)
        {
            Gain[i] = 1.0;
            Offset[i] = 0.0;
        }
        else {
            if (Exponent)
            {
                if (psf[i].UnitId == 1)
                {      // if volts
                    Gain[i] = (psf[i].UnitGain * (pow(10.0, Exponent))) *
                      ↪  1000000;      // in uV
                    Offset[i] = (psf[i].UnitOffSet * (pow(10.0, Exponent)))
                      ↪  * 1000000;
                }
            }
            else {
                mDigitalChannel = i + 1;
                Gain[i] = 1.0;
                Offset[i] = 0.0;
            }
        }
    }
}

void TMSiADC::OnPublish()
{
    BEGIN_PARAMETER_DEFINITIONS
        //"Source:TMSiADC int       SourceCh=            16      19       1
          ↪  % // number of digitized and stored channels",
```

```
        //"Source:TMSiADC int       SampleBlockSize=     12      10       1
          ↪ % // number of samples transmitted at a time (multiples of 6
          ↪ seem smoothest)",
        //"Source:TMSiADC int       SamplingRate=       256Hz   256     256
          ↪ 2048 // SampleRate of the Input Device",
        "Source:TMSiADC intlist   PhysicalChannels= 16  1 2 3 4 5 6 7 8 9 10
          ↪ 11 12 13 14 15 16 "
        "1       1    % // hardware indices of channels to record",
        "Source:TMSiADC floatlist SourceChGain=     16  0.0715 0.0715
          ↪ 0.0715 0.0715 0.0715 0.0715 0.0715 0.0715 0.0715 0.0715
          ↪ 0.0715 0.0715 0.0715 0.0715 0.0715 "
        "0.0715  0    % // ",
        "Source:TMSiADC floatlist SourceChOffset=   16  0 0 0 0 0 0 0 0 0 0
          ↪ 0 0 0 0 0 0 "
        "0       %    % // ",
    END_PARAMETER_DEFINITIONS
}

void TMSiADC::OnPreflight(SignalProperties& outputProperties) const
{
    int MaxSamplingRate = 2048; //Hz
    unsigned int BytesPerSample;

    BytesPerSample = mHardwareCh * sizeof(long);
    if (BytesPerSample == 0)
        bcierr << "BytesPerSample == 0" << endl;
    else
        bciout << "BytesPerSample == " << BytesPerSample << endl;

    // Check amount of source channels
    int softwareCh = Parameter("SourceCh");
    if (softwareCh > static_cast<int>(mHardwareCh))
        bcierr << "Trying to read more channels than available" <<
          ↪ std::endl;
    else
        bciout << "SourceCh = " << softwareCh << endl;


    outputProperties = SignalProperties(softwareCh,
        Parameter("SampleBlockSize"),
        SignalType::float32);

    if (Parameter("PhysicalChannels")->NumValues() != softwareCh)
        bcierr << "Number of elements in PhysicalChannels must match
          ↪ SourceCh" << endl;
    else
        bciout << "Number of PhysicalChannels = " <<
          ↪ Parameter("PhysicalChannels")->NumValues() << endl;

    bool goodVals = true;
    bool gainWarning = false;
    bool offsetWarning = false;
    for (int i = 0; i < softwareCh; ++i)
    {
        //SIZE_T ind = int(Parameter("PhysicalChannels")(i)) - 1; //gave
          ↪ warning
```

```
        SIZE_T ind = SIZE_T(Parameter("PhysicalChannels")(i)) - 1;

        if (ind < 0) bcierr << "A PhysicalChannels index of " << ind + 1 <<
         ↪  "is illegal" << endl;
        if (ind >= mHardwareCh) bcierr << "A PhysicalChannels index of " <<
         ↪  ind + 1 << "exceeds the hardware's maximum of " << mHardwareCh
         ↪  << endl;

       //Impedance has gain/offset of 1/0, warn if different but set anyway
        if (mMeasureImpedance)
        {
            if (float(Parameter("SourceChGain")(i)) != Gain[ind] &&
             ↪  !gainWarning)
            {
                bciout << "Using user-defined SourceChGain values for
                 ↪  impedance measurement, and assuming that you have
                 ↪  correctly calibrated these ";
                bciout << "(e.g. channel " << i + 1 << " has gain " <<
                 ↪  Parameter("SourceChGain")(i) << ")." << std::endl;
                gainWarning = true;
            }
            if (float(Parameter("SourceChOffset")(i)) != Offset[ind] &&
             ↪  !offsetWarning)
            {
                bciout << "Using user-defined SourceChOffset values for
                 ↪  impedance measurement, and assuming that you have
                 ↪  correctly calibrated these ";
                bciout << "(e.g. channel " << i + 1 << " has offset " <<
                 ↪  Parameter("SourceChOffset")(i) << ")." << std::endl;
                offsetWarning = true;
            }
        }
        //Check whether BCI Gains are correct
        else
        {
            double prmVal = Parameter("SourceChGain")(i);
            double hardwareVal = Gain[ind];
            bool same = (1e-3 > ::fabs(prmVal - hardwareVal) / (hardwareVal
             ↪  ? hardwareVal : 1.0));
            goodVals &= same;
            if (!same) bciout << "The amp driver says the gain of"
                << " channel " << ind + 1
                << " is " << hardwareVal
                << " whereas the corresponding value in the"
                << " SourceChGain parameter is " << prmVal << endl;

        }
    }
    if (!goodVals)
        bcierr << "The SourceChGain values "
        << "must match the hardware channel resolutions"
        << endl;
    else bciout << "SourceChGain = " << Parameter("SourceChGain") << endl;

    //Sampling Rate has to be 8Hz (Check for other Amps?), warn if data is
     ↪  updated less than once a second
```

```
    if (mMeasureImpedance)
    {
        bciout << "Impedance Mode is ON" << endl;
        int trueSamplingRate = cImpedanceRate;
        if (int(Parameter("SamplingRate").InHertz()) != trueSamplingRate) {
            bciout << "For impedance measurement, using a Sampling rate of
             ↪  " << trueSamplingRate << "Hz, rather than " <<
             ↪  int(Parameter("SamplingRate").InHertz()) << "Hz." <<
             ↪  std::endl;
        }
        if (int(Parameter("SampleBlockSize")) > 8) {
            bciout << "SampleBlockSize=" <<
             ↪  int(Parameter("SampleBlockSize")) << " is very big for a
             ↪  SamplingRate of " << trueSamplingRate << "\n";
        }
    }
    else
    {
        bciout << "Normal Sampling Mode is ON" << endl;


        if (int(Parameter("SamplingRate").InHertz()) > MaxSamplingRate)
        {
            bciout << "Sampling Rate is too high, max sampling rate =" <<
             ↪  MaxSamplingRate << "Hz, rather than " <<
             ↪  int(Parameter("SamplingRate").InHertz()) << "Hz." <<
             ↪  std::endl;
        }
    }
}

void TMSiADC::OnInitialize(const SignalProperties&)
{
    bciout << "Call Initialize" << endl;
    ULONG PercentFull, Overflow;
    ULONG SampleRateInMilliHz;
    ULONG SignalBufferSizeInSamples;

    Ohm = IC_OHM_010; // Impedance limit can be set to 2, 5, 10, 20, 50,
     ↪  100 and 200 kOhm

    mSoftwareCh = Parameter("SourceCh");
    mSampleBlockSize = Parameter("SampleBlockSize");
    mSampleRate = static_cast<unsigned
     ↪  int>(Parameter("SamplingRate").InHertz());

    bciout << "SourceCh = " << mSoftwareCh << endl;
    bciout << "SampleBlockSize = " << mSampleBlockSize << endl;
    bciout << "Sampling rate = " << mSampleRate << endl;

    // Find maximal sample rate and buffer size
    SampleRateInMilliHz = MAX_SAMPLE_RATE;
    SignalBufferSizeInSamples = MAX_BUFFER_SIZE;

    if (fpSetSignalBuffer(Handle, &SampleRateInMilliHz,
     ↪  &SignalBufferSizeInSamples) != TRUE)
```

```
{
    ErrorCode = fpGetErrorCode(Handle);
    bciout << "SetSignalBuffer 1 failed, error message =" <<
     ↪ fpGetErrorCodeMessage(Handle, ErrorCode) << endl;
}
else
    bciout << "Max sample rate = " << SampleRateInMilliHz / 1000 <<
     ↪ "Hz, Max buffer size = " << SignalBufferSizeInSamples <<
     ↪ "samples." << endl;

// Set sample rate and buffer size
mBufferSize = 50* mSampleBlockSize; // in samples!: in waitfordata
 ↪ endblock is linked to this...
//mBufferSize = mBufferMulti * mSampleBlockSize * mHardwareCh;
//mBufferSize = mSampleBlockSize; // in samples
mSrate = 1000 * mSampleRate;           // samplerate in mHz

mPhysChanInd.clear();
for (SIZE_T i = 0; i < mSoftwareCh; ++i)
 ↪ mPhysChanInd.push_back(int(Parameter("PhysicalChannels")(i)) - 1);

if (fpSetSignalBuffer(Handle, &mSrate, &mBufferSize) == FALSE)
{
    ErrorCode = fpGetErrorCode(Handle);
    bcierr << "SetSignalBuffer 2 failed, error message =" <<
     ↪ fpGetErrorCodeMessage(Handle, ErrorCode) << endl;
}
else
{
    bciout << "Buffersize in samples set in SetSignalBuffer = " <<
     ↪ mBufferSize << endl;
    bciout << "SampleRateInMilliHz set in SetSignalBuffer = " << mSrate
     ↪ << endl;
}

//(comment copied from old code:) I wonder where the difference to
 ↪ mBuffersize is...
//mValuesToRead = 50* mSampleBlockSize * mHardwareCh * 4;
mValuesToRead = (mSampleBlockSize + 16) * mHardwareCh * 4;
//mValuesToRead = 1 * mHardwareCh * 4;   // sizeof type? type
 ↪ dependent? TMS appears to only give 4 byte values.
bciout << "Selected buffersize for GetSamples in bytes = " <<
 ↪ mValuesToRead << endl;

if (mMeasureImpedance)
{
    // Set measuring mode to "impedance"
    if (fpSetMeasuringMode(Handle, MEASURE_MODE_IMPEDANCE_EX, Ohm))
    {
        mSampleRate = cImpedanceRate;
        bciout << "Set to impedance measuring mode. impedance limit set
         ↪ to" << Ohm << endl;
    }
    else
    {
        ErrorCode = fpGetErrorCode(Handle);
```

```
            bcierr << "Unable to set impedance mode, errorcode=" << ErrorCode
               ↪  << "," << fpGetErrorCodeMessage(Handle, ErrorCode) << endl;
        }
    }

}

//void
//TMSiADC::Halt()
//{
//    // STILL TO IMPLEMENT:  'if device is running'
//    // [............]
//
//    fpStop(Handle);
//    bciout << "Call Halt to Stop the device" << endl;
//}

void TMSiADC::OnStartAcquisition()
{
    // Start the device
    if (fpStart(Handle))
    {
        bciout << "Started device" << endl;
        //fpGetBufferInfo(Handle, &Overflow, &PercentFull);
    }
    else
    {
        ErrorCode = fpGetErrorCode(Handle);
        bcierr << "Could NOT start device, errorcode = " << ErrorCode <<
          ↪  "," << fpGetErrorCodeMessage(Handle, ErrorCode) << endl;
    }
}

void TMSiADC::OnStopAcquisition()
{
    fpStop(Handle);
}

void TMSiADC::DoAcquire(GenericSignal& outputSignal)
{
    //bciout << "&mSignalBuffer[0] = " << &mSignalBuffer[0] << endl;
    if (WaitForData(&mSignalBuffer[0], mValuesToRead) == TMSIOK)
    {
        const LONG* base = mSignalBuffer;
        /*for (unsigned int sample = 0; sample < mSampleBlockSize;
          ↪  ++sample)*/
        for (unsigned int sample = 0; sample < outputSignal.Elements();
          ↪  ++sample)
        {
           for (unsigned int channel = 0; channel < mSoftwareCh; ++channel)
            {
                int phys = mPhysChanInd[channel];
                if (phys == mDigitalChannel - 1) outputSignal(channel,
                  ↪  sample) = (float)((UCHAR)(~base[phys]));
                else outputSignal(channel, sample) = (float)(base[phys]);
            }
```

```
            base += mHardwareCh;
        }
        /*bciout << "reading data!!" << endl;*/
    }
    else
        bcierr << "Error reading data" << endl;
}


//-------------------------------------------------------------------------
 ↪  ----------------------------
//
//  25/10/05 MMS
//
//  WaitForData:
//
//  Blocking function that fills the SignalBuffer with 'size' samples.
//
//
//
//-------------------------------------------------------------------------
 ↪  ----------------------------
int
TMSiADC::WaitForData(LONG* SignalBuffer, ULONG size)
{
    ULONG PercentFull, Overflow;
    static unsigned int mOverflow = 0;
    ULONG endblock = static_cast<ULONG>(100.00 / mBufferMulti);
    ULONG BytesReturned = NULL;

    fpGetBufferInfo(Handle, &Overflow, &PercentFull);

    while (PercentFull < 2)
    //while (PercentFull < endblock)
    {
        Sleep(mSleepMsec);
        fpGetBufferInfo(Handle, &Overflow, &PercentFull);
        //bciout << "Overflow :" << Overflow << " % Full= " << PercentFull
         ↪  << endl;
    }

    if (Overflow > mOverflow)
    {
        mOverflow = Overflow;
        bciout << "Overflow occurred: " << Overflow << " % Full= " <<
         ↪  PercentFull << endl;
        // allow other applications some extra process time
        //Sleep(10);
        //bciout << "Sleep 10 sec" << endl;
    }


    BytesReturned = fpGetSamples(Handle, (PULONG)SignalBuffer, size);
     ↪  //size = signal buffer size in bytes
```

```
    if (BytesReturned != size)
    {
        bciout << "BytesReturned: " << BytesReturned << "; wanted: " <<
          ↪  size << endl;
        if (BytesReturned == 0)
            bciout << "BytesReturned: " << BytesReturned << "; no new data
              ↪  is available." << endl;
        if (BytesReturned < 0)
        {
            ErrorCode = fpGetErrorCode(Handle);
            bcierr << "BytesReturned: " << BytesReturned << "; wanted: " <<
              ↪  size << ", Error code" << ErrorCode << ", error message = "
              ↪  << fpGetErrorCodeMessage(Handle, ErrorCode) << endl;
        }

    }
    else
        bciout << "BytesReturned: " << BytesReturned << endl;


    return TMSIOK;


}
```