UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

ANDRÉ GUASTI LOZER

INTERFACE PARA PROCESSAMENTO DIGITAL DE SINAIS ELETROENCEFALOGRÁFICOS

VITÓRIA – ES
DEZEMBRO/2015

ANDRÉ GUASTI LOZER

INTERFACE PARA PROCESSAMENTO DIGITAL DE SINAIS ELETROENCEFALOGRÁFICOS

Parte manuscrita do Projeto de Graduação do aluno **André Guasti Lozer**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Profa. Dra. Eliete Maria de Oliveira Caldeira

Coorientador:
Dr. Javier Ferney Castillo García

VITÓRIA – ES
DEZEMBRO/2015

ANDRÉ GUASTI LOZER

INTERFACE PARA PROCESSAMENTO DIGITAL DE SINAIS ELETROENCEFALOGRÁFICOS

Parte manuscrita do Projeto de Graduação do aluno **André Guasti Lozer**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovada em 22 de dezembro de 2015.

COMISSÃO EXAMINADORA:

_____

Profa. Dra. Eliete Maria de Oliveira Caldeira
Universidade Federal do Espírito Santo
Orientadora

_____

Prof. Dr. André Ferreira
Universidade Federal do Espírito Santo
Examinador

_____

Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo
Examinador

RESUMO

O objetivo do presente Projeto de Graduação é a implementação de uma ferramenta computacional com capacidade de auxiliar ao estudo de sinais biológicos, em ambiente acadêmico e clínico, através da implementação de funções de condicionamento de sinal e análises matemáticas com representações visuais e numéricas que possam fornecer informações úteis para melhor aplicar o conceito de processamento digital de sinais de forma acessível. Este projeto se destina a estudantes e pesquisadores das áreas de Engenharia Elétrica, Engenharia Biomédica e de Computação e outros interessados em Processamento Digital de Sinais biológicos, mais especificamente para sinais cerebrais. O desenvolvimento desta ferramenta surgiu da necessidade de estudantes da área biomédica (fisioterapeutas, biólogos, farmacêuticos, enfermeiros, etc.) de realizarem processamentos de sinais coletados em suas pesquisas no LAI (Laboratório de Automação Inteligente) e de dependerem grandemente dos estudantes de graduação e pós-graduação em Engenharia para realizar estas tarefas. O *software* desenvolvido recebe sinais de EEG divididos em canais, armazenados em matrizes em arquivo do MatLab®, podendo tanto as linhas quanto as colunas da matriz representar os canais, e é compatível com dados provenientes de equipamentos de aquisição de EEG, desde que os sinais estejam armazenados em arquivos no formato **.mat* (formato proprietário do MatLab® utilizado para armazenar dados matriciais). Dados provenientes da plataforma Emotiv foram utilizados para os testes, por ser o equipamento utilizado no LAI. Com esse programa, o usuário pode promover o condicionamento do sinal (com filtros espaciais e em frequência) e a extração de características do sinal (com fatores como coerência, correlação, fator de fase bloqueada, energia e análise espectral), sempre exibindo os resultados dos processamentos de forma intuitiva. Para validar a eficiência da ferramenta, foram realizados testes com dados reais, comparando-se os resultados esperados (teóricos) e os resultados obtidos (reais). Foi possível realizar condicionamento de sinais EEG com a rejeição de bandas de frequência indesejadas através de filtros digitais, bem como explicitar ao usuário, numericamente e através de figuras, características do sinal de EEG como coerência, fator de fase bloqueada, correlação e energia de canais de forma satisfatória.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

BCI *Brain-Computer Interface* (Interface Cérebro-Computador)

CAR *Commom Average Reference*

DCCN *Donders Centre for Cognitive Neuroimaging*

ECG Eletrocardiograma

EEG Eletroencefalograma

EEGLAB

EMG Eletromiograma

ERS *Event-Related Synchronization* (Sincronização Relacionada a Evento)

ERICA *Experimental Recording, Interactive Control and Analysis* (Registro de Experimentos, Controle Interativo e Análise)

DSP *Digital Signal Processing* (Processamento Digital de Sinais)

LAI Laboratório de Automação Inteligente

LED *Light-emitting diode* (Diodo emissor de luz)

PDS Processamento Digital de Sinais

PLF *Phase-locking Factor* (Fator de Fase Bloqueada)

SSVEP *Steady-state Visually Evoked Potential* (Potencial Evocado Visual de Regime Permanente)

UFES Universidade Federal do Espírito Santo

LISTA DE SÍMBOLOS

α	Alfa, representando uma classe de ritmos cerebrais.
θ	Teta, representando uma classe de ritmos cerebrais.
β	Beta, representando uma classe de ritmos cerebrais.
δ	Delta, representando uma classe de ritmos cerebrais.
γ	Gama, representando uma classe de ritmos cerebrais.
V	Volts, representa diferença de potenciais elétricos
Hz	Hertz, representando frequência de onda.

SUMÁRIO

1 INTRODUÇÃO

De acordo com OPPENHEIM (1999), até a década de 1950, algoritmos de processamento de sinais eram executados de maneira analógica, implementados em circuitos eletrônicos. No fim da década de 50, com o advento de computadores digitais, desenvolveu-se o interesse em realizar processamento de sinais no domínio temporal discreto, tendo em vista a grande flexibilidade destes dispositivos. Inicialmente, ainda segundo OPPENHEIM (1999), as aplicações de DSP computacionais eram limitadas a simulações de filtros analógicos, visto que a partir da implementação digital podia-se testar exaustivamente a eficiência dos filtros antes de implementá-los.

Um dos maiores marcos que contribuíram para o avanço do processamento digital de sinais foi o algoritmo conhecido como FFT (*Fast Fourier Transform* – Transformada rápida de Fourier). De acordo com OPPENHEIM (1999), o desenvolvimento da transformada rápida de Fourier gerou grandes avanços por ter sido um algoritmo eficiente para computar a transformada de Fourier em tempo discreto, sendo não apenas uma aproximação, mas um método exato de cálculo da transformada.

Além do desenvolvimento do método eficiente mencionado, outro fator que aumentou a aplicabilidade de computadores no processamento de sinais foram os avanços alcançados no desenvolvimento de circuitos integrados cada vez mais poderosos (EYRE, 1998), o que possibilitou processamento em tempo real de sinais. Segundo o autor, vários fatores evolutivos proporcionaram tal escalada em poder de processamento, como multiplicadores mais rápidos, múltiplas unidades de execução (*pipelines*), acesso à memória eficiente, formato de dados e conjunto de instruções especializado.

Até o presente ano de 2015, *hardware* e *software* evoluíram consideravelmente (PROAKIS, 1996) e muitas ferramentas de processamento surgiram e foram aprimoradas, tais como o MatLab®. Entretanto, pela natureza teórica e abstrata dos métodos de processamento de sinais, estudantes e profissionais de áreas exteriores às ciências exatas muitas vezes têm dificuldades em trabalhar com tais métodos, o que torna a acessibilidade à técnica citada restrita. Esse fato se torna evidente na UFES - Universidade Federal do Espírito Santo, no

LAI - Laboratório de Automação Inteligente, onde muitos estudantes de diferentes áreas trabalham com projetos diversos que unem engenharia e biotecnologia.

Em aplicações biomédicas trabalha-se com sinais de pequena magnitude e alta susceptibilidade a interferências, visto que estes sinais muitas vezes estão em patamares de amplitude abaixo de 100 μV e suas frequências não chegam a 100Hz (USAKLI, 2010). Assim é muito importante, de acordo com USAKLI (2010), o emprego de técnicas de processamento digital de sinais adequadas e eficientes para que se alcance resultados satisfatórios. Entretanto, como os estudantes da área biomédica possuem pouco ou nenhum conhecimento técnico relativo a sinais digitais, muitas vezes ficam dependentes de estudantes de engenharia para realizar o condicionamento dos sinais e a extração de características e assim tendem utilizar tempo que poderia ser dedicado a estudo e pesquisa aguardando pela disponibilidade de outros membros do laboratório para realizar o processamento.

Ao analisar o panorama do PDS - Processamento Digital de Sinais, percebe-se as limitações de uso das técnicas e o grande leque de possibilidades que sua implementação em diversas áreas de pesquisa e desenvolvimento poderia abrir. Pode-se identificar a necessidade de aumentar a utilização do PDS criando-se a possibilidade de pessoas sem o conhecimento técnico necessário usarem o processamento de sinais em seus projetos. Visando suprir essa necessidade no LAI da UFES, propõe-se o desenvolvimento de uma ferramenta computacional em forma de interface gráfica para auxílio no processamento de sinais.

1.1 Motivação

O processamento digital de sinais consiste em um conjunto de técnicas de modelagem, manipulação e operações matemáticas que possui como objetivo extrair informações importantes do sinal com o qual se está trabalhando, as quais podem ser úteis na solução de um problema ao qual um pesquisador pode se deparar (KUO, 2013). Com técnicas de processamento digital de sinais, muito está se desenvolvendo atualmente em áreas médicas, nas quais diagnósticos podem ser simplificados, bem como novas características de enfermidades podem ser observadas (TOMPKINS, 1993).

De acordo com TOMPKINS (1993), pela natureza técnica e matemática do processamento de sinais, sua acessibilidade pode se restringir a estudantes de graduação, mestrado e doutorado em Engenharia Elétrica e em Biotecnologia. O alto nível de abstração e conhecimento técnico exigido pelos métodos promove uma redução na acessibilidade de ferramentas computacionais que utilizam de técnicas de PDS (como o MatLab®), limitando assim a alocação destes métodos na resolução de problemas por profissionais de áreas fora das ciências exatas.

No Laboratório de Automação Inteligente (LAI), da UFES, muito se desenvolve na área de Robótica de Reabilitação, e entre os profissionais e estudantes envolvidos no projeto encontram-se estudantes de graduação, mestrado e doutorado em Engenharia Elétrica e em Biotecnologia. Estes últimos são graduados em Biologia, Enfermagem, Farmácia, Educação Física, Fisioterapia, enfim, cursos da área de saúde. Sendo assim, possuem pouco conhecimento em computação, matemática avançada e outros, tendem a ter dificuldades ao se deparar com ambientes computacionais que dependem fortemente de conhecimento de computação e engenharia por parte do operador. A Figura 1 mostra um dos atuais projetos em desenvolvimento pela equipe do laboratório na área de robótica de reabilitação. Este projeto se trata de um andador robótico, que tem como objetivo auxiliar na reabilitação de pacientes com dificuldades de locomoção. De acordo com LOTERIO (2014), o andador teve boa aceitação por pacientes com hemiparesias (paralisia parcial de um lado do corpo).

Figura 1 - Sistema robótico para reabilitação da marcha

Fonte: LOTERIO (2014)

Ao analisar o cenário de exclusão apresentado, identifica-se a necessidade de revertê-lo, de forma que mais pessoas se beneficiem das técnicas de processamento digital de sinais, aplicando-as em análises de sinais biológicos de forma simples, sem grande necessidade de conhecimento prévio no assunto, para que seja possível focar as pesquisas científicas em desenvolver-se propriamente novas soluções.

1.2 Projetos semelhantes

Existem alguns projetos com funcionalidade similar já disponíveis, que podem ser carregados da *internet* em suas respectivas páginas virtuais, porém em sua maioria são softwares de código aberto voltados a pessoas com conhecimentos técnicos mais avançados, com forte contribuição da comunidade científica no que se refere à integração de novas funcionalidades, solução de problemas e melhorias para os usuários finais.

1.2.1 EEGLAB

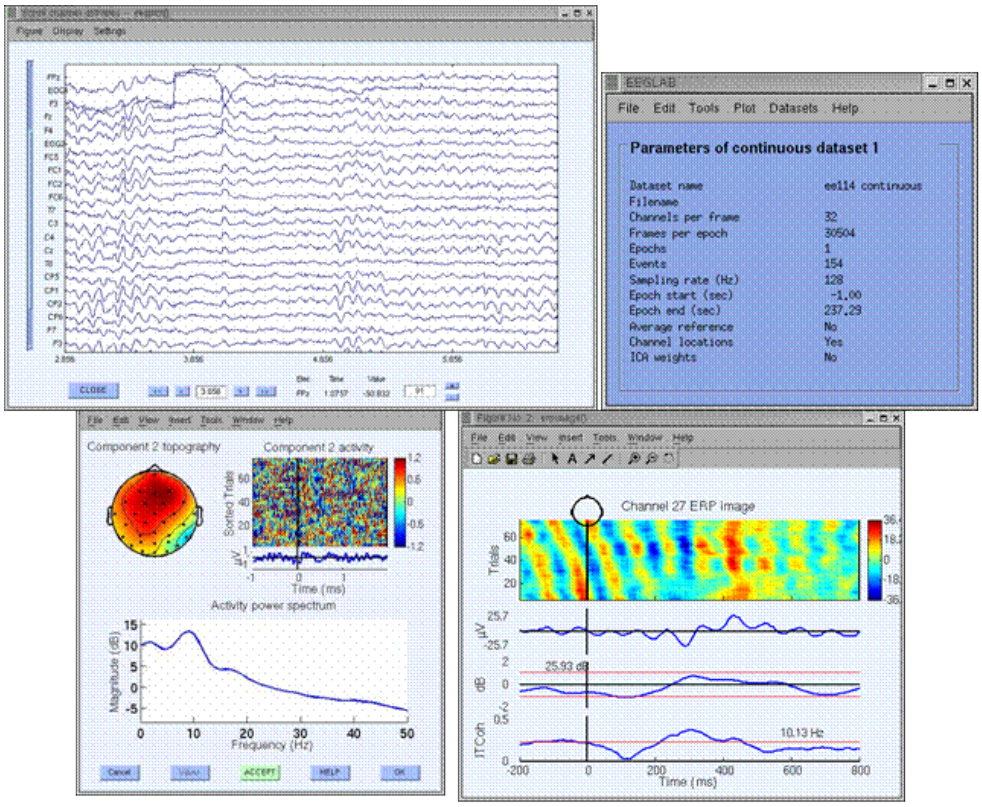
O primeiro projeto semelhante, desenvolvido de forma colaborativa pelo *Swartz Center for Computational Neuroscience* (SCCN), um centro do *Institute for Neural Computation*, da Universidade da Califórnia, é o EEGLAB – *Electroencephalogram Laboratory*. De acordo com a página virtual do projeto (EEGLAB, 2015), o EEGLAB começou a ser formulado em 1997, com o nome de *ICA Electrophysiology Toolbox (Independent Component Analysis Electrophysiology Toolbox*), uma ferramenta que podia realizar algumas poucas funções, como exibir sinais de EEG (Eletroencefalograma) armazenados em arquivos. Com o tempo e mudanças na equipe, o projeto foi se tornando maior e, em 23 de abril de 2001, a primeira versão do EEGLAB foi publicada.

Ao examinar-se a página do projeto, pode-se averiguar que, atualmente, a ferramenta se encontra na versão de número 12 (EEGLAB, 2015) e possui vasta gama de funcionalidades implementadas, mas apresenta-se como uma ferramenta avançada, com baixa acessibilidade a usuários leigos, tendo em vista a dificuldade em seu manuseio.

A ferramenta pode, entre outras funções, exibir e extrair características de sinais de EEG, medir e exibir graficamente características de atividade cerebral e dar características

qualitativas e quantitativas do sinal. Maiores informações podem ser encontradas na página do projeto. A Figura 2 mostra algumas telas do *software*, ilustrando algumas de suas funcionalidades.

Figura 2 - Imagem ilustrativa da interface do EEGLAB.



Fonte: SWARTZ CENTER FOR COMPUTATIONAL NEUROSCIENCE

1.2.2 ERICA

Outra ferramenta para captura e processamento de EEG está atualmente em desenvolvimento por pesquisadores da SCCN, concomitantemente ao EEGLAB, porém com foco em "registro de experimentos, controle interativo e análise" de EEG é a ERICA (*Experimental Recording, Interactive Control and Analysis*), aplicada a interfaces cérebro-computador (*Brain-Computer Interface* - BCI), monitoramento cognitivo, protocolos para jogos e experimentos sociais-neurais.

1.2.3 FieldTrip

O terceiro projeto semelhante também se encontra em desenvolvimento, porém já existe uma versão preliminar disponível. Esta ferramenta é desenvolvida pelo *Donders Centre for Cognitive Neuroimaging* (DCCN), da Holanda. De acordo com a página virtual do projeto (FIELDTRIP) a ferramenta *FieldTrip* é capaz de realizar análise na frequência e no tempo, exibir formas de onda e calcular grandezas estatísticas. O *FieldTrip* suporta formatos de dados de vários sistemas existentes. Existem várias limitações, porém, de acordo com (FIELDTRIP), no que diz respeito à compatibilidade com o MatLab® para que a ferramenta funcione, como a necessidade da presença de complementos específicos, além da complexidade de uso do *software* ser alta, por muito ainda ser feito através de linha de comando.

1.3 Descrição da Proposta

Desenvolvimento de ferramenta computacional intuitiva para preparação e processamento de sinais cerebrais (EEG) dotada de mecanismos para análise de diversas propriedades dos sinais, sendo estes:

- Análise espectral;
- Coerência entre sinais;
- Correlação entre sinais;
- Análise de fase bloqueada (*Phase-locking factor* – PLF);
- Análise temporal (análise espectral em fragmentos do sinal).

A interface também será dotada de mecanismos de condicionamento de sinal, para tratamento dos mesmos, promovendo redução de ruído e interferências, sendo eles:

- Filtros espaciais digitais do tipo *Common-Average Reference* (CAR) e Laplaciano;
- Filtros em frequência digitais do tipo FIR (*Finite Impulse Response*) e IIR (*Infinite Impulse Response*), havendo filtros para algumas faixas de frequência pré-projetados e ferramenta para projeto facilitado de filtros para outras faixas de frequência.

1.4 Objetivos

1.4.1 Geral

Deseja-se, ao concluir o desenvolvimento do presente projeto, fornecer uma ferramenta computacional amigável e intuitiva capaz de realizar processamento digital de sinais cerebrais, utilizando-se do ambiente computacional MatLab®, na qual profissionais e estudantes de áreas alheias às ciências exatas, especificamente da área da saúde, sejam capazes de realizar testes e estudos em dados coletados.

1.4.2 Específicos

Como objetivos específicos, estão em pauta os seguintes itens:

- Realizar condicionamento de sinais através de filtragem temporal e em frequência;
- Extrair características quantitativas do sinal, sendo elas Coerência, Correlação e PLF;
- Realizar análise temporal dos sinais;
- Fazer análise espectral do sinal.

Para tornar a ferramenta fácil de usar para profissionais da área de saúde, a ferramenta será direcionada a ter o mínimo de entrada de parâmetros pelo usuário, será dotada de instruções nas telas através de textos fixos e de textos de “dica”, apresentados quando o usuário deixa o cursor do mouse parado sobre um botão, menu, caixa de texto e outros elementos. Será também elaborado um manual explicando o passo a passo para realizar um processamento completo com o programa.

1.5 Estrutura do texto

No Capítulo 2 serão apresentados, com suas respectivas referências, todos os conceitos teóricos que serão utilizados como base para o desenvolvimento do trabalho, utilizando-se como fontes artigos, livros, publicações e trabalhos científicos. No Capítulo 3 será apresentada a ferramenta que será desenvolvida, com detalhamento de sua implementação e funcionamento, enquanto no Capítulo 4 serão mostrados os testes realizados para validação da mesma e, no Capítulo 5 por fim, será apresentada a conclusão do desenvolvimento do

trabalho, analisando-se o cumprimento dos objetivos e base para trabalhos futuros, com possibilidades de novas aplicações e utilização alternativa da ferramenta que será desenvolvida.

2 FUNDAMENTAÇÃO TEÓRICA

Alguns conceitos fundamentais devem ser introduzidos para boa compreensão e contextualização do trabalho a ser realizado. As subseções seguintes providenciarão fundamentos sobre sinais, sinais biológicos, sinais eletroencefalográficos, suas características e informações que podem ser obtidas deles, bem como técnicas de processamento digital de sinais aplicadas a sinais biológicos.

2.1 Sinais

Um sinal, como definido por HAYKIN (2001), um sinal pode ser de tempo contínuo ou de tempo discreto. O sinal é de tempo contínuo quando $x(t)$ for definido para qualquer valor de t, sendo esses sinais naturais, que surgem em forma de ondas físicas no mundo real. O sinal de tempo discreto é apenas definido para valores isolados de t, sendo estes sinais muitas vezes resultados de um processo de amostragem de um sinal contínuo. A amostragem do sinal, de acordo com TOMPKINS (1993), deve seguir o teorema de amostragem, inicialmente desenvolvido por Shannon, para garantir que o sinal original possa ser reconstruído a partir de suas amostras sem perda de informação. Segundo TOMPKINS (1993), o teorema da amostragem declara que, para um sinal contínuo no qual sua componente de maior frequência se encontra em:

$$f = f_c$$

Será necessária, para que haja representação sem distorção do sinal original, uma amostragem a uma frequência definida por:

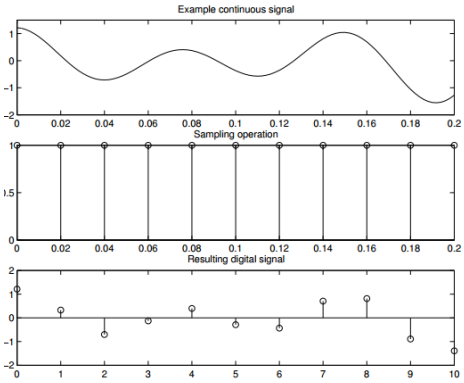
$$f_s = 2 * f_c$$

Outro conceito importante é o de sinal digital. O sinal digital, além de ter característica de tempo discreto, possui níveis de amplitude também discretos, no sentido de que ele não pode assumir qualquer nível de sinal em cada amostra, e sim um dos níveis predeterminados pela resolução da representação do sinal (quantidade de *bits* disponíveis para representar o valor da amostra) (WEEKS, 2011). De acordo com WEEKS, o processo de conversão de um sinal analógico para um sinal digital se inicia com a amostragem do sinal, tomando-se amostras do sinal a cada período de amostragem definido por:

$$T_s = \frac{1}{f_s}$$

Também é quantizada a amplitude do sinal, resultando em um sinal digital. Na Figura 3 pode-se ver um exemplo de digitalização de sinal analógico.

Figura 3 - Processo de conversão de sinal analógico para digital.



Fonte: (WEEKS, 2011, pg. 17)

2.2 Sinais biológicos

Sinais biológicos são definidos por NAÏT-ALI (2009), basicamente, como qualquer sinal adquirido do corpo humano. De acordo com o autor, sinais biológicos podem ser classificados como elétricos (Eletroencefalograma, Eletrocardiograma e Eletromiograma) ou não elétricos (respiração, movimentação, etc.). NAÏT-ALI (2009) ainda afirma que a coleta desses sinais tem grande importância não somente na área clínica, como também na industrial, citando como exemplo um sistema de avaliação de nível de alerta que pode ser implementado em um caminhão, que atuaria ao detectar que motorista esteja sonolento ou desconcentrado soando um alarme. Sinais biológicos são chave para diagnósticos médicos, como exemplo pode-se utilizar um sistema baseado em EEG para medir a atenção de um motorista, para diminuir riscos de acidentes automobilísticos.

2.2.1 Sinais cerebrais (EEG)

De acordo com SANEI (2007), a atividade neural do cérebro humano começa entre a décima sétima e vigésima terceira semana de desenvolvimento pré-natal. Segundo SANEI (2007), acredita-se que dessa fase inicial da vida sinais elétricos gerados pelo cérebro representam não

apenas funções cerebrais, mas também o status do corpo inteiro. Portanto, o cérebro reuniria informações coletadas por todos os sistemas sensoriais do corpo humano para formar uma espécie de autodiagnóstico quando alguma coisa não está correta, para promover ações corretivas automaticamente (SANEI, 2007).

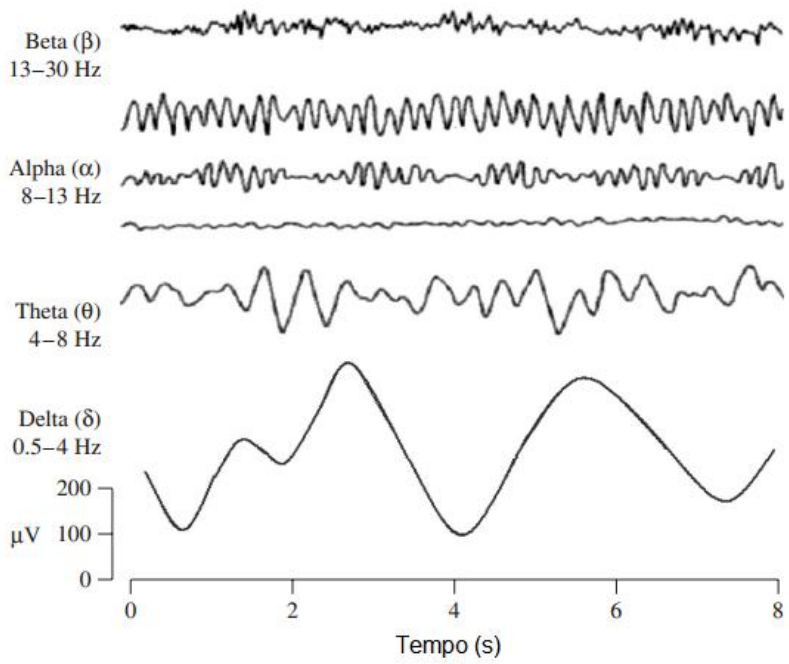
O conceito de neurofisiologia foi introduzido por Carlo Matteucci (1811 – 1868) e Emil Du Bois-Reymond (1818 – 1896), as primeiras pessoas a medir sinais elétricos emitidos por um nervo muscular, utilizando um galvanômetro (CATON, R, 1970; WALTER, 1964). De acordo com USAKLI (2010), uma das maiores dificuldades é a de projetar um sistema de instrumentação que meça satisfatoriamente a amplitude dos sinais a serem medidos e que compensem a grande susceptibilidade dos mesmos a ruído.

De acordo com USAKLI (2010), o cérebro humano gera sinais elétricos, chamados de EEG, que se relacionam a funções do corpo. Sinais EEG possuem amplitude que chegam a $100\mu V$ em amplitude e 100Hz em frequência, e dada a composição do crânio, a medida de sinais cerebrais é mais dificultada do que de outros sinais capturáveis com métodos não-invasivos, como eletromiograma e eletrocardiograma. Da faixa de frequência que abrange até 100Hz alcançada pelos sinais de EEG, subdivide-se em sub-faixas chamadas de Alfa (α), Teta (θ), Beta (β), Gama (γ) e Delta (δ) (LARSEN, 2011),

2.2.1.1 Ritmos cerebrais

De acordo com LARSEN, cada uma das classificações em ritmos cerebrais tem características de onda definidas e representam o estado das atividades cerebrais nos períodos nos quais ocorrem. Na Figura 4 pode ser observado o formato padrão de cada um destes diferentes ritmos cerebrais, que serão apresentados com maior profundidade nas seções que seguem.

Figura 4 - Padrões das formas de onda da maioria dos ritmos cerebrais.



Fonte: (SANEI, 2007)

De acordo com SANEI (2007), ondas alfa aparecem na metade posterior da cabeça e são usualmente encontradas na área acima da região occipital do cérebro. Sua frequência se encontra entre 8 e 13Hz, se apresentando geralmente como um sinal com forma senoidal, porém em alguns casos pode se manifestar como sinais agudos. Em tais casos a parte negativa do sinal tem forma aguda e a parte positiva arredondada. Acredita-se que as ondas alfa indicam uma forma de consciência relaxada, porém sem atenção ou concentração.

De acordo com LARSEN (2011), o ritmo alfa é o ritmo mais proeminente da atividade cerebral, sendo produzido enquanto um indivíduo está com seus olhos fechados, o que levou a acreditar que as ondas alfa nada mais são que um estado de espera produzido pela região cerebral responsável pela visão, e aumentam ao fechar os olhos. As ondas alfa têm maior amplitude na região occipital, sendo a mesma de até 50μV.

De acordo com SANEI (2007), o ritmo Beta se encontra na faixa de frequência entre 13 e 30Hz. Esse ritmo cerebral está associado ao pensamento ativo, atenção ativa, foco no ambiente, solução de problemas e é encontrado em adultos normais. Estas ondas são encontradas nas regiões central e frontal do cérebro e sua amplitude é, normalmente, menor

que 30µV (SANEI, 2007). Estas ondas podem, também, acontecer em locais onde há defeitos na estrutura óssea ou em regiões com tumores (STERMAN, 1972).

O ritmo teta é caracterizado por ter frequências na faixa entre 4 e 7,5Hz (SANEI, 2007). O ritmo teta se intensifica em estado de sonolência, nos instantes de consciência. Estas ondas têm sido associadas com o acesso ao material do subconsciente, a inspiração criativa e a meditação profunda. Ondas teta geralmente são acompanhadas de outras frequências, e parecem se relacionar com o nível de excitação.

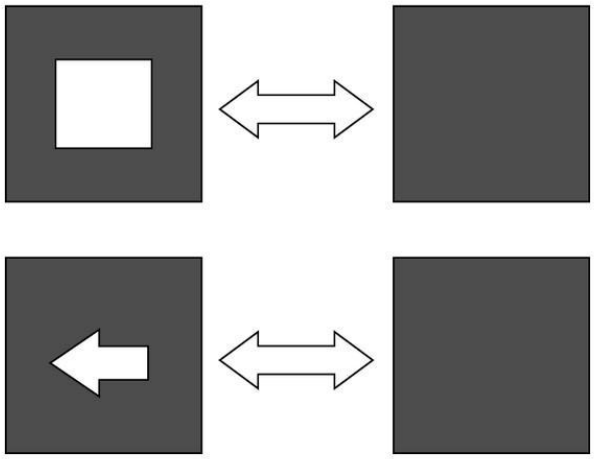
Frequências acima de 30Hz correspondem ao ritmo gama (PFURTSCHELLER, 1994). Sua amplitude é geralmente baixa em comparação com as outras classes, e sua ocorrência é rara, porém a detecção da ocorrência destas é importante para o diagnóstico de algumas doenças cerebrais. Sua ocorrência é maior nas áreas fronto-centrais. Também foi provado que essas ondas são uma boa indicação de sincronização relacionada a eventos (ERS – *Event-Related Synchronization*) do cérebro e pode ser usada para demonstrar a localização do movimento dos dedos indicadores direito e esquerdo, dedão direito e áreas bilaterais da língua (PFURTSCHELLER, 1994).

O ritmo delta engloba as menores frequência entre as classificações, estando entre 0.5 e 3 Hz (HAMMOND, 2006). Essas ondas se manifestam enquanto o indivíduo está dormindo, e se ocorrerem enquanto estiver acordado podem indicar problemas de saúde neurológica (LARSEN, 2011).

2.2.1.2 Potenciais evocados visuais

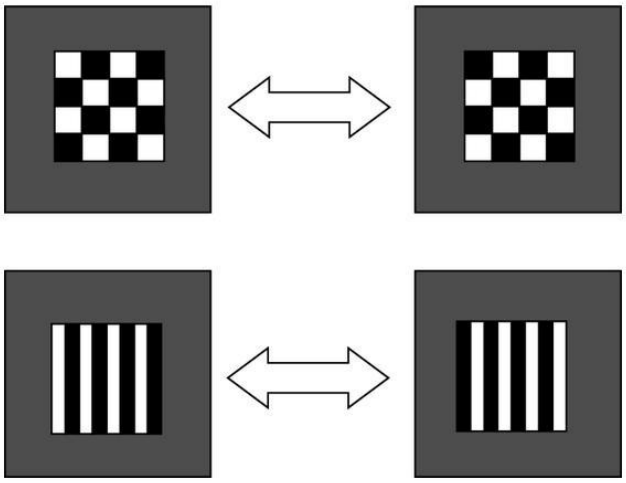
De acordo com ZHU (2009), Potenciais Evocados Visualmente de Regime Permanente (*Steady-State Visually Evoked Potentials* – SSVEP) são potenciais elétricos periódicos induzidos em regiões cerebrais através da exibição de um sinal visual periódico que consiste um elemento que alterne entre preto e branco numa frequência. Pode ser feito como indica a Figura 5, com um estímulo simples, onde numa tela de computador se exibe uma tela preta onde uma área da mesma alterna entre preto e branco periodicamente, ou com padrões mais complexos, como na Figura 6. Os potenciais evocados visuais são, geralmente, acompanhados de suas componentes harmônicas (ZHU, 2009).

Figura 5 - Exemplos de estímulos visuais simples.



Fonte: ZHU, 2009

Figura 6 - Exemplos de estímulos visuais por padrões reversos.



Fonte: ZHU, 2009

De acordo com ZHU (2009), há diversas maneiras de se induzir a ocorrência de SSVEP em indivíduos, ao mostrar-se padrões de estímulo luminoso, com um LED (*Light-Emitting Diodes*) ou outras fontes emissoras de luz pulsante em uma frequência, mostrando-se um estímulo gráfico simples, apenas um objeto piscando na tela de um computador (Figura 5), ou através de um gráfico de reversão de padrão, que consiste em um padrão que é invertido a cada meio período da frequência do sinal (Figura 6).

2.3 Processamento digital de sinais cerebrais

Como apresentado anteriormente, a coleta dos sinais cerebrais é dada em condições extremamente negativas. De acordo com SANEI (2007), os níveis medidos a partir dos eletrodos dos equipamentos dispostos no couro cabeludo dos indivíduos não representam o sinal cerebral em sua forma original, mas sim uma versão muito atenuada do mesmo. O sinal gerado no cérebro precisa ser transmitido por um canal de baixa condutividade (crânio e couro cabeludo) até ser detectado pelos sensores (SANEI, 2007). É evidente, portanto, que técnicas de processamento digital de sinais, junto com filtragem analógica, são necessárias para o condicionamento e análise do sinal EEG, tanto no momento da captura do mesmo (processamento *on-line*) ou para sinais armazenados (processamento *off-line*).

Dada a natureza e as condições de aquisição dos sinais cerebrais, a presença de ruído nos sinais coletados é muito grande. De acordo com CORREA (2011), os fatores que mais induzem ruídos em sinais cerebrais são interferência da rede elétrica (na frequência de 60Hz), vindo de fios elétricos, lâmpadas fluorescentes e outros equipamentos, que induzem tensões nos fios e eletrodos do sistema de captura de EEG. Outro fator que induz ruído, de acordo com VIGON (2000), são as piscadas de olhos que ocorrem durante a captura de EEG. Um terceiro fator que introduz ruído no sinal capturado, de acordo com CORREA (2011), são os batimentos cardíacos do paciente, por terem energia elétrica relativamente alta. Outros ruídos podem ser introduzidos por contrações musculares do paciente, voluntárias ou involuntárias.

Com o objetivo de reduzir esses ruídos e interferências externas no sinal capturado, para conseguir uma representação mais aproximada do sinal cerebral original, deve-se empregar técnicas de processamento de sinal (VIGON, 2000). Um sistema de captura de EEG, de acordo com NIEDERMEYER (2005), deve possuir filtros analógicos para eliminar efeito de *aliasing*, filtros analógicos na frequência da rede (60Hz, no Brasil) e filtros analógicos (ou digitais configuráveis) para reduzir ruído de contato (filtro na faixa de 0.01Hz a 5Hz). Porém, mesmo com esses filtros, de acordo com NIEDERMEYER (2005), ruídos comuns continuam presentes nos sinais e pode-se atenuá-los com técnicas de processamento digital de sinais.

2.3.1 Condicionamento de sinais

Para o condicionamento dos sinais, segundo NIEDERMEYER(2005), técnicas de filtragem espaciais e na frequência são necessárias. Como o sinal cerebral medido é de amplitude muito baixa, a mínima quantidade de ruído pode descaracterizar completamente a forma do mesmo. A frequência de 60Hz está presente em qualquer equipamento eletrônico, visto que é a frequência utilizada pela rede elétrica de alimentação no Brasil (ANEEL, 2010). Outras frequências indesejadas devem ser eliminadas por processo de filtros rejeitadores de banda, dependendo da faixa de frequência que seja pertinente à análise a ser desenvolvida (VIGON, 2000).

De acordo com MCFARLAND (1997), filtros espaciais conseguem melhorar a condição dos sinais provenientes de EEG, atenuando componentes de frequência indesejáveis provenientes de ruídos nos sinais. Exemplos muito utilizados são os filtros CAR (*Common-Average Reference*) e Laplaciano (MCFARLAND, 1997).

2.3.1.1 Filtro em frequência

Para supressão de bandas de frequência de sinais, utilizam-se técnicas de filtro em frequência. De acordo com OPENHEIN (2009), filtros de seleção de frequências são sistemas que deixam passar certas faixas de frequência e rejeitam totalmente todas as outras. Os filtros a serem utilizados na ferramenta são, basicamente, filtros IIR (*Infinite Impulse Response*) e filtros FIR (*Finite Impulse Response*). Essa classificação, de acordo com (RORABAUGH, 1993) é feita de acordo com a duração de sua resposta ao impulso.

De acordo com SANEI (2007), filtros FIR têm a resposta ao impulso finita, e não possuem recursividade em sua natureza, sendo puramente digitais, podendo ser projetados para terem uma vasta gama de respostas (passa-baixas, passa-altas, passa-faixas, etc). Um filtro FIR tem a função de transferência digital como:

$$H(z) = \sum_{i=0}^{O} a_i z^{-i}$$

Onde:

- O é a ordem do filtro;
- a_i é o i-ésimo coeficiente do filtro.

Filtros IIR, de acordo com OPENHEIN (2009), tem o nome *Infinite Impulse Response* pela sua resposta ao impulso ser infinita. Filtros IIR são filtros analógicos discretizados (levados do domínio de Laplace para o domínio em Z). Um filtro IIR tem a função de transferência digital como:

$$H(z) = \frac{\sum_{i=0}^{P} b_i z^{-i}}{\sum_{j=0}^{Q} a_j z^{-j}}$$

Onde:

- P é a ordem do filtro em alimentação positiva;
- b_i é o i-ésimo coeficiente do filtro em alimentação positiva*;*
- Q é a ordem do filtro em realimentação;
- a_i é o i-ésimo coeficiente do filtro em realimentação*;*

Por terem polos em sua função de transferência, dados os termos recursivos da função, a estabilidade dos filtros IIR não é garantida. Segundo RORABAUGH (1993), as vantagens dos filtros FIR são a possibilidade de atraso constante e a garantia de estabilidade para filtros não-recursivos. Porém, apesar das vantagens apresentadas pelos filtros FIR, algumas desvantagens apresentadas por esses filtros, como a necessidade de uma resposta ao impulso muito longa para conseguir boas rejeições nas frequências de corte e a maior dificuldade de projeto desse tipo de filtro quando comparados aos IIR (RORABAUGH, 1993).

2.3.1.1.1 Média móvel simples

GUIÑÓN (2007) afirma que a média móvel é uma técnica que consiste em calcular a média de um número de pontos, de forma recursiva, caracterizando um filtro digital IIR. De acordo com GUIÑÓN (2007), a média móvel é capaz de reduzir oscilações bruscas em sinais, melhorando a representação dos mesmos. GUIÑÓN ainda afirma que, pela simplicidade dessa técnica, sua eficiência se limita a sinais no domínio do tempo.

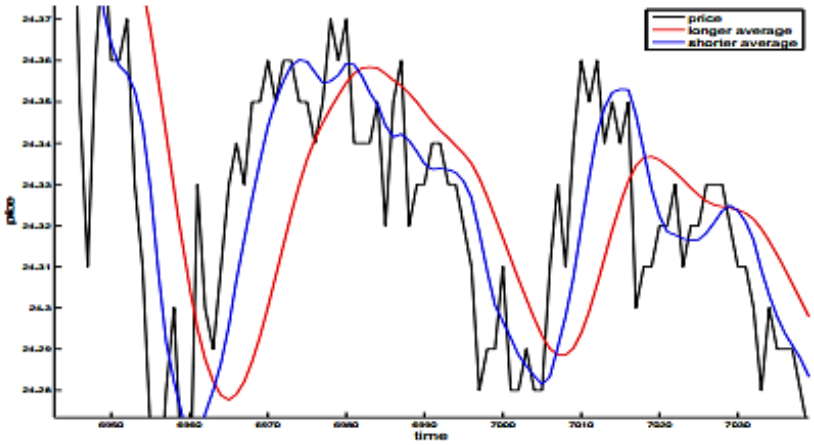
De acordo com RAUDYS (2013), a média móvel pode ser calculada pela fórmula:

$$x_m[t+1] = \frac{1}{n}\sum_{i=t+1-n}^{t} x[t]$$

Onde x_m é o sinal filtrado, t é a amostra, n é o tamanho da janela de tempo utilizada (número de amostras) e x é o sinal original. O tamanho da janela, de acordo com RAUDYS (2013),

influencia o nível de suavização resultante do processamento. Como pode ser observado na Figura 7, sem a média móvel (em preto) há muitas oscilações bruscas, enquanto com a média móvel de janela mais curta (azul) se tem uma suavização boa e com a janela mais longa (vermelho) a suavização é ainda maior.

Figura 7 - Efeito da média móvel no sinal



Fonte: (RAUDYS, 2013)

Como desvantagens da média móvel, GUIÑÓN (2007) cita distorções significantes e a possibilidade de reduzir a intensidade do sinal. Tendo em vista que a amplitude dos sinais EEG já é muito baixa, dados de EEG não serão submetidos à técnica de média móvel. Apenas dados extraídos desses sinais que necessitarem de filtragem serão submetidos a essa técnica, como os valores de coerência obtidos entre canais de EEG, que possuem muitas oscilações e prejudicam a análise visual.

2.3.1.2 Filtros espaciais

Filtros espaciais, de acordo com MOURIÑO (2001), são cruciais para avaliar atividade cerebral, visto que aumentam a qualidade do sinal ao atenuar ruídos comuns (presentes em todos os canais). MOURIÑO (2001) afirma que filtros espaciais aumentam a relação sinal/ruído. São exemplos de filtros espaciais populares os filtros CAR e o filtro Laplaciano (MCFARLAND, 1997).

2.3.1.2.1 Filtro Common Average Reference

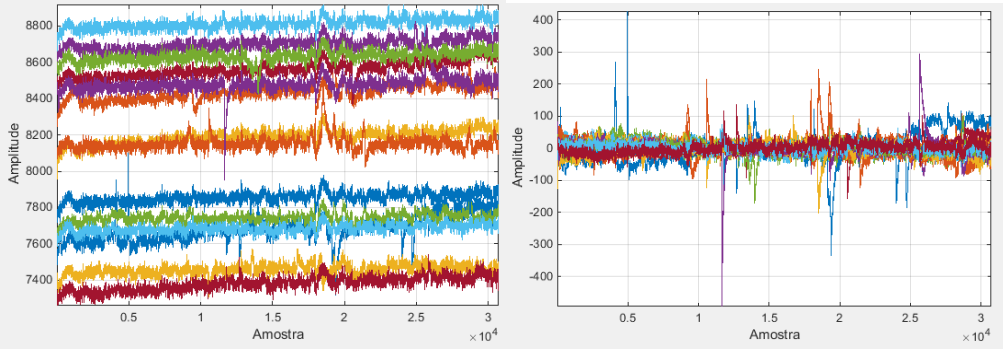
De acordo com GARCIA-MOLINA (2011), o filtro CAR é um método comum de filtragem espacial, consistindo em realizar a subtração da média de todos os sinais gravados para cada canal do sinal. Sua fórmula, de acordo com GARCIA-MOLINA (2011), é:

$$x_{fi}[n] = x_i[n] - \frac{\sum_{j=1}^{N} j[n]}{N}$$

Onde x_{fi} é o i-ésimo canal filtrado, x_i é o i-ésimo canal antes do filtro e N é o número de canais. Segundo GARCIA-MOLINA (2011), em sistemas EEG, se o número de sinais gravados é maior que 16 e esses sinais foram capturados por eletrodos uniformemente distribuídos pela cabeça do paciente, o processamento por filtro CAR consegue levar o sinal a um condicionamento muito melhor.

Esse tipo de filtro é capaz de remover sinais comuns a todos os sinais sendo filtrados, e como muitas vezes sinais comuns em EEG de múltiplos canais caracterizam ruídos (PFURTSCHELLER, 1994), essa técnica pode reduzir artefatos nos sinais processados, como os ruídos previamente discutidos provenientes de piscada de olhos, rede elétrica, batimentos cardíacos e contrações musculares, assim condicionando-os a melhor representarem os sinais cerebrais reais. A Figura 8 mostra o resultado da aplicação de um filtro CAR a um sinal de EEG, em todos os canais. Percebe-se que as componentes de corrente contínua dos sinais são eliminadas, levando-o a representar mais fielmente os sinais originais de EEG. Nota-se que no gráfico à esquerda os sinais apresentam uma componente contínua (DC), que foi filtrada, resultando em sinais centralizados em zero volt no gráfico à direita.

Figura 8 – Comparativo entre sinal original e sinal filtrado com filtro CAR.



Fonte: Nossa autoria.

2.3.1.2.2 Filtro Laplaciano

O filtro laplaciano é outro tipo de filtro espacial que, de acordo com MCFARLAND (1997), também possui desempenho bom em atenuação de ruído em sinais EEG. De acordo com TANDONNET (2005), o filtro laplaciano faz uma estimativa da densidade de corrente do couro cabeludo ao estimar a superfície laplaciana e aplicá-la ao sinal EEG. Segundo TANDONNET (2005), a superfície laplaciana pode ser computada aproximadamente pelo método de Hjorth pela fórmula:

$$SL = \frac{4}{3}[3V_N - (V_A + V_B + V_C)]$$

Onde SL é a superfície laplaciana, V_N é o potencial no eletrodo atual e V_A, V_B e V_C são os potenciais dos eletrodos próximos a V_N. De acordo com TANDONNET (2005), o método do filtro laplaciano resulta em redução nos erros de medida. Porém, de acordo com MCFARLAND (1997), entre os filtros espaciais o filtro CAR ainda é mais eficiente em análise de EEG.

2.3.2 Avaliação de sinais

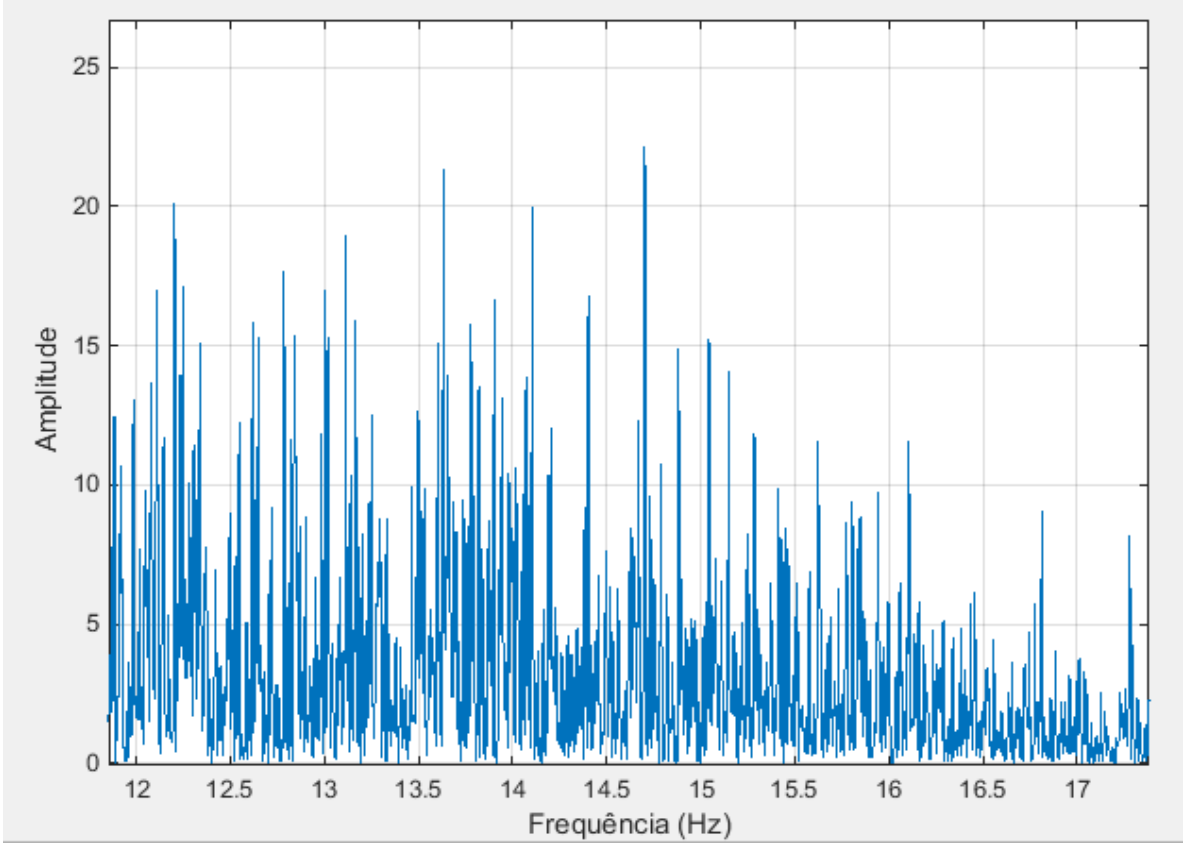
Os métodos de pré-processamento apresentados brevemente até aqui se limitam ao condicionamento do sinal para posterior análise. De acordo com SANEI (2007), para avaliação de sinais EEG, são necessárias ferramentas que extraiam características do sinal gerando informações fáceis de serem analisadas, sendo necessárias assim técnicas de processamento de sinal que trabalhem com os dados de forma a se obter um número, gráfico ou alguma espécie de representação visual.

2.3.2.1 Avaliação baseada na resposta em frequência do sinal

Uma técnica de avaliação de sinais muito utilizada é a resposta em frequência. De acordo com MITRA (2006), a resposta em frequência é a análise do comportamento da amplitude de um sinal em relação a suas componentes de frequência, sendo representada normalmente por um gráfico que relaciona as amplitudes (em escala linear ou logarítmica, em dB) pelas frequências, onde são mostradas as componentes de frequência presentes no sinal, relacionadas com a amplitude de cada componente. Este gráfico possui forma semelhante ao gráfico mostrado na Figura 9. Pode-se ver que o sinal (de EEG) tem vários picos de amplitude

ao longo do espectro de frequências, característica de sinais EEG que possuem alta presença de ruídos externos.

Figura 9 - Formato de gráfico de resposta em frequência.



Fonte: Nossa autoria

2.3.2.2 Avaliação baseada na coerência

Outra técnica utilizada em pesquisas para a avaliação de sinais cerebrais (PETERS, 2013) é a análise da coerência entre sinais de canais de EEG. Esta técnica quantifica a intensidade das ligações entre as diferentes áreas cerebrais, em função das componentes de frequência de cada sinal. Esta técnica resulta num índice numérico entre 0 e 1 para cada par de sinais calculado, ao longo do espectro de frequência dos sinais. A fórmula para o cálculo da coerência é:

$$C_{xy}(f) = \frac{\left|P_{xy}(f)\right|^2}{P_{xx}(f) * P_{yy}(f)}$$

Onde:

- $C_{xy}(f)$ é a coerência entre o sinal x e o sinal y na frequência f;
- $P_{xy}(f)$ é a densidade espectral cruzada dos sinais x e y na frequência f;
- $P_{xx}(f)$ é a densidade espectral do sinal x na frequência f;

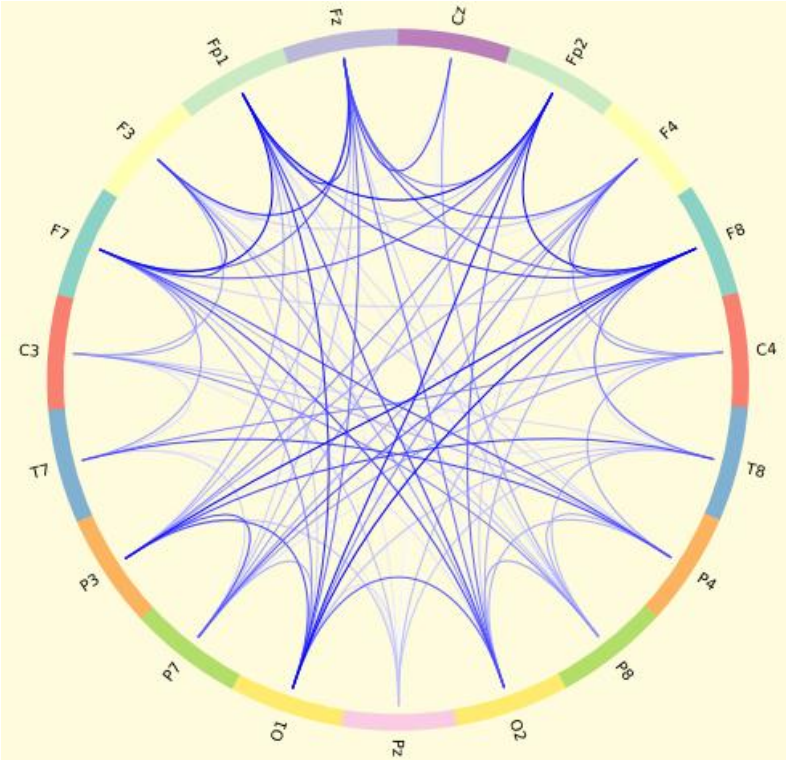
- $P_{yy}(f)$ é a densidade espectral do sinal y na frequência f.

De acordo com PETERS (2013), a densidade espectral cruzada dos sinais x e y $P_{xy}(f)$ pode ser calculada como a transformada de Fourier da correlação cruzada entre os sinais x e y. De acordo com BELMONT (1997), a correlação cruzada entre os canais é calculada por:

$$C_{xy}[n] = \sum_{m=-\infty}^{\infty} x^*[m]\, y[n+m]$$

Ao calcular-se o valor da coerência para cada par de canais pode-se montar um gráfico em forma de mapa que representa a coerência média de cada par de canais, utilizando-se a coerência média entre todas as frequências de cada par de canais, como o mostrado na Figura 10. Assim, pode-se averiguar visualmente qual par de canais possui maior coerência (linha mais grossa, como P3 e F8 na figura) e menor coerência (linha mais fina, como F7 e F8 na figura).

Figura 10 - Mapa neural de conexões entre canais feito relacionando as coerências



Fonte: PETERS, 2013, nossa edição.

2.3.2.3 Avaliação baseada no índice de fase bloqueada

Por fim, outra técnica utilizada na avaliação de sinais cerebrais é a análise do índice de fase bloqueada (*Phase-Locking Factor* – PLF). De acordo com MCFARLAND (1997), este fator representa uma relação de semelhança a partir das fases entre diferentes sinais cerebrais, em

uma frequência de análise. Pode-se, por exemplo, calcular o PLF numa frequência de ocorrência de ritmos alfa, beta, ou qualquer outro, com a finalidade de analisar a correlação entre as frequências de ocorrência de sinais nos diferentes testes. Esta informação gera um valor para cada canal analisado, e pode ser feita uma representação gráfica quando calculado o valor do fator para vários canais (QUIROGA, 1999).

O cálculo do PLF é feito a partir da equação (LACHAUX, 1999):

$$PLF(t) = \frac{1}{N}\left|\sum_{n=1}^{N} e^{j\theta(t,n)}\right|$$

Onde:

- $PLF(t)$ é o valor do fator de fase bloqueada;
- N é o número de sinais com os quais se está trabalhando (2 no caso de um par de canais de EEG);
- $\theta(t,n)$ é a diferença entre a fase do sinal x_n e a fase do sinal x_m no instante de tempo t.

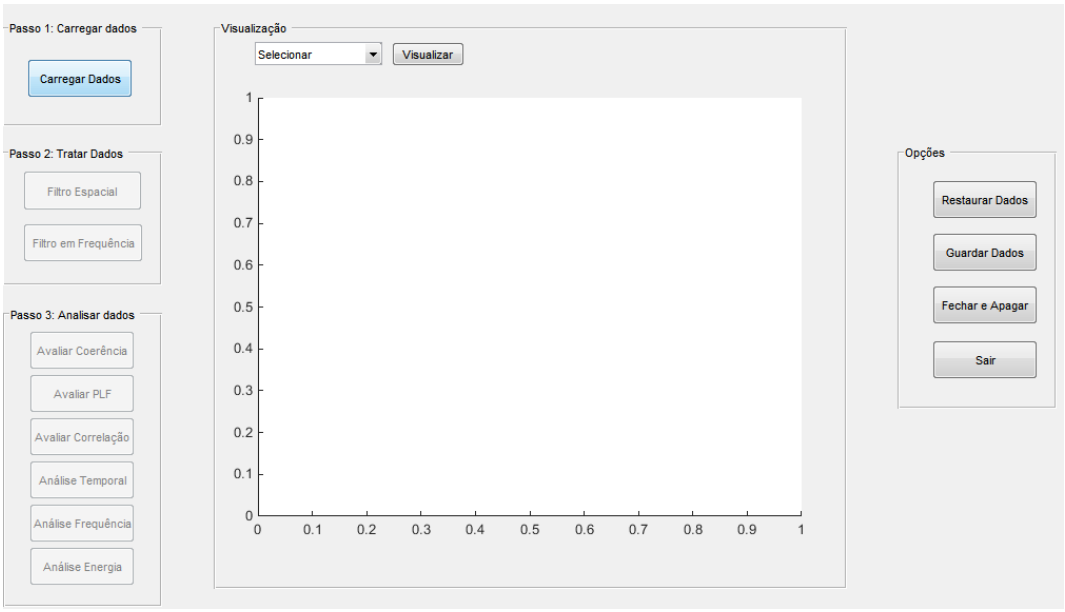
3 DESENVOLVIMENTO

Dada a quantidade de funções previamente disponíveis no *software* MatLab® para aquisição e processamento de sinais, além de uma grande quantidade de ferramentas para desenvolvimento da interface, esta foi a plataforma escolhida. Há uma interface principal, da qual são chamadas as outras componentes do programa. O método utilizado para a passagem de parâmetros e dados de uma componente para a outra foi a utilização do *workspace* ‘base‘ (onde os comandos dados pelo terminal do MatLab® são executados). Assim, ao salvar uma variável, o programa o faz no *worskpace* e, ao abrir outra componente do programa, o novo componente importa todos os dados necessários a partir do *workspace*. Nas próximas seções será detalhado o funcionamento do programa e de cada componente do mesmo.

3.1 Interface principal

A interface principal pode ser vista na Figura 11. Ela consiste em uma tela com botões à esquerda, uma área para gráficos no centro e mais botões à direita. À esquerda se encontram as várias opções de processamento possíveis, divididas em subconjuntos chamados de “passos”. A divisão por passos foi implementada para garantir que o processo de condicionamento do sinal tenha ordem respeitada (filtro espacial primeiro, depois filtro em frequência), para que o processamento seja mais eficiente, mesmo que o usuário seja da área de saúde, e por conseguinte, sem muito conhecimento do processamento em si.

Figura 11 - Representação da interface principal.



Fonte: Nossa autoria

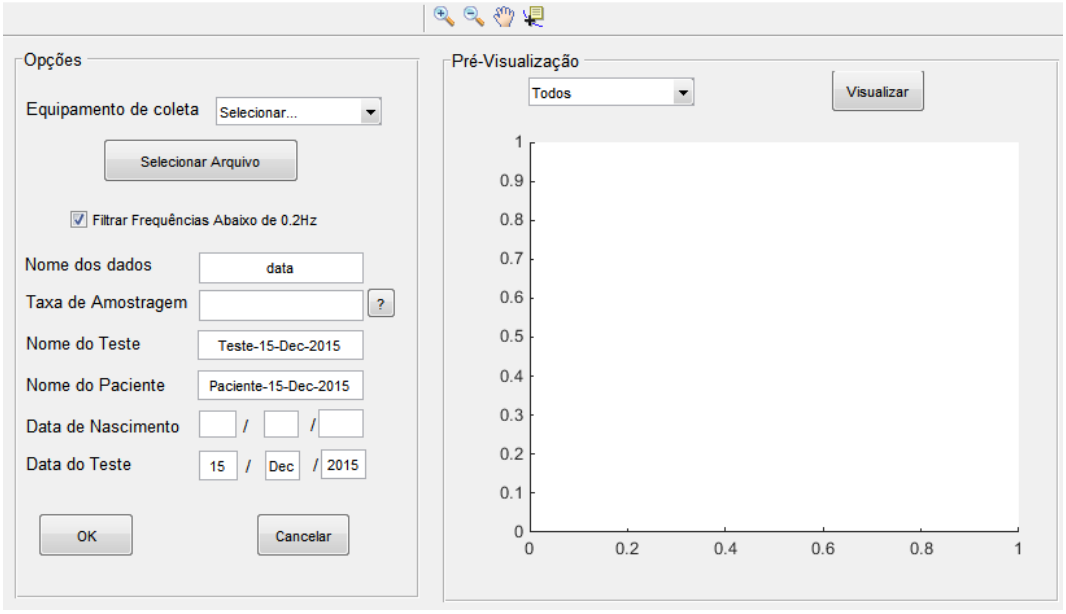
O primeiro passo é o carregamento dos dados, e nenhuma função pode ser executada na interface antes desta etapa ser realizada. Como é altamente indicado passar o sinal a ser utilizado por um filtro espacial antes de qualquer outra operação (MCFARLAND, 1997), este é o segundo passo na interface principal. Após filtragem espacial, o usuário pode ou não realizar filtragem em frequência, no sentido de que não há mecanismo de monitoramento no *software* que indica se o sinal foi filtrado ou não.

A área com espaço para gráficos ao centro da interface serve para o usuário visualizar o sinal em escala de tempo, podendo selecionar o canal a ser exibido ou exibir todos os canais no mesmo gráfico. Por fim, o usuário pode salvar, restaurar e voltar a uma seção salva anteriormente, bem como fechar a interface descartando todo o progresso feito.

3.2 Interface de carga

Para a interface de carga, foi desenvolvido um *layout* simples. Há um botão de "abrir arquivo", que abre uma caixa de diálogo com a qual se aponta para o arquivo, algumas caixas de entrada de texto, onde se coloca informações sobre os dados, e uma área onde o usuário pode pré-visualizar os dados.

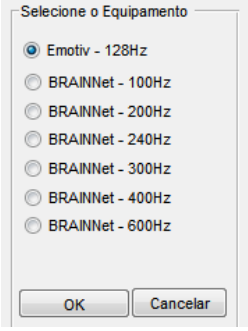
Figura 12 - Tela da interface de carregamento de dados



Fonte: Nossa autoria

Vale notar o campo “taxa de amostragem” individualmente, que é de extrema importância para o funcionamento correto da interface, visto que todo processamento em frequência realizado dentro da interface depende estritamente da frequência de amostragem para realizar os processamentos de forma correta, tendo como exemplo o projeto de filtros, que depende da frequência de amostragem para produzir resultado correto. Neste campo, foi inserido um botão que, ao ser pressionado, apresenta ao usuário a interface mostrada na Figura 13, com botões para seleção de valores de taxa de amostragem de acordo com o equipamento, ajudando assim o usuário inserir o valor adequado. Caso o usuário não selecione ou não digite valor no campo específico, o programa mostra uma caixa de diálogo na qual questiona ao usuário se deve utilizar o valor padrão de 128Hz, valor utilizado no equipamento Emotiv.

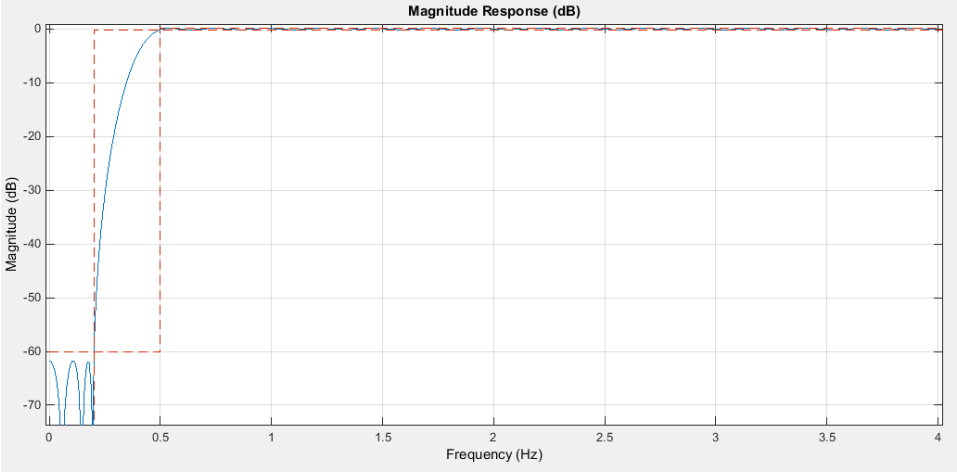
Figura 13 - Interface de ajuda para escolha da taxa de amostragem.



Fonte: Nossa autoria

Uma opção presente na interface ativada por padrão é a filtragem de sinais de corrente contínua (mais precisamente até 0,2Hz), realizado por um filtro FIR passa-altas sintonizado em 0,2Hz, com *ripple* de 0,3dB na banda passante. Esse filtro visa atenuar ruídos de contato presentes em sinais EEG antes de realizar outros processamentos no sinal. A resposta em frequência desse filtro, com *zoom*, pode ser vista na Figura 14. Foi colocada uma banda de transição entre 0,2 e 0,5Hz para evitar problemas na resposta do filtro.

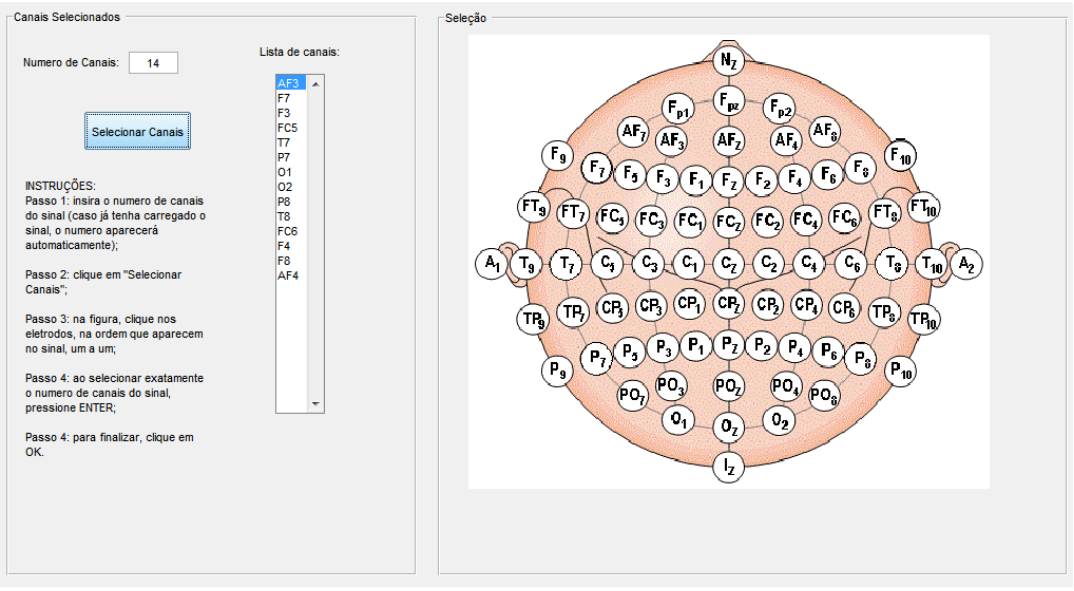
Figura 14 - Resposta em frequência do filtro para eliminar ruídos de contato



Fonte: Nossa autoria.

Outro parâmetro de entrada de usuário é o equipamento com o qual o sinal foi coletado. A partir desse parâmetro, foi possível ampliar o processamento do sinal de 14 canais para o total do protocolo 10-20, de 75 canais. Isso foi feito com a tela de seleção de canais mostrada na Figura 15. Seguindo as instruções exibidas em tela, o usuário inicia digitando o número de canais do sinal no campo correspondente (caso já tenha carregado os dados será preenchido automaticamente) e clica em "Selecionar canais". A interface então exibe um cursor diferente, que é utilizado para o usuário marcar os canais correspondentes para então pressionar *Enter* no teclado. O programa então armazenará o vetor de canais a serem processados.

Figura 15 - Tela de seleção de canais

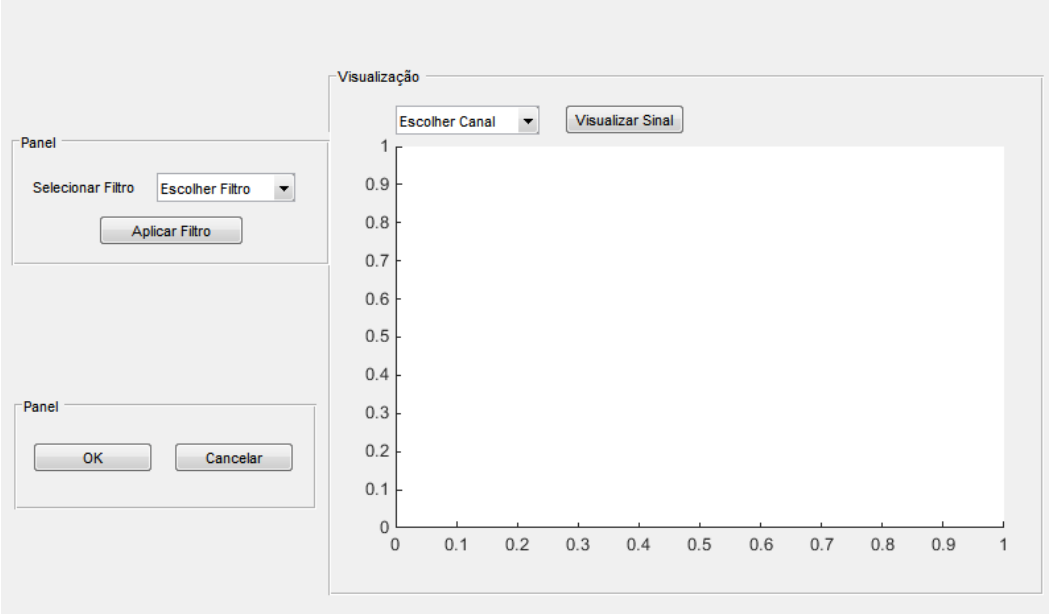


Fonte: Nossa autoria.

3.3 Interface de filtragem espacial

A interface dedicada ao filtro espacial é simples. Nela, o usuário deve selecionar o método (filtro) a ser aplicado e clicar em aplicar filtro. O usuário pode visualizar o sinal a qualquer momento. Caso não haja filtro selecionado, o sinal exibido na janela de visualização será o sinal original. Caso o usuário esteja com um filtro selecionado no menu, será exibido uma pré-visualização do sinal com o filtro aplicado. Para efetivamente aplicar o filtro ao sinal, o usuário deve clicar no botão correspondente.

Figura 16 - Tela da interface dedicada à filtragem espacial.



Fonte: Nossa autoria

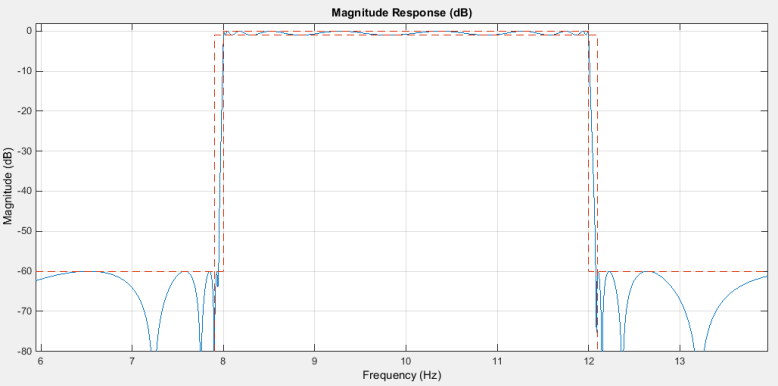
3.3.1 Filtros inclusos

Os filtros implementados para essa etapa de processamento foram o filtro CAR e o filtro laplaciano. Como já apontado na seção 2.3.1, de acordo com a pesquisa de MCFARLAND (1997), o filtro CAR é o filtro espacial de melhor desempenho geral para processamento de sinais cerebrais. O filtro laplaciano também gera resultados satisfatórios, apesar de inferiores, mas foi incluído pela possibilidade de ser útil em outros processamentos e constar em artigos como possibilidade.

3.4 Interface para filtro em frequência

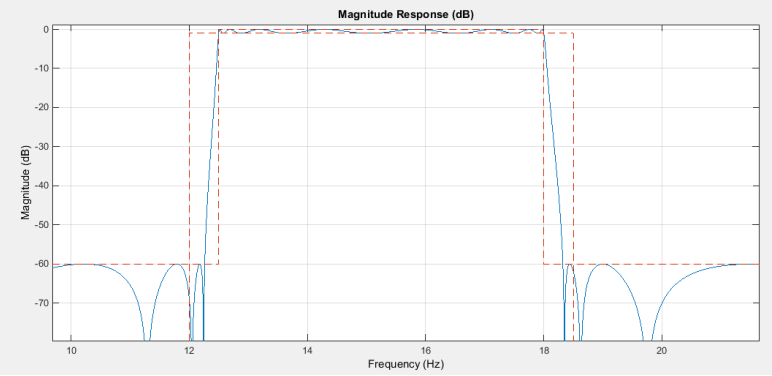
Esta interface é semelhante à anterior, porém existe uma gama muito maior de filtros. Foram selecionados alguns filtros julgados básicos, de uso bastante comum, que um usuário possa necessitar muito frequentemente, que foram denominados “Filtro alfa” e “Filtro beta”. Estes filtros básicos tem característica de banda passante de passa-faixa, portanto deixam passar bandas de ondas alfa e beta, respectivamente, assim para qualquer teste nas quais as frequências estejam nas bandas compreendidas entre os valores nominais destas frequências estarão cobertos pela ferramenta, sem necessidade de modificações. O Filtro Alfa é um filtro IIR passa-faixa, elíptico, com frequências de passagem entre 8 e 12 Hz, atenuação de 60dB na banda de rejeição e ripple de 0,3dB na banda de passagem, já o Filtro Beta difere na banda de frequência, que fica entre 12Hz e 18Hz. As respostas em frequência dos filtros podem ser vistas na Figura 17 e na Figura 18.

Figura 17 - Resposta em frequência do filtro Alfa



Fonte: Nossa autoria

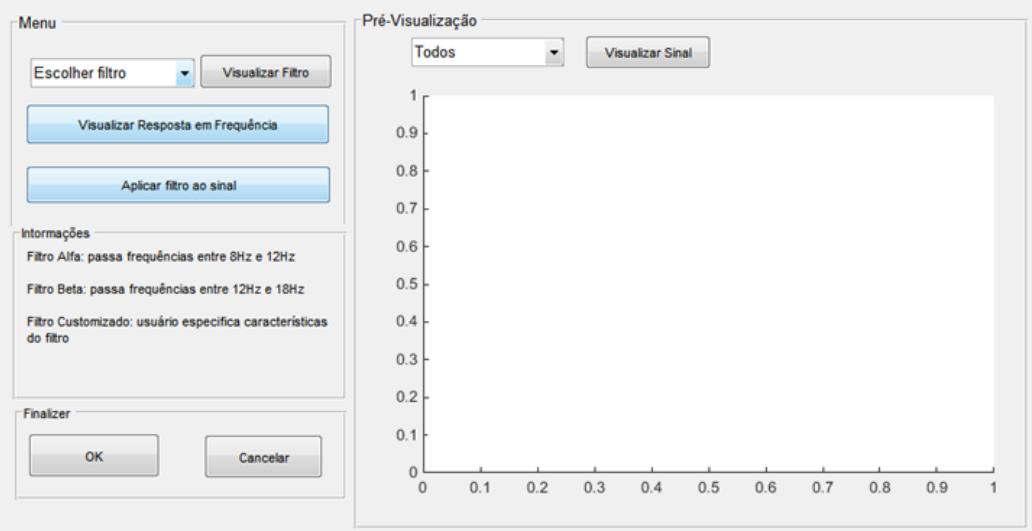
Figura 18 - Resposta em frequência do Filtro Beta



Fonte: Nossa autoria.

Na Figura 19 pode-se ver a tela da interface. Ao clicar em “Visualizar Filtro”, a interface apresenta a resposta em frequência do filtro, enquanto em “Visualizar Resposta em Frequência” a interface apresenta a resposta em frequência do sinal com o filtro aplicado. Para finalizar, o usuário deve clicar em “Aplicar filtro ao sinal”, para então fechar a interface. Há instruções na tela de ajuda com um passo-a-passo de como aplicar um filtro ao sinal corretamente.

Figura 19 - Tela da interface dedicada à filtragem em frequência.



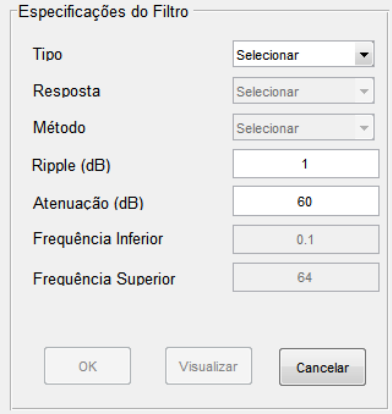
Fonte: Nossa autoria

Caso a faixa de frequências que o usuário esteja interessado tenha valores fora das faixas nas quais já existam filtros predefinidos, existe a opção de criação de um filtro customizado pelo usuário, utilizando a interface mostrada na Figura 20. Nesta opção, o usuário escolhe o tipo do filtro digital (FIR ou IIR), o tipo de resposta do filtro (Passa Faixas, Passa Baixa, Passa Alta ou Rejeita Faixa), o método do filtro (*Butterworth*, *Chebyshev* ou *Elíptico*, apenas para IIR), o valor de *Ripple* (na banda de rejeição), a atenuação (na banda de rejeição) e a faixa de frequência de passagem (ou de rejeição) e a interface então projeta um filtro que atenda aos requisitos definidos pelo usuário.

A interface preenche dinamicamente os valores ao longo da seleção do usuário nos menus, com valores padrão de *Ripple* em 1dB, atenuação em 60dB, servindo como guia para que o usuário não faça uma seleção indevida. Há verificações para que o usuário não defina parâmetros de projeto do filtro que retornariam em erros, como a utilização de frequência de corte acima da frequência máxima do sinal (definida em duas vezes a frequência de

amostragem), bloqueio dos botões e menus posteriores mediante a não seleção de menus superiores e verificação de estabilidade do filtro resultante, diferenciando esta interface do *toolbox* de projeto de filtros do MatLab®.

Figura 20 - Tela da interface dedicada à criação de filtros customizados



Fonte: Nossa autoria

3.5 Interface para análise de coerência

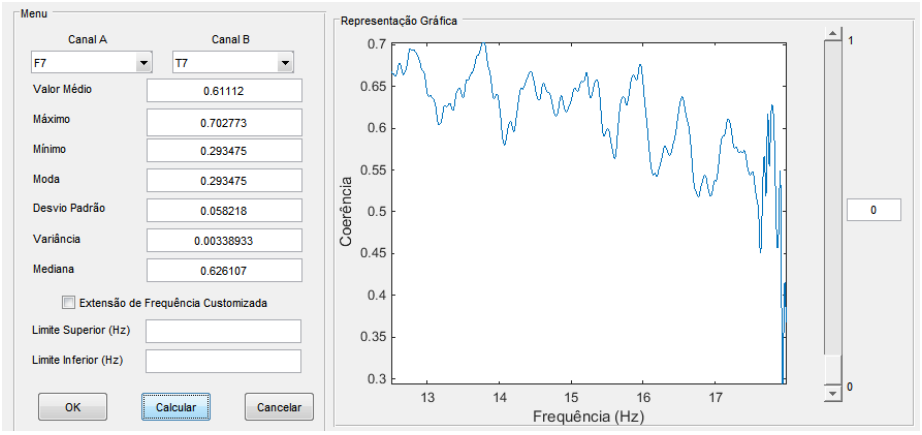
A partir desta área, acaba o condicionamento do sinal e começa o levantamento de características do mesmo. A coerência pode ser avaliada para um par de canais, separadamente, ou para todos os canais, em combinação. Na primeira opção, a ferramenta gera um gráfico relacionando frequência e coerência entre os dois sinais, além de informações estatísticas sobre o vetor de coerência, mostrando média, mediana, desvio padrão, valores mínimo e máximo e moda. Esta forma de representação é apresentada na Figura 21.

Por padrão, o programa utiliza para o cálculo e para a exibição gráfica os limites de frequência utilizados pelo último filtro em frequência que o usuário aplicou ao sinal, pois a coerência dos sinais fora desses limites se apresenta muito oscilatória e não representa uma informação útil ao usuário. Pode-se, também utilizar limites de frequência definidos pelo usuário, caso seja desejável.

Vale notar que o gráfico apresentado, bem como os valores estatísticos calculados, passam por um filtro de média móvel para reduzir oscilações bruscas de valores causados por ruídos ainda presentes no sinal processado, suavizando bastante o resultado. O efeito provocado pela

média móvel, porém, é mais visual que prático, pois os valores estatísticos calculados não mudam significantemente com a aplicação do filtro.

Figura 21 - Tela da interface dedicada à análise da coerência entre sinais-Par de canais



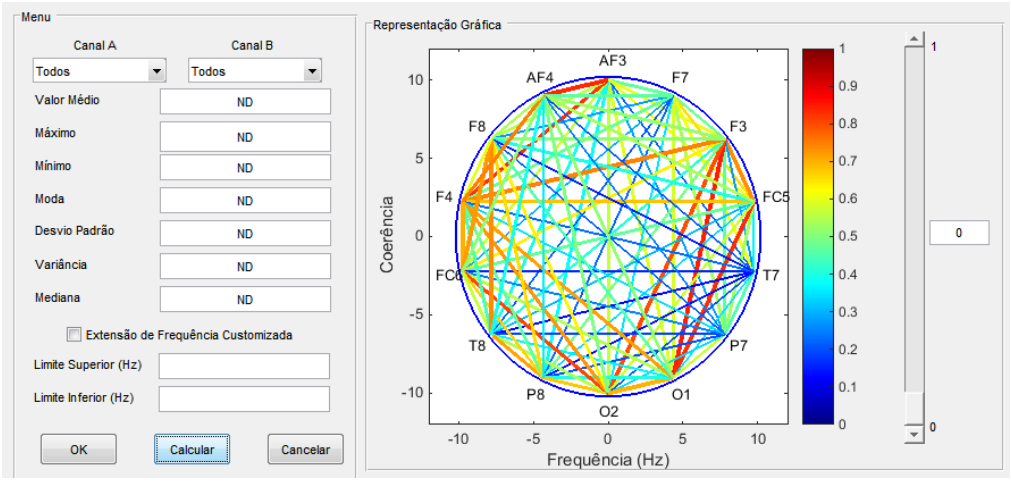
Fonte: Nossa autoria

A segunda forma de representação da coerência é através do cálculo combinado da coerência de todos os pares de canais do sinal. O programa calcula os vetores de coerência (em frequência), realiza o corte dos mesmos na banda de frequência de interesse (definida pelo último filtro aplicado ou pelo usuário), passa cada vetor resultante por média móvel e calcula o valor médio de cada vetor, resultando em uma matriz de valores médios de coerências entre cada par de canais.

A partir dos valores de coerência de cada par de canais, é plotado um grafo representando a intensidade das ligações entre os canais, como mostrado na Figura 22. A interpretação da representação gráfica é intuitiva: quanto maior a correlação entre os sinais, mais espessa a linha e mais quente é a cor da linha, sendo exibido ao lado um guia colorido com a graduação das cores e os valores numéricos correspondentes. Assim, pode-se fazer uma análise visual rápida da coerência global entre todos os canais.

O botão deslizante (*slider*) pode ser utilizado pelo usuário para limitar a quantidade de informação que será exibida na tela. Ele define o valor mínimo que a coerência entre um par de canais precisa ter para que ela seja representada no gráfico. Assim, o usuário pode visualizar apenas as interações com intensidade maior que 0,5, por exemplo, e ter um gráfico menos carregado de informações.

Figura 22 - Tela da interface dedicada à análise da coerência entre sinais (Combinação de 14 sinais)

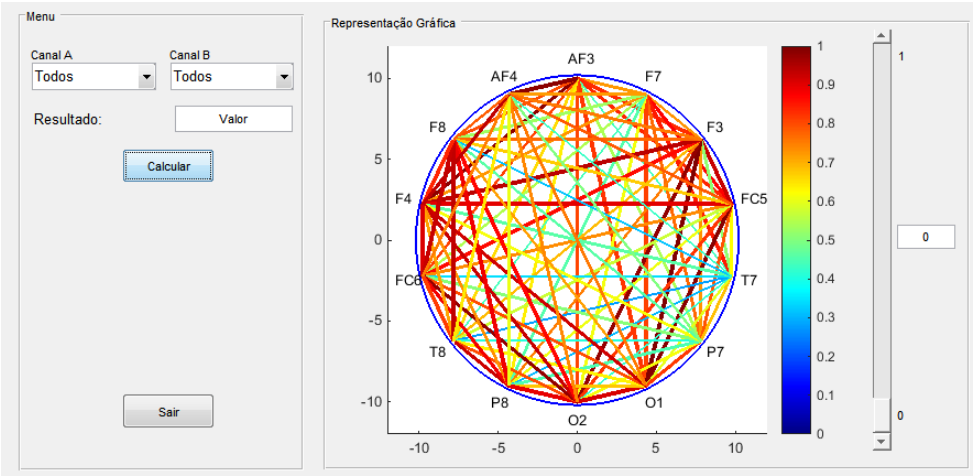


Fonte: Nossa autoria.

3.6 Interface para correlação

O funcionamento da correlação é semelhante à coerência, porém a função utilizada para o cálculo da mesma no MatLab® retorna apenas um índice por par de canais, não um vetor, portanto não existem cálculos de outras características em cima da correlação. Caso o usuário selecione um par de canais específicos para ser realizado o cálculo da correlação, o programa a calcula e exibe no campo “Resultado”. Caso contrário, o programa calcula a correlação entre cada par de canais e exibe um gráfico que relaciona a intensidade da ligação de todos os canais, assim como é feito na coerência. A Figura 23 mostra este gráfico, assim como foi feito na coerência. Também há o botão deslizante para seleção de limiar mínimo de amplitude de coerência entre os pares de canais para exibir no gráfico.

Figura 23 - Tela da interface dedicada à análise da correlação entre sinais-Interpolação de 14 sinais

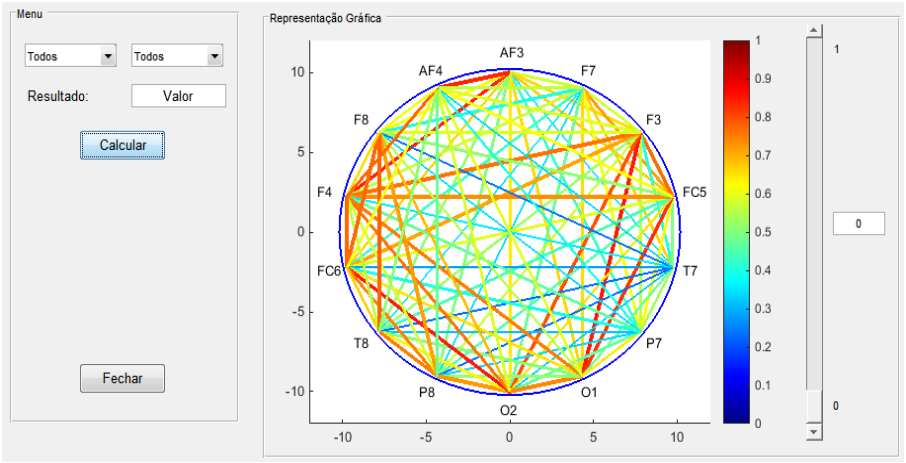


Fonte: Nossa autoria

3.7 Interface para PLF

A interface para a análise do valor de fase bloqueada é simples, semelhante à interface de correlação: se o usuário selecionar um par de canais o programa calcula o valor do PLF daquele par, caso contrário o programa gera o gráfico que relaciona a intensidade da ligação de todos os canais, representando os valores do PLF de cada par de canais, assim como é mostrado na Figura 24. Assim como era feito na coerência, o botão deslizante (*slider*) da interface pode ser utilizado pelo usuário para limitar a quantidade de informação que será exibida na tela, definindo o valor mínimo que PLF entre um par de canais precisa ter para que ela seja representada no gráfico.

Figura 24 - Tela da interface dedicada à análise do PLF entre sinais (combinação de 14 canais)



Fonte: Nossa autoria

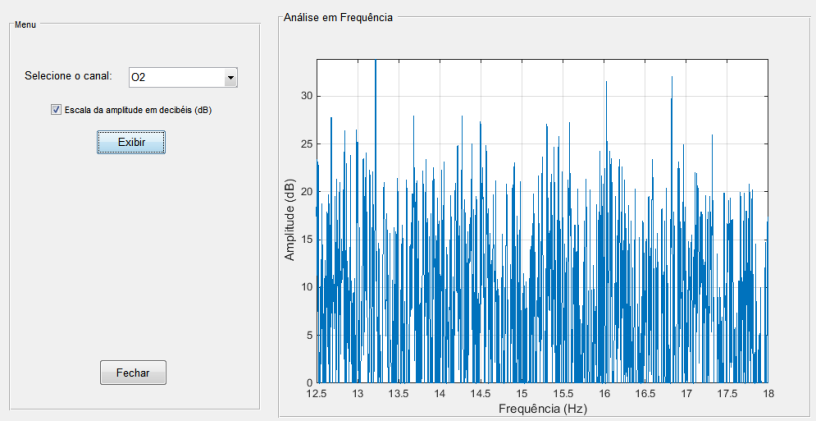
3.8 Interface para análise espectral

A representação do espectro de frequência do sinal é feita através do cálculo da densidade espectral com a transformada rápida de Fourier (FFT – *Fast Fourier Transform*), através da função *periodogram*, do MatLab®. A partir da seleção do canal, quando o usuário clica em "Calcular", o programa calcula e exibe na área gráfica o espectro de frequência do sinal. Há ferramentas de zoom, avaliação de valores e movimento do gráfico, também sendo possível salvar o gráfico em uma imagem.

Na Figura 25 pode-se ver a tela da interface mostrando o espectro de frequência de EEG na faixa do ritmo alfa. A área gráfica é limitada pelos parâmetros do último filtro utilizado pelo

usuário. No caso da Figura 25, o sinal foi filtrado para passar as frequências entre 8Hz e 12Hz, assim o eixo será aproximado para esse intervalo apenas, visando exibir apenas informação relevante ao usuário. Outra observação é a possibilidade de plotar o gráfico em escala linear ou logarítmica (decibéis).

Figura 25 - Tela da interface dedicada à análise espectral de sinais



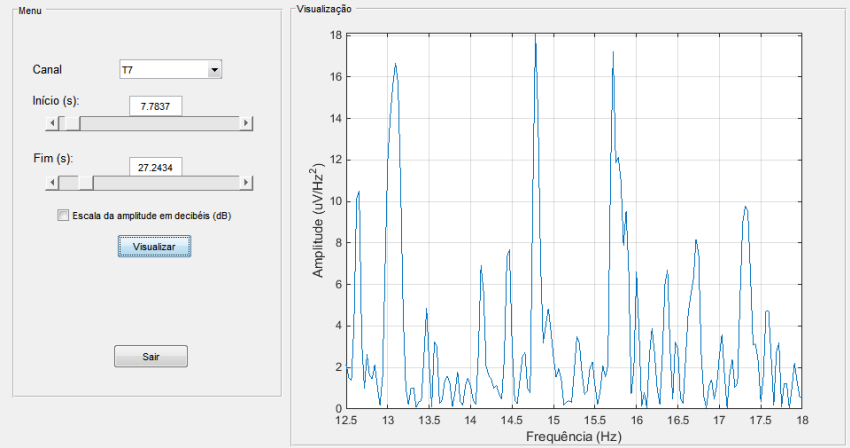
Fonte: Nossa autoria

3.9 Interface para análise temporal

Essa última janela tem como objetivo específico fazer a análise de um fragmento do sinal. Nela, o usuário pode escolher o intervalo em segundos do sinal que deseja analisar e visualizar o espectro de frequência de tal fragmento. Esta função é interessante para aplicações como o SSVEP (*Steady-State Visually-Evoked Potentials* - Potenciais de Regime Permanente Evocados Visualmente), pois se pode avaliar o tempo de resposta do cérebro para a indução de uma determinada frequência internamente a partir da exibição do padrão de alternação de cores característico do SSVEP para os olhos do indivíduo.

O funcionamento desta componente é simples: o usuário seleciona o canal e seleciona os valores de início e fim do fragmento do sinal a ser avaliado (através do teclado ou dos botões deslizantes) e clica em “Calcular”. O programa, então, exibe o espectro de frequência do sinal de forma gráfica. A interface pode ser vista na Figura 26. O sinal mostrado na Figura 26 representa a densidade espectral de um sinal de EEG proveniente de um teste de SSVEP realizado em laboratório, calculada pelo MatLab® através da função *periodogram*. Nesta interface também existe a escolha da escala logarítmica ou linear.

Figura 26 - Tela da interface dedicada à análise temporal de sinais

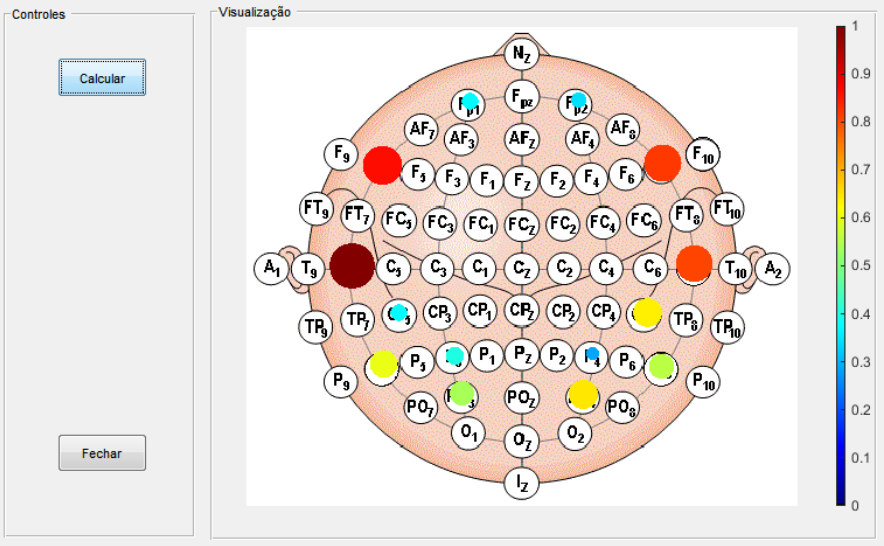


Fonte: Nossa autoria

3.10 Interface para análise de energia

Por fim, foi desenvolvida uma interface para a análise de sinais cerebrais que quantifica e representa visualmente a energia de cada canal de EEG separadamente. Na Figura 27, o sinal utilizado foi o mesmo que nas outras interfaces, provenientes do Emotiv. Em cada um dos pontos onde se localizam os 14 canais do Emotiv, o programa calcula a energia do sinal e faz um círculo preenchido com raio proporcional ao valor da energia do sinal capturado naquele ponto, além de graduar a cor do círculo pelo sistema de cores RGB: cores mais frias (azuladas) representam valores de energia mais baixos e cores mais quentes (avermelhadas) representam valores de energia mais altos. Esse sistema funciona para qualquer sistema que utilize o protocolo 10-20, porém se o dado não for proveniente do Emotiv ou do BrainNet BNT-36 o usuário deverá selecionar os canais manualmente.

Figura 27 - Análise de energia de canais de EEG



Fonte: Nossa autoria

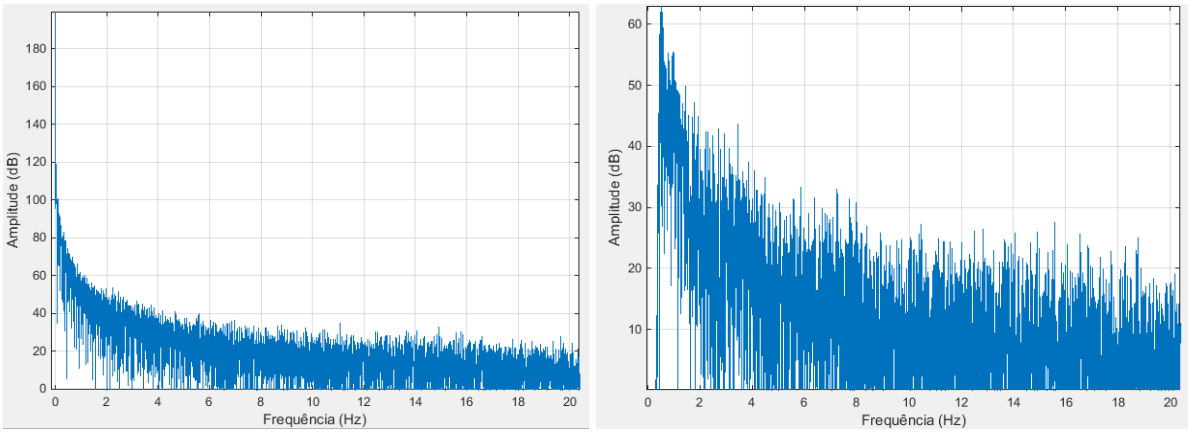
4 TESTES E RESULTADOS

Para a validação dos resultados foram realizados testes que comprovam os resultados esperados de todas as etapas de processamento do sistema. Serão apresentados nas seções seguintes os processamentos que a interface é capaz de realizar, seus resultados esperados e obtidos.

4.1 Condicionamento de sinais

No condicionamento de sinais, existem basicamente filtros em frequência e filtros espaciais. Dos filtros em frequência, o filtro de componentes abaixo de 0.2Hz se destaca pela importância da sua aplicação, visto que atenua ruídos de contato. A Figura 28 mostra o antes e o depois da aplicação desse filtro. É visível que o ruído existente inicialmente nas frequências baixas está em patamares muito elevados, e mesmo com aplicação de *zoom* não é possível ver as outras frequências do sinal. Já com o filtro aplicado, o sinal fica muito menos ruidoso, sendo possível visualizar as outras componentes de frequência.

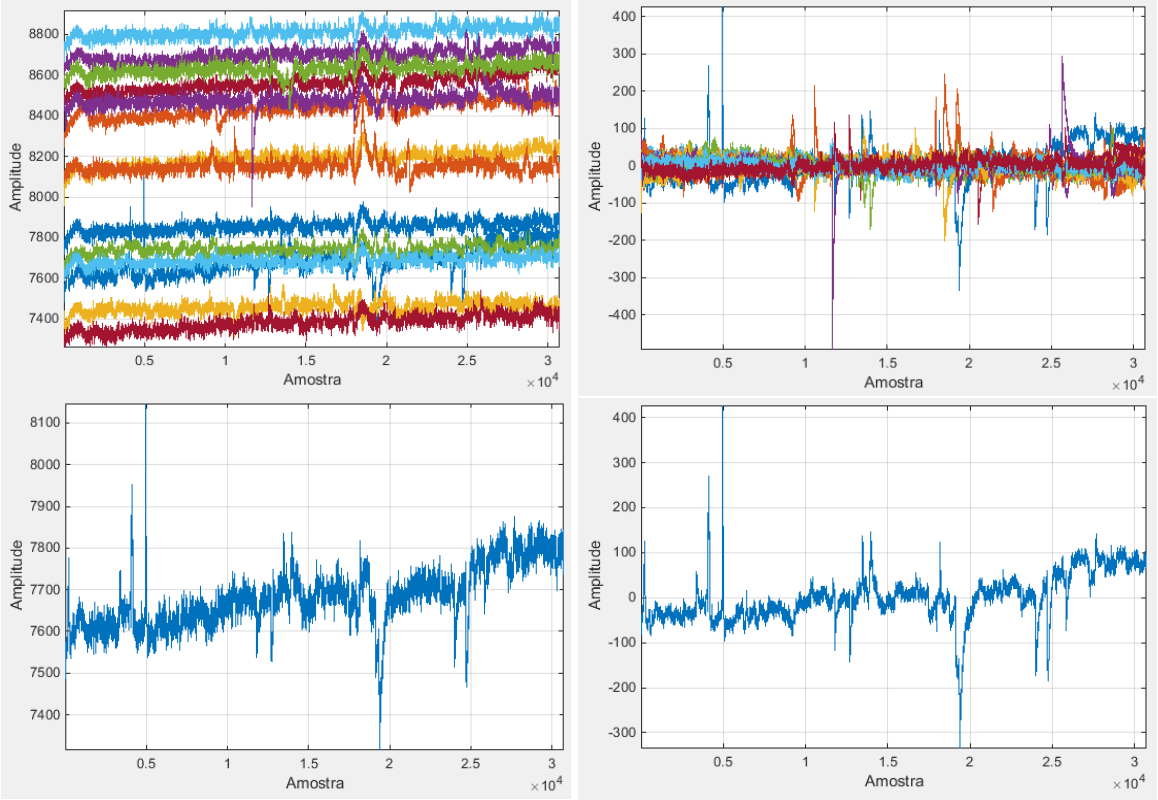
Figura 28 - Comparativo entre sinal não filtrado (esquerda) filtrado (direita)



Fonte: Nossa autoria.

Outro processamento de condicionamento de sinal que foi aplicado é o filtro CAR. Esse tipo de filtro, de acordo com a literatura, realiza atenuação em ruído comum nos diversos canais do sinal. A Figura 29 mostra o resultado da filtragem por método CAR no sinal. Basicamente, as componentes contínuas são eliminadas, trazendo todos os sinais para a origem do eixo. Outro fator observado é a diminuição das oscilações rápidas nos sinais, fazendo-os assumir uma forma mais definida.

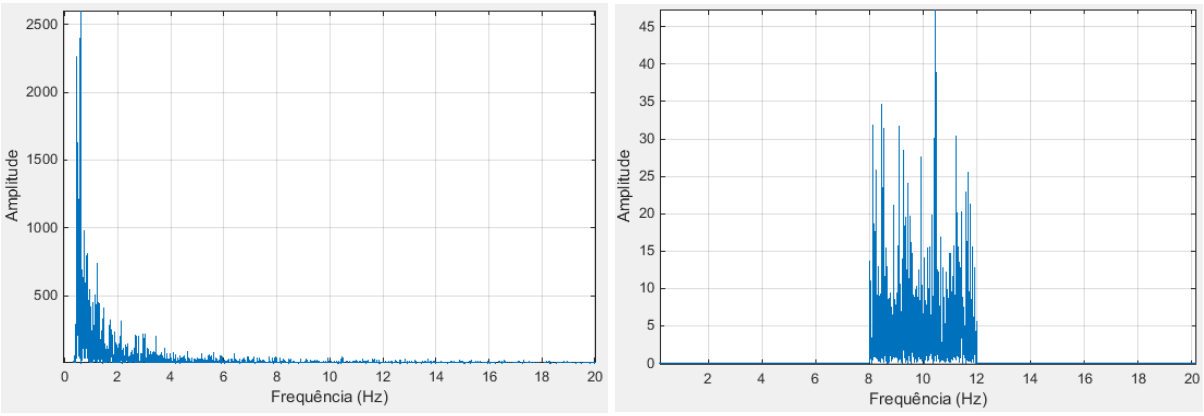
Figura 29 - Aplicação do filtro CAR, antes (esquerda) e depois (direita)



Fonte: Nossa autoria

A próxima técnica a ser testada é a de filtro em frequência. Serão isoladas as frequências da banda alfa utilizando-se um filtro passa-faixas entre 8 e 12Hz. A Figura 30 mostra esse processo. Na esquerda, pode-se ver alta densidade de componentes de frequência anteriores aos 8Hz, enquanto na direita apenas as frequências entre 8 e 12Hz continuaram presentes no sinal, caracterizando o bom funcionamento do filtro.

Figura 30 - Teste do filtro em frequência, sinal sem filtrar (esquerda) e sinal filtrado (direita)



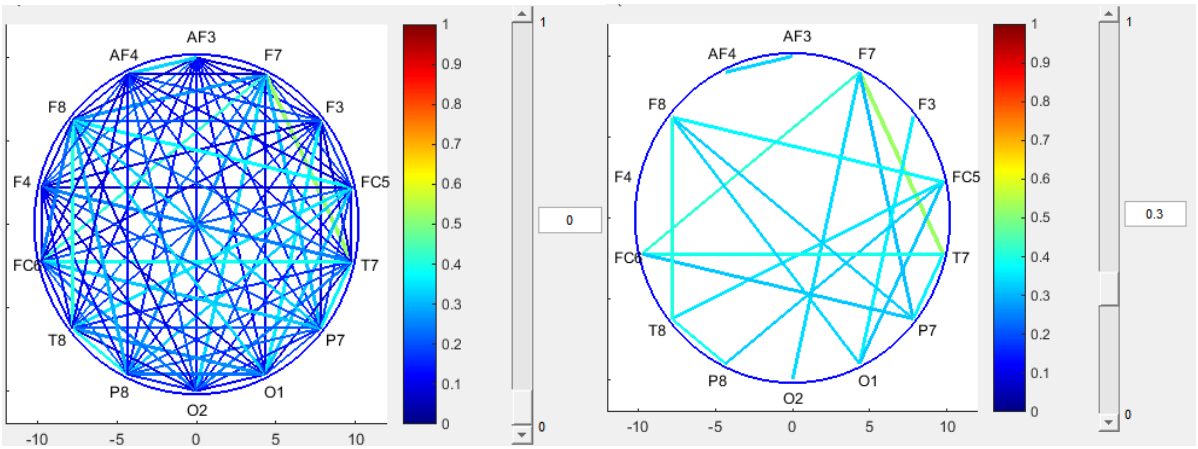
Fonte: Nossa autoria.

4.2 Avaliação de sinais

A avaliação de sinais é basicamente a extração de características dos sinais que podem ser úteis em testes de laboratório. A primeira característica é a correlação entre sinais. Com um conjunto de 14 canais, foi proposto inicialmente fazer um grafo que representasse a coerência entre os sinais em pares. Esse mapa cerebral foi construído, a partir da média da coerência de cada par de canais, e pode ser visto na Figura 31. À esquerda, é possível identificar os pares de canais com maiores valores de coerência, como F7 e T7 e com menores valores, como FC5 e T7. Os valores calculados dessas coerências são 0,52 para o par F7-T7 e 0,106 para o par FC5-T7, mostrando assim que o método é eficiente para informar ao usuário o panorama geral das coerências entre canais.

Ainda se referindo à Figura 31, à direita, é mostrado o efeito do botão deslizante ao selecionar um limiar de 0,3 para a coerência mínima. Selecionando este valor mínimo, apenas as interações com maiores coerências são exibidas, facilitando a visualização do usuário.

Figura 31 - Representação visual da coerência entre 14 canais de EEG

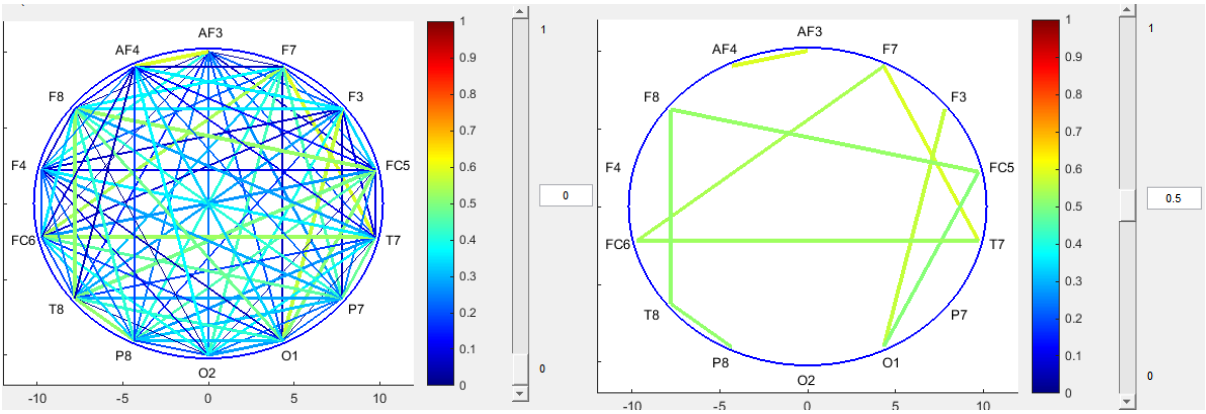


Fonte: Nossa autoria

O mesmo princípio foi aplicado ao PLF. No mesmo sinal, foi produzida a mesma imagem, porém com índices do PLF, resultando no que mostra a Figura 32. À esquerda, pode-se ver pela figura que os maiores valores de coerência estão nos pares de canais AF4-AF3 e F7-T7, enquanto os menores estão em AF3-F8. Os valores de AF4-AF3 e F7-T7 ficaram em 0,59, enquanto AF3-F8 ficou em 0,026, provando a eficiência do método. A mesma técnica foi usada para realizar o gráfico da correlação, sendo obtido resultado similar. O *slider* novamente se prova útil ao se observar a Figura 32, à direita. Ao definir o valor mínimo de

PLF para representação em 0,5, apenas 7 ligações são representadas, facilitando ao usuário identificar os valores mais relevantes.

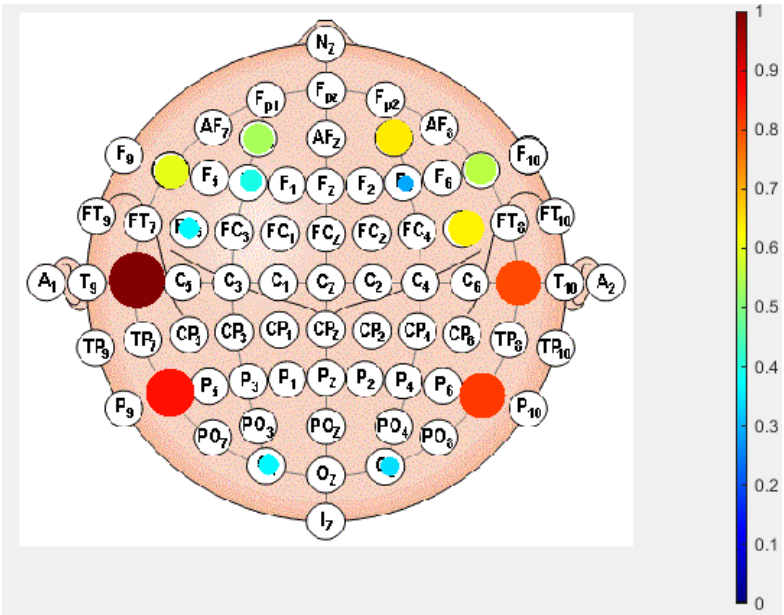
Figura 32 - Representação visual do PLF entre 14 canais de EEG



Fonte: Nossa autoria

Por fim, a energia dos sinais é um parâmetro importante para averiguar a "ativação" de cada região cerebral. Para a exibição desse parâmetro, foi utilizado um novo mapa das regiões cerebrais, que mostra todas as regiões do sistema 10-20 e o preenche com os canais que estão sendo processados, resultando num cenário de 14 canais do sistema Emotiv na Figura 33. Pode-se ver quais canais possuem maior potência de sinal (T7, T8, F7 e F8) e quais tem menores potências de sinal (P3, P4, Fp1 e Fp2), provando ser uma maneira eficaz de representar visualmente a energia de sinais EEG.

Figura 33 - Representação da energia nos diferentes canais do sinal



Fonte: Nossa autoria

5 CONCLUSÃO E TRABALHOS FUTUROS

Após desenvolvimento, implementação computacional e testes concluídos com sucesso, pode-se afirmar que os objetivos colocados inicialmente em pauta foram alcançados, portanto o problema identificado foi solucionado, suprindo assim a carência que inicialmente foi identificada. No que tange à acessibilidade da ferramenta desenvolvida, o objetivo de conseguir alcançar facilidade de uso da aplicação foi alcançado, visto que toda a parte de processamento e comandos do MatLab® fica transparente ao usuário, que se concentra apenas a comandar a ferramenta por cliques e avaliar resultados.

No que se refere aos processamentos propostos a serem realizados na ferramenta, foi possível abranger todos com a implementação desenvolvida. Em matéria de condicionamento de sinal, foram integradas as técnicas mais populares no processamento de sinais biológicos, principalmente em relação a EEG. No que tange à análise de sinais, foi possível implementar todos os métodos propostos, de acordo com as demandas mais vistas no ambiente de pesquisa do Laboratório de Automação Inteligente.

Uma limitação presente na ferramenta desenvolvida está atrelada ao formato de dados compatível. Para que se possa trabalhar com os sinais no *software*, os dados devem estar armazenados como um arquivo proprietário do Martlab® (extensão *.mat*), em uma matriz com um canal por linha ou um canal por coluna (o *software* adapta o dado automaticamente). Assim, caso um usuário queira utilizar diretamente um dado coletado por um equipamento que não colete os dados diretamente no MatLab® terá que adaptá-los antes de realizar qualquer processamento com a ferramenta, fazendo o MatLab® carregar os dados para um arquivo compatível. Um projeto futuro pode ser, portanto, um banco de dados de formatos de dados para convertê-los para o formato do MatLab®.

Outra limitação da interface é ter filtros prontos para condicionamento apenas de EEG. Assim, um usuário leigo que possa ter dificuldade em desenvolver seu próprio filtro com a ferramenta inclusa na interface pode não conseguir um resultado satisfatório por não haver suporte nativo para sinais de EMG ou ECG, por exemplo. Assim, uma implementação que se pode realizar futuramente é a inclusão de mais filtros específicos para outros tipos de sinais biológicos.

Por fim, para averiguar a usabilidade do *software* desenvolvido com usuários reais, pode-se realizar com um grupo de potenciais usuários das áreas de Biotecnologia uma avaliação com SUS - *System Usability Scale*, para avaliar quantitativamente a facilidade de uso da ferramenta pelas pessoas para as quais a interface foi projetada.

6 REFERÊNCIAS BIBLIOGRÁFICAS

OPPENHEIM, Alan V.; SCHAFER, Ronald W.; BUCK, John R. **Discrete-Time Signal Processing**. 1999.

COOLEY, James W.; TUKEY, John W. An algorithm for the machine calculation of complex Fourier series. **Mathematics of computation**, v. 19, n. 90, p. 297-301, 1965.

EYRE, Jennifer; BIER, Jeff. The evolution of DSP processors. **IEEE Signal Processing Magazine**, v. 17, n. 2, p. 43-51, 2000.

PROAKIS, John G.; MANOLAKIS, Dimitris G. Digital Signal Processing, principles, algorithms, and applications. **Pentice Hall**, 1996.

USAKLI, Ali Bulent; **Improvement of EEG Signal Acquisition: An Electrical Aspect for State of the Art of Front End;** Hindawi Publishing Corporation Computational, Intelligence and Neuroscience; v. 2010, 7p, 2010, disponível em: <http://downloads.hindawi.com/journals/cin/2010/630649.pdf>, Acessado em 8 de outubro de 2015.

KUO, Sen M.; LEE, Bob H.; TIAN, Wenshun. **Real-Time Digital Signal Processing: Fundamentals, Implementations and Applications**. John Wiley & Sons, 2013.

TOMPKINS, Willis J. **Biomedical digital signal processing**. Editorial Prentice Hall, 1993.

LOTERIO, F. A. et al. AVALIAÇÃO DA APLICABILIDADE DE ANDADOR ROBÓTICO PARA IN-DIVÍDUOS HEMIPARÉTICOS ATRAVÉS DE ELETROMIOGRAFIA, **XXIV Congresso Brasileiro de Engenharia** Biomédica – CBEB, 2014

SWARTZ CENTER FOR COMPUTATIONAL NEUROSCIENCE - EEGLAB, an open source environment for electrophysiological signal processing, disponível em: < http://sccn.ucsd.edu/eeglab/>. Acessado em 28 de Outubro de 2015

FIELDTRIP WIKI, DISPONÍVEL EM <http://www.fieldtriptoolbox.org>, Acessado em 28 de Outubro de 2015;

HAYKIN, SIMON S.; VAN VEEN, Barry. **Sinais e sistemas**. Bookman, 2001.

WEEKS , Michael. Digital Signal Processing Using Matlab and Wavelets. **Pearson publications, ISBN–81-297-0272-X**, v. 2, n. 13, 2011.

NAÏT-ALI, Amine (Ed.). **Advanced biosignal processing**. Springer Science & Business Media, 2009.

SANEI, Saeid; CHAMBERS, J A; **EEG Signal Processing**, 1st ed., West Sussex, John Wiley & Sons, 2007

CATON, R, **The electric currents of the brain**, Br. Med. J., 2, 1875, 278.

WALTER, W. Grey. Slow potential waves in the human brain associated with expectancy, attention and decision. **European Archives of Psychiatry and Clinical Neuroscience**, v. 206, n. 3, p. 309-322, 1964.

LARSEN, Erik Andreas. Classification of EEG Signals in a Brain-Computer Interface System. 2011.

STERMAN, M. B.; FRIAR, L. Suppression of seizures in an epileptic following sensorimotor EEG feedback training. **Electroencephalography and clinical neurophysiology**, v. 33, n. 1, p. 89-95, 1972.

MITRA, Sanjit Kumar; KUO, Yonghong. **Digital signal processing: a computer-based approach**. New York: McGraw-Hill, 2006.

PFURTSCHELLER, Gert; FLOTZINGER, Doris; NEUPER, Christa. Differentiation between finger, toe and tongue movement in man based on 40 Hz EEG. **Electroencephalography and clinical neurophysiology**, v. 90, n. 6, p. 456-460, 1994.

HAMMOND, D. Corydon. What is neurofeedback?. **Journal of Neurotherapy**, v. 10, n. 4, p. 25-36, 2007.

ZHU, Danhua et al. A survey of stimulation methods used in SSVEP-based BCIs. **Computational intelligence and neuroscience**, v. 2010, p. 1, 2010.

PRABHU, K. M. M. **Window functions and their applications in Signal Processing**. CRC Press, 2013.

CORREA, M. Agustina Garcés; LEBER, Eric Laciar. **Noise removal from EEG signals in polisomnographic records applying adaptive filters in cascade**. INTECH Open Access Publisher, 2011.

VIGON, L. et al. Quantitative evaluation of techniques for ocular artefact filtering of EEG waveforms. **IEE Proceedings-Science, Measurement and Technology**, v. 147, n. 5, p. 219-228, 2000.

NIEDERMEYER, Ernst; DA SILVA, FH Lopes (Ed.).**Electroencephalography: basic principles, clinical applications, and related fields**. Lippincott Williams & Wilkins, 2005.

ANEEL, RESOLUÇÃO NORMATIVA Nº 398, DE 23 DE MARÇO DE 2010.

RORABAUGH, C. Britton. **Digital filter designer's handbook: featuring C routines**. McGraw-Hill Professional, 1993.

GUIÑÓN, José Luis et al. Moving average and Savitzki-Golay smoothing filters using Mathcad. In: **International Conference on Engineering Education**. 2007. p. 3-7.

MOURIÑO, Josep et al. Spatial filtering in the training process of a brain computer interface. In: **Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE**. IEEE, 2001. p. 639-642.

MCFARLAND, Dennis J. et al. Spatial filter selection for EEG-based communication. **Electroencephalography and clinical Neurophysiology**, v. 103, n. 3, p. 386-394, 1997.

GARCIA-MOLINA, Gary; ZHU, Danhua. Optimal spatial filtering for the steady state visual evoked potential: BCI application. In: **Neural Engineering (NER), 2011 5th International IEEE/EMBS Conference on**. IEEE, 2011. p. 156-160.

RAUDYS, Aistis; LENČIAUSKAS, Vaidotas; MALČIUS, Edmundas. Moving Averages for Financial Data Smoothing. In: **Information and Software Technologies**. Springer Berlin Heidelberg, 2013. p. 34-45.

TANDONNET, C. et al. Spatial enhancement of EEG traces by surface Laplacian estimation: comparison between local and global methods. **Clinical Neurophysiology**, v. 116, n. 1, p. 18-24, 2005.

PETERS, Jurriaan M. et al. **Brain functional networks in syndromic and non-syndromic autism: a graph theoretical study of EEG connectivity**. BMC medicine, v. 11, n. 1, p. 54, 2013.

BELMONT, M. R.; HOTCHKISS, A. J. Generalized cross-correlation functions for engineering applications, part i: basic theory. **Journal of applied mechanics**, v. 64, n. 2, p. 321-326, 1997.

LACHAUX, Jean-Philippe et al. Measuring phase synchrony in brain signals.Human brain mapping, v. 8, n. 4, p. 194-208, 1999.

QUIROGA, R. Quian et al. Phase locking of event-related alpha oscillations.**Chaos in Brain**, p. 301-304, 2000.

7 APÊNDICE A – CÓDIGO DAS INTERFACES

7.1 Interface principal

```
function varargout = Interface(varargin)
   gui_Singleton = 1;
   gui_State = struct('gui_Name',       mfilename, ...
                      'gui_Singleton',  gui_Singleton, ...
                      'gui_OpeningFcn', @Interface_OpeningFcn, ...
                      'gui_OutputFcn',  @Interface_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

function Interface_OpeningFcn(hObject, eventdata, handles, varargin)
   handles.Interface = hObject;
   guidata(hObject, handles);

function varargout = Interface_OutputFcn(hObject, eventdata, handles)
   varargout{1} = handles.Interface;

function BotaoRestaurar_Callback(hObject, eventdata, handles)
   if evalin('base','exist ("data")')==1
      evalin('base','data=data_bkp;');
   else
      warndlg('Dados nï¿½o carregados!');
   end;

function BotaoGuardar_Callback(hObject, eventdata, handles)
   [file,path] = uiputfile('*.mat','Salvar trabalho');
   FileName=strcat(path,file);
   assignin('base','FileName',FileName);
   evalin('base','save(FileName);');
   evalin('base','clear FileName;');
   clear file;
   clear path;

function BotaoFechar_Callback(hObject, eventdata, handles)
   evalin('base','clear all');
   evalin('base','close all');
   evalin('base','clc');
   evalin('base','close Interface');
```

```
function BotaoSair_Callback(hObject, eventdata, handles)
   evalin('base','close Interface');

function SelecionaCanal_Callback(hObject, eventdata, handles)

function SelecionaCanal_CreateFcn(hObject, eventdata, handles)
   if evalin('base','exist ("ListaCanais")')
      ListaCanais=evalin('base','ListaCanais');
      n=size(ListaCanais,1);
      Canais{1,1}='Todos';
      for i=1:n
         Canais{i+1,1}=ListaCanais(i,:);
      end
      set(hObject,'String',Canais);
   else
      set(hObject,'String','Selecionar');
   end
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
   end

function BotaoVisualizar_Callback(hObject, eventdata, handles)
   if evalin('base','exist ("data");')==0
      warndlg('Dados não carregados!');
      return
   else
      %NOP
   end
   Canal=get(handles.SelecionaCanal, 'Value');
   Canal=Canal-1;
   data=evalin('base','data');
   if Canal==0
      plot(data);
   else
      plot(data(:,Canal));
   end
   axis('tight');
   grid on;
   xlabel('Amostra');
   ylabel('Amplitude');

function BotaoCoerencia_Callback(hObject, eventdata, handles)
   run ('InterfaceCoerencia.m')

function BotaoPLF_Callback(hObject, eventdata, handles)
   run ('InterfacePLF.m')

function BotaoCorrelacao_Callback(hObject, eventdata, handles)
   run ('InterfaceCorrelacao.m')

function BotaoAnaliseTemporal_Callback(hObject, eventdata, handles)
```

```
    run ('InterfaceAnaliseTempo.m');

function BotaoFiltraEspacial_Callback(hObject, eventdata, handles)
    run ('InterfaceFiltroEspacial.m')

function BotaoFiltraFrequencia_Callback(hObject, eventdata, handles)
    run ('InterfaceFiltro.m')

function BotaoCarregar_Callback(hObject, eventdata, handles)
    run ('InterfaceCarga.m')

function BotaoAnaliseFrequencia_Callback(hObject, eventdata, handles)
    run ('InterfaceAnaliseFrequencia.m')


function BotaoAnaliseEnergia_Callback(hObject, eventdata, handles)
    run ('InterfaceEnergia.m')

function BotaoFiltraEspacial_CreateFcn(hObject, eventdata, handles)
    if(evalin('base','exist("data");'))==1
        set(hObject,'Enable','on');
    else
        set(hObject,'Enable','off');
    end

function BotaoFiltraFrequencia_CreateFcn(hObject, eventdata, handles)
if(evalin('base','exist("data");'))==1
    set(hObject,'Enable','on');
else
    set(hObject,'Enable','off');
end

function BotaoCoerencia_CreateFcn(hObject, eventdata, handles)
    if(evalin('base','exist("data");'))==1
        set(hObject,'Enable','on');
    else
        set(hObject,'Enable','off');
    end

function BotaoPLF_CreateFcn(hObject, eventdata, handles)
if(evalin('base','exist("data");'))==1
    set(hObject,'Enable','on');
else
    set(hObject,'Enable','off');
end

function BotaoCorrelacao_CreateFcn(hObject, eventdata, handles)
if(evalin('base','exist("data");'))==1
    set(hObject,'Enable','on');
else
    set(hObject,'Enable','off');
end
```

```
function BotaoAnaliseTemporal_CreateFcn(hObject, eventdata, handles)
if(evalin('base','exist("data");'))==1
   set(hObject,'Enable','on');
else
   set(hObject,'Enable','off');
end

function BotaoAnaliseFrequencia_CreateFcn(hObject, eventdata, handles)
if(evalin('base','exist("data");'))==1
   set(hObject,'Enable','on');
else
   set(hObject,'Enable','off');
end

function BotaoAnaliseEnergia_CreateFcn(hObject, eventdata, handles)
if(evalin('base','exist("data");'))==1
   set(hObject,'Enable','on');
else
   set(hObject,'Enable','off');
end
```

7.2 Interface de carga

```
function varargout = InterfaceCarga(varargin)
   gui_Singleton = 1;
   gui_State = struct('gui_Name',       mfilename, ...
                      'gui_Singleton',  gui_Singleton, ...
                      'gui_OpeningFcn', @InterfaceCarga_OpeningFcn, ...
                      'gui_OutputFcn',  @InterfaceCarga_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

function InterfaceCarga_OpeningFcn(hObject, eventdata, handles, varargin)
   handles.Carga = hObject;
   guidata(hObject, handles);

function varargout = InterfaceCarga_OutputFcn(hObject, eventdata, handles)
   varargout{1} = handles.Carga;

function fs_Callback(hObject, eventdata, handles)
```

```
fs=get(hObject,'String');
fs=str2double(fs)
assignin('base','fs',fs);

function fs_CreateFcn(hObject, eventdata, handles)
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
   end

function nometeste_Callback(hObject, eventdata, handles)

function nometeste_CreateFcn(hObject, eventdata, handles)
   set(hObject,'String',['Teste-',date]);
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
   end

function nomepaciente_Callback(hObject, eventdata, handles)

function nomepaciente_CreateFcn(hObject, eventdata, handles)
   set(hObject,'String',['Paciente-',date]);
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
   end

function BotaoCarregar_Callback(hObject, eventdata, handles)
   load(uigetfile)
   NomeDados=get(handles.NomeDados,'string');
   data=eval(NomeDados);

   if size(data,1)<100
      data=data';
   else
      size(data,1)
   end;
   NumeroCanais=size(data,2);

   %Envia dados para workspace
   assignin('base','data',data);
   assignin('base','NumeroCanais',NumeroCanais);

function BotaoOK_Callback(hObject, eventdata, handles)
   %Avalia se os dados foram carregados
   if evalin('base','exist ("data");')==0
      warndlg('Dados não carregados!');
      return
   else
      %NOP
   end
   data=evalin('base','data');
   fs=get(handles.fs,'String');
   fs=str2double(fs);
```

```
if (isnan(fs)==1)
   Help = questdlg('Utilizar valor padrão para amostragem (128Hz)?', ...
   'Aviso', ...
   'Sim','Não','Sim');
% Handle response
   switch Help
      case 'Sim'
      fs=128;

      case 'Não'
         return;
   end
else
   fs=str2double(get(handles.fs,'String'));
end;
   assignin('base','fs',fs);
   nometeste=get(handles.nometeste,'String');
   assignin('base','nometeste',nometeste);

   nomepaciente=get(handles.nomepaciente,'String');
   assignin('base','nomepaciente',nomepaciente);


datanascimento=[get(handles.dianascimento,'String'),'/',get(handles.mesnascimento,'String'),'/',get(handles.anona
scimento,'String')];
   assignin('base','datanascimento',datanascimento);

   datateste=get(handles.diateste,'String');
   datateste=[get(handles.diateste,'String'),'/',get(handles.mesteste,'String'),'/',get(handles.anoteste,'String')];

   if get(handles.FiltraFreqZero,'Value')==1
         watchon
         drawnow
         if(evalin('base','exist("fdata")'))==1
            data=evalin('base','fdata');
         else
            Filtro = designfilt('highpassfir', 'StopbandFrequency', 0.2, 'PassbandFrequency', 0.5,
'StopbandAttenuation', 60, 'PassbandRipple', 0.3, 'SampleRate', fs);
            data=filtfilt(Filtro,data);
         end
         watchoff
         drawnow
      else
         %NOP
   end
Tempo=(size(data,1)/fs);
assignin('base','Tempo',Tempo);
assignin('base','data',data);
evalin('base','clc');
evalin('base','close Interface');
evalin('base','run("Interface.m");');
evalin('base','close InterfaceCarga');
```

```
function BotaoCancelar_Callback(hObject, eventdata, handles)
   clc;
   evalin('base','close InterfaceCarga');

function BotaoVisualizar_Callback(hObject, eventdata, handles)
   watchon
   drawnow
   if evalin('base','exist ("data");')==0
      watchoff;
      warndlg('Dados não carregados!');
      return;
   else
      %NOP
   end
   data = evalin('base','data');
   if get(handles.FiltraFreqZero,'Value')==1
      if (isnan(str2double(get(handles.fs,'String')))==1)
         Help = questdlg('Utilizar valor padrão para amostragem (128Hz)?', ...
         'Aviso', ...
         'Sim','Não','Sim');
      % Handle response
         switch Help
            case 'Sim'
               fs=128;
               set(handles.fs,'String','128');

            case 'Não'
               watchoff
               warndlg('Necessário valor da frequência de amostragem para aplicar pré-filtro!')
               return;
         end
      else
         fs=str2double(get(handles.fs,'String'));
      end
      %Filtra componentes de frequencia 0
      Filtro = designfilt('highpassfir', 'StopbandFrequency', 0.2, 'PassbandFrequency', 0.5, 'StopbandAttenuation',
60, 'PassbandRipple', 0.3, 'SampleRate', fs);
      data=filtfilt(Filtro,data);
      assignin('base','fdata',data);
   else
      %NOP
   end;
   Canal = get(handles.CanalVisualizacao,'Value')-1;
   if Canal==0
      plot(data);
   else
      plot(data(:,Canal));
   end
   axis('tight');
   grid on;
   xlabel('Amostra');
```

```
    ylabel('Amplitude');
    watchoff

function FiltraFreqZero_Callback(hObject, eventdata, handles)

function CanalVisualizacao_Callback(hObject, eventdata, handles)

function CanalVisualizacao_CreateFcn(hObject, eventdata, handles)
    if evalin('base','exist (''ListaCanais'')')
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP;
    end
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function BotaoAjudaAmostragem_Callback(hObject, eventdata, handles)
    run('InterfaceAmostragem.m');

function diateste_Callback(hObject, eventdata, handles)

function diateste_CreateFcn(hObject, eventdata, handles)
    temp=date;
    temp=temp(1:2);
    set(hObject,'String',temp);
    clear temp;
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function mesteste_Callback(hObject, eventdata, handles)

function mesteste_CreateFcn(hObject, eventdata, handles)
    temp=date;
    temp=temp(4:6);
    switch temp
        case 'Jan'
            temp='01';
        case 'Fev'
            temp='02';
        case 'Mar'
            temp='03';
        case 'Apr'
            temp='04';
        case 'May'
```

```
        temp='05';
    case 'Jun'
        temp='06';
    case 'Jul'
        temp='07';
    case 'Ago'
        temp='08';
    case 'Set';
        temp='09';
    case 'Oct'
        temp='10';
    case 'Nov'
        temp='11';
    case 'Dez'
        temp='12';
end;
set(hObject,'String',temp);
clear temp;
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function anoteste_Callback(hObject, eventdata, handles)

function anoteste_CreateFcn(hObject, eventdata, handles)
temp=date;
temp=temp(8:11);
set(hObject,'String',temp);
clear temp;
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function dianascimento_Callback(hObject, eventdata, handles)

function dianascimento_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function mesnascimento_Callback(hObject, eventdata, handles)

function mesnascimento_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function anonascimento_Callback(hObject, eventdata, handles)

function anonascimento_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
    end

function NomeDados_Callback(hObject, eventdata, handles)

function NomeDados_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end


% --- Executes on selection change in MenuEquipamento.
function MenuEquipamento_Callback(hObject, eventdata, handles)
    Equipamento=(get(hObject,'Value'));
    assignin('base','Equipamento',Equipamento);
    load('BrainChannels.mat');
    if evalin('base','exist("Equipamento")')==1
        Equipamento=evalin('base','Equipamento');
        load('BrainChannels.mat');
        if Equipamento==1
            Canais=[AF3; F7; F3; FC5; T7; P7; O1; O2; P8; T8; FC6; F4; F8; AF4];
            ListaCanais=['Canal 1 '; 'Canal 2 '; 'Canal 3 '; 'Canal 4 '; 'Canal 5 '; 'Canal 6 '; 'Canal 7 '; 'Canal 8 '; 'Canal
9 '; 'Canal 10'; 'Canal 11'; 'Canal 12'; 'Canal 13'; 'Canal 14'];
            assignin('base','Canais',Canais);
            assignin('base','ListaCanais',ListaCanais);
        elseif Equipamento==2
            Canais=[AF3; F7; F3; FC5; T7; P7; O1; O2; P8; T8; FC6; F4; F8; AF4];
            ListaCanais=['AF3'; 'F7 '; 'F3 '; 'FC5'; 'T7 '; 'P7 '; 'O1 '; 'O2 '; 'P8 '; 'T8 '; 'FC6'; 'F4 '; 'F8 '; 'AF4'];
            assignin('base','Canais',Canais);
            assignin('base','ListaCanais',ListaCanais);

        elseif Equipamento==3
            Canais=[FP2;FP1;F8;F4;Fz;F3;F7;A2;C6;C4;Cz;C3;C5;A1;P8;P4;Pz;P3;P7;O2;Oz;O1];
            ListaCanais=['FP2';'FP1';'F8 ';'F4 ';'Fz ';'F3 ';'F7 ';'A2 ';'C6 ';'C4 ';'Cz ';'C3 ';'C5 ';'A1 ';'P8 ';'P4 ';'Pz ';'P3
';'P7 ';'O2 ';'Oz ';'O1 '];
            assignin('base','Canais',Canais);
            assignin('base','ListaCanais',ListaCanais);

        elseif Equipamento==4
            run ('InterfaceSelecionaCanais.m')
        end
    else
        Equipamento=get(hObject,'Value');
        assignin('base','Equipamento',Equipamento);
    end
    n=size(ListaCanais,1);
        Lista{1,1}='Todos';
        for i=1:n
            Lista{i+1,1}=ListaCanais(i,:);
        end
    set(handles.CanalVisualizacao,'String',Lista);

function MenuEquipamento_CreateFcn(hObject, eventdata, handles)
```

```
   if evalin('base','exist("Equipamento")')==1
      Equipamento=evalin('base','Equipamento');
      load('BrainChannels.mat');
      if Equipamento==1
         Canais=[AF3; F7; F3; FC5; T7; P7; O1; O2; P8; T8; FC6; F4; F8; AF4];
         ListaCanais=['Canal 1 '; 'Canal 2 '; 'Canal 3 '; 'Canal 4 '; 'Canal 5 '; 'Canal 6 '; 'Canal 7 '; 'Canal 8 '; 'Canal
9 '; 'Canal 10'; 'Canal 11'; 'Canal 12'; 'Canal 13'; 'Canal 14'];
         assignin('base','Canais',Canais);
         assignin('base','ListaCanais',ListaCanais);
      elseif Equipamento==2
         Canais=[AF3; F7; F3; FC5; T7; P7; O1; O2; P8; T8; FC6; F4; F8; AF4];
         ListaCanais=['AF3'; 'F7 '; 'F3 '; 'FC5'; 'T7 '; 'P7 '; 'O1 '; 'O2 '; 'P8 '; 'T8 '; 'FC6'; 'F4 '; 'F8 '; 'AF4'];
         assignin('base','Canais',Canais);
         assignin('base','ListaCanais',ListaCanais);

      elseif Equipamento==3
         Canais=[FP2;FP1;F8;F4;Fz;F3;F7;A2;C6;C4;Cz;C3;C5;A1;P8;P4;Pz;P3;P7;O2;Oz;O1];
         ListaCanais=['FP2';'FP1';'F8 ';'F4 ';'Fz ';'F3 ';'F7 ';'A2 ';'C6 ';'C4 ';'Cz ';'C3 ';'C5 ';'A1 ';'P8 ';'P4 ';'Pz ';'P3
';'P7 ';'O2 ';'Oz ';'O1 '];
         assignin('base','Canais',Canais);
         assignin('base','ListaCanais',ListaCanais);

      elseif Equipamento==4
         run ('InterfaceSelecionaCanais.m')
      end
   else
      Equipamento=get(hObject,'Value');
      assignin('base','Equipamento',Equipamento);
   end
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

7.3 Interface Coerência

```
function varargout = InterfaceCoerencia(varargin)
   gui_Singleton = 1;
   gui_State = struct('gui_Name',       mfilename, ...
                      'gui_Singleton',  gui_Singleton, ...
                      'gui_OpeningFcn', @InterfaceCoerencia_OpeningFcn, ...
                      'gui_OutputFcn',  @InterfaceCoerencia_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

function InterfaceCoerencia_OpeningFcn(hObject, eventdata, handles, varargin)
handles.Coerencia = hObject;
guidata(hObject, handles);

function varargout = InterfaceCoerencia_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.Coerencia;

function BotaoCancelar_Callback(hObject, eventdata, handles)
clc;
evalin('base','close InterfaceCoerencia');

function BotaoCalcular_Callback(hObject, eventdata, handles)
watchon
drawnow
ListaCanais=evalin('base','ListaCanais');
data=evalin('base','data');
n=size(data,2);
fs=evalin('base','fs');
Ch1=get(handles.MenuCanal1,'Value')-1;
Ch2=get(handles.MenuCanal2,'Value')-1;
if get(handles.CustomRange,'Value')==1
      FreqSuperior=get(handles.LimiteSuperior,'String');
      FreqSuperior=str2double(FreqSuperior);

      FreqInferior=get(handles.LimiteInferior,'String');
      FreqInferior=str2double(FreqInferior);
else
   if evalin('base','exist("FreqSuperior")')==1
      FreqSuperior=evalin('base','FreqSuperior');
      FreqInferior=evalin('base','FreqInferior');
   else
      FreqSuperior=fs/2;
      FreqInferior=0;
   end
end
if (Ch1~=0)&(Ch2~=0)

   [CoerenciaXY,Fxy]=Coerencia(data(:,Ch1),data(:,Ch2),fs);
   FreqSuperior=round((2*FreqSuperior/fs)*length(CoerenciaXY));
   FreqInferior=round((2*FreqInferior/fs)*length(CoerenciaXY))+1;
   CoerenciaXY=CoerenciaXY(FreqInferior:FreqSuperior);

   CoerenciaXY=MediaMovel(CoerenciaXY);

   Fxy=Fxy(FreqInferior:FreqSuperior);
   plot(Fxy,CoerenciaXY);
   axis('tight');
   xlabel('Frequência (Hz)');
```

```
    ylabel('Coerência');

    assignin('base','CoerenciaXY',CoerenciaXY);
    CoerenciaMedia=mean(CoerenciaXY);
    CoerenciaMedia=num2str(CoerenciaMedia);
    set(handles.ValorCoerenciaMedia,'string',CoerenciaMedia);
    Maximo=max(CoerenciaXY);
    set(handles.ValorMaximo,'string',Maximo);
    Minimo=min(CoerenciaXY);
    set(handles.ValorMinimo,'string',Minimo);
    Moda=mode(CoerenciaXY);
    set(handles.ValorModa,'string',Moda);
    DP=std(CoerenciaXY);
    set(handles.ValorDP,'string',DP);
    Variancia=var(CoerenciaXY);
    set(handles.ValorVariancia,'string',Variancia);
    Mediana=median(CoerenciaXY);
    set(handles.ValorMediana,'string',Mediana);

elseif (Ch1==0)&(Ch2==0)
    FreqSup=FreqSuperior;
    FreqInf=FreqInferior;
    CoerenciaXY=zeros(n);
    for i=1:1:n
       for j=(i+1):1:n
          if i~=j
          Temp=Coerencia(data(:,i),data(:,j),fs);
          FreqSuperior=round((2*FreqSup/fs)*length(Temp));
          FreqInferior=round((2*FreqInf/fs)*length(Temp))+1;
          Temp=Temp(FreqInferior:FreqSuperior);
          Temp=MediaMovel(Temp);
          CoerenciaXY(i,j)=mean(Temp);
          end
       end
    end
    assignin('base','CoerenciaXY',CoerenciaXY);
    Limiar=get(handles.SelecionaLimiar,'Value');
    set(handles.ValorCoerenciaMedia,'string','ND');
    set(handles.ValorMaximo,'string','ND');
    set(handles.ValorMinimo,'string','ND');
    set(handles.ValorModa,'string','ND');
    set(handles.ValorDP,'string','ND');
    set(handles.ValorVariancia,'string','ND');
    set(handles.ValorMediana,'string','ND');
    axes(handles.axes2);
    cla;
    PlotBrainMap (CoerenciaXY,ListaCanais,Limiar);

else
    watchoff
    drawnow
    warndlg('Selecione os canais corretamente!');
```

```
    end
    watchoff

function MenuCanal1_Callback(hObject, eventdata, handles)

function MenuCanal1_CreateFcn(hObject, eventdata, handles)
    if evalin('base','exist("ListaCanais")')==1
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP
    end
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function MenuCanal2_Callback(hObject, eventdata, handles)

function MenuCanal2_CreateFcn(hObject, eventdata, handles)
    if evalin('base','exist("ListaCanais")')==1
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP
    end
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function ValorCoerenciaMedia_Callback(hObject, eventdata, handles)

function ValorCoerenciaMedia_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function BotaoOK_Callback(hObject, eventdata, handles)
    clc;
    evalin('base','close InterfaceCoerencia');

function ValorMaximo_Callback(hObject, eventdata, handles)
```

```
function ValorMaximo_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function ValorMinimo_Callback(hObject, eventdata, handles)

function ValorMinimo_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function ValorModa_Callback(hObject, eventdata, handles)

function ValorModa_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function ValorDP_Callback(hObject, eventdata, handles)

function ValorDP_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function ValorVariancia_Callback(hObject, eventdata, handles)

function ValorVariancia_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function ValorMediana_Callback(hObject, eventdata, handles)

function ValorMediana_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function CustomRange_Callback(hObject, eventdata, handles)

function LimiteSuperior_Callback(hObject, eventdata, handles)

function LimiteSuperior_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function LimiteInferior_Callback(hObject, eventdata, handles)

function LimiteInferior_CreateFcn(hObject, eventdata, handles)
```

```
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
   end

function SelecionaLimiar_Callback(hObject, eventdata, handles)
   set(handles.Limiar,'String',num2str(get(hObject,'Value')));

function SelecionaLimiar_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function Limiar_Callback(hObject, eventdata, handles)
   if isnan(str2double(get(hObject,'String')))
      warndlg('Apenas números de 0 a 1 são válidos!');
      set(hObject,'String',num2str(0));
   elseif str2double(get(hObject,'String'))>1
      set(hObject,'String',num2str(1));
   elseif str2double(get(hObject,'String'))<0
      set(hObject,'String',num2str(0));
   else
      set(handles.SelecionaLimiar,'Value',str2double(get(hObject,'String')));
   end
function Limiar_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

7.4 Interface de Correlação

```
function varargout = InterfaceCorrelacao(varargin)
   gui_Singleton = 1;
   gui_State = struct('gui_Name',       mfilename, ...
                      'gui_Singleton',  gui_Singleton, ...
                      'gui_OpeningFcn', @InterfaceCorrelacao_OpeningFcn, ...
                      'gui_OutputFcn',  @InterfaceCorrelacao_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

function InterfaceCorrelacao_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
    handles.Correlacao = hObject;
    guidata(hObject, handles);

function varargout = InterfaceCorrelacao_OutputFcn(hObject, eventdata, handles)
    varargout{1} = handles.Correlacao;

function MenuCanalA_Callback(hObject, eventdata, handles)

function MenuCanalA_CreateFcn(hObject, eventdata, handles)
    if evalin('base','exist("ListaCanais")')==1
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP
    end
    set(hObject,'String',Canais);
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function MenuCanalB_Callback(hObject, eventdata, handles)

function MenuCanalB_CreateFcn(hObject, eventdata, handles)
    if evalin('base','exist("ListaCanais")')==1
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP
    end
    set(hObject,'String',Canais);
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function BotaoCalcular_Callback(hObject, eventdata, handles)
    data=evalin('base','data');
    n=size(data,2);
    ListaCanais=evalin('base','ListaCanais');
    CanalA=get(handles.MenuCanalA,'Value')-1;
    CanalB=get(handles.MenuCanalB,'Value')-1;

    if (CanalA==0)&(CanalB==0)
```

```
        Correlacao=zeros(n);
          for i=1:1:n
            for j=(i+1):1:n
              if i~=j
                Correlacao(i,j)=xcorr(data(:,i),data(:,j),0,'coeff');
              else
                %NOP
              end
            end
          end
          Limiar=get(handles.SelecionaLimiar,'Value');
          Correlacao=Correlacao+abs(min(min(Correlacao)));
          Correlacao=Correlacao/(max(max(Correlacao)));
          axes(handles.axes1);
          cla;
          PlotBrainMap (Correlacao,ListaCanais,Limiar);

    elseif (CanalA==0)&(CanalB~=0)||(CanalA~=0)&(CanalB==0)
        warndlg ('Selecionar combinação de canais válida!');
        return;
    else
        Correlacao=xcorr(data(:,CanalA),data(:,CanalB),0,'coeff');
        set(handles.ValorCorrelacao,'String',num2str(Correlacao));
    end;

function BotaoSair_Callback(hObject, eventdata, handles)
    evalin('base','close InterfaceCorrelacao');

function ValorCorrelacao_Callback(hObject, eventdata, handles)

function ValorCorrelacao_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function SelecionaLimiar_Callback(hObject, eventdata, handles)
set(handles.Limiar,'String',num2str(get(hObject,'Value')));

function SelecionaLimiar_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function Limiar_Callback(hObject, eventdata, handles)
    if isnan(str2double(get(hObject,'String')))
        warndlg('Apenas números de 0 a 1 são válidos!');
        set(hObject,'String',num2str(0));
    elseif str2double(get(hObject,'String'))>1
        set(hObject,'String',num2str(1));
    elseif str2double(get(hObject,'String'))<0
```

```
    set(hObject,'String',num2str(0));
  else
    set(handles.SelecionaLimiar,'Value',str2double(get(hObject,'String')));
  end
function Limiar_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
  set(hObject,'BackgroundColor','white');
end
```

7.5 Interface de Energia

```
function varargout = InterfaceEnergia(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @InterfaceEnergia_OpeningFcn, ...
                   'gui_OutputFcn',  @InterfaceEnergia_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

function InterfaceEnergia_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = InterfaceEnergia_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function BotaoCalcular_Callback(hObject, eventdata, handles)
  Canais=evalin('base','Canais');
  data=evalin('base','data');
  pRMS = CalculaPotenciaSinal (data);
  axes(handles.axes1)
  map=PlotBrainMap2(pRMS,Canais);
  axes(handles.axes2)
  set(gca,'Ydir','reverse')
  colormap(map);
  colorbar;
  set(handles.axes2,'visible','off')
```

```
   set(handles.axes2,'XtickLabel',[],'YtickLabel',[]);
   axes(handles.axes2)
   set(gca,'Ydir','Normal')
function BotaoFechar_Callback(hObject, eventdata, handles)
evalin('base','close InterfaceEnergia');

function MenuSelecionaCanal_Callback(hObject, eventdata, handles)

function MenuSelecionaCanal_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

7.6 Interface Filtro

```
function varargout = InterfaceFiltro(varargin)
   gui_Singleton = 1;
   gui_State = struct('gui_Name',       mfilename, ...
                      'gui_Singleton',  gui_Singleton, ...
                      'gui_OpeningFcn', @InterfaceFiltro_OpeningFcn, ...
                      'gui_OutputFcn',  @InterfaceFiltro_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

function InterfaceFiltro_OpeningFcn(hObject, eventdata, handles, varargin)
   evalin('base','Filtrado=0;');
   handles.Filtro = hObject;
   guidata(hObject, handles);

function varargout = InterfaceFiltro_OutputFcn(hObject, eventdata, handles)
   varargout{1} = handles.Filtro;

function BotaoOK_Callback(hObject, eventdata, handles)
   Filtrado=evalin('base','Filtrado');
   if Filtrado>0
      evalin ('base','if exist("fdata");clear fdata;end');
      evalin ('base','if exist("Filtro");clear Filtro;end');
      evalin ('base','if exist("Filtrado");clear Filtrado;end');
      evalin('base','close InterfaceFiltro');
   else
      Help = questdlg('Nenhum filtro foi aplicado, deseja fechar?', ...
```

```
        'Aviso', ...
        'Sim','Não','Sim');
    % Handle response
        switch Help
            case 'Sim'
                evalin('base','close InterfaceFiltro');
            case 'Não'
                return;
        end
        evalin ('base','if exist("fdata");clear fdata;end');
        evalin ('base','if exist("Filtro");clear Filtro;end');
        evalin ('base','if exist("Filtrado");clear Filtrado;end');
    end
function BotaoCancelar_Callback(hObject, eventdata, handles)
    evalin ('base','if exist("Filtro");clear Filtro;end');
    evalin ('base','if exist("fdata");clear fdata;end');
    evalin ('base','if exist("Filtrado");clear Filtrado;end');
    evalin('base','close InterfaceFiltro');

function BotaoVisualizaSinal_Callback(hObject, eventdata, handles)
    watchon
    drawnow
    data=evalin ('base','data');
    TipoFiltro=get(handles.MenuEscolheFiltro,'Value');
    Canal=get(handles.MenuCanal,'Value')-1;
    F=evalin('base','exist (''Filtro'');');
    if (F==1)&(TipoFiltro~=1)
        Filtro=evalin('base','Filtro');
        fdata=filtfilt(Filtro,data);
        if Canal==0
            plot(fdata);
        else
            plot(fdata(:,Canal));
        end;
        axis('tight');
        grid on;
        xlabel('Amostra');
        ylabel('Amplitude');
        text(round(length(data(:,1))/1.2),0,'PRE-VISUALIZAÇÃO-Filtro ainda não aplicado ao
sinal!','HorizontalAlignment','right');
    else
        if Canal==0
            plot(data);
        else
            plot(data(:,Canal));
        end;
        axis('tight');
        grid on;
        xlabel('Amostra');
        ylabel('Amplitude');
        text(round(length(data(:,1))/2.5),0,'Nenhnum filtro aplicado!');
    end;
```

```
    watchoff

function BotaoPeriodograma_Callback(hObject, eventdata, handles)
    watchon
    drawnow
    fs=evalin('base','fs');
    data=evalin('base','data');
    Canal=get(handles.MenuCanal,'Value')-1;
    TipoFiltro=get(handles.MenuEscolheFiltro,'Value');
    F=evalin('base','exist (''Filtro'');');

    if (TipoFiltro==1)|(F~=1)
        if Canal~=0
            [Pxx,f]=Periodograma(data,fs,Canal);
        else
            watchoff
            drawnow
            warndlg('Selecione um canal!');
            %NOP
        end
    else
        if evalin('base','exist (''Filtro'')')==1
            Filtro=evalin('base','Filtro');
            fdata=filtfilt(Filtro,data);
            if Canal==0
                watchoff
                drawnow
                warndlg('Selecione um canal!');
            else
                [Pxx,f]=Periodograma(fdata,fs,Canal);
            end
        else
            [Pxx,f]=Periodograma(data,fs,Canal);
    end;
    end
    if exist('Pxx')
        plot(f,Pxx);
        axis ('tight');
        grid on;
        xlabel('Frequência (Hz)');
        ylabel('Amplitude');
    else
        %NOP;
    end;
    watchoff

function MenuEscolheFiltro_Callback(hObject, eventdata, handles)
    watchon;
    drawnow;
    TipoFiltro=get(hObject,'Value');
    if TipoFiltro==4
        InterfaceFiltroCustom;
```

```
    elseif TipoFiltro==2
        evalin('base','Filtro=FiltroAlfa(fs);');
        FreqInferior=8;
        FreqSuperior=12;
        assignin('base','FreqInferior',FreqInferior);
        assignin('base','FreqSuperior',FreqSuperior);
    elseif TipoFiltro==3
        evalin('base','Filtro=FiltroBeta(fs);');
        FreqInferior=12.5;
        FreqSuperior=18;
        assignin('base','FreqInferior',FreqInferior);
        assignin('base','FreqSuperior',FreqSuperior);
    else
        %NOP;
    end;
    watchoff;
function MenuEscolheFiltro_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function BotaoVisualizaResposta_Callback(hObject, eventdata, handles)
    evalin('base','if exist("Filtro")==1;fvtool(Filtro);else;warndlg("Filtro não inicializado");end');

function BotaoAplicar_Callback(hObject, eventdata, handles)
    watchon;
    drawnow;
    TipoFiltro=get(handles.MenuEscolheFiltro,'Value');
    data=evalin('base','data');
    if (TipoFiltro==1)
        watchoff;
        warndlg('Selecione um filtro!');
    else
        Filtro=evalin('base','Filtro');
        fdata=filtfilt(Filtro,data);
        assignin('base','data',fdata);
        watchoff;
        msgbox('Filtro aplicado com sucesso!');
        evalin('base','Filtrado=Filtrado+1;');
    end;

function MenuCanal_Callback(hObject, eventdata, handles)

function MenuCanal_CreateFcn(hObject, eventdata, handles)
    if evalin('base','exist ("ListaCanais")')
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
```

```
    else
        %NOP;
    end
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
```

7.7 Interface Filtro Custom

```
function varargout = InterfaceFiltroCustom(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name',       mfilename, ...
                       'gui_Singleton',  gui_Singleton, ...
                       'gui_OpeningFcn', @InterfaceFiltroCustom_OpeningFcn, ...
                       'gui_OutputFcn',  @InterfaceFiltroCustom_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

function InterfaceFiltroCustom_OpeningFcn(hObject, eventdata, handles, varargin)
    handles.Custom = hObject;
    guidata(hObject, handles);

function varargout = InterfaceFiltroCustom_OutputFcn(hObject, eventdata, handles)
    varargout{1} = handles.Custom;

function BotaoCancelar_Callback(hObject, eventdata, handles)
    clc;
    delete(handles.Custom);

function Tipo_Callback(hObject, eventdata, handles)
if get(hObject,'Value')==1
    set(handles.Metodo,'Enable','off');
    set(handles.Resposta,'Enable','off');
    set(handles.FreqSuperior,'Enable','off');
    set(handles.FreqInferior,'Enable','off');
    set(handles.Metodo,'Value',1);
    set(handles.Resposta,'Value',1);
    set(handles.BotaoOK,'Enable','off');
    set(handles.BotaoVisualizar,'Enable','off');
elseif get(hObject,'Value')==2
    set(handles.Metodo,'Enable','off');
    set(handles.Resposta,'Enable','on');
```

```
elseif get(hObject,'Value')==3
   if(get(handles.Metodo,'Value')==1)||(get(handles.Metodo,'Value')==1)
      set(handles.BotaoOK,'Enable','off');
      set(handles.BotaoVisualizar,'Enable','off');
   end
   set(handles.Metodo,'Enable','on');
   set(handles.Resposta,'Enable','on');
   %set(handles.FreqSuperior,'Enable','off');
   %set(handles.FreqInferior,'Enable','off');
end
function Tipo_CreateFcn(hObject, eventdata, handles)
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
   end

function Resposta_Callback(hObject, eventdata, handles)
fs=evalin('base','fs');
if get(hObject,'Value')==1
   set(handles.FreqSuperior,'Enable','off');
   set(handles.FreqInferior,'Enable','off');
   set(handles.BotaoOK,'Enable','off');
   set(handles.BotaoVisualizar,'Enable','off');
   set(handles.Metodo,'Value',1);
elseif get(hObject,'Value')==2
   set(handles.FreqSuperior,'Enable','on');
   set(handles.FreqInferior,'Enable','off');
   set(handles.FreqSuperior,'String',0.98*fs/2);

elseif get(hObject,'Value')==3
   set(handles.FreqSuperior,'Enable','off');
   set(handles.FreqInferior,'Enable','on');
   set(handles.FreqInferior,'String',0.5);
else
   set(handles.FreqSuperior,'Enable','on');
   set(handles.FreqInferior,'Enable','on');
end

if (get(handles.Tipo,'Value')==2)&(get(hObject,'Value')>1)
   set(handles.BotaoOK,'Enable','on');
   set(handles.BotaoVisualizar,'Enable','on');
end
function Resposta_CreateFcn(hObject, eventdata, handles)
   set(hObject,'Enable','off');
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
   end

function Ripple_Callback(hObject, eventdata, handles)

function Ripple_CreateFcn(hObject, eventdata, handles)
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
```

```
    end

function Atenuacao_Callback(hObject, eventdata, handles)

function Atenuacao_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function FreqInferior_Callback(hObject, eventdata, handles)

function FreqInferior_CreateFcn(hObject, eventdata, handles)
    set(hObject,'Enable','off');
    set(hObject,'String',num2str(0.5));
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function FreqSuperior_Callback(hObject, eventdata, handles)

function FreqSuperior_CreateFcn(hObject, eventdata, handles)
    fs=evalin('base','fs');
    set(hObject,'Enable','off');
    set(hObject,'String',num2str((0.98*fs/2)));
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function BotaoVisualizar_Callback(hObject, eventdata, handles)
    watchon;
    drawnow;
    fs=evalin('base','fs;');
    fmax=0.98*fs/2;
    if (get(handles.Tipo,'Value'))==1
        watchoff;
        warndlg('Selecione os parametros do filtro!');
        return
    else
        %NOP
    end
    Metodo=evalin('base','Metodo');
    Tipo=get(handles.Tipo,'Value')-1;
    Caracteristica=get(handles.Resposta,'Value')-1;

    FreqInferior=str2double(get(handles.FreqInferior,'string'));
    FreqSuperior=str2double(get(handles.FreqSuperior,'string'));
    Ripple=str2double(get(handles.Ripple,'string'));
    Atenuacao=str2double(get(handles.Atenuacao,'string'));
    if((isnan(FreqInferior))|(isnan(FreqSuperior))|(isnan(Ripple))|(isnan(Atenuacao)))==0
        if (FreqInferior<0.5)|(FreqSuperior>fmax)
            watchoff
            fmax=num2str(fmax);
```

```
        warndlg(strcat('Uma das frequências está muito próxima do limite mínimo de 0,2Hz ou máximo
de',fmax));
        return
      else
        %NOP
      end
      Filtro=MontaFiltro(Tipo,Caracteristica,FreqInferior,FreqSuperior,Atenuacao,Ripple,fs,Metodo);
      watchoff
      fvtool(Filtro);
   else
      watchoff
      warndlg('Preencha os campos que faltam!');
   end

function BotaoOK_Callback(hObject, eventdata, handles)
      watchon;
   drawnow;
   fs=evalin('base','fs;');
   if (get(handles.Tipo,'Value'))==1
      watchoff;
      warndlg('Selecione os parametros do filtro!');
      return
   else
      %NOP
   end
   Metodo=evalin('base','Metodo');
   Tipo=get(handles.Tipo,'Value')-1;
   Caracteristica=get(handles.Resposta,'Value')-1;

   FreqInferior=str2double(get(handles.FreqInferior,'string'));
   FreqSuperior=str2double(get(handles.FreqSuperior,'string'));
   Ripple=str2double(get(handles.Ripple,'string'));
   Atenuacao=str2double(get(handles.Atenuacao,'string'));
   if((isnan(FreqInferior))|(isnan(FreqSuperior))|(isnan(Ripple))|(isnan(Atenuacao)))==0
      Filtro=MontaFiltro(Tipo,Caracteristica,FreqInferior,FreqSuperior,Atenuacao,Ripple,fs,Metodo);
      watchoff
   else
      watchoff
      warndlg('Preencha os campos que faltam!');
      return;
   end
      if(isstable(Filtro))==1
         assignin('base','FreqInferior',FreqInferior);
         assignin('base','FreqSuperior',FreqSuperior);
         assignin('base','Filtro',Filtro);
         evalin('base','clear Metodo');
         clc;
         watchoff
         evalin('base','close InterfaceFiltroCustom');
      else
         warndlg('Filtro não estável!');
      end
```

```
function Metodo_Callback(hObject, eventdata, handles)
   if get(hObject,'Value')==1
      Metodo=0;
      set(handles.BotaoOK,'Enable','off');
      set(handles.BotaoVisualizar,'Enable','off');
   elseif get(hObject,'Value')==2
      Metodo='butter';
      set(handles.BotaoOK,'Enable','on');
      set(handles.BotaoVisualizar,'Enable','on');
   elseif get(hObject,'Value')==3
      Metodo='cheby1';
      set(handles.BotaoOK,'Enable','on');
      set(handles.BotaoVisualizar,'Enable','on');
   elseif get(hObject,'Value')==4
      Metodo='cheby2';
      set(handles.BotaoOK,'Enable','on');
      set(handles.BotaoVisualizar,'Enable','on');
   elseif get(hObject,'Value')==5
      Metodo='ellip';
      set(handles.BotaoOK,'Enable','on');
      set(handles.BotaoVisualizar,'Enable','on');
   else
      %NOP
   end;
   assignin('base','Metodo',Metodo);
function Metodo_CreateFcn(hObject, eventdata, handles)
   Metodo=0;
   assignin('base','Metodo',Metodo);
   set(hObject,'Enable','off');
   if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
   end

function BotaoOK_CreateFcn(hObject, eventdata, handles)
set(hObject,'Enable','off');

function BotaoVisualizar_CreateFcn(hObject, eventdata, handles)
set(hObject,'Enable','off');
```

7.8 Interface Filtro Espacial

```
function varargout = InterfaceFiltroEspacial(varargin)
   gui_Singleton = 1;
   gui_State = struct('gui_Name',       mfilename, ...
                      'gui_Singleton',  gui_Singleton, ...
                      'gui_OpeningFcn', @InterfaceFiltroEspacial_OpeningFcn, ...
                      'gui_OutputFcn',  @InterfaceFiltroEspacial_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

function varargout = InterfaceFiltroEspacial_OutputFcn(hObject, eventdata, handles)
    varargout{1} = handles.FiltroEspacial;

function InterfaceFiltroEspacial_OpeningFcn(hObject, eventdata, handles, varargin)
    evalin('base','Filtrado=0;');
    handles.FiltroEspacial = hObject;
    guidata(hObject, handles);

function BotaoOK_Callback(hObject, eventdata, handles)
    Filtrado=evalin('base','Filtrado');
    if Filtrado>0
        evalin ('base','if exist("fdata");clear fdata;end');
        evalin('base','close InterfaceFiltroEspacial');
    else
        Help = questdlg('Nenhum filtro foi aplicado, deseja fechar?', ...
        'Aviso', ...
        'Sim','Não','Sim');
    % Handle response
        switch Help
            case 'Sim'
                evalin('base','close InterfaceFiltroEspacial');
            case 'Não'
                return;
        end
    end


function BotaoCancelar_Callback(hObject, eventdata, handles)
    evalin ('base','if exist("fdata");clear fdata;end');
    evalin('base','close InterfaceFiltroEspacial');

function MenuCAR_Callback(hObject, eventdata, handles)

function MenuCAR_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function BotaoAplicar_Callback(hObject, eventdata, handles)
    FiltroEspacial=get(handles.MenuCAR,'Value');
    if evalin('base','exist("data")')==1
        data=evalin('base','data');
    else
```

```
        warndlg('Dados não carregados!');
    end;

    if FiltroEspacial==1
        warndlg('Selecione um filtro!');
    elseif FiltroEspacial==2
        fdata=FiltroCAR(data);
        assignin('base','data',fdata);
        msgbox('Filtro CAR aplicado com sucesso!');
        evalin('base','Filtrado=Filtrado+1;');
    elseif FiltroEspacial==3
        fdata=filter2(fspecial('laplacian'),data);
        assignin('base','data',fdata);
        msgbox('Filtro Laplaciano aplicado com sucesso!');
        evalin('base','Filtrado=Filtrado+1;');
    end;

function MenuCanal_Callback(hObject, eventdata, handles)
if evalin('base','exist ("ListaCanais")')
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP;
    end
function MenuCanal_CreateFcn(hObject, eventdata, handles)
if evalin('base','exist ("ListaCanais")')
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP;
    end
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function BotaoVisualizaSinal_Callback(hObject, eventdata, handles)
    if evalin('base','exist("data")')==1
        data=evalin('base','data');
    else
        warndlg('Dados não carregados!');
    end;
    TipoFiltro=get(handles.MenuCAR,'Value');
```

```
   Canal=get(handles.MenuCanal,'Value')-1;
   if TipoFiltro==1
      fdata=data;
   elseif (TipoFiltro==2)
      fdata=FiltroCAR(data);
   elseif (TipoFiltro==3)
      fdata=filter2(fspecial('laplacian'),data);
   end;

   if Canal==0
      plot(fdata);
   else
      plot(fdata(:,Canal));
   end;
   axis('tight');
   grid on;
   xlabel('Amostra');
   ylabel('Amplitude');
```

7.9 Interface PLF

```
function varargout = InterfacePLF(varargin)
   gui_Singleton = 1;
   gui_State = struct('gui_Name',       mfilename, ...
                      'gui_Singleton',  gui_Singleton, ...
                      'gui_OpeningFcn', @InterfacePLF_OpeningFcn, ...
                      'gui_OutputFcn',  @InterfacePLF_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

function InterfacePLF_OpeningFcn(hObject, eventdata, handles, varargin)
   handles.PLFInterface = hObject;
   guidata(hObject, handles);

function varargout = InterfacePLF_OutputFcn(hObject, eventdata, handles)
   varargout{1} = handles.PLFInterface;

function MenuCH1_Callback(hObject, eventdata, handles)

function MenuCH1_CreateFcn(hObject, eventdata, handles)
   if evalin('base','exist("ListaCanais")')==1
```

```
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP
    end
    set(hObject,'String',Canais);
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function MenuCH2_Callback(hObject, eventdata, handles)

function MenuCH2_CreateFcn(hObject, eventdata, handles)
    if evalin('base','exist("ListaCanais")')==1
        ListaCanais=evalin('base','ListaCanais');
        n=size(ListaCanais,1);
        Canais{1,1}='Todos';
        for i=1:n
            Canais{i+1,1}=ListaCanais(i,:);
        end
        set(hObject,'String',Canais);
    else
        %NOP
    end
    set(hObject,'String',Canais);
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function BotaoCalcula_Callback(hObject, eventdata, handles)
    data=evalin('base','data');
    n=size(data,2);
    ListaCanais=evalin('base','ListaCanais');
    Ch1=get(handles.MenuCH1,'Value')-1;
    Ch2=get(handles.MenuCH2,'Value')-1;
    if (Ch1==0)&(Ch2==0)
        PLV=zeros(n);
        for i=1:1:n
            for j=(i+1):1:n
                if i~=j
                    PLV(i,j)=FuncaoPLF(data(:,i),data(:,j));
                else
                    %NOP
                end
            end
        end
        Limiar=get(handles.SelecionaLimiar,'Value');
```

```
        axes(handles.axes1);
        cla;
        PlotBrainMap (PLV,ListaCanais,Limiar);

    elseif (Ch1==0)&(Ch2~=0)||(Ch1~=0)&(Ch2==0)
        warndlg('Selecionar par de canais válido!');

    else
        PLV=FuncaoPLF(data(:,Ch1),data(:,Ch2));
        set(handles.ValorPLF,'String',num2str(PLV));
    end

function BotaoFechar_Callback(hObject, eventdata, handles)
    evalin('base','close InterfacePLF');
    clc;
    clear all;

function ValorPLF_Callback(hObject, eventdata, handles)

function ValorPLF_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function SelecionaLimiar_Callback(hObject, eventdata, handles)
    set(handles.Limiar,'String',num2str(get(hObject,'Value')));

function SelecionaLimiar_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function Limiar_Callback(hObject, eventdata, handles)
    if isnan(str2double(get(hObject,'String')))
        warndlg('Apenas números de 0 a 1 são válidos!');
        set(hObject,'String',num2str(0));
    elseif str2double(get(hObject,'String'))>1
        set(hObject,'String',num2str(1));
    elseif str2double(get(hObject,'String'))<0
        set(hObject,'String',num2str(0));
    else
        set(handles.SelecionaLimiar,'Value',str2double(get(hObject,'String')));
    end
function Limiar_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

7.10 Interface Seleção de Canais

```
function varargout = InterfaceSelecionaCanais(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @InterfaceSelecionaCanais_OpeningFcn, ...
                   'gui_OutputFcn',  @InterfaceSelecionaCanais_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

function InterfaceSelecionaCanais_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;
guidata(hObject, handles);

function varargout = InterfaceSelecionaCanais_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


function BotaoOK_Callback(hObject, eventdata, handles)
if (evalin('base','exist("ListaCanaisSelecionados")'))==1
   evalin('base','Canais=CanaisSelecionados;ListaCanais=ListaCanaisSelecionados;clear
CanaisSelecionados;clear ListaCanaisSelecionados;');
else
   warndlg('Canais nao selecionados!');
   return
end
evalin('base','close InterfaceSelecionaCanais');

function BotaoCH_Callback(hObject, eventdata, handles)
   Brain=imread('Brain.png');
   imshow(Brain);
   NumeroCanais=str2double(get(handles.NumeroCanais,'String'));
   [c,r,P] = impixel(Brain);
   CoordenadasCanais(:,1)=c;
   CoordenadasCanais(:,2)=r;
   if (size(CoordenadasCanais,1)~=NumeroCanais)
      warndlg('Numero de pontos selecionados diferente do numero de canais!');
   else
      load('BrainChannels.mat');
      IDX = knnsearch(Canais,CoordenadasCanais);
      for i=1:NumeroCanais
         CanaisSelecionados(i,:)=Canais(IDX(i),:);
         ListaCanaisSelecionados(i,:)=ListaCanais(IDX(i),:);
      end
   end
   set(handles.ListaCanais,'String',ListaCanaisSelecionados);
```

```
   assignin('base','ListaCanaisSelecionados',ListaCanaisSelecionados);
   assignin('base','CanaisSelecionados',CanaisSelecionados);


function NumeroCanais_Callback(hObject, eventdata, handles)

function NumeroCanais_CreateFcn(hObject, eventdata, handles)
if evalin('base','exist ("NumeroCanais")')
      NumeroCanais=evalin('base','NumeroCanais');
      set(hObject,'String',num2str(NumeroCanais));
else
   %NOP
end

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ListaCanais_Callback(hObject, eventdata, handles)

function ListaCanais_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

7.11 Função Média Móvel

```
function [VetorM]=MediaMovel(Vetor)
          Tamanho=length(Vetor);
          VetorM=Vetor;
          TFiltro=100;
          for i=1:1:Tamanho-TFiltro
                    for j=1:1:TFiltro
                              VetorM(i)=VetorM(i)+VetorM(j+i);
                    end
                    VetorM(i)=VetorM(i)/(TFiltro+1);
          end
return;
```

7.12 Função monta filtro

```
function
[FiltroMontado]=MontaFiltro(Tipo,Caracteristica,FreqInferior,FreqSuperior,Atenuacao,Ripple,Amostragem,Met
odo)

   %A partir do tipo e frequÃªncias de corte do filtro, projeta um filtro que pode ser utilizado para qualquer sinal

   if Tipo==1
      if Caracteristica==1
         FiltroMontado =
designfilt('lowpassfir','Passbandfrequency',0.99*FreqSuperior,'Stopbandfrequency',FreqSuperior,'StopbandAtten
uation', Atenuacao, 'PassbandRipple', Ripple,'SampleRate', Amostragem);
      elseif Caracteristica==2
```

```
        FiltroMontado =
designfilt('highpassfir','StopbandFrequency',FreqInferior,'PassbandFrequency',0.1+FreqInferior,'StopbandAttenu
ation', Atenuacao, 'PassbandRipple', Ripple,'SampleRate', Amostragem);
      elseif Caracteristica==3
        FiltroMontado =
designfilt('bandpassfir','StopbandFrequency1',FreqInferior,'PassbandFrequency1',0.1+FreqInferior,'PassbandFreq
uency2',0.99*FreqSuperior,'StopbandFrequency2',FreqSuperior,'StopbandAttenuation1', Atenuacao,
'PassbandRipple', Ripple, 'StopbandAttenuation2', Atenuacao, 'SampleRate', Amostragem);
      elseif Caracteristica==4
        FiltroMontado =
designfilt('bandstopfir','PassbandFrequency1',FreqInferior,'StopbandFrequency1',0.1+FreqInferior,'StopbandFreq
uency2',0.99*FreqSuperior,'PassbandFrequency2',FreqSuperior,'StopbandAttenuation',
Atenuacao,'PassbandRipple1', Ripple, 'PassbandRipple2', Ripple, 'SampleRate', Amostragem);
      end;
      %fvtool(FiltroMontado);

   elseif Tipo==2
      if Caracteristica==1
        FiltroMontado =
designfilt('lowpassiir','Passbandfrequency',0.99*FreqSuperior,'Stopbandfrequency',FreqSuperior,'StopbandAtten
uation', Atenuacao, 'PassbandRipple', Ripple,'SampleRate', Amostragem,'DesignMethod', Metodo);
      elseif Caracteristica==2
        FiltroMontado =
designfilt('highpassiir','StopbandFrequency',FreqInferior,'PassbandFrequency',0.1+FreqInferior,'StopbandAttenu
ation', Atenuacao, 'PassbandRipple', Ripple,'SampleRate', Amostragem),'DesignMethod', Metodo;
      elseif Caracteristica==3
        FiltroMontado =
designfilt('bandpassiir','StopbandFrequency1',FreqInferior,'PassbandFrequency1',0.1+FreqInferior,'PassbandFreq
uency2',0.99*FreqSuperior,'StopbandFrequency2',FreqSuperior,'StopbandAttenuation1', Atenuacao,
'PassbandRipple', Ripple, 'StopbandAttenuation2', Atenuacao, 'SampleRate', Amostragem,'DesignMethod',
Metodo);
      elseif Caracteristica==4
        FiltroMontado =
designfilt('bandstopiir','PassbandFrequency1',FreqInferior,'StopbandFrequency1',0.1+FreqInferior,'StopbandFreq
uency2',0.999*FreqSuperior,'PassbandFrequency2',FreqSuperior,'StopbandAttenuation',
Atenuacao,'PassbandRipple1', Ripple, 'PassbandRipple2', Ripple, 'SampleRate', Amostragem,'DesignMethod',
Metodo);
      end;
      %fvtool(FiltroMontado);
   else
      disp('Tipo Invalido');
      FiltroMontado=[];
   end;

   %FiltroMontado Ã© o filtro personalizado a ser utilziado para tratar o sinal
return;
```

7.13 Função Densidade Espectral

```
function [Pxx,Fp] = Periodograma(data,fs,ch)
          ni = size(data,1);      % Tamanho do sinal de entrada
          nf = 2^(nextpow2(ni));
          [Pxx,Fp] = periodogram(data(:,ch),hamming(size(data,1)),nf,fs);
return;
```

7.14 Função PlotBrainMap

```
function [] = PlotBrainMap (MatrizFatores,ListaCanais,Limiar)
   r=10;
   hold on
   th = 0:pi/50:2*pi;
   xunit = 1.02*r*cos(th);
   yunit = 1.02*r*sin(th);
   plot(xunit,yunit,'blue','linewidth',1.5);
   n=size(ListaCanais,1);
   segmento=360/n;
   N=max(max(MatrizFatores));

   for i=1:1:n
      x(i)=r*sin((2*pi)*(segmento*(i-1))/360);
      y(i)=r*cos((2*pi)*(segmento*(i-1))/360);
      text((x(i)*1.12)-0.6,y(i)*1.12,ListaCanais(i,:));
      axis([-12 12, -12 12]);
   end

   map=colormap(jet(256));
   map(1,:)=[0 0 0];
   colormap(map)
   colorbar

   for i=1:1:n
         for j=(i+1):1:n
            if i~=j
            Cor=round(255*(MatrizFatores(i,j))+1);
            if (MatrizFatores(i,j))>=Limiar
               Linha=1+2*abs(MatrizFatores(i,j))/N;
               p=plot([x(i) x(j)],[y(i) y(j)],'Color',map(abs(Cor),:), 'linewidth',Linha);
            else
               %NOP
            end
         else
            %NOP
         end
      end
   end
   hold off;
return;
```

7.15 Função PlotBrainMat2

```
function [map] = PlotBrainMap2 (VetorEnergia,Canais)
   n=length(VetorEnergia);

   hold on

   Brain=imread('Brain.png');
```

```
   imshow((Brain));

   map=colormap(jet(256));
   map(1,:)=[0 0 0];



   Cor=round((255*(VetorEnergia)+1)/max(VetorEnergia));
   for i=(1:n)
      plot(Canais(i,1),Canais(i,2),'.', 'MarkerSize',VetorEnergia(i), 'MarkerEdgeColor',map(abs(Cor(i)),:))
   end
   hold off

return;
```

7.16 Função Filtro CAR

```
function [signalout]=CARn(signal)

num_chans=size(signal,2);
%Remove componente CC
signal=signal-repmat(mean(signal,1),size(signal,1),1)
spatfiltmatrix=[];
spatfiltmatrix=eye(num_chans)-ones(num_chans)/num_chans;
signalout=double(signal)*spatfiltmatrix;
```

7.17 Função Coerência

```
function [CoerenciaXY,Fxy]=Coerencia(data1,data2,fs)
          ni = length(data1);      % Tamanho do sinal de entrada
          N = 2^(nextpow2(ni));
          [CoerenciaXY,Fxy] = mscohere(data1,data2,hamming(length(data1)/8),[],N,fs);
return;
```