Assessment of Fatigue in Brain Computer Interface Users

A Thesis

Submitted to the Faculty

of

Drexel University

by

Vincent J. Petaccio II

in partial fulfillment of the
requirements for the degree

of

Master of Science in Biomedical Engineering

September 2013
Dedications

This thesis is dedicated to those who live each day unable to communicate with their friends, loved ones, and caregivers. May this work aid in the effort to renew your voice.

To Matthew R. Pattley, who persists through his influence on the lives of those who knew and love him. You are missed dearly.
Acknowledgements

To my parents and siblings, for their unending and unconditional support.

To Theresa Vaughan, Jonathan Wolpaw, and the team at the Wadsworth Center, for their innovation in creating BCI2000 and their help and support in my time working the BCI.

To Dr. Fred Allen, for his years of support and aid in my academic endeavors.

To Dr. Terry Heiman-Patterson, Sara Feldman, and the team at the ALS Hope Foundation, for giving me the opportunity and support to make this work a reality and for showing me what it truly means to provide compassionate care.

To Dr. Hasan Ayaz, for serving as my thesis advisor, for providing unending support and advice, and for challenging me to perform at my best.
# Table of Contents

List of Tables .............................................................................................................. viii

List of Figures ............................................................................................................. ix

Abstract ......................................................................................................................... xi

1. Clinical Need ........................................................................................................... 1

2. Specific Aims .......................................................................................................... 2

3. Literature Review and Background ....................................................................... 3

3.1 The Brain Computer Interface ............................................................................ 3

3.1.1 Definition of the Brain Computer Interface .................................................... 3

3.1.2 Components of the Brain Computer Interface ............................................... 4

3.1.2.1 Signal Acquisition ..................................................................................... 5

3.1.2.2 Signal Processing ..................................................................................... 5

3.1.2.3 Output Device .......................................................................................... 6

3.2 The Electroencephalography Signal .................................................................. 6

3.2.1 The Source of the EEG Signal ..................................................................... 7

3.2.1.1 Nerve Cell Potentials .......................................................................... 7

3.2.1.2 Summation of Cortical Potentials ....................................................... 8

3.2.2 Recording EEG .............................................................................................. 9

3.2.2.1 Electrode Design .................................................................................. 9

3.2.2.2 Electrode Placement ............................................................................. 10

3.2.2.3 Digital EEG Instruments .................................................................... 11

3.2.3 EEG Artifacts .................................................................................................. 13

3.2.3.1 Artifacts from Equipment .................................................................... 13
6.3.2 PVT .................................................................................................................. 33
6.3.3 Chalder Fatigue Scale ....................................................................................... 33
6.3.4 BCI Protocol ..................................................................................................... 34
6.4 Data Analysis ......................................................................................................... 35

7. Results .................................................................................................................... 36
   7.1 ERP Results ........................................................................................................ 36
   7.2 PVT Results ....................................................................................................... 40
   7.3 Self-Reported Measures ..................................................................................... 43
      7.3.1 Modified Chalder Fatigue Inventory .............................................................. 43
      7.3.2 Other Self-Reported Measures .................................................................... 47

8. Discussion .............................................................................................................. 49
   8.1 ERP Discussion .................................................................................................. 49
   8.2 PVT Discussion ................................................................................................ 50
   8.3 Self-Reported Measures Discussion ................................................................ 52
      8.3.1 Modified Chalder Fatigue Inventory Discussion ........................................ 52
      8.3.2 Other Self-Reported Measures Discussion ................................................ 52

9. Conclusions and Recommendations to the Field .............................................. 54
   9.1 Conclusions ..................................................................................................... 54
   9.2 Recommendations to the Field ....................................................................... 55

List of References ...................................................................................................... 56

Appendix A: Adapted Chalder Fatigue Inventory ................................................... 60

Appendix B: FREERP MATLAB Functions ............................................................. 61
   B.1 ERP Quality Assessment GUI ........................................................................... 61
B.2 Quality Data Consolidation GUI ............................................................... 65

B.3 ERP Selection Function ........................................................................... 67

B.4 ERP Analysis GUI ....................................................................................... 67

Appendix C: Detailed Protocol ......................................................................... 76

Appendix D: Screening Interview Questionnaire .............................................. 77
List of Tables

Table 1: Contents of the matrix created by the ERP quality assessment GUI. The output is an $n$-by-212 matrix, where the $n$ rows correspond to the $n$ analyzed ERPs in the recording. ............................................................................................................................................................................. 27

Table 2: Mean and standard error of the mean (SEM) values in microvolts ($\mu$V) for three EEG channels for each BCI trial across eleven subjects. Results are separated into two groups by BCI recording session................................................................. 36

Table 3: Mean and standard error of the mean (SEM) values in milliseconds (ms) for three EEG channels for each BCI trial across eleven subjects. Results are separated into two groups by BCI recording session................................................................. 37

Table 4: Group mean PVT response times in milliseconds, with standard error of the mean (SEM) values for all subjects for each PVT trial................................................................. 41

Table 5: Group mean PVT accuracy as percentages, with standard error of the mean (SEM) values for all subjects for each PVT trial ................................................................. 42

Table 6: Subject PVT accuracy as percentages for each subject for each PVT trial................................................................. 42

Table 7: Modified Chalder fatigue inventory response scores for each question for all subjects with error values. See above for score interpretation................................................................. 47

Table 8: Self-reported measures on a 0-10 scale for each subject for each measure, with mean and error values for each measure. Caffeine consumption in the 24 hours before the experimental protocol and clinical depression diagnosis history are on a binary yes/no scale......................................................................................................................................................... 48

Table 9: Age and handedness values for each subject, with mean and error values for age ......................................................................................................................................................... 48
List of Figures

Figure 1: A general schematic of a brain computer interface. Signals generated by the user’s brain are collected by a signal acquisition system before being digitized and processed to produce device commands. These commands control an output device such as a computer, wheelchair, or environmental control unit. Adapted from [1]................................................................. 4

Figure 2: The International 10-20 system modified for use in collecting EEG data for the P300 wave. The view is from the top, with the nose at the top of the image. .. 11

Figure 3: Equivalent circuit diagrams for A) a high-pass filter and B) a low-pass filter. ................................................................. 13

Figure 4: P300 waves elicited by single-stimulus presentation (top) and by the oddball paradigm (bottom). From [4] ................................................................. 16

Figure 5: The typical P300 wave as seen at Pz. Note that the y-axis is inverted. From [1] ........................................................................................................................................ 17

Figure 6: P300 amplitude results with linear regressions from 10 trial blocks, each consisting of 20 auditory ERPs. Amplitude decreases following a proposed habituation period, and then increases during a suggested “dishabituation” period. From [10]........................................................................................................ 20

Figure 7: The GUI created for assessing ERP quality. Here, the user has marked the ERP recorded at around 241 seconds into the session as low quality due to contamination by high-frequency noise, with the selection denoted by the red bar within the ERP time window. The ERP recorded at about 245 seconds has been marked as high quality, denoted by the green bar. The colored vertical bars show the placement of the mouse when the user selected the ERP windows - this feature helps to prevent accidental selections. ........................................................................................................ 26

Figure 8: The UseQuality GUI used for generating the consolidated quality matrix. It is shown with quality matrices loaded and ready for consolidation................... 28

Figure 9: The ERP Analysis GUI after plotting ERP responses for channel six. The y-axis has been inverted, and the range containing the P300 signal has been selected for analysis. The analysis values for the range are shown beneath the plot window. ................................................................................................................................................. 31

Figure 10: Mean P300 amplitude for frontal (Fz), central (Cz), and parietal (Pz) channels for each BCI trial (that is, for each word spelled). Also shown is the mean of all three channels. Results are separated into two groups by BCI recording session.......... 36
Figure 11: Mean P300 latencies for frontal (Fz), central (Cz), and parietal (Pz) channels for each BCI trial. Also shown is the mean of all three channels. Results are separated into two groups by BCI recording session. ................................................................. 37

Figure 12: Group mean ERP waveforms for all subjects for channels Fz and Cz for each BCI trial. .......................................................................................................................................................... 38

Figure 13: Group mean ERP waveforms for all subjects for channels Pz and Oz for each BCI trial. .......................................................................................................................................................... 39

Figure 14: Group mean ERP waveforms for all subjects for all BCI trials for the Fz, Cz, Pz, and Oz channels ................................................................................................................................................ 40

Figure 15: Group mean PVT response times for all subjects for each PVT trial. ..... 41

Figure 16: Group mean PVT accuracy for all subjects for each PVT trial. .......... 42

Figure 17: Group mean Chalder scores for question 1 for all subjects. See above for score interpretation ........................................................................................................................................ 44

Figure 18: Group mean Chalder scores for question 2 for all subjects. See above for score interpretation ........................................................................................................................................ 44

Figure 19: Group mean Chalder scores for question 3 for all subjects. See above for score interpretation ........................................................................................................................................ 45

Figure 20: Group mean Chalder scores for question 4 for all subjects. See above for score interpretation ........................................................................................................................................ 45

Figure 21: Group mean Chalder scores for question 5 for all subjects. See above for score interpretation ........................................................................................................................................ 46

Figure 22: Group mean total response scores for all subjects for each Chalder fatigue inventory trial. Error is shown as standard error of the mean (SEM) for eleven subjects. ........................................................................................................................................ 47
Abstract
Assessment of Fatigue in Brain Computer Interface Users
Vincent J. Petaccio II
Dr. Hasan Ayaz, PhD

The brain computer interface (BCI) is an alternative communication method for those living with physically disabling conditions such as neuromuscular disorders, traumatic brain injuries, and stroke that uses a cap with embedded electrodes to read electroencephalography (EEG) signals and uses them to control a computer. Specifically, the P300, a naïve response to target stimuli, is used to select letters from a matrix. The system provides those with no neuromuscular control with a novel channel for communication.

To date, little to no work has investigated the effects of mental fatigue arising from short-term BCI use on the signals that are used to control the BCI itself. Methods typically used to characterize fatigue in EEG signals employ spectral analysis techniques that are confounded by periodic stimulus presentation in the BCI protocol. The characterization of the effects of fatigue could enable developers to create BCI systems that compensate for or prevent fatigue, which could improve overall usability.

Eleven healthy subjects from the Philadelphia region participated in trials using the P300-based BCI alongside cognitive and self-reported measures of fatigue under informed consent. The changes in EEG signals related to increased mental fatigue occurred in a predictable manner, forming a “checkmark” pattern in P300 amplitude across BCI trials, mirroring the results of a 2006 study investigating fatigue’s effects on the P300 signal specifically. Future work may incorporate this pattern into adaptive
BCI systems capable of detecting and compensating for mental fatigue during long-term BCI use.

Additionally, a software package was designed and distributed to help accelerate and standardize research investigating event-related potentials (ERPs) such as the P300 wave. The package, created in MATLAB, allows the user to rapidly assess ERP quality and perform in-depth analysis of large quantities of data.
1. Clinical Need

There is a need among those with neuromuscular disorders and traumatic brain injuries for a communication method that does not rely upon neuromuscular control. Brain computer interfaces (BCIs) have demonstrated potential to fill this need by creating an alternative channel for communication. To date, little to no work has been done to determine the mentally fatiguing effects of short-term BCI use. If these effects are too severe, then BCI use could become difficult for the disabled. With the characteristics and severity of fatigue effects resulting from BCI use documented, adaptive BCIs could be created to compensate for or to prevent these fatigue effects.
2. Specific Aims

2.1. Identify the effects of mental fatigue resulting from BCI use on electroencephalography (EEG) signals collected from the BCI system.

Fatigue can be directly measured by spectral analysis of EEG signals, but the periodic presentation of BCI stimuli confounds this method of analysis, rendering it ineffective. Measuring fatigue during BCI use through established objective measures alongside self-reported measures will allow for the characterization of correlated changes in EEG signals as collected by the BCI system.

2. Design and distribute an event-related potential analysis package.

Most analysis of event-related potentials (ERPs), such as the P300 signal used by many BCIs, is performed using analysis methods designed *ad hoc*. This has the potential to slow research progress and to make comparisons across studies complicated. By creating and distributing an analysis package for ERP signals, research involving the analysis of these brain signals may be accelerated and, to some degree, standardized.
3. Literature Review and Background

3.1 The Brain Computer Interface

The brain computer interface (BCI) is an assistive device that does not rely upon neuromuscular control, creating an alternative channel for communication and environmental control for those who are afflicted with disorders affecting voluntary muscle control.

3.1.1 Definition of the Brain Computer Interface

Generally, the brain computer interface is defined as a system that bypasses typical peripheral output pathways in order to communicate commands originating in the central nervous system. For example, a BCI that employs the electroencephalogram (EEG) uses commands encoded within the electrical activity recorded on the scalp. A dependent BCI does not use the brain’s typical output pathways to carry output messages, but the brain signals used by the BCI depend upon the activity in those pathways. By contrast, an independent BCI does not in any way rely upon typical output pathways. While many dependent BCI systems are useful, disabled individuals are likely to benefit more from independent systems, as the only requirement is signal generation that is a function of the user’s intent. [1]

The BCI aims to accomplish the intent of the user as would occur if the typical output channels were intact. Nerves and muscles are replaced by the combination of electrophysiological signals, a decoding process, and software and hardware that translate these signals into action. A successful BCI will also provide some degree of
feedback, as typical output pathways depend upon feedback for proper function. It will also successfully integrate adaptations resulting from the brain’s response to this feedback. The performance of a BCI system, then, depends upon the interaction between the brain producing the electrophysiological signals and the system decoding these signals. [1]

3.1.2 Components of the Brain Computer Interface

For complete operation, the BCI requires input from the user, an element for translating the input into an output, the output itself, and a means for controlling the onset, offset, and timing of operation. [1] In practice, this includes the user’s brain signals, a system for signal acquisition, a signal processing scheme, and an output device. Figure 1 below depicts the organization of these elements.

![Diagram of Brain Computer Interface](image)

Figure 1: A general schematic of a brain computer interface. Signals generated by the user’s brain are collected by a signal acquisition system before being digitized and processed to produce device commands. These commands control an output device such as a computer, wheelchair, or environmental control unit. Adapted from [1].
3.1.2.1 Signal Acquisition

The signal acquisition phase includes the recording of the physiological signals that define the input to the BCI. These signals are recorded by electrodes or sensors and are typically amplified and subsequently digitized by a digital-analog converter (DAC) unit.

3.1.2.2 Signal Processing

In the signal processing phase, the digitized brain signals are subjected to feature extraction procedures such as voltage amplitude measurements, spectral analysis, and spatial filtering. The goal of these procedures is to elucidate the information containing the user’s commands. A BCI may use features in either the time or frequency domain, while the use of both domains may enhance the BCI’s performance. Signal features used to control many BCIs reflect identifiable brain events. Alternatively, a BCI may employ signal features that correlate with the user’s intent but do not correlate with specific brain events. It is important to ensure that the signals are not compromised by artifacts not originating in the central nervous system, including electromyogram (EMG) and electrooculogram (EOG) potentials. [1]

The first stage of signal processing extracts the features of the signal used to control the BCI. The next stage includes the translation algorithm, which translates these extracted features into commands used to carry out the intent of the user. This translation from signal feature to command output may be carried out using either linear or nonlinear methods.
In order to maximize the effectiveness of these translation algorithms, a procedure should be included to allow the algorithm to adapt to the user. This is necessary due to variations in signal parameters that are linked to time of day, circadian rhythms, environmental factors, fatigue, and illness. In ideal conditions, the user’s range of signal feature characteristics will vary within the available range of command values even over an extended period of time. [1]

The adaptation of the BCI system does not consider the interaction of the BCI with the brain of the user. With practice, the brain signals are modified in order to maximize BCI performance, representing the final level of adaptation. [1]

3.1.2.3 Output Device

The most fundamental output of the BCI is the selection of a target stimulus by the user. This stimulus can be in auditory, visual, tactile, or some other form. The output device in a BCI is a computer, the response of which may manifest as cursor movement, communication protocols, or control of a prosthesis or a mechanical system. [1]

3.2 The Electroencephalography Signal

Electroencephalography, or EEG, is a method of measuring electrical activity of cortical neurons using electrodes placed noninvasively upon the scalp. It is the control modality most often employed in BCIs, as it offers high temporal resolution, simple setup, and effective operation.
3.2.1 The Source of the EEG Signal

The EEG signal is recorded by placing at least one electrode on the scalp and another elsewhere on the body and monitoring the difference in voltage between these two locations. The voltage recorded is generated by the movement of ions across neuron membranes.

3.2.1.1 Nerve Cell Potentials

EEG potentials are primarily produced by cortical postsynaptic potential changes that alter the density of ionic charges present across pyramidal cell membranes. The membrane potential of these cells is equivalent to the voltage difference between the interior of the cell and the extracellular space.

During periods of relative inactivity, the potential across the cell membrane is known as the resting potential. The cell can be hyperpolarized or depolarized by other cells through excitation or inhibition respectively. The EEG signal that is recorded on the scalp is a temporal and spatial summation of thousands of inhibitory and excitatory postsynaptic potentials of the pyramidal cells.

In addition to these postsynaptic potentials there are also intrinsic cellular currents resulting from ionic movement across the cellular membrane. These currents, and the rebound potentials that result from them, can cause the cell to fire in bursts that may play a larger role in affecting the EEG signal than single action potentials.

An example of the effect of ionic currents on burst firing is the role of adenosine in the sleep homeostat. During waking hours, glycogen in neural tissues is depleted,
resulting in transient deficits in energy density. This causes metabolism of cAMP and a buildup of adenosine. This adenosine activates cortical A1 receptors, which through a membrane delimited g-protein increases potassium conductance ($g_{K^+}$). With the resulting influx of positive ions, the membrane is depolarized, resulting in burst firing. This firing mode ends when Clions enter the cell to direct the membrane potential towards equilibrium, and the process begins again. [2]

### 3.2.1.2 Summation of Cortical Potentials

Pyramidal cells are oriented in vertical columns to ease spatial summation and to allow pyramidal dendrites to reach both deep and superficial layers of the cortex. Groups of these pyramidal neurons receive similar inputs and respond with changes in potential that are similar in timing and in current flow direction. Only a single afferent axon is needed to stimulate several thousand cortical pyramidal neurons due to extensive interconnection.

The currents that are created by pyramidal cell potentials are summed within the extracellular space. Although the majority of this current is kept within the cortex, with amplitude being attenuated exponentially over distance, a small amount is able to penetrate to the scalp. While these currents may result in only small potentials of 10 to 100 μV, they can be recorded through two electrodes on the scalp to define the EEG.

EEG is the summed result of potential changes occurring within large groups of neurons. For this reason, it is not possible to determine whether the EEG signals recorded at the scalp are the consequence of inhibitory or excitatory post-synaptic
potentials (IPSPs and EPSPs). Although both of these events produce current flow in opposite directions, if they are located at opposing ends of the vertical pyramidal cell, then the current seen by a scalp electrode will appear to have identical polarity.

3.2.2 Recording EEG

EEG recorded on the scalp is the result of not only the summed electrical potentials generated by cortical pyramidal neurons, but also of physiological and environmental artifacts and occasionally of signals generated in distant brain regions. The amplitude of the recorded signal depends upon the intensity of the electrical source and its distance from and orientation to the electrode, electrical resistance between the two, and the capacitance of the medium separating the electrode and the source.

There is a difference between EEG signals recorded at the scalp and brain signals recorded at the surface of the cortex itself, with EEG signals of lower amplitude and sometimes distorted in shape. The skull tends to act as a low pass filter, attenuating high frequencies more than lower ones. While EEG may be of lower signal integrity, it is often preferred to cortical brain recordings (electrocorticography, or ECoG) due to its noninvasive nature.

3.2.2.1 Electrode Design

The electrodes used for recording EEG signals for BCIs are generally metal disc or cup electrodes having a diameter between 4 and 10 millimeters (mm). Smaller electrodes may not make sufficient contact with the scalp, while larger ones may be subject to spatial smearing of EEG signals. Disc electrodes should be made of a material that does not interfere with scalp electrolytes, such as gold, tin, silver/silver
chloride, or platinum. Each electrode is connected to a colored insulated wire to aid in easy identification of wires and their corresponding electrodes. These wires are sometimes shielded to prevent contamination from surrounding electrical interference.

### 3.2.2.2 Electrode Placement

The placement of EEG electrodes is standardized under the International 10-20 System of electrode placement. [3] This system uses anatomical landmarks - the inion, nasion, and preauricular points - to define a scalable set of electrode locations that provides uniform coverage of the scalp. These points are used to define a set of lines that run across the head, intersecting at intervals of 10 or 20 percent of the distance between the landmarks. Electrodes are named using abbreviations representing the recorded brain region - Fp (prefrontal), F (frontal), C (central), P (parietal), O (occipital), T (temporal), and A (auricular). These letters are accompanied by a number - odd numbers indicate lateral placement on the left and even numbers on the right - or a "z" to denote placement on the sagittal midline. The numbers increase with distance from the anterior-posterior midline on the head.

The electrodes used in collection of the P300 wave, which is used in many BCI applications, are a subset of eight of the electrodes from the 10-20 system. This set includes Fz, Cz, Pz, P3, P4, PO7, PO8, and Oz. They are highlighted in blue in Figure 2 below.
3.2.2.3 Digital EEG Instruments

A digital EEG system is similar to an analog system with an analog-digital converter (DAC) unit that allows EEG data to be viewed, processed, and stored on a computer. While a digital system lacks the resolution of analog system, options for montages, data processing and analysis are greatly enhanced.

Amplifiers for recording EEG signals use common mode rejection to amplify only the difference in electrical potential between two electrodes. Amplification by this method
allows for the removal electrical noise and potentials generated outside of the brain itself.

The sensitivity of the digital system is the ratio between input voltage and output signal deflection in units of μV/mm. A value of 7 μV/mm is common, but this value can be specified by channel with values between 1 and 1000 μV/mm.

Gain is defined as the ratio of output signal voltage to input signal voltage. Generally, EEG is amplified with a gain of about one million. Gain, unlike sensitivity, increases with increased amplification. Since gain cannot be directly measured, it is not useful in a clinical setting.

Filters are used to exclude particular frequencies outside the range of interest (approximately 1-30 Hz) and to improve signal quality. The EEG signal is amplified by the differential amplifier, and then passed to a single-ended amplifier, which simply applies gain without comparing the signal to another. The signal then passes through filters before being amplified by a final single-ended amplifier. A high-pass filter attenuates slower frequencies, while a low-pass filter attenuates faster frequencies. A notch filter attenuates frequencies within a specific range. Figure 3 below shows equivalent circuit diagrams for both high-pass and low-pass filters.
3.2.3 EEG Artifacts

Artifacts are any recorded EEG signals that do originate from the intended source of recording: the subject’s brain. These are sometimes easily identifiable in the EEG traces by their amplitude or frequency characteristics, which may differ significantly from the EEG signal. However, some artifacts are difficult to detect simply through inspection of the recordings. While artifacts may be the result of interference from electromagnetic radiation emanating from lights or other equipment, variations in power supply, or other sources, they most commonly result from issues with the recording equipment or physiological potentials originating outside of the central nervous system.

3.2.3.1 Artifacts from Equipment

Artifacts from recording equipment can arise when the electrodes are temporarily moved or bumped, causing a physical separation from the scalp. These transient changes in electrical contact can result in short-lived discontinuities in the EEG signal. Such an artifact can be the result of a broken wire connection, a lack of conductive gel, or of faulty wiring of the EEG recording system.
The most common artifact caused by equipment is 60 Hz interference, resulting from electrical interference originating in power lines and equipment. In settings where alternating current (AC) is used to power equipment, there may be some unavoidable interference introduced electrostatically through unshielded power cables or electromagnetically by cables carrying strong AC currents. A notch filter with a cutoff frequency of 60 Hz is often used to remove these artifacts from the EEG signals.

### 3.2.3.2 Artifacts from the Subject

Blinking and other movements of the eye (electrooculogram, or EOG) can cause strong deflections in the frontal electrodes, with the effects sometimes extending to central, temporal, or even parietal electrodes. These potentials were traditionally attributed to the electrical dipole existing between the cornea and retina, though it is now commonly thought to result from a static charge generated by the eyelid rubbing the surface of the eyeball. While these artifacts are often easy to detect within the EEG, electrodes can be placed near the eye to identify timing of eye blinks for more accurate artifact removal.

Muscle activity (electromyogram, or EMG) can generate short duration potentials that often occur in either clusters or in periodic runs. The spikes may occur as discrete potentials and sometimes resemble cerebral spikes or they may occur in rapid bursts and contaminate the cerebral spike activity. Artifacts arising from the scalp and facial muscles are seen mostly in the frontal and temporal electrode recordings, and can
be prevented by ensuring subject comfort and requesting that the subject remain still and refrain from chewing, speaking, or clenching the jaw.

While EOG and EMG artifacts account for the majority of artifacts originating from physiological phenomena, other artifacts may occasionally arise as a result of cardiac activity (EKG), body movements, or changes in skin conductance.

### 3.3 The P300 Wave

The P300 is a robust neural signal that presents in individuals of all ages. It is a naïve response to an observed, desired stimulus. Because it requires no initial training, it is often used as a control signal for BCI systems. [1] It is included in the group of brain signals known as event-related potentials (ERPs), as it is typically seen as a neural response to a stimulus or occurrence.

The method often used to elicit the P300 signal is called the oddball. In a single-stimulus trial, the target stimulus is presented infrequently in the absence of any other stimuli. In a two-stimulus trial, an infrequent target stimulus is presented amongst a background of frequent nontarget stimuli; this is the most common method of stimulation and is also called the oddball paradigm. The subject must respond to the target stimuli either mentally or physically. The elicited brain potential is an increase in amplitude from the frontal to parietal electrodes. [4] Figure 4 shows the single-stimulus and oddball P300 signals as they may appear within the recorded EEG.
The amplitude of the P300 signal indexes brain functioning when a mental representation of the stimulus environment is updated or changed, and the amplitude and latency of the waveform may relate to the probability of an event being remembered. [5] In general, the P300 presents as a positive peak in the EEG trace, occurring approximately three hundred milliseconds following the presentation of a target stimulus. It is from its polarity (P, positive) and latency (300, 300ms) that the P300 gets its name.

The neuroelectric processes determining P300 generation arise from the interactions between the frontal lobe and hippocampal/temporal-parietal function. Studies employing functional magnetic resonance imaging (fMRI) to examine the effects of
the oddball task show patterns consistent with a frontal-to-temporal and parietal lobe activation pattern. Once the incoming stimulus undergoes frontal lobe processing, activity propagates between the cerebral hemispheres and across the corpus callosum to produce the overall waveform of the P300, [4] as shown below in Figure 5.

![Figure 5: The typical P300 wave as seen at Pz. Note that the y-axis is inverted. From [1].](image)

The specific guidelines used for defining the P300 in practice are that it has a latency longer than 275 ms, has a positive polarity at all midline electrodes, has maximal values at parietal and central electrodes, and has an amplitude that is affected by the subjective probability of target presentation and relevance of the stimulus to the task. Scalp distribution typically increases in magnitude along the midline electrodes from frontal to parietal. [4]
3.4 Fatigue

Fatigue is one of the most common symptoms that is observed affecting multiple patient populations. It is viewed as both a symptom and a syndrome. It is often a symptom of medical conditions such as infections, cancer, heart disease, injury to the brain including traumatic brain injury (TBI), multiple sclerosis (MS), Parkinson Disease, stroke, psychiatric disorders, medications, and unhealthy lifestyles. As a syndrome, fatigue is considered to be a major component in a class of little-understood conditions including chronic fatigue syndrome (CFS), neurasthenia, and others. [6]

While the subject of fatigue is one that has been studied rather extensively, [7] a literature search among more than 130 scholarly databases (not all indexing relevant work) and Google Scholar returned no publications in which mental fatigue resulting from the use of a P300-based BCI was studied. About one dozen publications directly mentioned fatigue and BCI use, but many of these studied other BCI modalities and none measured fatigue resulting from BCI use.

3.4.1 Definitions of Fatigue

An inclusive definition for fatigue has proved difficult to obtain over the last 100 years of fatigue research, as self-reported measures of subjective fatigue have typically not correlated well with objective indices of fatigue. [6] The Multiple Sclerosis Council for Practice Guidelines defines fatigue as “a subjective lack of physical and/or mental energy that is perceived by the individual or caregiver to interfere with usual and desired activities,” [8] though some cognitive researchers choose to forego this
subjective definition in favor of direct behavioral observation. However, there seems to be a general consensus that fatigue can be divided into the subcategories of physical and mental fatigue, sometimes called peripheral and central fatigue respectively.

3.4.2 Mechanism of Mental Fatigue

While a mechanism for fatigue has not been fully identified, one proposed mechanism for mental fatigue specifically has implicated modulation of activity in the striato-thalamo-cortical loop connecting the basal ganglia, thalamus, and prefrontal cortex as a possible affector of the condition. The proposed mechanism suggests that the subthalamic nucleus (STN) and the thalamus, which operate in a state of reciprocal activation, may experience a shift in their activation levels. This shift, which may be caused by increased inhibition of STN signals in the thalamus or a decrease in available dopamine in the thalamus, could result in less motivational input to the prefrontal cortex, thus causing an increase in experienced mental fatigue. [9]

3.4.3 Fatigue and the P300

While few studies have focused on fatigue effects in BCI use, the topic has been investigated in ERP research. In 2006, Kececi et al [10] suggested that P300 event-related potentials exhibit a habituation period during which the subject learns the task and the amplitude of the ERP is decreased, followed by a dishabituation period in which the amplitude is increased. This was then demonstrated in healthy controls.

Kececi suggests that the user, after a period of habituation, must employ more cognitive effort in order to maintain a constant level of performance after fatiguing.
More cognitive effort, he argues, results in greater neural activation and greater P300 amplitude. [10] The results from this work, which demonstrate such patterns of P300 amplitude, are shown below in Figure 6.

![Figure 6](image)

Figure 6: P300 amplitude results with linear regressions from 10 trial blocks, each consisting of 20 auditory ERPs. Amplitude decreases following a proposed habituation period, and then increases during a suggested "dishabituation" period. From [10].

Kececi’s results, which in many ways agree with data from other studies, [11] may provide a simple, clear template for detecting fatigue in a P300-based BCI. The “checkmark” pattern of P300 amplitude may be a simple, easily identified, objective marker of fatigue in a system utilizing the signal.

3.5 The Psychomotor Vigilance Task

The psychomotor vigilance task (PVT) is a simple oddball paradigm developed in 1985 by Dr. David Dinges of the University of Pennsylvania that has been used extensively since its publication to record reaction time as a marker for mental fatigue.
It is free of practice effects and is easily and quickly performed using simple computer equipment. [16]

A common implementation of the PVT task places the subject in front of a computer screen with a computer mouse. Integers are displayed on the screen, one at a time, colored white. A rare “target” integer will occasionally appear, colored red. When this red target integer appears, the subject presses the computer mouse button. The response time, measured as the time elapsed between the presentation of the red integer and the pressing of the mouse button, is then recorded. The response times are then compared across trials as a marker for fatigue, in which greater response time values suggest greater mental fatigue. Lapses, or missed targets, may also be correlated with greater mental fatigue. [14]

### 3.6 Chalder Fatigue Scale

The Chalder Fatigue Scale is a self-reported 11-question survey designed to provide a scaled rating of both physical and mental fatigue. It enables subjects to quantify their subjective experience of each form of fatigue separately, and to combine these values into a single score for overall fatigue. [17] Despite the brevity of the questionnaire, it was found to be both reliable and valid. A goal of the authors was to develop a scale that could represent the continuous nature (as opposed to a “binary” yes/no) of fatigue severity. They suggest that it is feasible to remove questions as needed, and that while the scale has good reliability and discriminability, it should be used alongside other forms of fatigue indexing. [17] The set of questions used in the work described here can be found in Appendix A.
3.7 Summary
For those who have been affected by insults to the body that prevent communication through typical channels, an EEG-based BCI utilizing the P300 signal may provide a new channel for communication using only brain data. Fatigue, which may have negative impacts on the usability of such systems, has only very infrequently been studied as a factor in the use of BCIs. The effect of fatigue on the P300 has been studied, and the results from these works may offer a unique opportunity for examining the effects of mental fatigue on the usability of a P300-based BCI. With these effects documented, flexible BCI systems may mitigate or prevent fatiguing effects.
4. Proposed Method

4.1 Hypothesis
While mental fatigue results in complex changes to electrophysiological brain signal generation, the characteristics of the EEG signals that control the BCI system should change in a predictable manner.

4.2 Technique
To hasten the in-depth analysis of large quantities of ERP data and to facilitate the standardization of related research, a MATLAB-based ERP analysis package will be designed and distributed. The software will allow users to quickly survey the quality of individual ERPs and to perform waveform analysis on the data.

The PVT and Chalder Fatigue Inventory will be used to determine the effects of a lengthy BCI session on objective and self-reported fatigue. The EEG data collected during the session will be analyzed to determine the related changes in EEG and ERP activity. The review of the data will take place using the ERP analysis package generated in MATLAB.
5. ERP Analysis Package Design and Distribution

5.1 Rationale

Because ERP research often requires detailed and time-consuming analysis and verification of data, tools must be created to allow the experimenter to access, process, and prioritize ERP information. This is generally done *ad hoc*, with custom tools designed by individual researchers or institutions, sometimes being used only for a single study. The intention of the ERP analysis package described here is to provide researchers with a powerful, flexible, and accessible package for the analysis of ERP data, thereby accelerating research in the field and making it more readily approachable to those with little programming or signal processing experience.

5.2 Design

The goals for the design of this analysis package were to use MATLAB software to create and freely distribute a set of easy-to-use tools to enable researchers to quickly assess and store individual ERP quality, extract specific ERPs from multi-channel EEG recordings, and to perform in-depth analysis of ERP latencies and amplitudes. The resulting toolset contains four MATLAB functions, each of which can be found in detail in Appendix B.

5.2.1 ERP Quality Assessment

The quality of an ERP signal describes the degree of artifact or noise contamination of the EEG signal within the time window containing the ERP. Assessing the quality of the data requires individual analysis of ERP time windows to detect noise or
artifacts such as those described in section 3.2.3 above. While this process can be automated to a degree [19], many researchers prefer to perform this crucial step of data analysis manually in order to ensure that the data being used in downstream processing is of good quality.

This analysis of individual ERP quality can be time demanding, as ERP studies often involve a substantial number of ERPs; for example, the present work required the review of 79,200 ERP time windows. For this reason, a MATLAB graphical user interface (GUI) was developed to expedite this process.

To use the GUI, the user loads the ERPQuality.m file in MATLAB. The user interface opens, and instructs the user to select the channel of data to load, and to then select the data file to open. Once this has been done, the EEG trace opens in the plot window. ERP time windows are automatically marked on the trace by vertical line markers and the letters “ERP” above them. The user may then scroll through the EEG trace, selecting a quality level and then clicking directly in the plot window to mark individual ERPs as low, medium, or high quality. A colored bar appears in the ERP time window to denote the selected quality. To expedite review, all ERPs are by default marked as high quality when the EEG trace is loaded. This GUI is depicted in Figure 7.
Figure 7: The GUI created for assessing ERP quality. Here, the user has marked the ERP recorded at around 241 seconds into the session as low quality due to contamination by high-frequency noise, with the selection denoted by the red bar within the ERP time window. The ERP recorded at about 245 seconds has been marked as high quality, denoted by the green bar. The colored vertical bars show the placement of the mouse when the user selected the ERP windows - this feature helps to prevent accidental selections.

To further accelerate the process of quality assessment, a number of options have been provided for scrolling through the EEG trace. Arrow buttons are provided for slow and fast scrolling in each direction, as well as for jumping to the beginning or end of the recording. Alternatively, the scroll wheel on a computer mouse can be used to shuttle through the data. If these options are not sufficient, the user may also use the “C” and “S” keys to scroll forward and backward through the data respectively. Also, the “1,” “2,” and “3” keys correspond to low, medium, and high quality respectively. The use of hotkeys enables the user to perform nearly every action on the keyboard with one hand while selecting ERPs in the plot area with the other, allowing for very rapid assessment of ERP data quality.
When the user has finished designating the quality of each individual ERP, a number must be entered into the **Subject #** and **Run #** boxes in order to enable the **Save and Close...** button. Pressing this button allows the user to designate a destination for the output matrix containing information about the quality and attributes of the ERPs analyzed, which is saved as a .mat file for use in MATLAB. The contents of this matrix are summarized in Table 1 below.

Table 1: Contents of the matrix created by the ERP quality assessment GUI. The output is an \( n \)-by-212 matrix, where the \( n \) rows correspond to the \( n \) analyzed ERPs in the recording.

<table>
<thead>
<tr>
<th>Column Number</th>
<th>Column Value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8:212</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Number</td>
<td>Subject Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run Number</td>
<td>Run Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel Number</td>
<td>Channel Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Letter Number</td>
<td>Letter Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash Number</td>
<td>Flash Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERP Start Sample</td>
<td>ERP Start Sample</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality Rating</td>
<td>Quality Rating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All ERP Samples</td>
<td>All ERP Samples</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Since each EEG trace assessed using this utility results in a unique output matrix, an additional utility was generated to consolidate all of these matrices into a single matrix containing all of the information from all ERPs evaluated. In order to create this matrix, the user runs the UseQuality.m MATLAB function, which opens a GUI (Figure 8) with two buttons, one of which is disabled. The only available button allows the user to select the quality matrices to consolidate. Once the user has made the selection, the names of these files are used to populate a list box on the GUI and the **Create Data Matrix** button is enabled. If the user is satisfied with the list of files, clicking this button will generate a consolidated matrix in the output location specified by the user. This consolidated matrix is of the same format shown in Table 1. Once this consolidated quality matrix has been created, the user can begin to extract ERPs meeting specific criteria from the recorded ERP data.
5.2.2 ERP Extraction

With each ERP assessed for its quality, and the consolidated quality matrix generated, the user now has access to a data matrix of considerable size, containing a substantial quantity of information. In order to ease the selection of specific ERPs, a MATLAB function was created to enable rapid selection of ERPs meeting specific criteria.

Using the erpselect.m function, users can extract from the consolidated quality matrix ERPs from specific subjects, runs, channels, letters, target flashes, and of specific quality ratings. Any combination of these parameters may be specified. Additionally, the function allows the user to normalize each ERP to its own maximum amplitude.

In order to use the erpselect.m function, the user types the commands in standard MATLAB format within the Command Window. The function accepts the inputs in the following order:

```matlab
erpselect(qualityMatrix, subjects, runs, channels, letters, flashes, quality, normalize)
```
where \textit{qualitymatrix} is the consolidated quality matrix, \textit{normalize} is a logical integer of value 0 or 1 (1 normalizes the output while 0 does not), and each of the other inputs are vectors containing the desired values. To select all of the available values for any input type, the user may enter a 0. For example, if the user wishes to extract only the high quality, normalized ERPs from all subjects, runs, letters, and flashes for channels 2, 6, 12, and 16, and assign the recorded ERP values to the output variable entitled \textit{erpdata}, then the usage will be:

\begin{verbatim}
    erpdata = erpselect(qualitymatrix,0,0,[2 6 12 16],0,0,[3],1);
\end{verbatim}

The output will in this case be an \textit{n}-by-205 matrix, with each row containing the EEG data for the specified ERP time windows. This matrix can then be averaged, plotted, or processed as the user chooses.

\textbf{5.2.3 ERP Analysis}

With the quality of the ERP data assessed, the data can be processed in a number of ways. Common metrics for analyzing ERPs include onset latency, peak amplitude, and spectral analysis.

In order to analyze ERPs that have been assessed for their quality, a third MATLAB GUI was generated to allow the user to load the consolidated quality matrix, and then plot ERPs selected by subject, channel, run, letter, flash, and quality rating. Plots can optionally be generated with normalized ERP data. By clicking on the resulting plot, the user can determine values including maximum, minimum, mean, median, and latency values. The trace data and analysis values can be exported in .xls format for
further processing in Microsoft Excel. The user can also export a screenshot of the plot for use in publications or otherwise.

To use the ERP Analysis GUI, the user first runs the ERPAnalysis.m MATLAB function. Next, the user selects the consolidated quality matrix generated by using the UseQuality GUI, and then chooses the parameters to plot. Pressing the Plot Responses button will generate the plot specified. Next, the user can determine the aforementioned analysis values by clicking on the plot to select a range of the ERP, displaying the values beneath the plot window. Figure 9 below shows the ERP Analysis GUI in this state.

With these values generated, the user can add them to a list of values to be added to an analysis spreadsheet by clicking on the Queue to .xls button. After the analysis has been completed, the entire list of values can be written to a spreadsheet by clicking the Commit to .xls button. Similarly, the Export plot data... button will enable the user to specify a location for a .xls file containing the x and y values for the ERP trace, and the Save Screenshot button allows the user to save a screenshot of the GUI to a specified location in .jpeg, .bmp, or .tiff formats.
Figure 9: The ERP Analysis GUI after plotting ERP responses for channel six. The y-axis has been inverted, and the range containing the P300 signal has been selected for analysis. The analysis values for the range are shown beneath the plot window.

5.3 Distribution

The software package is to be distributed freely and publicly in order to make it accessible to as many researchers as possible. The first method of distribution will be through direct transfer to collaborators and colleagues. Additionally, the analysis package was made available as a free download from the MATLAB File Exchange on the MathWorks website, allowing MATLAB users to download the files as .m files. Finally, a website (www.freerponline.weebly.com) was created from which any internet user can download the software package. The package itself has been named FREERP (pronounced as “Free R P”), and has been given its own branding.
6. Methods

6.1 Subjects
Eleven able-bodied individuals from the Philadelphia area participated in the present investigational trials. Subjects had varying experience using the brain computer interface. Each subject provided informed consent, and the work was approved by the Drexel University College of Medicine Institutional Review Board.

6.2 Data Acquisition
EEG signals were recorded using an EEG cap manufactured by Electro-Cap International, Inc. containing 16 embedded electrodes placed at the F3, F2, F4, T7, C3, C2, C4, T8, CP3, CP4, P3, Pz, P4, PO7, PO8, and O2 electrode locations. All channels were grounded to the right mastoid and referenced to the left mastoid. The EEG signal was filtered using a bandpass filter with cutoffs at 0.1 and 30 Hz, as well as a notch filter at 60 Hz, and was then amplified with a Guger Technologies amplifier before being digitized at a rate of 256 Hz and stored. All aspects of data collection were controlled by the BCI2000 system [20]. Acquisition occurred in a clinic room operated by Drexel University College Medicine, Department of Neurology.

6.3 Experimental Protocol
For a diagrammatic display of the experimental protocol, along with approximate durations of each step of the procedure, see Appendix C.
6.3.1 Screening Interview

Before any data collection began, a screening interview was administered in order to determine the initial cognitive state of the subject by interviewing the subject with regards to previous night’s sleep (amount and quality), recent ingestion of stimulants such as caffeine, history of diagnosis of depression, and experience with brain-computer interface use. Each of these parameters has been shown to affect the results of EEG-based investigation, [21-24] and could therefore impact the use of a P300-based brain-computer interface. The questionnaire that was used can be found in Appendix D.

6.3.2 PVT

For this protocol, a simple task was created in which the subject continually monitored a computer screen on which single randomized integers were displayed in 1s intervals, colored white. Targets were indicated by a red integer, and the subject responded to targets with a left-click of the computer mouse. The task lasted for 5 minutes, including 300 total integers, and was implemented following each step of the protocol to track the influence of each task towards the subject’s objective cognitive performance.

6.3.3 Chalder Fatigue Scale

For this protocol, the physical fatigue questions were omitted, as well as any non-relevant mental fatigue questions, as suggested by the authors. [17] This scale was included in order to elucidate the subjective effects of BCI use on the subject’s experience. Another intention for using this questionnaire alongside the PVT task was
to increase internal validity with a minimal addition to the length of the session. The list of questions can be found in Appendix A.

6.3.4 BCI Protocol

BCI2000 software [20] was used to power the brain computer interface. The subjects were placed in a chair approximately one meter from a computer monitor. Within the BCI2000 software, an 8x9 matrix of white characters was displayed on a black background, following the work documented in [25].

The subject’s task was to complete a copy spelling session in which the system specified the target letter in order to facilitate offline analyses. The subject was instructed to attend to one letter of the matrix and count the number of times that target letter was intensified. The characters within the matrix were intensified in random groupings in a random sequence designed to prevent adjacent letters from flashing simultaneously to help prevent proximity effects among them [25]. At the start of each run, the first letter of the target word was displayed in parentheses at the end of the word above the matrix, designating this letter as the target letter. After ten intensifications (epochs) the classifier made a decision on the letter chosen. The sampling rate was 256 Hz, and there was a 4 second delay before and after the test sequence with a stimulus duration of 3 sample blocks and an interstimulus interval of 4 sample blocks. After five seconds, the next target character in the word was presented in parentheses. This process continued until the subject spelled the entire target word.
For the first half of the BCI protocol, the subjects spelled “DREXEL” then “UNIVERSITY” and finally “BRAIN”. For the second half of the protocol, the subjects spelled “COMPUTER” then “INTERFACE” and finally “8675309”. The words were to provide a large body of EEG data and to lengthen the session in order to elicit fatigue.

6.4 Data Analysis

The FREERP analysis package previously generated was used to analyze the ERP signals. The P300 recordings were measured for their amplitude, latency, and other statistics. While fatigue has previously been characterized in EEG signals using spectral analysis, [26] the spectral information contained within the EEG signals produced by a P300 BCI are contaminated due to the periodic presentation of stimuli. Therefore, the analysis carried out in this work was restricted to the time domain. PVT data was consolidated using MATLAB software and analyzed by measuring response time and accuracy (defined more by the number of missed targets). All self-reported measures were entered into a computer to enable more rapid analysis, also using MATLAB software.
7. Results

7.1 ERP Results

For comparison with the results from [10], shown in Figure 6 in section 3.4.3, the P300 amplitudes—each defined as the maximum value within an 800ms time window following stimulus presentation—were plotted as a function of BCI trial number. Figure 10 below shows the results, which are also summarized with errors in Table 2.

![Group P300 Amplitude by BCI Trial (n=11)](image)

Figure 10: Mean P300 amplitude for frontal (Fz), central (Cz), and parietal (Pz) channels for each BCI trial (that is, for each word spelled). Also shown is the mean of all three channels. Results are separated into two groups by BCI recording session.

### Table 2: Mean and standard error of the mean (SEM) values in microvolts (µV) for three EEG channels for each BCI trial across eleven subjects. Results are separated into two groups by BCI recording session.

<table>
<thead>
<tr>
<th>Trial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SE M</td>
<td>Mean</td>
<td>SE M</td>
<td>Mean</td>
<td>SE M</td>
</tr>
<tr>
<td>Fz</td>
<td>4.54</td>
<td>0.42</td>
<td>4.01</td>
<td>0.31</td>
<td>4.49</td>
<td>0.47</td>
</tr>
<tr>
<td>Cz</td>
<td>5.11</td>
<td>0.49</td>
<td>4.60</td>
<td>0.51</td>
<td>4.95</td>
<td>0.29</td>
</tr>
<tr>
<td>Pz</td>
<td>4.58</td>
<td>0.48</td>
<td>4.06</td>
<td>0.32</td>
<td>4.09</td>
<td>0.25</td>
</tr>
</tbody>
</table>
The P300 latency was defined as the time in milliseconds (ms, 1/1000 of one second) between stimulus presentation and the P300 amplitude as defined above. Because the EEG signals were digitized at a sampling rate of 256 Hz, the recordings have a resolution of 3.91 ms. The latencies values for three channels are also shown above in Figure 11, and are summarized with errors in Table 3.

Table 3: Mean and standard error of the mean (SEM) values in milliseconds (ms) for three EEG channels for each BCI trial across eleven subjects. Results are separated into two groups by BCI recording session.

<table>
<thead>
<tr>
<th>Trial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SE</td>
<td>Mean</td>
<td>SE</td>
<td>Mean</td>
<td>SE</td>
</tr>
<tr>
<td>FZ</td>
<td>308.6</td>
<td>46.9</td>
<td>275.7</td>
<td>45.0</td>
<td>333.8</td>
<td>43.6</td>
</tr>
<tr>
<td>CZ</td>
<td>296.9</td>
<td>37.9</td>
<td>268.6</td>
<td>26.4</td>
<td>238.8</td>
<td>21.7</td>
</tr>
<tr>
<td>PZ</td>
<td>266.4</td>
<td>17.9</td>
<td>280.6</td>
<td>25.9</td>
<td>265.7</td>
<td>24.3</td>
</tr>
</tbody>
</table>

The mean P300 waveforms for the central channels (FZ, CZ, PZ, and OZ) for all subjects are presented for each BCI trial on the following pages.
Figure 12: Group mean ERP waveforms for all subjects for channels Fz and Cz for each BCI trial.
Figure 13: Group mean ERP waveforms for all subjects for channels Pz and Oz for each BCI trial.
The mean P300 waveforms for all subjects for all BCI trials for channels Fz, Cz, Pz, and O2 are shown below in Figure 14.

**Group Mean ERP Waveforms (n=11)**

PVT data was analyzed for response time and for accuracy, with accuracy defined using the number of lapsed responses. Before analysis began, statistical outliers, defined as values that varied by more than two standard deviations from the mean, were removed from the data. Lapses, where response time was greater than the stimulus presentation duration (that is, one second) were also removed from the data. PVT response time for all subjects for each PVT trial is shown below in Figure 15, with values and errors summarized in Table 4.
Figure 15: Group mean PVT response times for all subjects for each PVT trial.

Table 4: Group mean PVT response times in milliseconds, with standard error of the mean (SEM) values for all subjects for each PVT trial.

<table>
<thead>
<tr>
<th>Trial</th>
<th>Mean RT (ms)</th>
<th>SEM (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>401.5</td>
<td>9.18</td>
</tr>
<tr>
<td>2</td>
<td>397.2</td>
<td>8.73</td>
</tr>
<tr>
<td>3</td>
<td>386.9</td>
<td>11.34</td>
</tr>
<tr>
<td>4</td>
<td>391.2</td>
<td>13.23</td>
</tr>
</tbody>
</table>

PVT accuracy was calculated by dividing the total number of targets presented by the number of targets for which a response was recorded. Lapses where no response was recorded counted against accuracy. These lapses represent brief periods of fatigue-related microsleep [16] and are believed to result from behavioral habituation to fatiguing tasks. Figure 16 below shows group mean accuracy for each PVT trial throughout the experimental protocol, and Table 5 shows these results with errors. Table 6 shows accuracy for each subject for each PVT trial.
Figure 16: Group mean PVT accuracy for all subjects for each PVT trial.

Table 5: Group mean PVT accuracy as percentages, with standard error of the mean (SEM) values for all subjects for each PVT trial.

<table>
<thead>
<tr>
<th>Trial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Accuracy (%)</td>
<td>99.39</td>
<td>99.60</td>
<td>95.56</td>
<td>94.95</td>
</tr>
<tr>
<td>SEM (%)</td>
<td>0.61</td>
<td>0.27</td>
<td>4.44</td>
<td>3.98</td>
</tr>
</tbody>
</table>

Table 6: Subject PVT accuracy as percentages for each subject for each PVT trial.

<table>
<thead>
<tr>
<th>Trial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>S2 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>S3 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>S4 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>97.8</td>
</tr>
<tr>
<td>S5 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>S6 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>95.6</td>
</tr>
<tr>
<td>S7 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>S8 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>S9 Accuracy (%)</td>
<td>100</td>
<td>97.8</td>
<td>100</td>
<td>95.6</td>
</tr>
<tr>
<td>S10 Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>S11 Accuracy (%)</td>
<td>93.3</td>
<td>97.8</td>
<td>51.1</td>
<td>55.6</td>
</tr>
</tbody>
</table>
7.3 Self-Reported Measures

A number of self-reported measures were collected from each subject during the course of the experimental protocol. These measures included the modified Chalder fatigue inventory, previous night’s sleep quantity and quality, caffeine ingestion, age, BCI experience, and history of clinical depression.

7.3.1 Modified Chalder Fatigue Inventory

The Chalder fatigue inventory was collected four times throughout the experimental protocol immediately following each PVT trial, and was modified to include five questions targeting cognitive fatigue, which were:

1. Do you feel sleepy or drowsy?
2. Are you lacking in energy?
3. Do you have difficulty concentrating?
4. Do you have problems thinking clearly?
5. How is your memory?

Each question offered four responses which were identical for each question and were taken from [17]. They were better than usual, no more than usual, worse than usual, and much worse than usual. For the sake of analysis, they were assigned scores of one to four respectively, with higher scores corresponding to greater quantities of experienced fatigue. The scores for each question are shown on the following pages.
Figure 17: Group mean Chalder scores for question 1 for all subjects. See above for score interpretation.

Figure 18: Group mean Chalder scores for question 2 for all subjects. See above for score interpretation.
Figure 19: Group mean Chalder scores for question 3 for all subjects. See above for score interpretation.

Figure 20: Group mean Chalder scores for question 4 for all subjects. See above for score interpretation.
Figure 21: Group mean Chalder scores for question 5 for all subjects. See above for score interpretation.

Table 7 below shows the Chalder response scores for each question for each trial, with mean and error values for each question for all subjects. Figure 22 on the following page shows the group mean total scores for each Chalder trial. These values are the mean across all questions across all subjects for each trial, with error bars shown as standard error of the mean for eleven subjects.
Figure 22: Group mean total response scores for all subjects for each Chalder fatigue inventory trial. Error is shown as standard error of the mean (SEM) for eleven subjects.

Table 7: Modified Chalder fatigue inventory response scores for each question for all subjects with error values. See above for score interpretation.

<table>
<thead>
<tr>
<th>Trial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SEM</td>
<td>Mean</td>
<td>SEM</td>
</tr>
<tr>
<td>Q1 Score</td>
<td>2.18</td>
<td>0.12</td>
<td>2.36</td>
<td>0.15</td>
</tr>
<tr>
<td>Q2 Score</td>
<td>2.18</td>
<td>0.12</td>
<td>2.18</td>
<td>0.12</td>
</tr>
<tr>
<td>Q3 Score</td>
<td>2.00</td>
<td>0.00</td>
<td>2.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Q4 Score</td>
<td>2.00</td>
<td>0.00</td>
<td>2.09</td>
<td>0.09</td>
</tr>
<tr>
<td>Q5 Score</td>
<td>2.00</td>
<td>0.00</td>
<td>2.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>2.07</td>
<td>0.03</td>
<td>2.13</td>
<td>0.05</td>
</tr>
</tbody>
</table>

7.3.2 Other Self-Reported Measures

The remaining self-reported measures include sleep quantity, sleep quality, caffeine consumption, history of clinical diagnosis of depression, and BCI experience. Subjects reported each value on a continuous scale by marking a 10 cm long line. The score were then determined by measuring from the beginning of the line, with
scores of 0-10 corresponding to the location of the mark created by the subject.

See Appendix D for the detailed form used by subjects to report these values. Table 8 below summarizes the results of these reported values, along with group means and errors.

Table 8: Self-reported measures on a 0-10 scale for each subject for each measure, with mean and error values for each measure. Caffeine consumption in the 24 hours before the experimental protocol and clinical depression diagnosis history are on a binary yes/no scale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Sleep Hours</th>
<th>Sleep Quality</th>
<th>Caffeine (y/n)</th>
<th>Depression (y/n)</th>
<th>BCI Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject 1</td>
<td>7.00</td>
<td>4.20</td>
<td>Y</td>
<td>N</td>
<td>3.50</td>
</tr>
<tr>
<td>Subject 2</td>
<td>5.50</td>
<td>4.80</td>
<td>Y</td>
<td>N</td>
<td>5.00</td>
</tr>
<tr>
<td>Subject 3</td>
<td>8.00</td>
<td>7.40</td>
<td>Y</td>
<td>N</td>
<td>8.50</td>
</tr>
<tr>
<td>Subject 4</td>
<td>5.00</td>
<td>2.50</td>
<td>Y</td>
<td>N</td>
<td>0.00</td>
</tr>
<tr>
<td>Subject 5</td>
<td>8.00</td>
<td>9.00</td>
<td>Y</td>
<td>Y</td>
<td>0.00</td>
</tr>
<tr>
<td>Subject 6</td>
<td>7.00</td>
<td>5.00</td>
<td>Y</td>
<td>N</td>
<td>0.00</td>
</tr>
<tr>
<td>Subject 7</td>
<td>6.75</td>
<td>6.70</td>
<td>Y</td>
<td>N</td>
<td>0.00</td>
</tr>
<tr>
<td>Subject 8</td>
<td>7.00</td>
<td>4.60</td>
<td>Y</td>
<td>N</td>
<td>0.00</td>
</tr>
<tr>
<td>Subject 9</td>
<td>6.00</td>
<td>5.00</td>
<td>Y</td>
<td>N</td>
<td>0.00</td>
</tr>
<tr>
<td>Subject 10</td>
<td>8.00</td>
<td>7.00</td>
<td>Y</td>
<td>N</td>
<td>5.00</td>
</tr>
<tr>
<td>Subject 11</td>
<td>6.80</td>
<td>7.80</td>
<td>N</td>
<td>N</td>
<td>0.00</td>
</tr>
<tr>
<td>Mean Value</td>
<td>6.82</td>
<td>5.82</td>
<td>N/A</td>
<td>N/A</td>
<td>2.00</td>
</tr>
<tr>
<td>SEM Value</td>
<td>0.30</td>
<td>0.57</td>
<td>N/A</td>
<td>N/A</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Additional values reported by each subject include age and handedness. These values are summarized below in Table 9, along with mean and error values.

Table 9: Age and handedness values for each subject, with mean and error values for age.

<table>
<thead>
<tr>
<th>Value</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
<th>S10</th>
<th>S11</th>
<th>Mean</th>
<th>SEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age (yrs)</td>
<td>26</td>
<td>61</td>
<td>48</td>
<td>57</td>
<td>49</td>
<td>24</td>
<td>55</td>
<td>49</td>
<td>59</td>
<td>57</td>
<td>22</td>
<td>46.09</td>
<td>4.47</td>
</tr>
<tr>
<td>Handedness</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
8. Discussion

8.1 ERP Discussion

The ERP results have great potential to affect the implementation of BCI protocols, as they represent data that is already collected and available in all existing ERP-based BCI systems.

As expected, the channel showing the most positive P300 amplitude and, by inspection, the most prominent peak, is channel Cz. This is true for all but one BCI trial (trial number five), where the difference is within the margin of error. In this case, the channel with the most positive amplitude is channel Fz. The characteristics of the P300 signals recorded generally match the accepted definition of the P300 potential.

Perhaps the most notable result arising from the analysis of the ERP signals is the pattern of P300 amplitudes occurring across each BCI session. Each session, comprised of three BCI trials, shows an initial decrease of peak ERP amplitude, followed by a subsequent increase in amplitude. Additionally, the initial peak amplitude is lowered for the second BCI session with respect to the first. These results, shown in Figure 10, are in direct agreement with the results shown in [10], and mirror the trend in peak amplitudes shown in Figure 6 almost exactly. This suggests strongly that the effects of prolonged BCI use on P300 amplitude manifest as a habituation and subsequent dishabituation to the task.

The second BCI session begins with an initial ERP amplitude that is lesser in magnitude than the initial amplitude recorded during the first session. Likewise, the
second trial in session two shows decreased amplitude compared to the second trial in session one; this trend continues for the third trial in each session. Since this protocol included two BCI sessions, it is not known whether the amplitudes arising from a third session would continue this downward trend or if they would mirror the inter-session trend shown in Figure 10. If the mechanisms suggested in [10] are implicated in the production of this trend, though, it is likely that a third BCI session would exhibit a continuation of the downward trend in P300 amplitudes.

The P300 latency results reveal very little upon initial inspection, and inclusion of the error among the results further confirms the absence of meaningful information from the latency data collected here. Since each channel shows a unique pattern of latency activity, it is difficult to identify any behavioral brain activation patterns that may be occurring as a result of experienced fatigue.

8.2 PVT Discussion

The inclusion of the PVT task was intended to confirm objectively that mental fatigue results from prolonged use of a P300-based BCI system. Target response time and accuracy were the metrics measured from the PVT.

Interestingly, the three PVT response time results collected before the first BCI trial, before the second BCI trial, and after the second BCI trial (PVT trials 2, 3, and 4 respectively) also mirror the “checkmark” pattern of P300 amplitude shown in [10] and in the ERP results. While PVT response times are generally thought to directly correlate to mental fatigue, it is difficult to ignore the influence of other cognitive tasks on the results of the PVT task. These results are, at least fundamentally, in line with
Kececi’s suggested mechanism driving P300 amplitudes during mental fatigue. That is, an initial fatiguing BCI session may result in a lowered mean response time after allowing the user to “warm up,” but a subsequent compensatory increase in cognitive effort during a second fatiguing BCI trial may result in lengthened response times as the user becomes mentally fatigued.

PVT accuracy, in contrast to response time, followed a downward trend throughout the experimental protocol. This accuracy, related directly to mental fatigue, may be less susceptible to the effects of the cognitive task of the BCI since it is a measure of the impact of lapses in attention, and not a measure of behavioral performance.

It is important to note, however, that the errors among both PVT measures are substantial, and—especially in the case of accuracy—may be attributed primarily to the results from a minority of subjects.
8.3 Self-Reported Measures Discussion

8.3.1 Modified Chalder Fatigue Inventory Discussion

The modified Chalder fatigue inventory was collected in order to elucidate the impact of prolonged BCI use on the user’s phenomenological experience of mental fatigue. The user’s experience of fatigue is, perhaps, more important than objective reporting of fatigue, as it is the subjective experience that may impact usability of BCI systems.

The scores, which correspond directly to the level of experienced fatigue, increase notably among the duration of the protocol, with the combined score showing a significant increase from 2.07 to 2.49, a change of more than twenty percent. This change suggests that there was a significant increase in the level of fatigue experienced by the subjects participating in this study as the protocol was administered.

While every question showed an increase in response score throughout the duration of the protocol, the first question (“Do you feel sleepy or drowsy?”) showed the most substantial change in score, rising from 2.18 to 2.82. This may represent the phenomenon most readily experienced by users of a BCI system.

8.3.2 Other Self-Reported Measures Discussion

The other values reported by the subjects enrolled in this study included age, handedness, quantity and quality of the previous night’s sleep, consumption of caffeine in the last 24 hours, history of diagnosis of clinical depression, and previous experience with BCI systems of any type. These values, shown above in section 7.3.2,
were not found to correlate with any other results collected. They are presented here without analysis.

One trend of note from these self-reported values is that just one of eleven subjects had not consumed caffeine within the 24 hours prior to the protocol. It is not unreasonable to expect a different set of results from subjects who had not consumed caffeine. However, since the results of the subject who had not consumed caffeine are not significantly different from those of the other subjects, this claim cannot be analyzed here.
9. Conclusions and Recommendations to the Field

9.1 Conclusions

From the results collected here, it is clear that the objectives to create a fatiguing condition for the users of a BCI system, and to identify a corresponding change in ERP signals, were successfully achieved. In addition, a set of accessible and useful tools for powerful ERP analysis was generated for use in the ubiquitous MATLAB platform and widely distributed.

With the correspondence of the ERP results collected here with those shown in [10], and the notable increases in reported subjected fatigue, there is further evidence of a clear, easily obtained indicator of fatigue within the P300 signals. This indicator is the occurrence of a “checkmark” pattern of P300 amplitudes, in which an initial group of P300 amplitudes is followed by a less positive group of amplitudes, and then a subsequent group of amplitudes which is less positive than the first, but more positive than the second.

This indicating pattern of P300 amplitudes is readily identified among data collected by existing ERP-based BCI systems. It requires only that the system have the capability to identify, store, and analyze P300 amplitudes; these capabilities are present in most, if not all, ERP-based systems.

Due to large variance, it is difficult to reach conclusions with regards to the PVT scores. Like much of the self-reported data, they are presented here as results without analysis or conclusion.
9.2 Recommendations to the Field

The results of this work may provide future BCI work with a powerful means for identifying subjective fatigue without collecting any data beyond that which is already collected by an ERP-based BCI system. It is therefore recommended that future implementations of BCI systems intended for long-term use include the necessary means to identify the “checkmark” pattern of P300 amplitudes first identified by Kececi, and later confirmed here.

The inclusion of detectors of this indicator may allow future BCI implementations to identify users’ fatigue, allowing the systems to adapt or to shut off, providing relief from mental fatigue. Assessment of user engagement [27-28] and hybrid signal processing techniques [29] with independent classifiers that run in parallel and inform the BCI have already shown potential. Also, other neuroimaging techniques such as functional near infrared (fNIR) spectroscopy, which has been shown to identify cognitive workload, [30-31] and for BCI, [32-33] can be integrated with EEG for building improved multi-modal systems [34]. Such adaptation could make BCI systems significantly more user-friendly, ensuring that the technology continues to thrive and develop into a life-changing means of assistive technology.

Future work may attempt to enlarge the scale of this work, with the intention of generating greater statistical significance in all measurements. Future work may also attempt to replicate these results in users with pathologies that may benefit from the use of a brain-computer interface system, as central nervous system disorders may affect the mechanisms underlying central fatigue.
List of References


[14] I. S. Lee, W. A. Bardwell, S. Ancoli-Israel, and J. E. Dimsdale, "Number of lapses during the psychomotor vigilance task as an objective measure of fatigue," *Journal of clinical sleep medicine : JCSM : official publication of the American Academy of Sleep Medicine*, vol. 6, pp. 163-8, Apr 15 2010.


Appendix A: Adapted Chalder Fatigue Inventory

Chalder Fatigue Inventory

Subject ID: _________________________  Date: ________________

1. Do you feel sleepy or drowsy?
   □ Better than usual  □ No more than usual
   □ Worse than usual  □ Much worse than usual

2. Are you lacking in energy?
   □ Better than usual  □ No more than usual
   □ Worse than usual  □ Much worse than usual

3. Do you have difficulty concentrating?
   □ Better than usual  □ No more than usual
   □ Worse than usual  □ Much worse than usual

4. Do you have problems thinking clearly?
   □ Better than usual  □ No more than usual
   □ Worse than usual  □ Much worse than usual

5. How is your memory?
   □ Better than usual  □ No more than usual
   □ Worse than usual  □ Much worse than usual

Session #: ______
Appendix B: FREERP MATLAB Functions

B.1 ERP Quality Assessment GUI

```matlab
function varargout = ERPQuality(varargin)
% ERPQUALITY M-file for ERPQuality.fig
% ERPQUALITY, by itself, creates a new ERPQUALITY or raises the existing
% singleton*.  
% M = ERPQUALITY returns the handle to a new ERPQUALITY or the handle to
% the existing singleton*.
% ERPQUALITY('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in ERPQUALITY.M with the given input arguments.
% ERPQUALITY('Property','Value',...) creates a new ERPQUALITY or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before ERPQuality_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to ERPQuality_OpeningFcn via varargin.
% See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help ERPQuality
% Last Modified by GUIDE v2.5 17-Aug-2012 14:01:12
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct(
    ' gui_Name',       mfilename,
    ' gui_Singleton',  gui_Singleton,
    ' gui_OpeningFcn', @ERPQuality_OpeningFcn,
    ' gui_OutputFcn',  @ERPQuality_OutputFcn,
    ' gui_LayoutFcn',  [] ,
    ' gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before ERPQuality is made visible.
function ERPQuality_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Choose default command line output for ERPQuality
global erpdat
erpdat.start=[];
erpdat.end=[];
erpdat.letters=[];
erpdat.quality=[];
erpdat.cdbak=cd;
erpdat.cdnew=cd;
handles.output = hObject;
set(handles.plotaxes, 'xtick', [], 'ytick', []);
text(.37,.5, 'Select a channel, then choose data to plot.');
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes ERPQuality wait for user response (see UIRESUME)
uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = ERPQuality_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT)
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in rightbutton.
function rightbutton_Callback(hObject, eventdata, handles)
% hObject    handle to rightbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```
global ylims
ylims = ylims;
set(handles.plotaxes,'ylim',get(handles.plotaxes,'ylim')+128);

% Executes on button press in leftbutton.
function leftbutton_Callback(hObject, eventdata, handles)
    % hObject    handle to leftbutton (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global ylims
    ylim(ylims);
    set(handles.plotaxes,'xlim',get(handles.plotaxes,'xlim')+128);
    
    % Executes on button press in lowquality.
    function lowquality_Callback(hObject, eventdata, handles)
        % hObject    handle to lowquality (see GCBO)
        % eventdata  reserved - to be defined in a future version of MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        % Hint: get(hObject,'Value') returns toggle state of lowquality
        if (get(hObject,'Value'))
            set(hObject,'Value',0);
        set(hObject,'Value',1);
    end
    
    % Executes on button press in midquality.
    function midquality_Callback(hObject, eventdata, handles)
        % hObject    handle to midquality (see GCBO)
        % eventdata  reserved - to be defined in a future version of MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        % Hint: get(hObject,'Value') returns toggle state of midquality
        if (get(hObject,'Value'))
            set(hObject,'Value',0);
        set(hObject,'Value',1);
    end
    
    % Executes on button press in highquality.
    function highquality_Callback(hObject, eventdata, handles)
        % hObject    handle to highquality (see GCBO)
        % eventdata  reserved - to be defined in a future version of MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        % Hint: get(hObject,'Value') returns toggle state of highquality
        if (get(hObject,'Value'))
            set(hObject,'Value',0);
        set(hObject,'Value',1);
    end
    
    % Executes on button press in choosedata.
    function choosedata_Callback(hObject, eventdata, handles)
        % hObject    handle to choosedata (see GCBO)
        % eventdata  reserved - to be defined in a future version of MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        global siglen ylims erpdat sigmin sigrange
        cd(erpdat.cdnew);
        [handles.traindatfiles,handles.traindatdir,check]=
            uigetfile(‘*.dat’,[‘Select the BCI2000 P300’...
            ‘ (.dat) training data file(s)’],’multiselect’,’off’);
        cd(erpdat.cdbak);
        if check ~= 0
            erpdat.cdnew=handles.traindatdir;
            set(handles.subject,’string’,’-‘);
            set(handles.run,’string’,’-‘);
            set(handles.saveclose,’enable’,’off’);
            handles.trainfile=handles.traindatfiles(1:length(handles.traindatfiles)-4);
            datastruc.file=[handles.trainfile];
            [signal,parse,state,parms]=getInfo(handles.traindatfiles,handles.traindatdir);
            siglen=length(signal);
            hold off
            plot([0 siglen],[0 0],’color’,[.7 .7 .7],’hittest’,’off’);
            hold on
            plot(signal(:,get(handles.chanbox,’value’)),’k’,’hittest’,’off’);
            title([datastruc.file ‘ Channel ’ num2str(get(handles.chanbox,’value’))]);
            erpdat.savefile=[datastruc.file ’ .mat’]
            xlabel(’Time (s)’);
            ylabel(’Amplitude (uV)’);
            set(gca,’ButtonDownFcn’,{’plotaxes_ButtonDownFcn’,handles});
            % Get target onset times
            flashes=find(state.Flashing(1:end-1)==0 & state.Flashing(2:end)==1);
            targettimes=flashes(1:length(handles.traindatfiles)-4);
            targettimes=flashes(1:length(handles.traindatfiles)-4);
            targetendtimes=targettimes+205;
            erpdat.flashnum=[
            erpdat.start=targettimes;
            erpdat.chan=get(handles.chanbox,’value’);
            erpdat.quality=3*ones(length(targettimes),1);
            erpdat.letters=zeros(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
            erpdat.letters=parse(length(targettimes),1);
            erpdat.save=parse(length(targettimes),1);
            erpdat.plots=erpdat.letters;
            erpdat.end=targetendtimes;
totalletters=length(targettimes)/flashesperletter;  
for i=1:totalletters  
    erpdat.flashnum((i-1)*flashesperletter+1: ...  
        (i-1)*flashesperletter+1: ...  
    end

xlim([0 205]);  
sigmin=min(signal(:,get(handles.chanbox,'Value')));  
sigmax=max(signal(:,get(handles.chanbox,'Value')));  
sigrange=sigmax-sigmin;  
ylimits=[sigmin-.05*sigrange sigmax+.05*sigrange];  
ylim(ylimits);  
set(gca,'xtick',[]);  
set(gca,'ytick',[]);  
set(gca,'xticklabel',num2cell(get(gca,'xtick')));  
set(gca,'yticklabel',num2cell(get(gca,'ytick')));  
extxticklabel='off';  
nextxticklabel='off';  
set(gca,'xtick',[targettimes(i)+60 (sigmax ...  
    'color',8);  
    'linewidth',8);  
n(1:*);  
set(handles.plotaxes,'ylabel',get(handles.chanbox,'Value'));  
end

% --- Executes on button press in saveclose.
function saveclose_Callback(hObject, eventdata, handles)  
% hObject handle to saveclose (see GCBO)  
% eventdata reserved - to be defined in a future version of MATLAB  
% handles structure with handles and user data (see GUIDATA)  
global erpdat  
cd(cdnew);  
cd(erpdat.cdbak);  
save(erpdat.savefile,'output');  
cd(cdnew);  
hold off  
plot([0 0],[0 0],'color','white');  
set(handles.plotaxes,'stick',[],'tick',[],'visible','off');  

% --- Executes on selection change in chanbox.
function chanbox_CreateFcn(hObject, eventdata, handles)  
% hObject handle to chanbox (see GCBO)  
% eventdata reserved - to be defined in a future version of MATLAB  
% handles structure with handles and user data (see GUIDATA)  

% Hint: contents = cellstr(get(hObject,'String')) returns chanbox contents as cell array  
% contents=get(hObject,'Value')) returns selected item from chanbox  

% --- Executes during object creation, after setting all properties.
function chanbox_Callback(hObject, eventdata, handles)  
% hObject handle to chanbox (see GCBO)  
% eventdata reserved - to be defined in a future version of MATLAB  
% handles structure with handles and user data (see GUIDATA)  
% Hint: popupmenu controls usually have a white background on Windows.  
% See IFIG and COMPUTER.  
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))  
set(hObject,'BackgroundColor','white');  
end

% --- Executes on button press in leftfast.
function leftfast_Callback(hObject, eventdata, handles)  
% hObject handle to leftfast (see GCBO)  
% eventdata reserved - to be defined in a future version of MATLAB  
% handles structure with handles and user data (see GUIDATA)  
global ylims  
ylim(ylims);  
set(handles.plotaxes,'ylim',get(handles.plotaxes,'ylim')-512);  

% --- Executes on button press in rightfast.
function rightfast_Callback(hObject, eventdata, handles)  
% hObject handle to rightfast (see GCBO)  
% eventdata reserved - to be defined in a future version of MATLAB  
% handles structure with handles and user data (see GUIDATA)  
global ylims  
ylim(ylims);  
set(handles.plotaxes,'ylim',get(handles.plotaxes,'ylim')+512);  

% --- Executes on button press in endbutton.
function endbutton_Callback(hObject, eventdata, handles)  
% hObject handle to endbutton (see GCBO)  
% eventdata reserved - to be defined in a future version of MATLAB  
% handles structure with handles and user data (see GUIDATA)  
global siglen ylims
global ylimits
ylim(ylimits);
set(handles.plotaxes, 'xlim', [siglen-2050 siglen]);

% Executed on button press in beginning.
function beginning_Callback(hObject, eventdata, handles)
% hObject    handle to beginning (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ylimits
ylim(ylimits);
set(handles.plotaxes, 'xlim', [0 2050]);

% Executed on mouse press over axes background.
function plotaxes_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to plotaxes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ylimits erpdat sigmin sigrange
xy=get(hObject, 'currentpoint');
hold on
xlimits=get(hObject, 'xlim');
diffvec=xy(1,1)-erpdat.start;
nearesterp=find(diffvec<205 & diffvec>0);
if ~isempty(nearesterp)
    if get(handles.lowquality, 'Value')
        if erpdat.plots(nearesterp) ~= 0
            delete(erpdat.plots(nearesterp));
        end
        erpdat.plots(nearesterp)= plot([xy(1,1) xy(1,1)], [ylimits(1) ylimits(2)], 'color', [.75 .2 .45]);
        plot([erpdat.start(nearesterp) erpdat.end(nearesterp)], [sigmin+.005*sigrange sigmin+.005*sigrange], 'color', [.75 .2 .45], 'linewidth', 8);
        xlim(xlimits);
        ylim(ylimits);
        erpdat.quality(nearesterp)=1;
    elseif get(handles.midquality, 'Value')
        if erpdat.plots(nearesterp) ~= 0
            delete(erpdat.plots(nearesterp));
        end
        erpdat.plots(nearesterp)= plot([xy(1,1) xy(1,1)], [ylimits(1) ylimits(2)], 'color', [.86 .73 .27]);
        plot([erpdat.start(nearesterp) erpdat.end(nearesterp)], [sigmin+.005*sigrange sigmin+.005*sigrange], 'color', [.86 .73 .27], 'linewidth', 8);
        xlim(xlimits);
        ylim(ylimits);
        erpdat.quality(nearesterp)=2;
    elseif get(handles.highquality, 'Value')
        if erpdat.plots(nearesterp) ~= 0
            delete(erpdat.plots(nearesterp));
        end
        erpdat.plots(nearesterp)= plot([xy(1,1) xy(1,1)], [ylimits(1) ylimits(2)], 'color', [.1 .7 .3]);
        plot([erpdat.start(nearesterp) erpdat.end(nearesterp)], [sigmin+.005*sigrange sigmin+.005*sigrange], 'color', [.1 .7 .3], 'linewidth', 8);
        xlim(xlimits);
        ylim(ylimits);
        erpdat.quality(nearesterp)=3;
    end
end

function subject_Callback(hObject, eventdata, handles)
% hObject    handle to subject (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if ~isnan(str2double(get(hObject, 'string')))
    if ~isnan(str2double(get(handles.run, 'string')))
        set(handles.saveclose, 'enable','on');
    end
else
    set(handles.saveclose, 'enable','off');
end

% Executed during object creation, after setting all properties.
function subject_CreateFcn(hObject, eventdata, handles)
% hObject    handle to subject (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

% Executed on scroll wheel click while the figure is in focus.
function figure1_WindowScrollWheelFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  structure with the following fields (see FIGURE)
%   VerticalScrollCount: signed integer indicating direction and number of clicks
% handles    structure with handles and user data (see GUIDATA)
% Hints: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if eventdata.VerticalScrollCount > 0
    set(handles.plotaxes, 'ylim', xlims(128));
else
    eventdata.VerticalScrollCount < 0
end
set(handles.plotaxes,'xlim',xlims-128);

% Executes on key press with focus on figure1 and none of its controls.
function figure1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  structure with the following fields (see FIGURE)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
% handles   structure with handles and user data (see GUIDATA)
if eventdata.Character=='1'
    set(handles.lowquality,'value',1);
    lowquality_Callback(handles.lowquality, eventdata, handles);
elseif eventdata.Character=='2'
    set(handles.midquality,'value',1);
    midquality_Callback(handles.midquality, eventdata, handles);
elseif eventdata.Character=='3'
    set(handles.highquality,'value',1);
    highquality_Callback(handles.highquality, eventdata, handles);
elseif strcmp(eventdata.Key,'rightarrow') || strcmp(eventdata.Key,'c')
    rightfast_Callback(handles.rightfast, eventdata, handles);
elseif strcmp(eventdata.Key,'leftarrow') || strcmp(eventdata.Key,'d')
    leftfast_Callback(handles.leftfast, eventdata, handles);
end

function run_Callback(hObject, eventdata, handles)
% hObject    handle to run (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of run as text
%        str2double(get(hObject,'String')) returns contents of run as a double
if ~isnan(str2double(get(hObject,'string'))) && ~isnan(str2double(get(handles.subject,'string')))
    set(handles.saveclose,'enable','on');
else
    set(handles.saveclose,'enable','off');
end

% Executes during object creation, after setting all properties.
function run_CreateFcn(hObject, eventdata, handles)
% hObject    handle to run (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Code created 8.9.12 by Vince Petaccio for use in assessing the quality of BCI2000 ERP data.

B.2 Quality Data Consolidation GUI

function varargout = UseQuality(varargin)
% USEQUALITY M-file for UseQuality.fig
% USEQUALITY, by itself, creates a new USEQUALITY or raises the existing singleton.
% H = USEQUALITY returns the handle to a new USEQUALITY or the handle to the existing singleton.
% USEQUALITY('CALLBACK',hObject,eventData,handles,...) calls the local function named CALLBACK in USEQUALITY.M with the given input arguments.
% USEQUALITY('Property','Value',...) creates a new USEQUALITY or raises the existing singleton. Starting from the left, property value pairs are applied to the GUI before UseQuality_OpeningFcn gets called. An unrecognized property name or invalid value makes property application stop. All inputs are passed to UseQuality_OpeningFcn via varargin.
% See GUIDE Options on GUIDE's Tools menu. Choose "GUI allows only one instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIDATA
% Edit the above text to modify the response to help UseQuality
% Last Modified by GUIDE v2.5 15-Aug-2012 21:53:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @UseQuality_OpeningFcn, ...
    'gui_OutputFcn', @UseQuality_OutputFcn, ...
    'gui_LayoutFcn', [], ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargin
    varargin{1:nargin} = gui_mainfcn(gui_State, varargin{:});
end
else
  gui_mainfcn(gui_State, varargin{:});
end  % End initialization code - DO NOT EDIT

% DO NOT EDIT
% Executes just before UseQuality is made visible.
function UseQuality_OpeningFcn(hObject, eventdata, handles, varargin)
  % This function has no output args, see OutputFcn.
  hObject  % handle to figure
  eventdata reserved - to be defined in a future version of MATLAB
  varargin command line arguments to UseQuality (see VARARGIND)

% Choose default command line output for UseQuality
global qualdata
handles.output = hObject;
qualdata,thisqual=zeros(1,3);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes UseQuality wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% Outputs from this function are returned to the command line.
function varargout = UseQuality_OutputFcn(hObject, eventdata, handles)
  % varargout  cell array for returning output args (see VARARGOUT);
  hObject    handle to figure
  eventdata   reserved - to be defined in a future version of MATLAB
  handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

% Executes on selection change in consolidatelist.
function consolidatelist_Callback(hObject, eventdata, handles)
  % hObject    handle to consolidatelist (see GCBO)
  eventdata reserved - to be defined in a future version of MATLAB
  handles    structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns consolidatelist contents as cell array
%        contents{get(hObject,'Value')} returns selected item from consolidatelist

% Executes during object creation, after setting all properties.
function consolidatelist_CreateFcn(hObject, eventdata, handles)
  % hObject    handle to consolidatelist (see GCBO)
  eventdata reserved - to be defined in a future version of MATLAB
  handles    empty - handles not created until after all CreateFcns called
% Hint: listbox controls usually have a white background on Windows.
  % See ISPC and COMPUTER.
  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
  end

% Executes on button press in creatematrix.
function creatematrix_Callback(hObject, eventdata, handles)
  % hObject    handle to creatematrix (see GCBO)
  eventdata reserved - to be defined in a future version of MATLAB
  handles    structure with handles and user data (see GUIDATA)
  global qualdata
  [qualfile,qualpath,check]=uiputfile('ERPQualityMatrix.m');
  if (check)
    QualityData=qualdata.consol;  %#ok<NASGU>
    save([qualpath qualfile],'QualityData');
    set(handles.consolidatelist,'string','');
    set(hObject,'enable','off');
  end

% Executes on button press in getqualdat1.
function getqualdat1_Callback(hObject, eventdata, handles)
  % hObject    handle to getqualdat1 (see GCBO)
  eventdata reserved - to be defined in a future version of MATLAB
  handles    structure with handles and user data (see GUIDATA)
  global qualdata
  [datfiles,datdir,check]=uigetfile('multiselect','on');
  if (check)
    set(handles.consolidatelist,'string',datfiles);
    qualdata.consol=[];
    if length(datfiles) > 1
      for i=1:length(datfiles)
        data=load(fullfile(datadir datfiles(i)));
        qualdata.consol=[qualdata.consol; data.output];
      end
    else
      data=load(fullfile(datadir datfiles));
      qualdata.consol=data.output;
    end
    set(handles.creatematrix,'enable','on');
    else
    set(handles.creatematrix,'enable','off');
  end

% Code created 8.11.12 by Vince Petaccio for use in assessing the quality
% of BCI2000 ERP data.
B.3 ERP Selection Function

```matlab
function erpdata=erpselect(erpmatrix,subjects,runs,channels,letters,flashes,quality,normalize)
% Inputs are vectors.
% Use a zero as the input for a parameter where all values should be
% included in the output data matrix
if subjects > 0
    erpdata5=[];
    for i=1:length(subjects)
        erpdata5=[erpdata5; erpmatrix(erpmatrix(:,1)==subjects(i),:)]; %#ok<AGROW>
    end
else
    erpdata5=erpmatrix;
end
if runs > 0
    erpdata4=[];
    for j=1:length(runs)
        erpdata4=[erpdata4; erpdata5(erpdata5(:,2)==runs(j),:)]; %#ok<AGROW>
    end
else
    erpdata4=erpdata5;
end
if channels > 0
    erpdata3=[];
    for k=1:length(channels)
        erpdata3=[erpdata3; erpdata4(erpdata4(:,3)==channels(k),:)]; %#ok<AGROW>
    end
else
    erpdata3=erpdata4;
end
if letters > 0
    erpdata2=[];
    for m=1:length(letters)
        erpdata2=[erpdata2; erpdata3(erpdata3(:,4)==letters(m),:)]; %#ok<AGROW>
    end
else
    erpdata2=erpdata3;
end
if flashes > 0
    erpdata1=[];
    for n=1:length(flashes)
        erpdata1=[erpdata1; erpdata2(erpdata2(:,5)==flashes(n),:)]; %#ok<AGROW>
    end
else
    erpdata1=erpdata2;
end
if quality > 0
    erpdatablastoff=[];
    for p=1:length(quality)
        erpdatablastoff=[erpdatablastoff; erpdata1(erpdata1(:,7)==quality(p),:)]; %#ok<AGROW>
    end
else
    erpdatablastoff=erpdata1;
end
if (normalize)
    erpdata=erpdatablastoff(:,8:212);
    erpdata=erpdata./repmat(max(erpdata')',1,205); %#ok<UDIM>
else
    erpdata=erpdatablastoff(:,8:212);
end
% Code created 8.12.12 by Vince Petaccio for use in assessing the quality
% of BC1200 ERP data.
```

B.4 ERP Analysis GUI

```matlab
function varargout = ERPAnalysis(varargin)
% Main Functions
% ERPANALYSIS M-file for ERPAnalysis.fig
% ERPANALYSIS, by itself, creates a new ERPANALYSIS or raises the existing
% singleton*.
% H = ERPANALYSIS returns the handle to a new ERPANALYSIS or the handle to
% the existing singleton*.  Starting from the left, property value pairs are
% applied to the GUI before ERPANALYSIS_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to ERPANALYSIS_OpeningFcn via varargin.
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
```
% instance to run (singleton)"
% See also: GUIDE, GUIDATA, GUIDATAS
% Last Modified by GUIDE v2.5 10-Sep-2012 06:32:04
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename,...
                 'gui_Singleton', gui_Singleton,...
                 'gui_OpeningFcn', ERPAnalysis_OpeningFcn,...
                 'gui_OutputFcn', ERPAnalysis_OutputFcn,...
                 'gui_LayoutFcn', [], ...
                 'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
%---
% Executes just before ERPAnalysis is made visible.
function ERPAnalysis_OpeningFcn(hObject, ~, handles, varargin)
    % This function has no output args, see OutputFcn.
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % varargin    command line arguments to ERPAnalysis (see VARARGIN)
    % Choose default command line output for ERPAnalysis
    global range rangeb rangebt bline eline ptype strbk datastruc fval ...
    plotbak plotbak=1;
    fval=1;
    handles.windlen=[0 800];
    handles.SF=1;
    handles.rndsmp=100;
    handles.output = hObject;
    range=1;
    handles.plot=0;
    rangeb=1;
    rangebt=0;
    bline=0;
    eline=0;
    ptype='';
    strbk='';
    datastruc.row=1;
    datastruc.array={};
    % Update handles structure
    guidata(hObject, handles);
    % UIWAIT makes ERPAnalysis wait for user response (see UIRESUME)
    uiwait(handles.figure1);
    %---
    % Outputs from this function are returned to the command line.
    function varargout = ERPAnalysis_OutputFcn(~, ~, handles)
    % varargout cell array for returning output args (see VARARGOUT)
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % Get default command line output from handles structure
    varargout{1} = handles.output;
    %---
    % Executes on selection change in chanbox.
    function chanbox_Callback(hObject, ~, handles)
    % hObject    handle to chanbox (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % Hints: contents = cellstr(get(hObject,'String')) returns chanbox contents as cell array
    %        contents{get(hObject,'Value')} returns selected item from chanbox
    set(handles.gobutton,'enable','on');
    %---
    % Executes during object creation, after setting all properties.
    function chanbox_CreateFcn(hObject, ~, handles)
    % hObject    handle to chanbox (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % Hints: contents = cellstr(get(hObject,'String')) returns chanbox contents as cell array
    %        contents{get(hObject,'Value')} returns selected item from chanbox
    set(handles.gobutton,'enable','on');
    %---
    % Executes during object creation, after setting all properties.
    function chanbox_CreateFcn(hObject, ~, handles)
    % hObject    handle to chanbox (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % Hints: contents = cellstr(get(hObject,'String')) returns chanbox contents as cell array
    %        contents{get(hObject,'Value')} returns selected item from chanbox
    set(handles.gobutton,'enable','on');
    %---
    % Executes on button press in choosedata.
    function choosedata_Callback(hObject, ~, handles)
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
if check ~= 0
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
if (gooddata)
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
else
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
end
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% hObject    handle to choosedata (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB

set(handles.trainfiletxt, 'String', ...
    ['Target Responses for ' handles.ermatrix]);
ptype=1;

function clearaxis(handles)
global eline bline
set(handles.exportdata, 'enable', 'off');
eline=0;
bline=0;
cla(handles.plot=0;
set(handles.p300axes,'xtick',[],'ytick',[]);
title(gca,'');
xlabel(gca,'');
ylabel(gca,'');
set(handles.brange,'string','');
set(handles.erange,'string','');

% --- Executes on button press in screenshot.
function screenshot_Callback(~, ~, ~)
% hObject    handle to screenshot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[fname, path, index] = uiputfile({
    '*.jpg', '*.jpg JPEG Image';
    '*.bmp', '*.bmp Bitmap Image';
    '*.tiff', '*.tiff Uncompressed TIFF Image'},...
    'Save as';
if index ~= 0
    set(gcf, 'InvertHardcopy', 'off');
    set(gcf, 'PaperPositionMode', 'auto');
    format = { 
    '-djpeg100', '-dbmp', '-dtiffn'};
    print(gcf, format{index}, '-r0', [path name]);
end

% --- Executes on mouse press over axes background.
function p300axes_ButtonDownFcn(hObject, ~, handles)
% hObject    handle to p300axes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global data range xp bline eline ptype datastruc fval
if (handles.plot)
    xy = get(hObject, 'currentpoint');
    x = round(xy(1,1));
    xl = get(gca, 'xlim');
    if x > xl(1) && x < xl(2)
        ylimits = get(gca, 'ylim');
        if (range)
            % First click in selection
            set(handles.spreadsheet, 'enable', 'off');
            if bline ~= 0
                delete(bline);
                bline = 0; % #ok<*NASGU>
            end
            if eline ~= 0
                delete(eline);
                eline = 0;
            end
            hold on
            bline = line([x x], [ylimits(1) ylimits(2)], 'color', [.65 .75 .65], ...
            'HitTest', 'off');
            hold off
            set(handles.brange, 'string', '');
            set(handles.erange, 'string', 'Select end of range.');
            set(handles.spreadsheet, 'enable', 'on');
            hold on
            eline = line([x x], [ylimits(1) ylimits(2)], 'color', [.75 .65 .65], ...
            'HitTest', 'off');
            hold off
            set(handles.brange, 'string', 'Select beginning of range.');
            range = 1;
            hold on
            eline = line([x x], [ylimits(1) ylimits(2)], 'color', [.65 .75 .65], ...
            'HitTest', 'off');
            hold off
            end
        end
    end
end
else % Second click in range selection
    set(handles.spreadsheet, 'enable', 'on');
    x2 = cellstr([x(2)], 'length(mp)');
    % Determine index of xvalue
datastruc.startlat = x2(1);
    set(handles.spreadsheet, 'enable', 'on');
    set(handles.erange, 'string', 'Select beginning of range.');
    range = 1;
    hold on
    eline = line([x x], [ylimits(1) ylimits(2)], 'color', [.75 .65 .65], ...
    'HitTest', 'off');
    hold off
    end
end

% Use old-version names for datastruc values
range = round(1000*mean(datastruc.startlat))/1000;
datastruc.startlat = range;
datastruc.endlat = range;
datastruc.startval = range;
datastruc.endval = range;

% Max/Min Latencies
maxloc = round((xl(2)*(datastruc.startlat + find(rdata==max(rdata)))/length(xp)));
set(handles.lpeak, 'string', ['Peak Latency: ' datastruc.peaklat ' ms']);
range = round(1000*median(rdata))/1000;
datastruc.median = range;
datastruc.startlat = range;
datastruc.endlat = range;
datastruc.startval = range;
datastruc.endval = range;

% Use old-version names for datastruc values
range = round(1000*max(rdata))/1000;
datastruc.startlat = range;
datastruc.endlat = range;
datastruc.startval = range;
datastruc.endval = range;

% Peak/Max Latency
range = round(1000*median(rdata))/1000;
datastruc.median = range;
datastruc.startlat = range;
datastruc.endlat = range;
datastruc.startval = range;
datastruc.endval = range;

% Use old-version names for datastruc values
range = round(1000*max(rdata))/1000;
datastruc.startlat = range;
datastruc.endlat = range;
datastruc.startval = range;
datastruc.endval = range;
```matlab
datastruc.peakval=rangemax;
rangemin=round(1000*min(rdata))/1000;
datastruc.endval=rangemin;
if (get(hObject,'string')=='Rows')
datastruc.type='Response';
set(handles.amp, 'string', ['Range Min= ' num2str(rangemin)]);
set(handles.amp, 'Value', num2str(rangemin));
set(handles.ramp, 'string', ['Range Median= ' num2str(rangemedian)]);
set(handles.ramp, 'Value', num2str(rangemedian));
set(handles.peak, 'string', ['Range Peak= ' num2str(rangemax)]);
else
    datastruc.type='Response';
    set(handles.amp, 'string', ['Range Min= ' num2str(rangemin) ' microV']);
    set(handles.ramp, 'string', ['Range Median= ' num2str(rangemedian) ' microV']);
    set(handles.peak, 'string', ['Range Peak= ' num2str(rangemax) ' microV']);
end
else
    % Second selection is earlier than first - exchange values
    x2bak=x2;
latabak=datastruc.startlat;
x2=latabak;
datastruc.startlat=x2bak;
rdata=datastruc.startlat:x2;
end
    % Max/Min Latencies
    % Peak Value Latency
    maxloc=round(max(rdata));
datastruc.peaklat=num2str(rangemean);
datastruc.peakval=rangemax;
    rangemax=round(1000*max(rdata))/1000;
    datastruc.endval=rangemin;
    rangemin=round(1000*min(rdata))/1000;
datastruc.endlat=num2str(rangemean);
datastruc.startlat=num2str(rangemin);
datastruc.peaklat=num2str(rangemean);
datastruc.peakval=num2str(rangemin);
    datastruc.type='Responses';
    set(handles.amp, 'string', ['Range Min= ' num2str(rangemin)]);
    set(handles.amp, 'Value', num2str(rangemin));
    set(handles.ramp, 'string', ['Range Median= ' num2str(rangemedian)]);
    set(handles.ramp, 'Value', num2str(rangemedian));
    set(handles.peak, 'string', ['Range Peak= ' num2str(rangemax)]);
    set(handles.peak, 'Value', num2str(rangemax));
else
    datastruc.type='Response';
    set(handles.amp, 'string', ['Range Min= ' num2str(rangemin) ' microV']);
    set(handles.amp, 'Value', num2str(rangemin) ' microV');
    set(handles.ramp, 'string', ['Range Median= ' num2str(rangemedian) ' microV']);
    set(handles.ramp, 'Value', num2str(rangemedian) ' microV');
    set(handles.peak, 'string', ['Range Peak= ' num2str(rangemax) ' microV']);
end
end
```

tempstruc=datastruc;
end
datastruc.array(datastruc.rows,:)=struct2cell(tempstruc);
datastruc.startlat=str2double(datastruc.startlat);
datastruc.endlat=str2double(datastruc.endlat);

% --- Executes on button press in commit.
function commit_Callback(hObject, ~, handles)
% hObject    handle to commit (see GCBO)
% eventdata  reserved
% handles    structure with handles and user data (see GUIDATA)
global datastruc
set(hObject, 'enable', 'off');

handlevector=[handles.choosedata handles.chanbox handles.spatialfilter ...
handles.plottype handles.gobutton handles.screenshot];
set(handlevector, 'enable', 'off');
strbk=get(handles.status, 'string');
pause(.05);
filecheck=exist(fullfile('ERP Data.xls'));
%#ok<*EXIST>

ordervec=[1 8 2 3 6 4 7 5 9];
if filecheck==2 %The ERP Data spreadsheet already exists in the cd
set(handles.status, 'string', 'Saving data to spreadsheet...');
pause(.15);
arrh=size(datastruc.array);
for i=1:9
    data2add(:,i)=datastruc.array(:,ordervec(i));
    [~,~,raw]=xlsread(fullfile('ERP Data.xlsx'));
writedata=vertcat(raw,data2add);
xlswrite(fullfile('ERP Data.xlsx'),writedata);
end
else %Make a new ERP Data spreadsheet
    set(handles.status, 'string', 'Saving data to a new sheet...');
    pause(.15);
    arrs=size(datastruc.array);
    writedata=cell(arrs(1),9);
    for i=1:9
        writedata(:,i)=datastruc.array(:,ordervec(i));
        xlswrite(fullfile('ERP Data.xlsx'),writedata);
    end
    datastruc=struct;
    datastruc.rows=1;
    datastruc.array={};
    datastruc.file=[handles.trainfile];
datastruc.array{1,1}={'Filename' 'Range Mean' 'Range Median' 'Peak Latency' ...
'Min Latency' 'Peak Value' 'Range Min' 'Data Type' 'Channels Analyzed'};
set(handlevector, 'enable', 'on');
set(handles.status, 'string', strbk);

% --- Executes on button press in exportdata.
function exportdata_Callback(~, ~, ~)
% hObject    handle to exportdata (see GCBO)
% eventdata  reserved
% handles    structure with handles and user data (see GUIDATA)
global gp data
if (size(gp) ~= (size(data)))
data=data';
end
exportdata=[gp',data'];
name, path, index = uiputfile(...
    '*.xls', '*.xls  Excel '97-''03 Spreadsheet';
    '*.txt', '*.txt  Comma-Separated Values'; ...
    'Save as');
if index ~= 0
    if index == 1
        aswrite([path name '.xls'],exportdata);
    else
        csvwrite([path name '.csv'],exportdata);
    end
end

%Version 2.11.11
%Created by Vince Petaccio at Drexel University College of Medicine,
%Department of Neurology and the ALS Hope Foundation using adaptations of
%work done by the Wadsworth Center of the New York State Department of
%Health for use in analyzing EEG data collected using BCI2000.

% --- Executes on selection change in subbox.
function subbox_Callback(hObject, eventdata, handles)
% hObject    handle to subbox (see GCBO)
% eventdata  reserved
% handles    structure with handles and user data (see GUIDATA)

datastruc=[4 2 1 2 3 4 5 2 9];
datastruc.array{1,1}={'Filename' 'Channel Mean' 'Channel Median' 'Peak Latency' ...
'Min Latency' 'Peak Value' 'Range Min' 'Data Type' 'Channels Analyzed'};
set(handles.status, 'string', 'Saving data to a new sheet');
pause(.15);
arrh=size(datastruc.array);
for i=1:9
    data2add(:,i)=datastruc.array(:,ordervec(i));
    [~,~,raw]=xlsread(fullfile('ERP Data.xlsx'));
writedata=vertcat(raw,data2add);
xlswrite(fullfile('ERP Data.xlsx'),writedata);
end

%HINT: contents = cellstr(get(hObject,'String')) returns subbox contents as cell array
%contents=get(hObject,'Value') returns selected item from subbox

% --- Executes during object creation, after setting all properties.
function subbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to subbox (see GCBO)
% eventdata  reserved
% handles    empty - handles not created until after all CreateF ons called

% Hint: Listbox controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end
% --- Executes on selection change in flashbox.
function flashbox_Callback(hObject, eventdata, handles)
    hObject    handle to flashbox (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    structure with handles and user data (see GUIDATA)
    % Hints: contents = cellstr(get(hObject,'String')) returns flashbox contents as cell array
    % contents(get(hObject,'Value')) returns selected item from flashbox

% --- Executes during object creation, after setting all properties.
function flashbox_CreateFcn(hObject, eventdata, handles)
    hObject    handle to flashbox (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    structure with handles and user data (see GUIDATA)
    % Hints: listbox controls usually have a white background on Windows.
    See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

% --- Executes on selection change in letbox.
function letbox_Callback(hObject, eventdata, handles)
    hObject    handle to letbox (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    structure with handles and user data (see GUIDATA)
    % Hints: contents = cellstr(get(hObject,'String')) returns letbox contents as cell array
    % contents(get(hObject,'Value')) returns selected item from letbox

% --- Executes during object creation, after setting all properties.
function letbox_CreateFcn(hObject, eventdata, handles)
    hObject    handle to letbox (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    empty - handles not created until after all CreateFcns called
    % Hint: listbox controls usually have a white background on Windows.
    See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

% --- Executes on selection change in runbox.
function runbox_Callback(hObject, eventdata, handles)
    hObject    handle to runbox (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    structure with handles and user data (see GUIDATA)
    % Hints: contents = cellstr(get(hObject,'String')) returns runbox contents as cell array
    % contents(get(hObject,'Value')) returns selected item from runbox

% --- Executes during object creation, after setting all properties.
function runbox_CreateFcn(hObject, eventdata, handles)
    hObject    handle to runbox (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    empty - handles not created until after all CreateFcns called
    % Hint: listbox controls usually have a white background on Windows.
    See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

% --- Executes on button press in hiqual.
function hiqual_Callback(hObject, eventdata, handles)
    hObject    handle to hiqual (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    structure with handles and user data (see GUIDATA)
    % Hint: get(hObject,'Value') returns toggle state of hiqual

% --- Executes on button press in medqual.
function medqual_Callback(hObject, eventdata, handles)
    hObject    handle to medqual (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    structure with handles and user data (see GUIDATA)
    % Hint: get(hObject,'Value') returns toggle state of medqual

% --- Executes on button press in lowqual.
function lowqual_Callback(hObject, eventdata, handles)
    hObject    handle to lowqual (see GCBO)
    eventdata  reserved - to be defined in a future version of MATLAB
    handles    structure with handles and user data (see GUIDATA)
    % Hint: get(hObject,'Value') returns toggle state of lowqual

% ---- Executing on selection change in flashbox.
function [range2,plotdata]=plotresponse(erpmatrix,handles)
    dat.subjects=str2num(get(handles.subbox,'string'));
    dat.subjects=dat.subjects(get(handles.subbox,'value'));
    dat.runs=str2num(get(handles.runbox,'string'));
    dat.runs=dat.runs(get(handles.runbox,'value'));

dat.channels=str2num(get(handles.chanbox,'string'));
dat.channels=dat.channels(get(handles.chanbox,'value'));
dat.letters=str2num(get(handles.letbox,'string'));
dat.letters=dat.letters(get(handles.letbox,'value'));
dat.flashes=str2num(get(handles.flashbox,'string'));
dat.flashes=dat.flashes(get(handles.flashbox,'value'));
dat.quality=[get(handles.lowqual,'value') 2*get(handles.medqual,'value') ...
3*get(handles.hiqual,'value')];
dat.quality(dat.quality==0)=[];
range2=0:204;
%Calculate EEG Responses
plotdata=mean(erpselect(erpmatrix,dat.subjects,dat.runs,dat.channels,
...dat.letters,dat.flashes,dat.quality,get(handles.norm,'value')));% mean of trials for each condition
xvals=1000*(0:length(plotdata)-1)/256;
range2=xvals;
% Plot!
zeroline=plot(range2,zeros(1,length(range2)),'color',[.9 .9 .9]);
hold on
if (get(handles.plotspectral,'value'))
    plotdata=ERPfreqmet(plotdata,get(handles.spectral,'value')+4);
dataline=plot(range2,plotdata,'b');
else
dataline=plot(range2,plotdata,'b');
end
hold off
set(zeroline,'HitTest','off');
set(dataline,'HitTest','off');

% --- Executes on button press in norm.
function norm_Callback(hObject, eventdata, handles)
% hObject    handle to norm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns toggle state of norm

% --- Executes on button press in yflip.
function yflip_Callback(hObject, eventdata, handles)
% hObject    handle to yflip (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (get(hObject,'Value'))
    set(gca,'YDir','reverse');
else
    set(gca,'YDir','normal');
end

% --- Executes on selection change in spectral.
function spectral_Callback(hObject, eventdata, handles)
% hObject    handle to spectral (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns spectral contents as cell array
%        contents{get(hObject,'Value')} returns selected item from spectral

function spectral_CreateFcn(hObject, eventdata, handles)
% hObject    handle to spectral (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hints: listbox controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in plotspectral.
function plotspectral_Callback(hObject, eventdata, handles)
% hObject    handle to plotspectral (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns toggle state of plotspectral

function pBand=ERPfreqmet(datain,process)
% Values for process:
% 1-4: Algorithms 1-4 from Jap et al
% 5-8: Percent band power for delta, theta, alpha, and beta respectively
if length(datain) >= 100
    for i=floor(length(datain)/25)
        data=datain((1+(i-1)*25):(25*i));
        fs=256;
        [p,f]=pspectrum(data,fs);
        calculate the FFT
        p=fft(data);
        p=abs(p(1:round(length(p)/2)));
        calculate the Power Spectrum
    end
    calculate
end

PSD=(\textit{p}^2)/(\textit{L}/2)^2; 

%perform a 3-point smoothing on the PSD
p=PSD;
for \textit{i}=3:((\textit{L}/2)-2)
\textit{p}(\textit{i})=\text{sum}(\text{PSD}(\textit{i}-1:i+1))/3;
end
PSD=p;

%calculate the locations of the frequency bins
\textit{f}=0:((\textit{fs}/64)-((\textit{fs}/L)));

%get Total Power for 0 - 30 Hz (excluding 0 Hz)
\text{TotalPower}=(\text{sum}(\text{PSD}(\text{tp}_{\text{band}})))

switch process
  case 1
    % algorithm 1 from Jap et al
    \text{pBand}=(\text{theta} + \text{alpha})/\text{beta};
    \text{t}_{\text{band}}=(\text{f}=4)(f<8);
    a_{\text{Theta}}=\text{sum}(\text{PSD}(\text{t}_{\text{band}}));
    a_{\text{band}}=(\text{f}=8)(f<13);
    a_{\text{Alpha}}=\text{sum}(\text{PSD}(\text{a}_{\text{band}}));
    b_{\text{band}}=(\text{f}=13)(f<30);
    a_{\text{Beta}}=\text{sum}(\text{PSD}(\text{b}_{\text{band}}));
    \text{pBand}(\text{ii})=(a_{\text{Theta}}+a_{\text{Alpha}})/a_{\text{Beta}};
  case 2
    % algorithm 2 from Jap et al
    \text{pBand}=(\text{alpha})/\text{beta};
    a_{\text{band}}=(\text{f}=8)(f<13);
    a_{\text{Alpha}}=\text{sum}(\text{PSD}(\text{a}_{\text{band}}));
    b_{\text{band}}=(\text{f}=13)(f<30);
    a_{\text{Beta}}=\text{sum}(\text{PSD}(\text{b}_{\text{band}}));
    \text{pBand}(\text{ii})=a_{\text{Alpha}}/a_{\text{Beta}};
  case 3
    % algorithm 3 from Jap et al
    \text{pBand}=(\text{theta} + \text{alpha})/(\text{alpha} + \text{beta});
    \text{t}_{\text{band}}=(\text{f}=4)(f<8);
    a_{\text{Theta}}=\text{sum}(\text{PSD}(\text{t}_{\text{band}}));
    a_{\text{band}}=(\text{f}=8)(f<13);
    a_{\text{Alpha}}=\text{sum}(\text{PSD}(\text{a}_{\text{band}}));
    b_{\text{band}}=(\text{f}=13)(f<30);
    a_{\text{Beta}}=\text{sum}(\text{PSD}(\text{b}_{\text{band}}));
    \text{pBand}(\text{ii})=(a_{\text{Theta}}+a_{\text{Alpha}})/(a_{\text{Alpha}}+a_{\text{Beta}});
  case 4
    % algorithm 4 from Jap et al
    \text{pBand}=(\text{theta})/\text{beta};
    \text{t}_{\text{band}}=(\text{f}=4)(f<8);
    a_{\text{Theta}}=\text{sum}(\text{PSD}(\text{t}_{\text{band}}));
    a_{\text{band}}=(\text{f}=8)(f<13);
    a_{\text{Alpha}}=\text{sum}(\text{PSD}(\text{a}_{\text{band}}));
    b_{\text{band}}=(\text{f}=13)(f<30);
    a_{\text{Beta}}=\text{sum}(\text{PSD}(\text{b}_{\text{band}}));
    \text{pBand}(\text{ii})=a_{\text{Theta}}/a_{\text{Beta}};
  case 5
    % get percent Delta
    d_{\text{band}}=(\text{f}=0)(f<4);
    a_{\text{Delta}}=\text{sum}(\text{PSD}(\text{d}_{\text{band}}));
    \text{pBand}(\text{ii})=100*a_{\text{Delta}}/\text{TotalPower};
  case 6
    % get percent Theta
    t_{\text{band}}=(\text{f}=4)(f<8);
    a_{\text{Theta}}=\text{sum}(\text{PSD}(\text{t}_{\text{band}}));
    \text{pBand}(\text{ii})=100*a_{\text{Theta}}/\text{TotalPower};
  case 7
    % get percent Alpha
    a_{\text{band}}=(\text{f}=8)(f<13);
    a_{\text{Alpha}}=\text{sum}(\text{PSD}(\text{a}_{\text{band}}));
    \text{pBand}(\text{ii})=100*a_{\text{Alpha}}/\text{TotalPower};
  case 8
    % get percent Beta
    b_{\text{band}}=(\text{f}=13)(f<30);
    a_{\text{Beta}}=\text{sum}(\text{PSD}(\text{b}_{\text{band}}));
    \text{pBand}(\text{ii})=100*a_{\text{Beta}}/\text{TotalPower};
end

% Code created 8.30.12 by Vince Petaccio for use in the analysis of ERP
% data analyzed using the PREERP analysis package
Appendix C: Detailed Protocol

<table>
<thead>
<tr>
<th>Task</th>
<th>Approximate Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screening Interview</td>
<td>5 min</td>
</tr>
<tr>
<td>PVT and Fatigue Questionnaire</td>
<td>10 min</td>
</tr>
<tr>
<td>BCI Setup</td>
<td>20 min</td>
</tr>
<tr>
<td>PVT and Fatigue Questionnaire</td>
<td>10 min</td>
</tr>
<tr>
<td>BCI Protocol-First Half</td>
<td>25 min</td>
</tr>
<tr>
<td>PVT and Fatigue Questionnaire</td>
<td>10 min</td>
</tr>
<tr>
<td>BCI Protocol-Second Half</td>
<td>25 min</td>
</tr>
<tr>
<td>PVT and Fatigue Questionnaire</td>
<td>10 min</td>
</tr>
</tbody>
</table>

Total: 1h 45m
Appendix D: Screening Interview Questionnaire

Screening Interview

Subject ID: __________________________       Date: _________________

How many hours of sleep did you get last night?

0  5  10+

How was the quality of your sleep last night?

Poor  Adequate  Perfect

Did you eat or drink any caffeinated products in the last 24 hours?

☐ Yes  ☐ No  ☐ Don’t Remember

Have you ever been diagnosed with any form of clinical depression?

☐ Yes  ☐ No

What amount of experience do you have with using a brain-computer interface (BCI) of any type?

None  Some  Substantial