Real-Time EEG Neurofeedback as a Tool to Improve Neural Entrainment to Speech

Francisco Javier Carrera Arias¹, Nicola Molinaro^{1,2}, Mikel Lizarazu¹

09/04/2020

¹ Basque Center for Cognition, Brain and Language (BCBL)

² Ikerbasque, Basque Foundation for Science Bilbao, Spain

Abstract

Neurofeedback represents a particular type of biofeedback whose aim is to teach self-control of brain function by measuring brain activity and presenting a feedback signal in real-time. Traditionally, neurofeedback has been used to complement interventions for various neuropsychological disorders through techniques like frequency training, which attempts to change the power ratio of certain EEG frequency bands. However, to date, there are no neurofeedback approaches that look directly into modulating the neural entrainment to speech. Speech-brain entrainment, which stands for the alignment of the neural activity to the envelope of the speech input, has been shown to be key to speech comprehension. In fact, atypical neural entrainment to speech seems to be consistently found in language development disorders such as dyslexia. Thus, making speech entrainment neurofeedback a promising technique to obtain behavioral improvements. In this work, we present the first open-source brain-computer interface system that can be reliably used to provide speech entrainment neurofeedback while still being flexible enough to deliver more traditional coherence-based neurofeedback. In addition, it has the potential of being an open-source alternative to deliver other types of neurofeedback if configured to do so.

Keywords: Brain-computer interface, Neurofeedback, Speech Entrainment, Dyslexia

Introduction

What is neurofeedback?

Electroencephalography-based neurofeedback (EEG-NF) represents a commonly used technique that involves the real-time processing and measurement of EEG signals while providing feedback to the same person simultaneously (Omejc, Rojc, Battaglini & Marusic, 2018). Through such a feedback loop, that person may attain better control over a neurophysiological parameter of interest by inducing changes in its brain functioning, which may, in turn, affect a given behavior (Omejc et al., 2018).

The origin of EEG-NF goes back to the 1960s where it initially arose a lot of interest due to its potential as a revolutionary clinical intervention (Kamiya, 2011). The technology experienced a decline in use afterwards since it did not meet expectations. However, as technology improved, it saw increasing use again. Today, it is now implemented in a wide array of private clinics over the world (Omejc et al., 2018).

The main function of EEG-NF is to learn how to self-regulate an electrophysiological marker as much as possible. Persons are taught how to increase or inhibit these electrophysiological markers through operant conditioning, which is the process where behavior prevalence is altered through immediate feedback and reinforcement. This premise is rooted on a causal hypothesis stipulating that the deviations in brain function cause the underlying behavioral symptoms leading to the disorders themselves (Sherlin et al., 2011; Sitaram et al., 2016; Enriquez-Geppert, Huster & Herrmann, 2017).

The whole process of EEG-NF works as a feedback loop through a brain-computer interface (BCI), which begins with the acquisition of EEG data from a subject (Figure 1). Immediately after, the EEG signal can be analyzed either offline or in real-time to gather an electrophysiological marker of interest. This marker is then presented back at the subject in visual, tactile, or even auditory form. Traditional examples include video games controlled by

the desired brain activity or bar plots displaying the brain activity itself along with a baseline that the subject must try to achieve/improve (Gruzelier, 2014; Enriquez-Geppert et al., 2017).

Figure 1: Traditional Neurofeedback BCI Diagram. The Acquisition Computer processes and gathers the EEG data stream. The buffer receives the data from the Acquisition Computer and makes it available to the Calculation Computer. Finally, the Calculation Computer performs the real-time EEG data analyses and presents the metrics back to the participant.

Currently, three major approaches to EEG-NF exist. The one that is by far the most commonly used focuses on frequency training. The aim of this approach is to modulate the power ratio of EEG on-going brain activity, which have been traditionally separated into frequency bands: delta (< 4 Hz), theta (4-7 Hz), alpha (8-13), beta (14-30 Hz) and gamma (> 30 Hz) (Noachtar et al., 1999). The logic behind this kind of neurofeedback modulation is the frequency-to-function mapping. That is, the proposed relationship between the power of certain frequency bands and their associated cognitive functions. The most common frequency training used today is the modulation of the theta/beta ratio that is used for treating attention deficit hyperactivity disorder (ADHD) (Leins et al., 2007).

Alongside frequency training, there are other approaches to EEG-NF. These are primarily the training of slow cortical potentials (SCPs) and coherence-based neurofeedback. SCPs aim to modulate concrete event-related potentials that can be either positive or negative (Birbaumer, 1999). They have been used for patients with epilepsy to increase their seizure threshold levels as well as an intervention for ADHD to improve attentional skills (Birbaumer, 1999; Mayer, Wyckoff & Strehl, 2012). Coherence-based neurofeedback aims to change the connectivity patterns between two or more brain regions defined by a given EEG channel layout (Decker, Fillmore & Roberts, 2017). Distorted connectivity has been shown to appear in several neurologic conditions such as epilepsy, autism spectrum disorder, and traumatic brain injuries, thus leading to the use of this EEG-NF modality in clinical practice (Walker & Kozlowski, 2005; Coben, Wright, Decker & Morgan, 2015; Rostami et al., 2017).

Despite the wide array of clinical applications that EEG-NF modalities possess, there are still gaps not addressed by any of them. One of these, is the fact that the software implementations of most EEG-NF systems are usually not open-source or flexible enough to accommodate more than one EEG-NF modality. They also do not tend to control the temporal delay between neurofeedback presentation with a sufficient level of detail, which is crucial since increased latency significantly affects the sense of agency during BCI control (Evans, Gale, Schurger, Blanke, 2015). More importantly for our purposes however, is that there are no neurofeedback modalities that currently exist to directly modulate speech-brain entrainment.

A Primer in Speech-Brain Entrainment

Speech-brain entrainment stands for the alignment of neural activity to the slow temporal fluctuations (envelope) of acoustic speech input. The speech signal includes slow temporal fluctuations within the 0.5-10 Hz band that are closely related to phrase and syllable features in the acoustic signal. Tracking such temporal structures, both at phrasal and syllabic

rates, is crucial for speech comprehension (Greenberg et al., 2003; Poeppel, 2003; Poeppel, Idsardi, & van Wassenhove, 2008). The phase of low-frequency delta (< 4 Hz) and theta (4 – 7 Hz) oscillations in the auditory cortex synchronizes to the phrasal and syllabic patterns of speech, respectively (Figure 2; Bourguignon et al., 2013, 2020; Molinaro & Lizarazu, 2018; Lizarazu, Lallier & Molinaro, 2019).

Figure 2: Cortical Tracking of Speech. Representation of the observed synchrony between delta and theta brain oscillations and the phrasal and syllabic patterns of speech respectively (for example: “todas las tardes mama me lleva al parque”/ “every afternoon mom takes me to the park”). On top, we have the original speech wave, followed by its spectrogram containing the envelope shown in yellow. We showcase how theta band oscillations synchronize to syllables and words as well how delta band oscillations entrain to prosodic information.

Speech-brain entrainment reflects to a great extent the bottom-up analysis of the speech signal. Yet, these mechanisms can also reflect top-down modulations of sound processing by using attention, linguistic knowledge, and expectation. Concretely, Park et al. (2015) found that the left inferior frontal and precentral gyri along with posterior temporal areas in the right hemisphere modulate the phase of low-frequency brain oscillations within the left auditory cortex depending on speech intelligibility. Moreover, they noted that speech-brain entrainment is improved as a function of stronger top-down modulation. Combined,

these results ultimately suggest that theta and delta brain oscillations are crucial in exerting top-down control during speech processing. These findings are further supported by a plethora of other studies that have also suggested that neural entrainment to speech correlates with the intelligibility of the speech signal (Ahissar et al., 2001; Ding & Simon, 2013; Doelling, Arnal, Ghitza & Poeppel, 2014; Gross et al., 2013; Peelle, Gross, & Davis, 2013; Pérez et al., 2015; Rimmele, Golumbic, Schröger, & Poeppel, 2015; Ghinst et al., 2016; Molinaro & Lizarazu, 2018; Lizarazu et al., 2021; Molinaro et al., 2021). For instance, in Ghinst et al (2016), participants were asked to pay attention to a speech stream embedded within a “cocktail party” multitalker background noise at different intensity levels. They found that delta and theta neural entrainment was stronger for the attended speech compared to the multitalker background. In addition, they showed that neural entrainment to the attended speech decreased as the level the multitalker background noise progressively increased.

Clinical Neurofeedback Applications of Speech-Brain Entrainment

Atypical entrainment to speech has been shown to play a role on various neuropsychological disorders such as dyslexia, stroke, and Broca’s aphasia (Goswami, 2011; Molinaro et al., 2016; Feenaughty et al., 2017; Liberto et al., 2018; Thors, 2019; Lizarazu et al., 2021b,c). For example, in terms of developmental dyslexia, Goswami (2011) theorized that atypical speech-brain entrainment could be part of the underlying cause. She stipulated that impairments in the temporal sampling of speech by brain oscillations could explain the phonological and perceptual difficulties that characterize individuals with dyslexia.

In this vein, numerous functional brain studies have shown atypical neural entrainment to speech in dyslexia (Hämäläinen, Salminen, & Leppänen, 2013; Lizarazu et al., 2015; Cutini et al., 2016; Molinaro et al., 2016; see Jiménez-Bravo, Marrero, & Benítez-Burraco, 2017 for a recent review). Molinaro and colleagues (2016) found that dyslexic

readers had impaired neural entrainment in the lower delta band range localized in both the left inferior frontal gyrus and right auditory cortex. Moreover, they found impaired functional connectivity between oscillations in the left frontal cortex and the right auditory cortex. Lizarazu and colleagues (2015) further reported atypical neural entrainment to non-linguistic auditory signals (amplitude modulated white-noise) in the theta band in dyslexics. Together, these results suggested that individuals with developmental dyslexia display reduced neural entrainment to low-frequency speech oscillations compared to controls while listening to speech.

This could be a clear example where speech entrainment neurofeedback could have a huge impact by training the ability of individuals to better synchronize to a given speech signal. Specifically, neurofeedback may be able to enhance the top-down modulation of speech-brain entrainment in auditory regions (Park et al., 2015). Therefore, to that end, we have developed a first of its kind open-source BCI toolbox in Matlab that is capable of providing real-time speech entrainment neurofeedback while participants listen to an audio signal. This required a very precise design capable of aligning EEG data to the exact segments of audio that the participant is listening to while performing all the neurofeedback calculations in real-time. Furthermore, we built the toolbox such that it is flexible enough to accommodate more traditional coherence-based neurofeedback with respect to any given EEG channel, as well as to provide other types of neurofeedback if configured to do so.

In the following sections, we will describe how this toolbox has been implemented and how it operates in detail under the simulated data framework we used to test it. Moreover, we also propose a speech-in-noise experiment to test its implementation using real data, which is crucial for any preliminary tests on clinical populations.

Methods

Overall, we conceptualized a complete neurofeedback session using our toolbox as a two-step process. The first step involves an offline localizer experiment that aims to find the EEG sensors showing the highest levels of speech entrainment. The second step is the neurofeedback session itself, which consists of continuously presenting the strength of the neural entrainment to speech. The objective is to maximize the neural entrainment to speech across the neurofeedback session. In the following sections, we will go into the detail of each of the steps showing how the toolbox works internally, as well as to provide a guide on how to conduct successful neurofeedback sessions. All the code developed for this toolbox can be seen in Appendix A as well as in the following URL to Matlab File Exchange: https://es.mathworks.com/matlabcentral/fileexchange/78170-real-time-speech-brain-entrainment-neurofeedback-toolbox?s_tid=LandingPageTabfx

EEG Data Acquisition

This toolbox has been designed to work with BrainAmp amplifiers and BrainVision Recorder software (BrainProducts, Germany). Based on our testing setup, we recommend using EEG systems with 32 electrodes positioned according to the international 10-20 system (Jasper, 1958). Moreover, we recommend placing the reference channels in the earlobes and keeping scalp-electrode impedance below 8 k Ω to ensure high-quality EEG recordings. The sampling rate of the EEG system should be at or above 500 Hz.

Part 1: Offline Localizer Session

The localizer part of the neurofeedback session behaves very much like a classical offline EEG experiment. The objective, as mentioned previously, is to locate the EEG sensors that display the highest level of neural entrainment with the speech envelope for a given frequency band. This step is optional, but highly recommended since it is customary for EEG-NF applications to focus on a few channels of interest (Omejc et al., 2018). By default, the frequency range of interest is between 0.5-7 Hz, which corresponds to the delta and theta

EEG bands. This default was chosen given the extensive literature highlighting their importance in speech-brain entrainment (Bourguignon et al., 2013; Molinaro et al., 2016; Molinaro and Lizarazu, 2018; Lizarazu, Lallier and Molinaro, 2019).

To run the localizer session, we developed the function *speech_brain_coherence.m*, which is built on top of Fieldtrip (Oostenveld, Fries, Maris & Schoffelen, 2011), and it is able to perform all the necessary steps. These include:

1. Preprocessing and trial segmentation of the offline EEG experiment data.
2. Calculation of the degree of neural entrainment based on the coherence between each EEG channel and the envelope of the speech presented across trials.
3. Gathering the n EEG channels with the highest average coherence between the range of frequencies of interest defined by the user.
4. Optionally, as an additional step, the user can choose to plot the coherence across the range of frequencies of interest. An example of this can be seen in Figure 3.

Figure 3: Localizer session. In this case, the range of frequencies of interest is from 0.5-7 Hz, which correspond to the delta and theta EEG bands. Results are shown using a 32-channel EasycapM7 layout. The coherence shown represents that obtained from calculating the coherence between simulated EEG data and the speech envelope.

The current implementation of the toolbox only segments the trials based on the trigger values and applies a FIR band-pass filter to the data using the frequency band of interest given by the user. Ideally, this frequency band will correspond to the frequencies tracked during the neurofeedback session. The *Trial_Segment.m*, function, which is in charge of this step, is very flexible and can easily be modified to accommodate more preprocessing steps like re-referencing as well as electrooculogram or motion artifact detection if desired.

Coherence

For this session, we used coherence to quantify the degree of neural entrainment to speech. Coherence measures the phase synchronization between two signals (i.e. speech

envelope and EEG data) in the frequency domain. This measure was chosen for the localizer since it provides a fine-grained view of entrainment at each individual frequency within the band of interest. This can be useful to pick sensors of interest as well as for standalone offline data analyses. The definition of coherence between two signals $x(t)$ and $y(t)$ can be seen in Equation 1:

$$Coherence_{xy}(f) = \frac{|G_{xy}(f)|^2}{G_{xx}(f)G_{yy}(f)}$$

Where f stands for frequency in Hz, $G_{xy}(f)$ stands for the cross-spectral density between the signal of the EEG channel and the speech envelope, $G_{xx}(f)$ and $G_{yy}(f)$ correspond to the spectral density of the EEG channels and the speech envelope with themselves, respectively (Bruña, Maestú & Pereda, 2018). For obtaining the speech envelope, we obtained the magnitude of the analytic signal of the speech after a Hilbert transformation. Following that, we downsampled the signal to the sampling frequency of the given EEG system, which in our case was 500 Hz. Coherence values range from 0 to 1 where 0 indicates no phase synchrony and 1 indicates perfect phase synchrony.

It is important to highlight that the user can select the n number of channels to keep for use during the neurofeedback session. As a guideline, we suggest selecting 3-5 sensors of interest at the most. It is likely that only a few would be needed given the custom in neurofeedback applications (Omejc et al., 2018). As an additional optional feature, the user can also generate interactive plots of the coherence spectrum for each channel (Figure 3). This allows for a thorough study of a participant's speech entrainment prior to any neurofeedback session if desired. Thus, even though the *speech_brain_coherence.m* function has been designed primarily with speech-brain entrainment neurofeedback in mind, it can also work as a reusable standalone function for offline experiments.

Part 2: Real-Time Speech Entrainment Neurofeedback Session

The bulk of the work conducted has been related to the implementation of a brain-computer interface (BCI) system that could make reliably speech entrainment neurofeedback with minimal delay. To that end, we developed a series of functions built on top of several toolboxes. These being Fieldtrip (Oostenveld et al., 2011), Psychtoolbox-3 (Brainard, 1997; Pelli, 1997; Kleiner, Brainard & Pelli, 2007) and a slightly modified version of the TCP/UDP/IP toolbox for Matlab (Rydesäter, 2019).

BCI Data Pipeline Setup

For running a successful neurofeedback session, the BCI data pipeline depicted in Figure 4 must be in place. The envisioned data pipeline begins with a real-time EEG data stream from a BrainAmp amplifier, which is recorded in the acquisition computer running BrainVision Recorder (BrainProducts, Germany). Prior to the neurofeedback session, the remote data access (RDA) server within BrainVision recorder's preferences must be activated (Figure 5). This allows the EEG data to be available through a TCP/IP connection to the data processing and presentation computer.

Figure 4: BCI setup in the present study. BV stands for BrainVision, RDA stands for remote data access, TCP/IP stands for transmission control protocol and internet protocol. GUI stands for graphical user interface.

Figure 5: Location of the RDA activation checkbox with BrainVision Recorder's Preferences.

The acquisition computer will also house the Matlab script contained within this toolbox to send the trigger pulses (*Trigger_Gen_Multi.m*). This will allow the data processing and presentation computer to synchronize the neurofeedback with the speech/audio stimuli. The current toolbox implementation uses a mex-file parallel port interface to send the triggers to the EEG system (Schieber, 2020). However, this can be a researcher's choice and other options could be used instead. The only requirement is that the software is able to send TTL pulses to an EEG system.

Moving on from the acquisition computer, we have the utilities that are housed within the data processing and presentation computer. The key utility that makes the BCI data pipeline implemented in this toolbox possible is Fieldtrip's data buffer. The buffer houses the data streamed from the RDA server in BrainVision Recorder. To collect the data from the RDA server, we used a compiled C utility that is provided by Fieldtrip called *rda2ft* (Oostenveld et al., 2011). In order to activate *rda2ft*, the IP of the acquisition computer is needed. Thus, it is key that the data processing computer and acquisition computer are in the same network. The port of the RDA server is also required, which for a x64 bit architecture

computer is port 51244. By default, once given those parameters, *rda2ft* instantiates the Fieldtrip buffer on the local data processing and presentation computer on port 1970. Being a C-built utility, *rda2ft* is extremely fast, and it is capable of collecting data from the RDA server every 2 ms. This allows the stimulus presentation and subsequent neurofeedback processes to begin immediately after the trigger is sent within the acquisition computer.

Phase-Locking Value

The aim of the neurofeedback session is to continuously present the strength of the neural entrainment to the speech, which during the neurofeedback session is quantified using phase-locking value (PLV). Specifically, we designed the toolbox to compute the average PLV between brain oscillations at the delta (0.5-4 Hz) and theta (4-7 Hz) frequency bands and the speech envelope.

Coherence was preferred for the localizer session since it provides a more detailed view of entrainment within each individual frequency inside a band. This can give a researcher more control in terms of selecting the EEG sensors of interest and standalone value as a reusable function for offline experiments. However, for the neurofeedback session we chose PLV because of the opposite. As implemented, it outputs a single value for an entire frequency band, which is desirable for the delta/theta average calculation we use for neurofeedback. It is also a slightly leaner calculation, which helps in terms of computation speed (Bressan et al. 2018). Since the neurofeedback session calculates the PLV in real-time, it was defined as shown in Equation 2 for the purposes of our toolbox:

$$PLV = \frac{1}{T} \left| \sum_{t=1}^T e^{-i(\varphi_i(t) - \varphi_j(t))} \right|$$

Where T is the data length (number of samples), $\varphi_i(t)$ is the instantaneous phase of the signal of each EEG channel at time t , and $\varphi_j(t)$ is the instantaneous phase of the speech envelope (default) or any reference channel of choice at time t . The instantaneous phases of

each of the signals and the speech envelope were obtained from the phase angle of the analytic signals after a Hilbert transformation and a FIR band-pass filter.

PLV is defined as the absolute value of the mean phase difference between two signals, and has values ranging from 0 indicating no phase alignment to 1 indicating complete phase alignment (Lachaux, Rodriguez, Martinerie & Varela, 1999; Bruña et al., 2018). The goal of a participant during the neurofeedback session, by default, will be to maximize as much as possible the theta/delta PLV average across the channels of interest gathered from the localizer over time.

Experimental Design and Neurofeedback BCI Loop

Once the data is being collected into the buffer through *rda2ft*, the heavy lifting of presenting the neurofeedback along with the speech/audio stimuli falls to the *ft_realtime_plv_fully_sync.m* function. As a general overview, *ft_realtime_plv_fully_sync.m* constantly checks the buffer for events (triggers) of interest selected by the researcher. While no triggers of interest are detected, a white fixation cross is displayed on the screen. During this time, participants can be invited to blink to reduce artifacts during the neurofeedback presentation since these data are not analyzed. Once a trigger of interest is detected, it proceeds to play a piece of audio that is randomly selected among the ones contained within the stimuli folder associated with the trigger value (see the documentation for *Audio_Processor.m* for further details into how stimuli folders must be named). The real-time neurofeedback itself will begin once there is enough EEG data in the buffer associated with the presentation of the speech/audio. It is important to note that the determination of whether there is enough EEG data is controlled by the chosen presentation frequency window. This is a key user-defined parameter that defines how often the neurofeedback should be presented.

Given that the speech/audio is an external signal to the EEG, we were extremely careful to align each segment of real-time EEG data to the exact segment of the speech/audio

that an individual listened to during data collection. As such, once an event of interest is detected, the *ft_realtime_plv_fully_sync.m* function calls *ft_trialfun_speechwindow.m* (a custom trial definition function) to define the boundaries of the neurofeedback presentation windows. We use the EEG data stream samples from the buffer to mark the beginning and the end of these presentation windows. Likewise, these boundaries are used to gather the exact segment of the speech/audio envelope that corresponds to what the individual listened to during that time in order to perform accurate PLV calculations.

It is crucial to note that the *ft_trialfun_speechwindow.m* function will only create windows that are exact divisions with respect to the audio length. That is, if we have an EEG system with a sampling rate of 500 Hz, speech stimuli lasting 10s, and a window size of 2s, the *ft_trialfun_speechwindow.m* will create 5 trials. All of these would contain 1000 samples and will cover the entirety of the audio envelope. However, if we have speech stimuli lasting 11s, the *ft_trialfun_speechwindow.m* will also create 5 trials containing 1000 samples each since 10 is the nearest exact division of 2. This is done to avoid confounding effects of window length by guaranteeing that the presentation windows are always consistent.

During the presentation of the neurofeedback, it is important to emphasize the global moving average baseline. This baseline is plotted as a horizontal line in the bar chart that is used as the default neurofeedback visualization method. We included this baseline in order to provide a clear improvement objective to the individual receiving the neurofeedback. The default instruction is to try to move that line as closer to 1 (perfect phase alignment) as possible.

Once the last presentation window is reached, the neurofeedback will remain in the screen for the duration of another presentation window. This way, the participant receiving the neurofeedback will have enough time to process the information presented in the last window. After this, the *ft_realtime_plv_fully_sync.m* will allow researchers to evaluate

participant behavior. By default, the toolbox comes with an intelligibility question in which the participant has to evaluate in a scale from 1-9 how intelligible they found a given speech/audio stimulus. However, other options are possible such as yes/no comprehension questions. In any case, the function waits for a keyboard key press as a response before returning to the white fixation cross. This fixation cross will remain in the screen until the next trigger of interest is detected and the cycle is repeated again.

The BCI loop will continue until all the stimuli have been presented. Once that point is reached, the BCI loop ends and the function returns the results of the neurofeedback session. These results are stored in a Matlab cell array format with as many rows as stimuli and columns containing the condition based on the trigger value, the answer to any behavioral questions, and the average theta/delta PLV at each presentation window. A visual representation of the BCI loop explained above can be seen in Figure 6.

Figure 6: BCI presentation diagram. The (pre) trigger pauses stand for the time in between triggers, which is a parameter controlled by *Trigger_Gen_Multi.m* in the acquisition computer. This pause serves to control the presentation of stimuli and provide time for participants to blink. The presentation windows, as mentioned, are how often the neurofeedback presentation is desired (i.e every 2s); the window pause is also the same amount. Lastly, behavioral questions (for example, a comprehension question) are controlled by key presses. As soon as the participant presses a key, the system goes back to fixation.

Neurofeedback Computational Performance

Our testing was run on an Alienware Area 51m laptop as the data processing and presentation computer. This computer contained an Ubuntu 20.04 operating system, a 9th generation Intel i9-9900k CPU, a Nvidia RTX 2060 GPU, 32 GB of RAM and a 17.3 inch screen with a 144 Hz refresh rate. Under these specifications, we obtained an overall average delay of 218.9 ms for all the signal processing and PLV calculations, which is a good real-time performance for a Matlab toolbox.

We obtained this value by testing across 3 different presentation windows, which we hypothesize to be the most likely to be used, and 5 channels of interest from the localizer. We do not expect that a researcher would use that many channels given that neurofeedback applications tend to focus on less (Omejc et al., 2018), but it allowed us to see how the toolbox would perform under harder circumstances. Figure 7 shows the details of the calculation timings for each of the presentation windows tested.

Figure 7: Computation Delays. Average computation delays in milliseconds by presentation window length. As can be expected, the longer the presentation window, the longer the average delay. Error bars are standard deviation representing the variability in computation times. These can be caused due to ongoing background processes within an operating system and Matlab's optimizations after the first time a given piece of code is executed.

Proposed Validation Experiment

Due to technical difficulties that arose during the time this work was produced (closure of the labs to prevent the spread of the virus COVID-19), we could only test and validate the operation of our neurofeedback toolbox using simulated EEG data. However, in order to better position speech-brain entrainment neurofeedback as a strong candidate for clinical application, a formal experiment with real data is needed.

In order to achieve this and provide a stronger rationale for using our neurofeedback approach with clinical populations in the future, we first propose a speech-in-noise experiment with healthy participants. This study would contain two groups. A control group, which would receive random neurofeedback, and an experimental group that would receive the true average theta/delta PLV neurofeedback as they listen to speech. Given the sample sizes of prior entrainment experiments, each group would contain 10 participants (Ghinst et al., 2016). Thus, resulting in a total sample size of 20 healthy subjects.

The experiment would consist of 4 speech-in-noise conditions that are randomly presented during the neurofeedback session. Clean speech would be presented during the localizer session. The stimuli would be 20 seconds long with neurofeedback presented in windows of 2s. After this presentation, participants would be asked to rate the intelligibility of the speech as well as comprehension questions to gather behavioral measures. It is important to note that individuals would be asked to attend to the voice belonging to the person whose clean speech was listened during the offline localizer session while in the neurofeedback session (Ghinst et al., 2016). This would be done in order to provide a clear attention target for top-down entrainment modulation. The noise would be a cocktail party noise of 6 people (3 females). A full diagram of the conditions that could be used for this experiment can be found in Figure 8.

Figure 8: Proposed Speech-in-noise conditions for neurofeedback validation in healthy participants. Adapted from Ghinst et al., 2016.

If neurofeedback can indeed modulate top-down speech-brain entrainment in our proposed experiment, we would expect results similar to those shown in Figure 9. That is, we would expect to observe strong positive correlations at the individual level between the mean delta/theta PLV and the intelligibility scores of each stimulus presented during mild noise conditions on participants receiving true neurofeedback. We also hypothesize that the participants in the control group will display weak correlations between their intelligibility scores and the mean delta/theta PLV across presentation windows. Furthermore, we expect neurofeedback to decrease in effectiveness as background noise increases given the decrease in entrainment observed in past studies (Ghinst et al., 2016).

Figure 9: Hypothesized effect of speech-entrainment neurofeedback on stimulus intelligibility over time in a single individual during a mild noise condition such as +5 dB.

Discussion

The BCI system presented as part of this work represents one of the firsts of its kind to be designed for real-time speech-brain entrainment neurofeedback. Even though it was not possible to conduct the proposed experiment at the time this is being written, we have tested that our toolbox is capable of providing accurate and reliable real-time speech entrainment neurofeedback with simulated data. Therefore, it is ready to be used in actual experimental settings.

Our toolbox has minimal delays in all aspects. From the 2 ms delay in data collection into the buffer through *rda2ft*, to the average 218.9 ms in terms of PLV calculations. This maximizes the sense of agency for the participant and, consequently, BCI control as described by Evans et al., (2015). This is essential in order to develop strategies for top-down modulation of entrainment within auditory regions (Park et al., 2015). Moreover, our toolbox allows for the possibility of compiling it into a standalone application for users with a limited programming background and to further reduce the computation delays.

Nevertheless, the most remarkable feature of the toolbox is that it is built to be flexible and to allow the user to select various hyperparameters such the neurofeedback presentation windows or the frequency bands of interest depending on the experimental question. For example, in terms of the frequency bands of interest, the default theta/delta PLV average metric was picked to cover the spectrum of past studies highlighting the role of theta and delta brain oscillations in speech-brain entrainment (Bourguignon et al., 2013; Molinaro & Lizarazu, 2018; Lizarazu, Lallier & Molinaro, 2019). We believe that this default may be an appropriate metric to begin testing our toolbox in settings such as our proposed speech-in-noise validation experiment. Moreover, it could be used to test the effects of our neurofeedback technique on disorders like dyslexia given the atypical neural entrainment found in studies such as Lizarazu et al. (2015) and Molinaro et al. (2016). However, this

default could be too broad and fail to capture specific entrainment features of either the delta or theta band for a given research question. In this vein, a researcher may choose to modify this default within the *RT_Entrainment_Toolbox_Main.m* script and elect to track just entrainment to the delta band by setting the frequency inputs to just the delta band instead of theta and delta for instance. In addition, for posterior clinical applications a user may even choose to track an entirely different frequency band such as the gamma band, which has been related to phoneme entrainment. This is relevant since phonemic oversampling has also been proposed as a potential cause of the phonological processing impairment characteristic of dyslexia (Giraud & Poeppel, 2012).

This level of flexibility within our novel neurofeedback technique thus opens up the possibility of several experimental paradigms that could become effective therapies for disorders such as the beforementioned dyslexia. Already, EEG-NF in general has already been shown to be successful in treating neuropsychological disorders ranging from ADHD (Sonuga-Barke et al., 2013) and epilepsy (Tan et al., 2009), to traumatic brain injury (Garzon, 2018) and autism spectrum disorder (Holtmann et al., 2011). Even in the case that speech-brain entrainment neurofeedback would be shown to be ineffective experimentally, the way we have built this toolbox also allows researchers and clinicians to swap the default audio comparison reference with an EEG channel of interest. Thus, making this toolbox relevant in all cases as it could apply traditional coherence-based neurofeedback, which has been already shown to be effective for epilepsy, autism spectrum disorder (ASD), and traumatic brain injury in previous studies (Coben, Wright, Decker & Morgan, 2015; Walker & Kozlowski, 2005; Rostami et al., 2017).

With that being said, this toolbox has its limitations. Some could be addressed with further development effort. Examples of these include the fact that it only supports BrainProducts technologies (BrainAmp amplifiers and BrainVision Recorder) along with the

need for a very powerful hardware setup with a Linux operating system for the data processing and presentation computer. Moreover, the real-time data preprocessing currently implemented is limited. We rely on FIR band-pass filtering to attenuate high-frequency artifacts and trigger presentation pauses to control for actions such as blinking when there are no stimuli being presented. Real-time applications are harder to preprocess than an offline experiment without a major loss in latency. However, there are techniques that could be implemented like artifact subspace reconstruction (ASR) that would allow for thorough and efficient artifact rejection in real-time (Chang, Hsu, Pion-Tonachini & Jung, 2018).

Lastly, there is an inherent limitation with all neurofeedback applications, which is their underlying controversy regarding their reliability as clinical tools. There are study reviews that claim that the findings showing that neurofeedback applications are effective have several limitations that range from small sample sizes to the lack of study randomization (Micoulaud-Franchi, 2015; Schoenberg & David, 2014). However, the biggest limitation is the unclear long-term effects of neurofeedback therapies. In this regard, previous studies have found mixed results. For instance, Janssen et al. (2020) did not find any specific long-term benefits of theta/beta neurofeedback in the treatment of ADHD compared to pharmacological and physical treatment. On the other hand, Kouijzer et al. (2009) found significant long-term effects of theta/beta modulation neurofeedback in ASD patients. This level of variability makes it hard to predict whether speech-brain entrainment neurofeedback will have robust long-term effects. However, given how atypical entrainment appears to be present in disorders such as dyslexia as shown by previous studies (Hämäläinen, Salminen, & Leppänen, 2013; Lizarazu et al., 2015; Molinaro et al., 2016), it is likely that speech-brain entrainment can lead to reliable behavioral improvements. In any case, the long-term effects of speech entrainment neurofeedback along with the other limitations are certainly aspects

that will be carefully looked at, and controlled for, in future studies looking to further evaluate our toolbox for clinical applications.

Conclusion

Overall, despite the testing limitations, we have implemented an open-source BCI toolbox that is a robust piece of software capable of delivering real-time speech-brain entrainment neurofeedback with minimal delays in all fronts. There are well-founded reasons to believe that speech-brain entrainment neurofeedback will be effective given the relevance of neural entrainment to speech in disorders such as developmental dyslexia, stroke, and Broca's aphasia. Thus, there is a great potential for this toolbox to become the core of a cutting-edge clinical intervention.

References

- Ahissar, E., Nagarajan, S., Ahissar, M., Protopapas, A., Mahncke, H., & Merzenich, M. M. (2001). Speech comprehension is correlated with temporal response patterns recorded from auditory cortex. *Proceedings of the National Academy of Sciences*, 98(23), 13367-13372.
- Birbaumer, N. (1999). Slow Cortical Potentials: Plasticity, Operant Control, and Behavioral Effects. *The Neuroscientist*, 5(2), 74–78.
- Bourguignon M, De Tiege X, de Beeck MO, Ligot N, Paquier P, Van Bogaert P, Goldman S, Hari R, Jousmäki V. (2013). The pace of prosodic phrasing couples the listener's cortex to the reader's voice. *Human Brain Mapping*, 34(2), 314-326.
- Bourguignon, M., Molinaro, N., Lizarazu, M., Taulu, S., Jousmäki, V., Lallier, M., ... & De Tiège, X. (2020). Neocortical activity tracks the hierarchical linguistic structures of self-produced speech during reading aloud. *NeuroImage*, 216, 116788.
- Brainard, D. H. (1997) The Psychophysics Toolbox, *Spatial Vision* 10:433-436.
- Bruña, R., Maestú, F., & Pereda, E. (2018). Phase locking value revisited: teaching new tricks to an old dog. *Journal of Neural Engineering*, 15(5), 056011.
- Chang, C.-Y., Hsu, S.-H., Pion-Tonachini, L., & Jung, T.-P. (2018). Evaluation of Artifact Subspace Reconstruction for Automatic EEG Artifact Removal. *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*.
- Coben, R., Wright, E. K., Decker, S., & Morgan, T. (2015). Impact of Coherence Neurofeedback on Reading Delays in Learning Disabled Children: A Randomized Controlled Study. *NeuroRegulation*, 2(4), 168–178.

- Cutini, S., Szűcs, D., Mead, N., Huss, M., & Goswami, U. (2016). Atypical right hemisphere response to slow temporal modulations in children with developmental dyslexia. *NeuroImage*, 143, 40–49.
- Decker, S. L., Fillmore, P. T., & Roberts, A. M. (2017). Coherence: The Measurement and Application of Brain Connectivity. *NeuroRegulation*, 4(1), 3–13.
- Ding, N., & Simon, J. Z. (2013). Adaptive temporal encoding leads to a background-insensitive cortical representation of speech. *Journal of Neuroscience*, 33(13), 5728–5735.
- Doelling, K. B., Arnal, L. H., Ghitza, O., & Poeppel, D. (2014). Acoustic landmarks drive delta–theta oscillations to enable speech comprehension by facilitating perceptual parsing. *Neuroimage*, 85, 761–768.
- Enriquez-Geppert, S., Huster, R. J., & Herrmann, C. S. (2017). EEG-Neurofeedback as a Tool to Modulate Cognition and Behavior: A Review Tutorial. *Frontiers in Human Neuroscience*, 11.
- Evans N, Gale S, Schurger A, Blanke O. (2015) Visual Feedback Dominates the Sense of Agency for Brain-Machine Actions. *PLOS ONE*, 10(6).
- Feenaughty, L., Basilakos, A., Bonilha, L., Ouden, D.-B. D., Rorden, C., Stark, B., & Fridriksson, J. (2017). Non-fluent speech following stroke is caused by impaired efference copy. *Cognitive Neuropsychology*, 34(6), 333–346.
- Garzón, M. E. R. (2018). Neurofeedback Training to Increase of Cognitive Skills in Patient with Traumatic Brain Injury (TBI). *Journal of Neurology & Stroke*, 8(1).
- Ghinst.V., Bourguignon, M., de Beeck, M. O., Wens, V., Marty, B., Hassid, S., ... & Goldman, S. (2016). Left superior temporal gyrus is coupled to attended speech in a cocktail-party auditory scene. *Journal of Neuroscience*, 36(5), 1596–1606.

- Giraud A. L., Poeppel D. (2012). Speech perception from a neurophysiological perspective, in *Human Auditory Cortex*, ed Poeppel D., Overath T., Popper A. N., Fay R. R. (New York, NY: Springer;), 225–259.
- Goswami U. (2011). A temporal sampling framework for developmental dyslexia. *Trends in cognitive sciences*, 15(1), 3–10.
- Greenberg S, Carvey H, Hitchcock L, Chang S. (2003). Temporal properties of spontaneous speech—a syllable-centric perspective. *Journal of Phonetics*, 31(3-4), 465-485.
- Gross, J., Hoogenboom, N., Thut, G., Schyns, P., Panzeri, S., Belin, P., & Garrod, S. (2013). Speech rhythms and multiplexed oscillatory sensory coding in the human brain. *PLoS biology*, 11(12).
- Gruzelier, J. H. (2014). EEG-neurofeedback for optimising performance. III: A review of methodological and theoretical considerations. *Neuroscience & Biobehavioral Reviews*, 44, 159–182.
- Hämäläinen, J. A., Salminen, H. K., & Leppänen, P. H. (2013). Basic auditory processing deficits in dyslexia: systematic review of the behavioral and event-related potential/field evidence. *Journal of learning disabilities*, 46(5), 413–427.
- Holtmann, M., Steiner, S., Hohmann, S., Poustka, L., Banaschewski, T., & Bölte, S. (2011). Neurofeedback in autism spectrum disorders. *Developmental medicine and child neurology*, 53(11), 986–993.
- Janssen, T., Geladé, K., Bink, M., Mourik, R. V., Twisk, J., Maras, A., & Oosterlaan, J. (2020). Long-term effects of theta/beta neurofeedback on EEG power spectra in children with attention deficit hyperactivity disorder. *Clinical Neurophysiology*, 131(6), 1332-1341.
- Jasper, H.H. (1958) The Ten-Twenty Electrode System of the International Federation. *Electroencephalography and Clinical Neurophysiology*, 10, 371-375.

- Jiménez-Bravo, M., Marrero, V., & Benítez-Burraco, A. (2017). An oscillopathic approach to developmental dyslexia: From genes to speech processing. *Behavioural brain research*, 329, 84–95.
- Kamiya, J. (2011). The First Communications About Operant Conditioning of the EEG. *Journal of Neurotherapy*, 15(1), 65–73.
- Kleiner M, Brainard D, Pelli D. (2007). What is new in Psychtoolbox-3? *Perception*, 36. ECVF Abstract Supplement.
- Kouijzer, M. E., Moor, J. M., Gerrits, B. J., Buitelaar, J. K., & Schie, H. T. (2009). Long-term effects of neurofeedback treatment in autism. *Research in Autism Spectrum Disorders*, 3(2), 496-501.
- Lachaux, J. P., Rodriguez, E., Martinerie, J., & Varela, F. J. (1999). Measuring phase synchrony in brain signals. *Human brain mapping*, 8(4), 194–208.
- Leins, U., Goth, G., Hinterberger, T., Klinger, C., Rumpf, N., & Strehl, U. (2007). Neurofeedback for Children with ADHD: A Comparison of SCP and Theta/Beta Protocols. *Applied Psychophysiology and Biofeedback*, 32(2), 73–88.
- Liberto, G. M. D., Peter, V., Kalashnikova, M., Goswami, U., Burnham, D., & Lalor, E. C. (2018). Atypical cortical entrainment to speech in the right hemisphere underpins phonemic deficits in dyslexia. *NeuroImage*, 175, 70–79.
- Lizarazu, M., Lallier, M., Molinaro, N., Bourguignon, M., Paz-Alonso, P. M., Lerma-Usabiaga, G., & Carreiras, M. (2015). Developmental evaluation of atypical auditory sampling in dyslexia: Functional and structural evidence. *Human brain mapping*, 36(12), 4986–5002.
- Lizarazu M, Lallier M, Molinaro N. (2019). Phase–amplitude coupling between theta and gamma oscillations adapts to speech rate. *Annals of the New York Academy of Sciences*, 1453(1), 140.

- Lizarazu, M., Carreiras, M., Bourguignon, M., Zarraga, A., & Molinaro, N. (2021a). Language Proficiency Entails Tuning Cortical Activity to Second Language Speech. *Cerebral Cortex*.
- Lizarazu, M., di Covella, L. S., van Wassenhove, V., Rivière, D., Mizzi, R., Lehongre, K., ... & Ramus, F. (2021b). Neural entrainment to speech and nonspeech in dyslexia: conceptual replication and extension of previous investigations. *Cortex*, 137, 160-178.
- Lizarazu, M., Lallier, M., Bourguignon, M., Carreiras, M., & Molinaro, N. (2021). Impaired neural response to speech edges in dyslexia. *Cortex*, 135, 207-218.
- Mayer, K., Wyckoff, S. N., & Strehl, U. (2012). One Size Fits All? Slow Cortical Potentials Neurofeedback. *Journal of Attention Disorders*, 17(5), 393–409.
- Micoulaud-Franchi, J. A., McGonigal, A., Lopez, R., Daudet, C., Kotwas, I., & Bartolomei, F. (2015). Electroencephalographic neurofeedback: Level of evidence in mental and brain disorders and suggestions for good clinical practice. *Neurophysiologie clinique = Clinical neurophysiology*, 45(6), 423–433.
- Molinaro, N., Lizarazu, M., Lallier, M., Bourguignon, M., & Carreiras, M. (2016). Out-of-synchrony speech entrainment in developmental dyslexia. *Human brain mapping*, 37(8), 2767–2783.
- Molinaro N, Lizarazu M. (2018). Delta (but not theta)-band cortical entrainment involves speech-specific processing. *European Journal of Neuroscience*, 48(7), 2642-2650.
- Molinaro, N., Lizarazu, M., Baldin, V., Pérez-Navarro, J., Lallier, M., & Ríos-López, P. (2021). Speech-brain phase coupling is enhanced in low contextual semantic predictability conditions. *Neuropsychologia*, 107830.
- Noachtar, S., Binnie, C., Ebersole, J., Mauguière, F., Sakamoto, A., & Westmoreland, B. (1999). A glossary of terms most commonly used by clinical electroencephalographers and proposal for the report form for the EEG findings. The International Federation of

- Clinical Neurophysiology. *Electroencephalography and clinical neurophysiology. Supplement*, 52, 21–41.
- Omejc, N., Rojc, B., Battaglini, P. P., & Marusic, U. (2018). Review of the therapeutic neurofeedback method using electroencephalography: EEG Neurofeedback. *Bosnian Journal of Basic Medical Sciences*.
- Oostenveld, R., Fries, P., Maris, E., & Schoffelen, J.-M. (2011). FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data. *Computational Intelligence and Neuroscience*, 2011, 1–9.
- Park, H., Ince, R. A., Schyns, P. G., Thut, G., & Gross, J. (2015). Frontal top-down signals increase coupling of auditory low-frequency oscillations to continuous speech in human listeners. *Current biology: CB*, 25(12), 1649–1653.
- Peelle, J. E., Gross, J., & Davis, M. H. (2013). Phase-locked responses to speech in human auditory cortex are enhanced during comprehension. *Cerebral cortex*, 23(6), 1378-1387.
- Pelli, D. G. (1997) The VideoToolbox software for visual psychophysics: Transforming numbers into movies, *Spatial Vision* 10:437-442.
- Pérez, A., Carreiras, M., Dowens, M. G., & Duñabeitia, J. A. (2015). Differential oscillatory encoding of foreign speech. *Brain and language*, 147, 51-57.
- Poeppel D. (2003). The analysis of speech in different temporal integration windows: cerebral lateralization as ‘asymmetric sampling in time’. *Speech Communication*, 41(1), 245-255.
- Poeppel D, Idsardi WJ, Van Wassenhove V. (2008). Speech perception at the interface of neurobiology and linguistics. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363(1493), 1071-1086.

- Rimmele, J. M., Golumbic, E. Z., Schröger, E., & Poeppel, D. (2015). The effects of selective attention and speech acoustics on neural speech-tracking in a multi-talker scene. *Cortex*, 68, 144-154.
- Rostami, R., Salamati, P., Yarandi, K. K., Khoshnevisan, A., Saadat, S., Kamali, Z. S., Ghiasi, S., Zaryabi, A., Ghazi Mir Saeid, S. S., Arjipour, M., Rezaee-Zavareh, M. S., & Rahimi-Movaghar, V. (2017). Effects of neurofeedback on the short-term memory and continuous attention of patients with moderate traumatic brain injury: A preliminary randomized controlled clinical trial. *Chinese journal of traumatology*, 20(5), 278–282. <https://doi.org/10.1016/j.cjtee.2016.11.007>
- Rydesäter, P. (2019). TCP/UDP/IP Toolbox 2.0.6 (<https://www.mathworks.com/matlabcentral/fileexchange/345-tcp-udp-ip-toolbox-2-0-6>), MATLAB Central File Exchange. Retrieved November 2019.
- Schieber, F. (2020). Io64 port interface. (<http://apps.usd.edu/coglab/psyc770/IO64.html>). Retrieved May 2020.
- Schoenberg, P. L., & David, A. S. (2014). Biofeedback for psychiatric disorders: a systematic review. *Applied psychophysiology and biofeedback*, 39(2), 109–135.
- Sherlin, L. H., Arns, M., Lubar, J., Heinrich, H., Kerson, C., Strehl, U., & Sterman, M. B. (2011). Neurofeedback and Basic Learning Theory: Implications for Research and Practice. *Journal of Neurotherapy*, 15(4), 292–304.
- Sitaram, R., Ros, T., Stoeckel, L., Haller, S., Scharnowski, F., Lewis-Peacock, J., ... Sulzer, J. (2016). Closed-loop brain training: the science of neurofeedback. *Nature Reviews Neuroscience*, 18(2), 86–100.
- Sonuga-Barke, E. J., Brandeis, D., Cortese, S., Daley, D., Ferrin, M., Holtmann, M., Stevenson, J., Danckaerts, M., van der Oord, S., Döpfner, M., Dittmann, R. W., Simonoff, E., Zuddas, A., Banaschewski, T., Buitelaar, J., Coghill, D., Hollis, C.,

- Konofal, E., Lecendreux, M., Wong, I. C., ... European ADHD Guidelines Group (2013). Nonpharmacological interventions for ADHD: systematic review and meta-analyses of randomized controlled trials of dietary and psychological treatments. *The American journal of psychiatry*, 170(3), 275–289.
- Tan, G., Thornby, J., Hammond, D. C., Strehl, U., Canady, B., Arnemann, K., & Kaiser, D. A. (2009). Meta-analysis of EEG biofeedback in treating epilepsy. *Clinical EEG and neuroscience*, 40(3), 173–179.
- Thors, H.(2019). *Speech Entrainment to Improve Spontaneous Speech in Broca's Aphasia..* (Doctoral dissertation). Retrieved from
- Walker, J. E., & Kozlowski, G. P. (2005). Neurofeedback treatment of epilepsy. *Child and adolescent psychiatric clinics of North America*, 14(1), 163–viii.

Appendix A: Matlab Scripts for the Real-Time Entrainment Toolbox

RT_Entrainment_Toolbox_Main.m

```
%{
Name: RT_Entrainment_Toolbox_Main.m
Desc: Main script to run a neurofeedback experiment/session with
entrainment measures using the real-time entrainment toolbox
Date: 10/06/2020
Authors: Francisco Javier Carrera Arias, Mikel Lizarazu
%}

% Import libraries and dependencies
```

```

close all;
clear all;
clc;
startup()

%% Localizer Experiment Analysis

% The audio files must be in the order that they were presented to the
% participant/patient. Since the localizer experiment/session only has the
% clear audio condition and participants go through it only once,
% randomization should not be needed
[localizer_audio, audio_fs] = Audio_Processor('S 2', 'all');

% Localizer struct definition
% Note: So far only tested with brainvision products
localizer.eeg_data = 'Test_Trigger.eeg';
localizer.stimulus = 'Stimulus';
localizer.value = 'S 2';
% Keep prestim at 0, baseline correction not as relevant for RT Entrainment
localizer.prestim = 0;
localizer.poststim = 9.59;
localizer.frequencies = [1 7];
localizer.top_n = 5;
localizer.audio = localizer_audio;
localizer.audio_fs = audio_fs;
localizer.plot = 0;
localizer.layout = 'easycapM7.mat';

% Gather top channels from localizer
[top_coherence, channel_names, indices] = speech_brain_coherence(localizer);

%% Define Parameters for BCI loop - Neurofeedback

% Define FIR filter specifications. Order in this case is 4 cycles
% of desired EEG frequency band. Calculated as follows:
% Delta band -> 2 Hz: 1/2 = 0.5/500 ms
% 500 ms x 4 = 2000 ms
% Sampling -> 500 Hz: 1/500 = 0.002/2 ms
% 2 ms * order = 2000 ms -> order = 1000
filtSpec_Freq1.order = 1000;
% The range is the desired frequency band. In this case theta
filtSpec_Freq1.range = [1,3];

% Define FIR filter specifications. Order in this case is 4 cycles
% of desired EEG frequency band. Calculated as follows:
% Theta band -> 6 Hz: 1/6 = 0.166/166.6 ms
% 166.6 ms x 4 = 666.4 ms
% Sampling -> 500 Hz: 1/500 = 0.002/2 ms
% 2 ms * order = 666 ms -> order = 333
filtSpec_Freq2.order = 333;
% The range is the desired frequency band. In this case theta
filtSpec_Freq2.range = [4,7];

% Initialize the cfg struct
cfg = [];
cfg.channel = channel_names; % Channel selection already obtained from
localizer
% Read data from local buffer at port 1972
cfg.dataset = 'buffer://localhost:1972';

```

```
% Trial Definition Parameters
cfg.trialfun = 'ft_trialfun_speechwindow';
cfg.trialdef.eventtype = 'Stimulus';
cfg.trialdef.eventvalue = {'S 8', 'S128'};
% Keep prestim at 0, baseline correction not as relevant for RT Entrainment
cfg.trialdef.prestim = 0;
cfg.trialdef.poststim = 9.59;
cfg.window = 4;
% Stimuli and design parameters
cfg.n_stimuli = 3;
cfg.reference_chan = "wav";

% Run Neurofeedback BCI - Make sure to execute rda2ft to create the buffer
% from the command line prior to running the line below
NF_results = ft_realtime_plv_fully_sync(1070,1920,filtSpec_Freq1,...
    filtSpec_Freq2,cfg);
```

Startup.m

```
%{
startup.m
Francisco Javier Carrera Arias
10/02/2019

Adds fieldtrip as well as the tcp_udp toolbox to matlab prior to
neurofeedback from the RT_Entrainment_Toolbox directory
%}

function startup()
% Adds fieldtrip to MATLAB's path and set everything to its defaults
addpath(sprintf("%s/%s",pwd,"fieldtrip-20200224"))
addpath(sprintf("%s/%s",pwd,"tcp_udp_ip"))
ft_defaults
end
```

Audio_Processor.m

```
%{
Audio_Processor.m
Desc: Loads audio files from stimuli folders based on trigger values
Author: Francisco Javier Carrera Arias
Date: 05/10/2020
```

Note: for this script to work each stimuli folder must have in its name the associated trigger name as well as the condition name separated by underscores

For example: if the trigger values expected for condition 'No Noise' are 'S 8', the folder must be named 'Stimuli_S8_NoNoise'

Inputs:

- trigger: a character array obtained from the event struct of Fieldtrip
- amount: a character array that can be either 'all' to obtain a cell array with all the audio files in a stimuli folder or 'one' to obtain an audio file at random from a given stimuli folder. The 'all' option is meant for the offline localizer session that expects each audio file to be presented only once since and not in a random order since this is only run once per participant. The 'one' option is meant for the neurofeedback session since the audio files of each condition will be presented at random

Outputs:

- audio: a cell array with the audio files if amount is 'all' or a numeric array if amount is 'one'
- audio_fs: an integer with the sampling frequency of the audio files if the sampling frequency of the audio files is not uniform, the localizer session will fail.
- condition: a character array with the condition name of the stimuli folder. If amount is 'all' this is 'localizer' otherwise it is obtained from the stimuli folder name.

%}

```
function [audio, audio_fs, condition] = Audio_Processor(trigger, amount)
    % Show folders in the current working directory
    folders = dir();
    folder_names = {folders(:).name};
    folder_flag = [folders(:).isdir];
    folder_names = folder_names(:,folder_flag);

    % Gather the folder associated with the trigger value
    trigger = trigger(~isspace(trigger)); % Remove any whitespace
    id = cellfun('isempty', regexp(folder_names, trigger));
    target_audio_folder = folder_names(~id);

    % Read all stimuli in the folder as presented in localizer session
    % or one by one at random for neurofeedback
    if isequal(amount, 'all')
        % Gather the audio stimuli folder and set up arrays
        stimuli = dir(sprintf("%s/%s", pwd, target_audio_folder{1}));
        audio = {};
        audio_fs = [];
        % Read the audio and sampling rate for all stimuli
        for k = 3:length(stimuli)
            [aud,fs] = audioread(sprintf("%s/%s", stimuli(k).folder, ...
                stimuli(k).name));
            audio{k-2} = aud.';
            audio_fs = [audio_fs fs];
        end
        % Gather unique sampling frequency
        audio_fs = unique(audio_fs);
        % Condition is localizer
        condition = 'localizer';
        % If the sampling frequency is not unique warn the user (the NF
will
        % fail)
        if length(audio_fs) > 1
            warndlg("The files contain more than 1 sampling frequency.
Please correct this before proceeding further")
        end
    elseif isequal(amount, 'one')
        % Gather the audio stimuli folder and set up arrays
        stimuli = dir(sprintf("%s/%s", pwd, target_audio_folder{1}));
        % Read single audio at random based on how many stimuli there are
        random = randi([3,length(stimuli)]);
        [audio,audio_fs] =
audioread(sprintf("%s/%s", stimuli(random).folder, ...
            stimuli(random).name));
        % Transpose audio
        audio = audio.';
        % Gather condition from the target folder
        condition = regexp(target_audio_folder{1}, '(?<= \w+_) \w+$', ...
```

```

        'once', 'match');
    end
end

```

Trial_Segment.m

```

%{
Name: Trial_Segment.m
Desc: Identifies the trials based on the stimulus for the localizer data
Date: 04/10/2020
Authors: Francisco Javier Carrera Arias
%}

%{
Inputs:
- localizer_data: a .eeg file as given by BrainVision Recorder that
  contains the EEG data from the localizer experiment
- trigger: Character with the name of the trigger type (e.g. Stimulus)
- trigger_name: character vector or cell array of character vectors
  with numbers or strings of the trigger values
- prestim: float, latency in seconds prior to the trigger
- poststim: integer or float with latency in seconds after the trigger
- frequencies: integer vector of dimensions 1x2 with the desired band
  pass frequencies
Outputs:
- data_clean: Preprocessed and epoched EEG data
%}

function data_clean = Trial_Segment(localizer_data, trigger, trigger_name,
    prestim, poststim, frequencies)

%Define the trials based on the function parameters
cfg = [];
cfg.dataset = localizer_data;
cfg.trialfun = 'ft_trialfun_general'; % this is the default
cfg.trialdef.eventtype = trigger;
cfg.trialdef.eventvalue = trigger_name;
cfg.trialdef.prestim = prestim; % in seconds
cfg.trialdef.poststim = poststim; % in seconds

cfg = ft_definetrial(cfg);

% Segment the data and preprocess it from all the channels
% By default we perform baseline correction and band pass
% filtering across the desired frequencies of the localizer
cfg.channel = 'all';
cfg.bpfilter = 'yes';
cfg.bpfreq = frequencies;
cfg.bpfiltertype = 'fir'; % Apply FIR filtering to be consistent with NF
data_clean = ft_preprocessing(cfg);
end

```

audio_envelope.m

```
%{
audio_envelope.m
Desc: Gathers the phase of the speech envelope downsamples to EEG
sampling frequency
Author: Francisco Javier Carrera Arias
Date: 05/25/2020

Inputs:
- audio: a float vector with an audiofile read with audioread
- eeg_fs: a number with the sampling frequency of the EEG system
- audio_fs: a number with the sampling frequency of the audio

Outputs:
- env_ds: a float vector with the resampled audio envelope
%}

function env_ds = audio_envelope(audio,eeg_fs,audio_fs)
% Gather the audio envelope by gathering the amplitude after a hilbert
% transform
env = abs(hilbert(audio));
% Downsample to EEG sampling frequency
env_ds = resample(env,eeg_fs,audio_fs);
end
```

speech_brain_coherence.m

```
%{
Name: speech_brain_coherence.m
Desc: Gathers EEG sensors with the highest mean coherence to speech/audio
between 2 frequencies for localizer experiment
Date: 10/10/2019
Authors: Francisco Javier Carrera Arias, Mikel Lizarazu

Inputs: Struct with the following parameters:
- localizer.eeg_data: character array with the EEG data from
  the offline localizer experiment (currently tested only for brainamp
  products)
- localizer.stimulus: character array with the trigger type for trial/epoch
  segmentation (e.g. 'Stimulus' or 'Response')
- localizer.value: character array with the trigger value
- localizer.prestim: float representing how many seconds to gather prior to
  the trigger onset
- localizer.poststim: float or integer representing how many seconds
  to gather after trigger onset
- localizer.frequencies: numeric vector of dimensions 1x2 containing the
  frequencies to calculate mean coherence
- localizer.top_n: integer indicating the number of channels desired after
  mean coherence calculation
- localizer.audio: cell array with audio files vectors (i.e. wav files
  after being read with audioread)
- localizer.audio_fs: float or integer representing the sampling frequency
  of the audio
- localizer.plot: 0 or 1 indicating whether you want to plot the coherence
  between your selected frequencies across an EEG map
- localizer.layout: character vector with your desired EEG map layout
  (default 32 channel easycapM7)
%}
```

```
function [top_coherence,channel_names,indices] =
speech_brain_coherence(localizer)

% Load the data
% Create Initial Variables
Fdata=[];
Faudio=[];

% Load localizer data
data_clean = Trial_Segment(localizer.eeg_data,localizer.stimulus,...
    localizer.value,localizer.prestim,localizer.poststim,...
    localizer.frequencies);

%% FFTs
% Obtain the fourier transforms of the EEG data and audio envelope
for i=1:size(data_clean.trial,2)
    for j=1:length(data_clean.label)
        Fdata(j,:,i)=fft(data_clean.trial{1,i}(j,:),[],2);
    end
    audio_en =
audio_envelope(localizer.audio{i},data_clean.fsampl,localizer.audio_fs);
    Faudio(:,i)=fft(audio_en,[],2);
end

%% Compute Coherence based on CSD
Fxx=[];
Fyy=[];
Fxy=[];
Faudio = permute(Faudio,[3 1 2]);
Fxx=mean(Fdata.*conj(Fdata),3);
Fyy=mean(Faudio.*conj(Faudio),3);
Fxy=mean(Fdata.*repmat(conj(Faudio),[length(data_clean.label) 1]),3);
coh_audio(:,:)=Fxy.*conj(Fxy)./(repmat(Fyy,[length(data_clean.label)
1]).*Fxx);

% Plot results if desired
%Select the frequency to plot, in this case 1 Hz
fs = data_clean.fsampl;
window = length(coh_audio);

C=[];
for i=1:length(data_clean.label)
    C.label{i,1}=data_clean.label{i,1};
end
C.fsampl=fs;
C.time{1,1}= 0:fs/window:fs-fs/window;
cfg=[];
C.trial{1,1}=squeeze(coh_audio(:,:));
Coh = ft_preprocessing(cfg,C);

% Calculate Channels with top average coherence over frequencies of
% interest
datapoints = Coh.time{1} >= localizer.frequencies(1) & Coh.time{1} <=
localizer.frequencies(2);
% Extract these datapoints from the coherence
coherence = Coh.trial{1}(:,datapoints);
% Gather mean coherence and the top channel names
mean_coherence = mean(coherence,2);
```

```
[top_coherence, indices] = maxk(mean_coherence,localizer.top_n);
channel_names = data_clean.label(indices);
% Reshape cell array for NF section
channel_names = reshape(channel_names,[1,length(channel_names)]);

if localizer.plot == 1
    cfg = [];
    cfg.layout = localizer.layout;
    cfg.hlim = localizer.frequencies;
    cfg.interactive = 'yes';
    cfg.showlabels = 'yes';
    figure; ft_multispectrogram(cfg,Coh);
end

end
```

Screen_Setup.m

```
%{
Name: Screen_Setup.m
Desc: Sets up the screen parameters for question output and fixation cross
Date: 04/15/2020
Authors: Francisco Javier Carrera Arias

Inputs:
- screen_height: Screen height in pixels
- screen_width: Screen width in pixels

Outputs:
- window: a Psychtoolbox window
- allCoords: the coordinates of the fixation cross
- lineWidthPix: the line width of the fixation cross in pixels
- white: white RGB values as given by Screen_Setup()
- xCenter: coordinates of the fixation cross center on the x axis
- yCenter: coordinates of the fixation cross center on the y axis
%}

function [window,white,xCenter,yCenter,...
    allCoords,lineWidthPix] = Screen_Setup(screen_height,screen_width)

    %clear the screen
    sca;

    % Get the screen numbers
    screens = Screen('Screens');

    % Select the external screen if it is present, else revert to the
native
    % screen
    screenNumber = max(screens);

    % Define black
    black = BlackIndex(screenNumber);
    white = WhiteIndex(screenNumber);

    % Open an on screen window and color it grey
```



```
[window, windowRect] = PsychImaging('OpenWindow', screenNumber,
black,...
    [50 50 screen_height screen_width], [], [], [], [], [],...
    kPsychGUIWindow);

% Set the blend function for the screen
Screen('BlendFunction', window, 'GL_SRC_ALPHA',
'GL_ONE_MINUS_SRC_ALPHA');

% Get the centre coordinate of the window in pixels
% For help see: help RectCenter
[xCenter, yCenter] = RectCenter(windowRect);

% Here we set the size of the arms of our fixation cross
fixCrossDimPix = 40;

% Now we set the coordinates (these are all relative to zero we will
let
% the drawing routine center the cross in the center of our monitor for
us)
xCoords = [-fixCrossDimPix fixCrossDimPix 0 0];
yCoords = [0 0 -fixCrossDimPix fixCrossDimPix];
allCoords = [xCoords; yCoords];

% Set the line width for our fixation cross
lineWidthPix = 4;
end
```

Sound_Setup.m

```
%{
Name: Screen_Setup.m
Desc: Sets up the sound parameters for audio presentation
Date: 12/20/2019
Authors: Francisco Javier Carrera Arias

Outputs:
- pahandle: a Psychtoolbox audio handle
%}

function pahandle = Sound_Setup()

% Initialize Sounddriver
InitializePsychSound(1);

% Number of channels and Frequency of the sound
nrchannels = 1;

% Open Psych-Audio port, with the follow arguments
% (1) [] = default sound device
% (2) 1 = sound playback only
% (3) 1 = default level of latency
% (4) Requested frequency in samples per second
% (5) 2 = stereo output
pahandle = PsychPortAudio('Open', [], 1, 1, [], nrchannels);
```

```
% Set the volume to half for this demo
PsychPortAudio('Volume', pahandle, 0.5);
end
```

Draw_Fixation.m

```
%{
Name: Draw_Fixation.m
Desc: Draws a fixation cross in the center of the screen while the
neurofeedback is not active
Date: 04/20/2020
Authors: Francisco Javier Carrera Arias

Inputs:
- window: a Psychtoolbox window as given by Screen_Setup()
- allCoords: the coordinates of the fixation cross as given by
Screen_Setup()
- lineWidthPix: the line width of the fixation cross in pixels. This
parameter is also given by Screen_Setup()
- white: white RGB values as given by Screen_Setup()
- xCenter: coordinates of the fixation cross center on the x axis as given
by Screen_Setup()
- yCenter: coordinates of the fixation cross center on the y axis as given
by Screen_Setup()
%}

function Draw_Fixation(window,allCoords,lineWidthPix,white,...
    xCenter,yCenter)
    % Draw the fixation cross in white, set it to the center of our screen
    % and set good quality antialiasing
    Screen('DrawLines', window, allCoords,...
        lineWidthPix, white, [xCenter yCenter], 2);
    % Flip to the screen
    Screen('Flip', window);
end
```

ft_trialfun_speechwindow.m

```
%{
Name: ft_trialfun_speechwindow.m
Desc: Splits a single speech stimulus into several trials for NF if desired
Date: 04/10/2020
Authors: Francisco Javier Carrera Arias
%}

%{
Inputs:
- cfg: a configuration struct with the following fields:
    - cfg.header: EEG header as given by fieldtrip's ft_read_header
    - cfg.event: Events as given by fieldtrip's ft_read_event
    - cfg.trialdef.eventvalue: The Trigger values to listen to
    - cfg.trialdef.prestim: Period in seconds before trigger onset for
    entrainment neurofeedback typically 0
    - cfg.trialdef.poststim: Period in seconds after trigger onset. This
```

```

    should be equal to the length of the audio file in seconds
    - cfg.window: a number indicating the number of neurofeedback
    presentation windows
Outputs:
- trl: An array with the starting and finishing samples of all the
neurofeedback presentation windows. If the audio length is not an exact
division by the window number, this will contain as many windows as the
nearest exact division (i.e window of 2 and audio length of 9 seconds
would be 4 windows of 2 seconds each)
- event: The event structure with the trigger values
%}

function [trl, event] = ft_trialfun_speechwindow(cfg)

% Gather header and events from configuration structure
ft_info('using the header from the configuration structure\n');
hdr = cfg.hdr;
ft_info('using the events from the configuration structure\n');
event = cfg.event;

trl = [];
if ~isempty(event)
    % Check if the event is in the desired list
    if any(strcmp(cfg.trialdef.eventvalue,event(end).value))
        % Gather sample of last desired event
        sample = event(end).sample;

        % determine the number of samples before and after the trigger
        pretrig = round(cfg.trialdef.prestim * hdr.Fs);
        posttrig = round(cfg.trialdef.poststim * hdr.Fs);

        % Define total length of audio trial based on post stimulus
        total_length = sample + posttrig;

        if isequal(cfg.window,'full')
            trlbegin = sample;
            trlend = total_length;
            offset = pretrig;
            newtrl = [trlbegin trlend offset];
            trl = [trl; newtrl];
        else
            % Break down in smaller trials based on chosen NF presnetation
            window
            % In cases where the length of the audio is not a perfect
            division
            % by the window size, the last window will be the up to the
            % last exact division (i.e. stimuli of length 9s with a window
            % of 2s will have 4 trials of 2 seconds each and the last
            % second will not be used)
            win_num = fix(cfg.trialdef.poststim/cfg.window);
            while size(trl,1) ~= win_num
                trlbegin = sample;
                trlend = sample + (cfg.window * hdr.Fs);
                offset = pretrig;
                newtrl = [trlbegin trlend offset];
                trl = [trl; newtrl];
                sample = trlend;
            end
        end
    end
end

```

```

    end
end
end

```

plv_realtime_fun.m

```

%{
Name: plv_realtime_fun.m
Desc: Displays plv average between EEG channels and/or audio in real-time
Date: 10/10/2019
Authors: Francisco Javier Carrera Arias, Mikel Lizarazu

Inputs:
- dat: an eeg dataset from Fieldtrip's buffer. Currently only tested with
  data from Brain Products Devices
- hdr: a hdr struct from Fieldtrip's ft_read_header
- filtSpec_Freq1: a struct with the order and frequencies in hertz of the
  first set of frequencies that want to track as part of the neurofeedback
- filtSpec_Freq2: a struct with the order and frequencies in hertz of the
  second set of frequencies that want to track as part of the neurofeedback
- reference_chan: a string or integer with "wav" if you want to use audio
  files or an integer with the channel index that you would like to use
  instead of audio files (this index must be taken from the cell array
  given by the localizer, not from all the channels)
- target_baseline: a float with the moving average baseline
- audio_env: an array with the audio envelope. Only used if reference_chan
  is
  "wav"
Outputs:
- A neurofeedback visualization displaying the average PLV between the two
  selected frequencies of interest (for example average entrainment within
  the theta and delta bands)
%}

function plv_avg = plv_realtime_fun(dat,hdr,filtSpec_Freq1,...
    filtSpec_Freq2,reference_chan,target_baseline,...
    audio_env)

% Set defaults if not using audio
if nargin < 7
    audio_env = [];
end

% if target_baseline is nan at the beginning place a 0
if isnan(target_baseline)
    target_baseline = 0;
end

% % (for testing) If window is even set channel 1 and 3 to be the same
% if mod(test_c,2) == 0
%     dat(1,:) = dat(2,:);
% end

if string(reference_chan) ~= "wav"
    % Calculate phase locking value (PLV) with respect to channel
    plv_freq1 = eegPLV_RT(dat,hdr.Fs,filtSpec_Freq1,reference_chan);
    plv_freq2 = eegPLV_RT(dat,hdr.Fs,filtSpec_Freq2,reference_chan);

```

```

else
    % Calculate phase locking value (PLV) with respect to audio
    plv_freq1 = eegPLV_RT(dat, hdr.Fs, filtSpec_Freq1, ...
        reference_chan, audio_env);
    plv_freq2 = eegPLV_RT(dat, hdr.Fs, filtSpec_Freq2, ...
        reference_chan, audio_env);
end

% Calculate average between bands and append to mother vector
freq1_avg = sum(plv_freq1)/length(plv_freq1);
freq2_avg = sum(plv_freq2)/length(plv_freq2);
plv_avg = (freq1_avg+freq2_avg)/2;

% Plot PLV for user feedback
Neurofeedback_Vis_II(plv_avg, target_baseline);
drawnow;
end

```

eegPLV_RT.m

```

function [plv] = eegPLV_RT(eegData, srate, filtSpec, ...
    compareChannel, audio_env)
% Computes the Phase Locking Value (PLV) for a signal over time
% with respect to a given channel
%
% Input parameters:
% - eegData: is a 2D matrix numChannels x numTimePoints
% - srate: is a double with the sampling rate of the EEG data
% - filtSpec: is the filter specification to filter the EEG signal in the
%   desired frequency band of interest. It is a structure with two
%   fields, order and range.
%   - Range specifies the limits of the frequency
%   band, for example, put filtSpec.range = [35 45] for gamma band.
%   - The order of the FIR filter in filtSpec.order. A useful
%   rule of thumb can be to include about 4 to 5 cycles of the desired
%   signal. For example, filtSpec.order = 50 for eeg data sampled at
%   500 Hz corresponds to 100 ms and contains ~4 cycles of gamma band
%   (40 Hz).
% - compareChannel: is a number with the reference channel to
%   used to calculate entrainment or "wav" if an audio file is used
% - audio_env: is a vector with the audio envelope
%   It is only used if compareChannel = "wav"
%
% Output parameters:
% plv is a 2D matrix -
%   numChannels x plv
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created By: Francisco Javier Carrera Arias, Mikel Lizarazu
% Credit to Praneeth Namburi

% Set defaults if audio is not used
if nargin < 5
    audio_env = [];
end

% Gather the number of channels
numChannels = size(eegData, 1);

```

```
% Run an FIR filter through the EEG data accross the time dimension
filtPts = fir1(filtSpec.order,filtSpec.range/(srate/2));
filteredData = filter(filtPts, 1, eegData, [], 2);

% Initilize an empty array for phase locking values and phases
plv = zeros(numChannels,1);
filteredData_Hi = zeros(numChannels,size(filteredData,2));

% Gather the phase from the analytic signal of each channel after
% passing the filtered EEG data through a Hilbert transform
for channelCount = 1:numChannels
    filteredData_Hi(channelCount, :) =
angle(hilbert(filteredData(channelCount, :)));
end

% Obtain the data of the compare channel
% If the compare channel is an audio wav file filter it and compute phase
% separately
if string(compareChannel) ~= "wav"
    compareChannelData = squeeze(filteredData_Hi(compareChannel, :));
else
    fprintf("Using Audio...")
    filteredAudio = filter(filtPts,1,audio_env,[],2);
    compareChannelData = squeeze(angle(hilbert(filteredAudio)));
end

% Calculate the phase locking value between all channels and the compare
% channel
for channelCount = 1:numChannels
    channelData = squeeze(filteredData_Hi(channelCount, :));
    % Phase locking value - the absolute value of the mean phase difference
    plv(channelCount,1) = abs(sum(exp(1i*(channelData -
compareChannelData)), 2))/size(filteredData_Hi,2);
end
return;
```

Neurofeedback_Vis_II.m

```
%{
Neurofeedback Visualization MKII
Desc: Plots entrainment neurofeedback
Author: Francisco Javier Carrera Arias
Date: 2/2/2020

Inputs:
- plv_avg: The entrainment average between the two frequency bands of
interest as given by eggPLV_RT()
- baseline: The entrainment moving average baseline
%}

function Neurofeedback_Vis_II(plv_avg,baseline)
    % Frequency bands average bar plot
    visual = bar(plv_avg);
    visual.FaceColor = "flat";
    % Add baseline horizontal line to the plot
```

```

yline(gca,baseline,'label',sprintf("%s: %s","Baseline",...
    string(round(baseline,2))), 'LabelHorizontalAlignment','center');
ydata = get(visual,'Ydata');
set(visual,'Cdata',ydata,'EdgeColor','none')
% Set up the color map
colormap autumn;
set(gcf,'color','w');
set(gca,'Ylim',[0 1], 'Xlim',[0 2], "YColor",[1 1 1],...
    'XColor',[1 1 1], 'Color',[1 1 1], 'CLim',[0 1]);
% Create colorbar
colorbar("YTick",[]);
% Create title
title(sprintf("Entrainment: %.2f",plv_avg))
% Signals
text(1.04,0.01,"-", "Units", "normalized", "FontSize", 24)
text(1.035,0.99,"+", "Units", "normalized", "FontSize", 20)
end

```

Intel_Question.m

```

%{
Name: Intel_Question.m
Desc: Presents the intellegibility question and gathers the answer
Date: 04/20/2020
Authors: Francisco Javier Carrera Arias

Inputs:
- window: a Psychtoolbox window as given by Screen_Setup()
- white: white RGB values as given by Screen_Setup()

Outputs:
- answer: Gathers the keyboard pressed for answering the question
%}

function answer = Intel_Question(window,white)
% Draw text
DrawFormattedText(window, ['Del 1 (imposible de entender) al 9
(perfectamente claro),'...
    ' califique la intelegibilidad del sonido'],...
    'center', 'center', white);
% Flip to the screen
Screen('Flip', window);

% % Wait for a key press and check key
KbStrokeWait;
[~,pressed] = KbQueueCheck;
[~, Index] = max(pressed);
pressed_key = regexp(KbName(Index), '\d', 'Match');
pressed_key = pressed_key{1};

% Gather the answer
answer = str2double(pressed_key);
end

```

ft_realtime_plv_fully_sync.m

```
%{
Name: ft_realtime_plv_avg_sync.m
Desc: Displays plv between EEG channels in real-time for BCI or
neurofeedback
Date: 10/10/2019
Authors: Francisco Javier Carrera Arias
%}

%{
Inputs:
- screen_height: a number with the screen height in pixels (i.e. 1080)
- screen_width: a number with the screen width in pixels (i.e. 1920)
- filtSpec_Freq1: a struct with the order and frequencies in hertz of the
  first set of frequencies that want to track as part of the neurofeedback
- filtSpec_Freq2: a struct with the order and frequencies in hertz of the
  second set of frequencies that want to track as part of the neurofeedback
- cfg: a configuration struct with the following fields:
  - cfg.channel: a cell array with the channel selections from the
    localizer
  - cfg.dataset: a character array with the address of the data buffer.
    Usually 'buffer://localhost:1972'
  - cfg.trialfun: a character array with name of the trial function
  - cfg.trialdef.eventtype: a character array with the name of the type
of
event that the neurofeedback should sync to. Usually 'Stimulus'
  - cfg.trialdef.eventvalue: a character array or cell array with the
    names of the events/triggers
  - cfg.trialdef.prestim: Period in seconds before trigger onset for
    entrainment neurofeedback typically 0
  - cfg.trialdef.poststim: Period in seconds after trigger onset. This
    should be equal to the length of the audio file in seconds
  - cfg.window: a number indicating the number of neurofeedback
    presentation windows
  - cfg.n_stimuli: a number indicating how many stimuli will be presented
    in the neurofeedback session
  - cfg.reference_chan: a string or integer with "wav" if you want to use
audio
files or an integer with the channel index that you would like to use
instead of audio files (this index must be taken from the cell array
given by the localizer, not from all the channels)
Outputs:
- NF_results: an array containing the results of the neurofeedback
session. It has n_stimuli rows and columns for the stimuli type, answers to
any experimental responses and the PLV value between the selected channels
and the reference channels for every presentation window.
%}

function NF_results = ft_realtime_plv_fully_sync(screen_height,
screen_width,...
    filtSpec_Freq1,filtSpec_Freq2,cfg)

% Setup screen parameters
[window,white,xCenter,yCenter,...
    allCoords,lineWidthPix] = Screen_Setup(screen_width,screen_height);

% Sound Setup Parameters
pahandle = Sound_Setup();
```



```
% Set up keyboard queue
KbQueueCreate;
KbQueueStart;

% Draw Fixation Prior to starting loop
Draw_Fixation(window,allCoords,lineWidthPix,white,xCenter,yCenter)
%% Setup Prior to BCI Loop

% translate dataset into datafile+headerfile
cfg = ft_checkconfig(cfg, 'dataset2files', 'yes');
cfg = ft_checkconfig(cfg, 'required', {'datafile' 'headerfile'});

% ensure that the persistent variables related to caching are cleared
clear ft_read_header

% start by reading the header from the realtime buffer
hdr = ft_read_header(cfg.headerfile, 'headerformat', cfg.headerformat,
'cache', true, 'retry', true);

% define a subset of channels for reading
cfg.channel = ft_channelselection(cfg.channel, hdr.label);
chanindx    = match_str(hdr.label, cfg.channel);
nchan       = length(chanindx);
if nchan==0
    ft_error('no channels were selected');
end

% Housekeeping Variables
prevSample = 0;
count = 0;
count_stimuli = 0;
%test_c = 0; % for testing only

% Obtain how many trials are going to be obtained for each stimulus
window_elements = fix(cfg.trialdef.poststim/cfg.window);

% Initialize master and individual PLV arrays, target and reference sensors
% and target baseline
master_plv_array = zeros(cfg.n_stimuli>window_elements);
condition_array = string();
answer_array = zeros(cfg.n_stimuli,1);
baseline_vec = []; % This ia a vector to calculate a global moving average
%% BCI Loop
while true

    % Detect Status of Audio Playback
    status = PsychPortAudio('GetStatus', pahandle);

    % determine latest header and event information
    event    = ft_read_event(cfg.dataset, 'minsample', prevSample+1); %
only consider events that are later than the data processed sofar
    hdr      = ft_read_header(cfg.dataset, 'cache', true); %
the trialfun might want to use this, but it is not required
    cfg.event = event; %
store it in the configuration, so that it can be passed on to the trialfun
    cfg.hdr   = hdr; %
store it in the configuration, so that it can be passed on to the trialfun
```

```

    % evaluate the trialfun, note that the trialfun should not re-read the
    events and header
    fprintf('evaluating '%s'' based on %d events\n', cfg.trialfun,
length(event));
    [trl,~] = feval(cfg.trialfun, cfg);

    % If trigger is detected in the latest sample,
    % play audio based on trigger value
    if isempty(event) == 0 && event(end).type == "Stimulus" &&
status.Active == 0
        % Randomly select a speech piece based on trigger value
        [audio, audio_fs, condition] =
Audio_Processor(event(end).value, 'one');
        % Obtain audio envelope if reference channel is wav
        if string(cfg.reference_chan) == "wav"
            audio_env = audio_envelope(audio, hdr.Fs, audio_fs);
        end
        % Load audio
        PsychPortAudio('FillBuffer', pahandle, audio);
        % Start audio playback
        PsychPortAudio('Start', pahandle, 1, 0, 1);
    end

    fprintf('processing %d trials\n', size(trl,1));

    if isempty(trl) == 0

        % Intialize trial PLV array
        trial_plv_array = [];

        % Initialize audio segmentation sample if needed
        audio_begsample = 1;

        % Plot Figure
        figure('units', 'normalized', 'outerposition', [0 0 1 1])

        for trllopp=1:size(trl,1)

            begsample = trl(trllopp,1);
            endsample = trl(trllopp,2);

            % remember up to where the data was read
            prevSample = endsample;
            count = count + 1;
            fprintf('-----\n');
            fprintf('processing segment %d from sample %d to %d\n', count,
begsample, endsample);

            % read data segment from buffer
            dat = ft_read_data(cfg.datafile, 'header', hdr,
'begsample', ...
            begsample + 1, 'endsample', endsample, ...
            'chanindx', chanindx, 'checkboundary', false, ...
            'blocking', 'yes', 'timeout', 60);

            tic;
            % Calculate global plv average
            target_baseline = mean(baseline_vec);

```

```

1);

    % Segment the audio envelope to match trial window if
    % needed
    if string(cfg.reference_chan) == "wav"
        audio_endsample = audio_begsample + (endsample-begsample);
        fprintf('processing audio from sample %d to %d\n',...
            audio_begsample, audio_endsample);
        audio_env_seg = audio_env(audio_begsample:audio_endsample-
            1);

        audio_begsample = audio_endsample;
    end

    if string(cfg.reference_chan) ~= "wav"
        % Calculate the iPLV of the target EEG sensors
        % with respect to the reference channel
        iplv = plv_realtime_fun(dat,hdr,filtSpec_Freq1,...
            filtSpec_Freq2,cfg.reference_chan,target_baseline);
    else
        % Calculate the iPLV of the target EEG sensors
        % with respect to audio
        iplv = plv_realtime_fun(dat,hdr,filtSpec_Freq1,...
            filtSpec_Freq2,cfg.reference_chan,target_baseline,...
            audio_env_seg);
    end

    % Append PLV of this window to the vector for the moving
    % average and trial plv array
    baseline_vec = [baseline_vec iplv];
    trial_plv_array = [trial_plv_array iplv];
    toc;
end

% Stop audio when complete
PsychPortAudio('Stop', pahandle, 1);

% Close the NF Figure - window pause for visualization
pause(cfg.window)
close all

% Add stimuli to tally and the PLV values to the master
% PLV array
count_stimuli = count_stimuli + 1;
master_plv_array(count_stimuli,:) = trial_plv_array;

% Intellegibility Question
answer = Intel_Question(window,white);

% further validation questions could be added at this point

% Flip Fixation cross back
Draw_Fixation(window,allCoords,lineWidthPix,white,...
    xCenter,yCenter)

% Collect answer
answer_array(count_stimuli,:) = answer;
% Collect condition
condition_array(count_stimuli,:) = string(condition);
end

```

```

    % Close the bci loop if all the stimuli have been presented
    if count_stimuli == cfg.n_stimuli
        % Concatenate all results
        NF_results = cat(2, condition_array, answer_array, master_plv_array);
        % Clear the screen
        sca;
        % Close the audio device
        PsychPortAudio('Close', pahandle);
        break
    end
end
end

```

Trigger_Gen_Multi.m

```

%{
Trigger_Gen_Multi.m
Desc: Sends triggers to Brain Vision Recorder through a parallel port
Date: 05/15/2020
Author: Francisco Javier Carrera Arias

Inputs:
- n_Trigger: a number indicating how many triggers to send
- n_Trigger_Type: a number indicating how many different trigger types to
send. Current parallel port drivers in this script allow for 3 different
types (i.e input a number between 1 and 3)
- Start_Pause: a number indicating the pause in seconds before the
function begins sending triggers
- Inter_Stimuli_Pause: a number indicating the pause in seconds in between
triggers. Bear in mind the length of each stimulus at the time of
determining this pause
- LPT1_Port: a character array indicating the hex LPT1 port address,
defaults to '4FF8' if nothing is given.
%}

function Trigger_Gen_Multi(n_Trigger,n_Trigger_Type,Start_Pause,...
    Inter_Stimuli_Pause,LPT1_Port)

% Set default LPT1 Port if none given
if nargin < 5
    LPT1_Port = '4FF8';
end

% Initialize parallel port setup
% initialize access to the inpoutx64 low-level I/O driver
config_io;
% optional step: verify that the inpoutx64 driver was successfully
initialized
global cogent;
if( cogent.io.status ~= 0 )
    error('inp/outp installation failed');
end
% write a value to the default LPT1 printer output port (at 0x378)
address = hex2dec(LPT1_Port);

% Pause for NF system startup
pause(Start_Pause)

```

```
% Send triggers to EEG based on stimuli duration
for k = 1:n_Trigger
    % Select trigger pin
    val = randi([1,n_Trigger_Type]);
    % Send Trigger
    outp(address,val);
    fprintf("Trigger\n");
    % Reset Parallel port
    outp(address,0);
    % Inter stimuli pause for questions
    pause(Inter_Stimuli_Pause)
end

% Close parallel port when done
clear all
end
```