UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA POLITÉCNICA

DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

REDUÇÃO DE RUÍDO ADITIVO: UMA VISÃO UNIFORMIZADA

Autor: Diego da Silva Rodrigues

Orientador: Sérgio Lima Netto, Ph.D.

Examinador: Gelson Vieira Mendonça, Ph.D.

Examinador: Vagner Luis Latsch, M.Sc.

DEL

Novembro de 2008

à minha mãe Angela pelo amor e dedicação,

aos meus padrinhos Emidio e Dora pelo carinho e apoio por toda a vida,

à Anna Carolinna pelo amor e companheirismo.

Agradecimentos

- A Deus.
- À minha família pela infra-estrutura que foi o suporte primordial para que eu alcançasse esta conquista. Agradeço pela educação e pelo incentivo que sempre me deram.
- Ao meu orientador, professor e amigo Sergio Lima Netto, pela paciência, pela clareza, pela simplicidade e pelo brilhantismo com que conduziu a orientação deste trabalho, agregando bastante valor à minha formação como engenheiro e como ser humano.
- Aos meus amigos pelo suporte contínuo e pelo companheirismo dentro e fora da universidade.
- Ao Departamento de Eletrônica e de Computação e à Escola Politécnica da UFRJ pelo curso de Engenharia oferecido e a todos os professores que garantem a excelência do ensino.

Resumo

Este trabalho aborda o tema supressão de ruído. O objetivo é reunir em módulos didáticos as principais técnicas de processamento de sinais aplicados ao problema. Cada módulo deverá conter uma apresentação para expôr o tema, uma apostila com o embasamento matemático e um toolbox desenvolvido no MATLAB implementando a técnica. Neste trabalho foram resumidos e correlacionados diversos métodos de processamento de sinais no escopo do tema do trabalho, realizando assim a modelagem dos módulos. Além disso, foram criadas apresentações e exemplos para cada método, foi desenvolvido um toolbox para o MATLAB implementando filtros digitais, acrescentado um algoritmo de redução de ruído musical ao toolbox de Filtragem de Wiener já existente e gerado um exemplo utilizando todos os toolboxes do conjunto.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Apresentação

Em diversas áreas da engenharia, estamos sempre nos deparando com um tipo peculiar de sinal: o ruído. Na maior parte das aplicações desta área há uma interferência inerente a cada uma das partes do sistema, seja na detecção, no processamento, na transmissão ou na recepção. Portanto, lidar com um mundo digital onde a quantidade de informação cresce a cada dia, é lidar com esse tipo característico de sinal cuja definição varia imensamente para cada modelo de interesse.

Nos sistemas de áudio costumamos encarar o ruído como qualquer barulho inconveniente que atrapalhe a compreensão do sinal de interesse. Seja o ruído de fundo numa gravação de música, seja o rotor do helicóptero interferindo no discurso do jornalista ao vivo. Numa transmissão de voz por telefone, temos o inconveniente do chiado da linha, dos ecos de voz. Vendo televisão, muitas vezes nos deparamos com fantasmas, com borrões, com interferência uniforme na imagem. E muitas vezes o ruído está em locais imperceptíveis aos sentidos humanos, como por exemplo, nos dados enviados para o outro lado do mundo por satélite, mas que a influência pode comprometer criticamente o funcionamento de diversos sistemas importantes para o mundo globalizado, como telefonia, internet, etc. Portanto, para trabalhar com sinais analógicos e digitais, é preciso compreender e tratar o ruído inerente ao sistema, reforçando o desempenho e a qualidade.

1.2 Objetivo

Este projeto é parte de um trabalho maior cujo principal objetivo é estudar e apresentar as principais técnicas de supressão/redução de ruído, de forma conjunta, didática e uniformizada. Isto é alcançado a partir da criação de diversos módulos, tendo cada um a função de explicar, implementar e exemplificar uma determinada técnica de processamento de sinais e sua aplicação no problema. Cada um dos módulos é composto por três partes:

- Uma apresentação em *PowerPoint* visando introduzir e cativar os espectadores ao estudo de determinada técnica.
- Uma apostila contendo o embasamento matemático no qual sustenta-se o método.
- *Toolboxes* desenvolvidos no *MATLAB* implementando as técnicas. Possuem interface GUI tornando-se uma ferramenta para desenvolvimento dos exemplos utilizados nas apresentações, e plataforma de aprendizado e implementação das técnicas.

Como cada técnica funciona melhor para casos diferentes, o objetivo é abrangir um escopo grande de problemas na área de processamento de sinais, sendo o resultado final um compêndio de métodos e aplicações de supressão de ruído. Desta forma, poderá servir de referência para qualquer trabalho na área de engenharia onde houver a necessidade de lidar com interferências deste tipo. As técnicas que serão vistas neste trabalho são:

- Análise espectral - Transformada de Fourier.
- Filtragem digital - Média móvel, FIR, IIR.
- Subtração espectral.
- Filtragem de Wiener.

- Filtragem Adaptativa.
- Wavelets.

Estas técnicas estão ordenadas de forma a cadenciar o estudo: cada módulo apresenta uma área de processamento de sinais que é enriquecida e complementada no próximo módulo, e assim por diante.

Concluindo, o conjunto: apresentação, apostila e exemplos para o pacote define um módulo abordando uma técnica. O conjunto de módulos junto com o pacote define o trabalho.

1.3 Resultados

Este projeto engloba o desenvolvimento dos *toolboxes*, das apresentações e exemplos que foram criados para cada uma das técnicas acima listadas. Para todos os módulos foram criadas apresentações, exceto Wavelets. Os exemplos foram escolhidos e desenvolvidos em scripts e funções do MATLAB, para justificar os casos onde tal modelo obtém melhor resultado, sendo eles amostras de áudio, voz e imagem.

O *toolbox* de filtragem digital foi completamente desenvolvido neste trabalho. O *toolbox* de filtragem de Wiener foi aprimorado com um algoritmo de redução de ruído musical que foi implementado durante o desenvolvimento da apresentação deste método, para geração dos exemplos. Para o *toolbox* de Wavelet não foi desenvolvido aprimoramento, mas ele foi utilizado no exemplo que demonstra a utilização dos *toolboxes* no capítulo 4. Para o *toolbox* de filtragem adaptativa não houve aprimoramentos, embora os algoritmos tenham sido implementados para geração dos exemplos das apresentações. Um exemplo foi desenvolvido para demonstrar a utilização conjunta dos pacotes no capítulo 4, de forma a ilustrar a capacidade do conjunto de técnicas em resolver problemas que fugiriam do escopo de uma técnica em particular.

O próximo capítulo descreve as técnicas de processamento de sinais que foram descritas nas apresentações e implementadas nos *scripts* e *toolboxes*. O capítulo 3

detalha as apresentações e os exemplos criados para cada uma. O capítulo 4 exibe os toolboxes desenvolvidos para o MATLAB e o exemplo que mostra as técnicas utilizadas em conjunto, e o capítulo 5 conclui o trabalho com um resumo dos resultados e possíveis continuações e futuros caminhos.

2 Técnicas de processamento de sinais

Este capítulo expõe as técnicas que compõem os módulos. Como descrito no capitulo anterior, o desenvolvimento das apresentações foi encadeado de forma ao aprendizado fluir seqüencialmente.

A primeira técnica vista é a transformada discreta de Fourier. Em seguida média móvel, filtros FIR e IIR. Utilizando uma abordagem estatística, subtração espectral e filtragem de Wiener, seguido por filtragem adaptativa e concluíndo com Wavelets.

2.1 Transformada de Fourier

A transformada de Fourier discreta no tempo leva um sinal do domínio do tempo para o domínio da freqüência, isto é, decompõe um sinal amostrado no tempo como uma soma infinita de senóides complexas [5]. Neste domínio, os sinais são representados no eixo das freqüências pelo módulo e pela fase. Abaixo temos a definição da transformada.

$$X(j\Omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\Omega Tn} \tag{2.1}$$

Onde $x(n)$ representa um sinal no tempo discreto, Ω representa a freqüência analógica e T o período de amostragem.

Para qualquer solução computacional para a transformada de Fourier, não contamos com sinais contínuos, seja no domínio do tempo ou no domínio da freqüência. Portanto, neste trabalho utilizamos a transformada de Fourier discreta

(DFT) [5] definida:

$$X(e^{j\frac{2\pi}{N}k}) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn} \tag{2.2}$$

O *MATLAB* utiliza um algoritmo rápido para o cálculo da DFT, que é conhecido como FFT (*Fast Fourier Transform*).

A DFT considera que o sinal de entrada é periódico, se repetindo a cada 2π. Logo, para uma sequência de tamanho N, geramos uma DFT de tamanho similar, possuíndo N componentes freqüenciais $\omega_k = \frac{2\pi}{N}k$. Portanto, o comprimento define a resolução espectral da transformada.

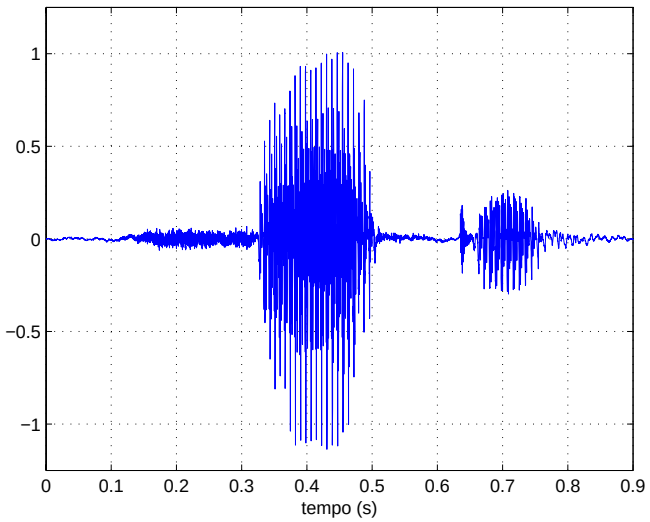

Figura 1: Sinal de voz no domínio do tempo.

Uma propriedade importante da DFT é a convolução circular:

$$X(k)H(k) = x(n) * \ h(n) \longleftrightarrow \sum_{l=0}^{N-1} x(l)h(n-l) = \sum_{l=0}^{N-1} x(n-l)h(n) \tag{2.3}$$

A convolução de dois sinais no tempo é igual ao produto de suas transformadas. Esta propriedade é útil, porque da convolução deriva o conceito de filtragem visto na próxima seção.

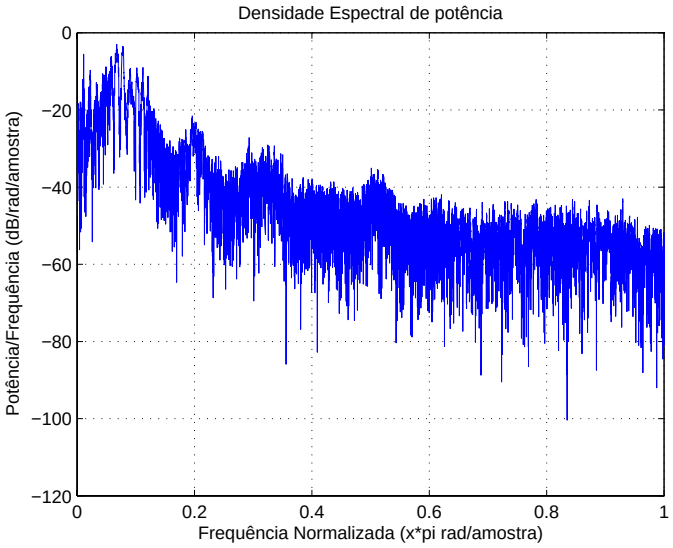

Figura 2: Espectro de potência do sinal de voz. Este gráfico foi obtido utilizando a função *pwelch*() do MATLAB, que calcula uma aproximação do espectro de potência de dado sinal. A freqüência de amostragem utlizada foi de 8 kHz, e o gráfico ilustra como as componentes espectrais da voz humana concentra-se nas baixas freqüências.

2.2 Filtragem digital

Utilizando a análise espectral descrita na parte anterior, temos uma visão das componentes freqüenciais que compõem um sinal. Como cada sinal tem características espectrais diferentes podemos tentar separar dois sinais misturados baseando-se nestas diferenças. Na figura 2 temos um exemplo de como se comportam as componentes de um sinal de voz no domínio da freqüência.

Um processo simples que ilustra a filtragem é o cálculo da média móvel de um sinal. Intuitivamente percebemos que calcular a média móvel num trecho onde o sinal varia rapidamente causa atenuação neste pedaço. Portanto, a média pode ser considerada um processo que reduz oscilações bruscas na amplitude de um sinal. O efeito da média móvel (2.4) é similar a de um filtro passa-baixas. Este tipo de filtro preserva as freqüências abaixo de determinada freqüência de corte, atenuando o resto. Uma explanação sobre tipos de filtro será dada adiante. A figura 3 apresenta o efeito da média móvel sobre um sinal senoidal contaminado com ruído.

$$y(n) = \frac{1}{M} \sum_{l=0}^{M-1} x(n-l) \tag{2.4}$$

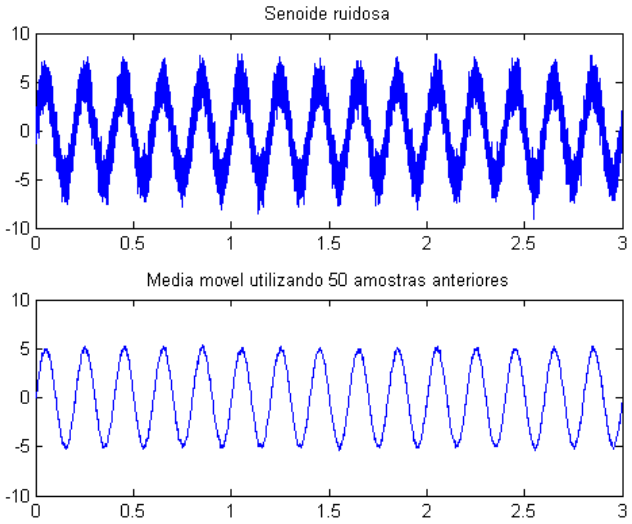

Figura 3: Efeito atenuante da média móvel sobre o ruído, sobre um sinal senoidal. O processamento atenua o ruído existente sobre a senóide.

Se compararmos a equação da média móvel (2.4) com a equação (2.3), para um $h(n)$ com todos coeficientes iguais a $\frac{1}{M}$, vemos que $y(n)$ é o resultado de uma convolução, ou seja, o produto da DFT de $x(n)$ pela DFT de $h(n) = [\frac{1}{M}\ \frac{1}{M} ...\ \frac{1}{M}]$. Sabendo que a convolução é o produto das transformadas, podemos avaliar a transformada de $h(n)$ para diferentes valores de M, que é número de amostras do sinal utilizado na média móvel.

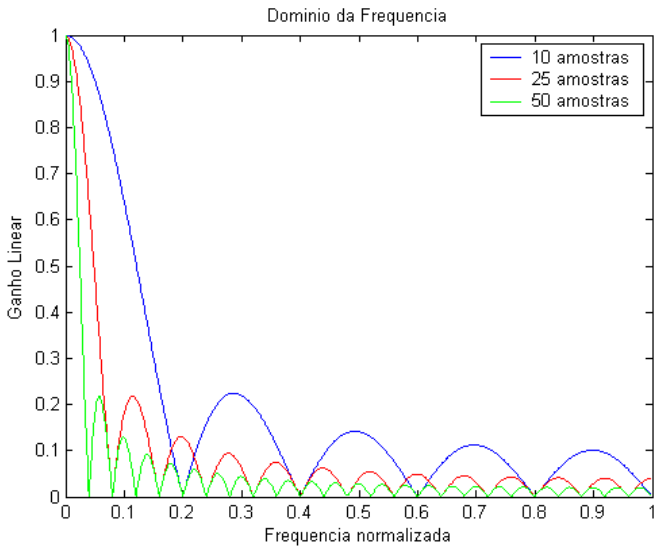

Figura 4: DFT de $h(n)$ para diferentes comprimentos. O aumento no número de amostras torna cada vez mais estreita a faixa de ganho.

Analisando o resultado (fig. 4), vemos que para freqüências baixas, o produto

pela transformada do filtro conserva as componentes (multiplicando por um ganho 1), atenuando as outras e isto é o que define o processo de filtragem.

Os filtros usualmente são caracterizados pelo intervalo de freqüências que preservam. Passa-baixas para conservar as baixas freqüências, passa-altas para conservar as altas freqüências, analogamente. O Filtro Passa-faixas preserva um intervalo específico dentro do espectro, enquanto o Rejeita-faixas faz o contrário: conserva todas as freqüências fora do intervalo determinado dentro do espectro. A figura 5 ilustra os quatro tipos.

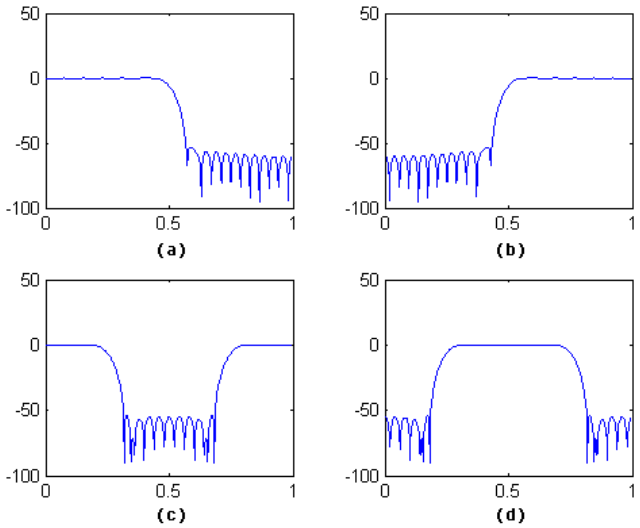

Figura 5: Filtros (a) passa-baixas, (b) passa-altas, (c) rejeita-faixa, (d) passa-faixa.

Existem duas arquiteturas diferentes utilizadas no projeto de filtros: FIR (*Finite Impulse Response*) e IIR (*Infinite Impulse Response*). Um filtro IIR apresenta uma função de transferência com pólos e zeros, enquanto o FIR apresenta uma função de transferência sem pólos (todos estão em zero).

$$H(z) = \sum_{m=0}^{M} b_m z^{-m} \tag{2.5}$$

$$H(z) = \frac{\sum_{m=0}^{M} b_m z^{-m}}{\sum_{n=1}^{N} a_n z^{-n}} \tag{2.6}$$

Os coeficientes b_m e a_n são projetados de acordo com especificações do filtro utilizando diferentes métodos. Um filtro passa-baixas ideal, comportaria-se formalmente como a figura (6). Atenuaria todas as frequências acima de uma frequência de corte definida, e conservaria as restantes. O problema é que para aproximar a discontinuidade em ω_c deste filtro, precisariamos convoluir o sinal de entrada por um sinal de comprimento infinito, a função $sinc(n)$, gerando infinitos coeficientes b_m, no caso de um filtro FIR por exemplo.

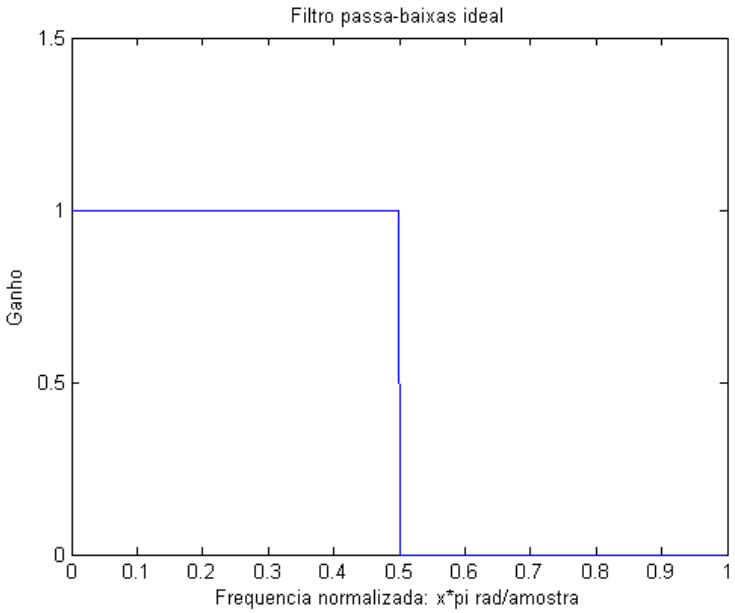

Figura 6: Filtros passa-baixas ideal: Preserva completamente as componentes menores que determinada freqüência de corte, atenuando todas as outras.

O truncamento da função $sinc(n)$ para criação de filtros, definindo os coeficientes b_m, gera um efeito indesejado na banda de passagem, conhecido como efeito de Gibbs (fig. 8).

Para atenuar o efeito Gibbs, é utilizado *Janelamento* sobre o truncamento da função $sinc(n)$. Os filtros FIR projetados pelo método de janelamento tem seus coeficientes b_m calculados pela multiplicação dos coeficientes da $sinc(n)$ por uma janela, sendo as mais conhecidas, Hanning, Hamming, Blackman. Cada um das janelas gera efeitos sobre a faixa de rejeição e de passagem que podem ser vistos na figura 9. Maiores detalhes podem ser encontrados em [5].

Para os filtros IIR, existem métodos que trazem implementações do domínio

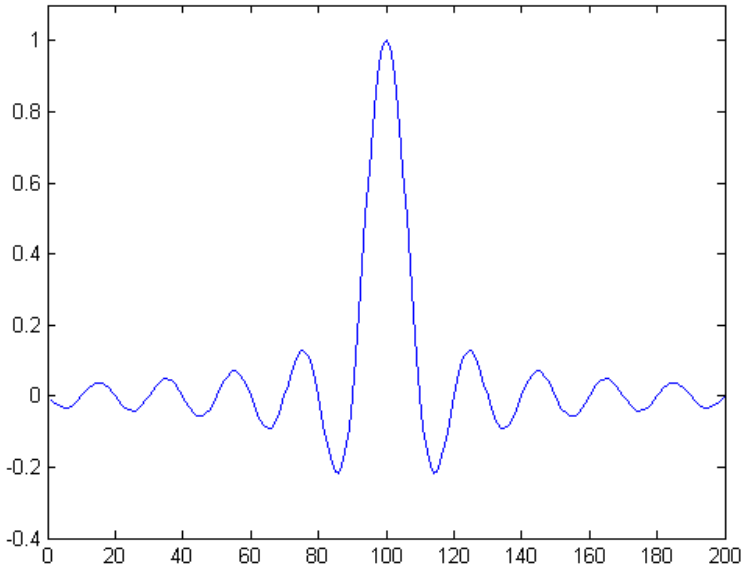

Figura 7: A função $sinc(n)$ é a transformada de Fourier inversa do filtro passa-baixas ideal da figura 6, de comprimento infinito.

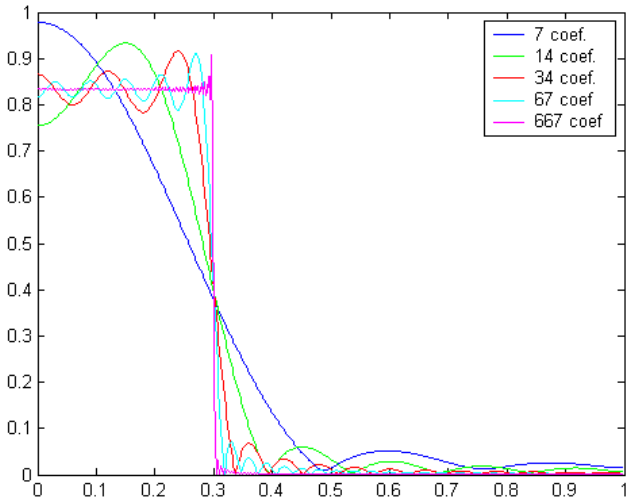

Figura 8: O efeito de Gibbs é a oscilação próxima a descontinuidade entre as faixas de passagem e rejeição, ilustrada nesta figura. O aumento do número de coeficientes aproxima o resultado do filtro ideal, juntamente com o efeito indesejado do Efeito de Gibbs.

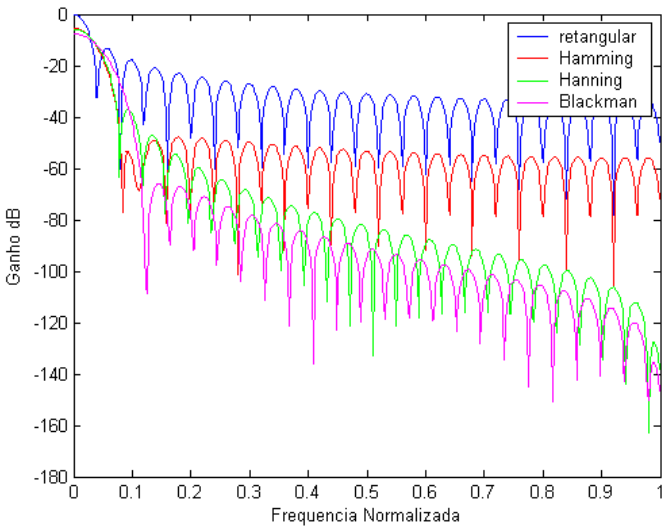

Figura 9: Nesta figura podemos ver o efeito da utilização de diferentes janelas no domínio da freqüência. A janela retangular apresenta o menor efeito atenuador, enquanto a janela de Blackman, o maior.

contínuo para o domínio discreto. Entre eles estão os métodos de Chebyshev, Elíptico, Butterworth. Em geral, o método aborda o posicionamento de pólos e zeros no plano complexo, para obter os resultados de posicionamento da(s) freqüência(s) de corte e a atenuação na banda de rejeição.

O *MATLAB* contém funções para geração de janelas, e para a geração dos coeficientes dos filtros, baseado nas condições desejadas de corte e atenuação. Os algoritmos são completos facilitando a implementação.

2.3 Subtração Espectral

A subtração espectral acrescenta à análise do domínio da freqüência a análise estatística dos sinais, adicionando ao conjunto noções de processos estocásticos. Ela funciona melhor para um tipo característico de ruído, conhecido como ruído de fundo. A modelagem mais simples de sinal adicionado de ruído pode ser vista em (2.7):

$$y(n) = x(n) + r(n) \tag{2.7}$$

Considerando os dois sinais estacionários, não-correlacionados e com média

zero, podemos aproximar a função densidade de potência da saída $y(n)$ por [1]:

$$S_{yy}(\omega) = S_{xx}(\omega) + S_{rr}(\omega) \tag{2.8}$$

A subtração espectral então é definida como:

$$S_{xx}(\omega) = S_{yy}(\omega) - S_{rr}(\omega) \tag{2.9}$$

Como o sinal contaminado por ruído $y(n)$ é o sinal analisado, e podemos obter o ruído $r(n)$ de trechos de silêncio no primeiro sinal, temos como estimar os dois espectros de densidade de potência. Uma estimativa satisfatória é $S_{yy}(\omega) = |Y(\omega)|^2$ sendo o segundo termo o módulo do quadrado da DFT do sinal (energia instantânea). Para o sinal $S_{rr}(\omega)$ utilizamos a mesma forma de aproximação com $|R(\omega)|^2$. Conseguimos então obter uma aproximação para $S_{xx}(\omega)$ fazendo:

$$|\hat{X}(\omega)|^2 = |Y(\omega)|^2 - |\hat{R}(\omega)|^2 \tag{2.10}$$

Tirando a raiz, e calculando a fase de $\hat{X}(\omega)$ a partir da fase de $Y(\omega)$, voltando para o domínio do tempo com a DFT inversa conseguimos obter o sinal limpo.

Para a subtração espectral funcionar, é ideal que o ruído seja branco, ou seja, tenha suas componentes espalhadas igualmente pelo espectro (suas componentes freqüenciais têm o mesmo módulo). Desta forma, o espectro de densidade de potência do sinal contaminado é igual ao do sinal de entrada mais uma constante, que é a densidade espectral de potência do ruído branco. É claro que na prática $\hat{R}(\omega)$ não equivale a uma constante, e o impacto da subtração espectral é a geração de alguns picos senoidais indesejados, conhecidos como "ruído musical".

Esta subtração abrupta no espectro é atenuada pela técnica vista na próxima seção.

2.4 Filtro de Wiener

Enquanto a subtração espectral retira abruptamente parte das componentes espectrais de um sinal a partir de uma estimação da DFT do ruído, a solução de Wiener considera um filtro que minimiza o erro médio quadrático (*Mean-Squared Error - MSE*) da relação entre o sinal de entrada e o ruído, de acordo com a equação (2.11)[6], considerando que o ruído e sinal de entrada são independentes:

$$H(\omega) = \frac{S_X(\omega)}{S_X(\omega) + S_N(\omega)} \tag{2.11}$$

$S_X(\omega)$ é a densidade espectral de potência do sinal e $S_N(\omega)$ é a densidade espectral de potência do ruído. Esta fórmula se aplica a sinais contínuos. Para trabalhar com sinais digitais, podemos calcular o ganho do filtro para cada componente (*bin* da DFT) fazendo a consideração abaixo [6], dentro de uma janela do sinal de entrada:

$$f(Y(m)) = \frac{S_X(m)}{S_X(m) + S_N(m)} Y(m), \tag{2.12}$$

ou seja, para cada componente senoidal correspondente ao índice m, obtemos $f(Y(m))$, que é o sinal corrompido $Y(m)$ multiplicado por um ganho definido por $\frac{S_X(m)}{S_X(m)+S_N(m)}$. Este ganho variável para cada componente define a filtragem de Wiener: quando a razão sinal ruído é alta para uma determinada componente m, o ganho é alto, e a componente é preservada. Conforme a razão sinal ruído diminui, o ganho diminui, eliminando as componentes ruidosas.

A equação 2.10 mostrou uma aproximação pro sinal $S_X(\omega)$ a partir de $Y(\omega)$ (ω aqui representado pelo *bin* da DFT m). Substituindo na equação 2.12 leva à equação abaixo:

$$f(Y(m)) = \frac{|Y(m)|^2 - |R(m)|^2}{|Y(m)|^2} Y(m) \tag{2.13}$$

[6] define $\rho(m)$ como razão-sinal-ruído de potência em cada *bin* frequencial. Esta definição encontra-se na equação abaixo:

$$\rho(m) = \frac{|Y(m)|^2 - |R(m)|^2}{|R(m)|^2} \tag{2.14}$$

Combinando as equações 2.14 e 2.13 temos a definição do filtro de Wiener. $\rho(m) > 0$ indica que a densidade espectral de potência do ruído é maior do que a do sinal de interesse para determinada componente m. Portanto, a filtragem de Wiener é definida abaixo:

$$f(Y(m)) = \begin{cases} \frac{\rho(m)}{1+\rho(m)} Y(m) & \rho(m) > 0, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.15}$$

A equação (2.15) deixa claro o explicado anteriormente: quanto maior a SNR para uma componente senoidal em m, mais preservada será aquela freqüência (isto é, o ganho do filtro tende a 1). Nos casos onde o ruído supera o sinal, a componente é zerada.

O filtro de Wiener está sujeito ao ruído musical da mesma forma que a subtração espectral, embora seja um método mais apurado, no que tange a análise da SNR para cada componente, ao invés de uma simples subtração abrupta sobre o espectro inteiro. Podemos ver nas figs. 10 e 11 o efeito dos dois métodos no espectro, e os picos freqüenciais que sobram do processamento e geram o ruído indesejado.

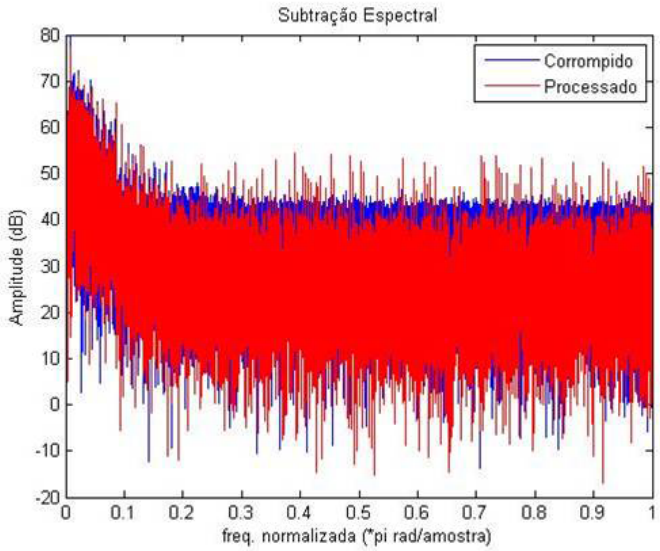

Figura 10: Efeito atenuador da subtração espectral no domínio da freqüência. Notar os diversos picos no sinal processado nas altas freqüências, responsáveis pelo ruído musical.

A forma mais simples de combater esta interferência de processamento é utilizando mascaramento. A primeira forma é multiplicar a estimativa da densidade

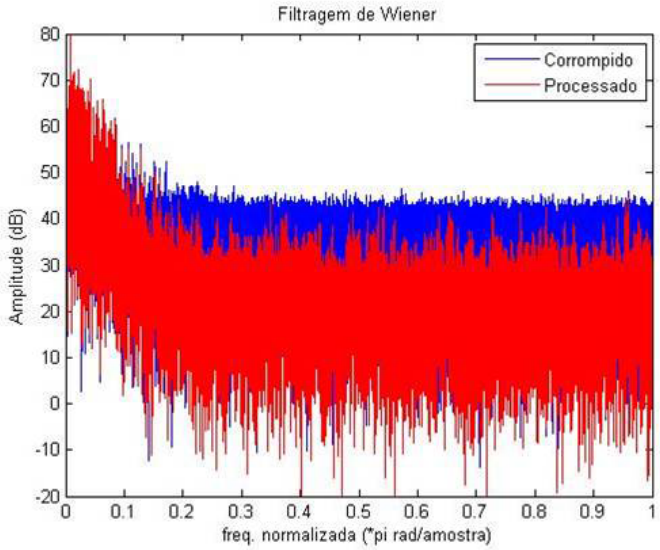

Figura 11: Efeito da filtragem de Wiener no domínio da freqüência. Percebe-se maior efeito de atenuação deste método em comparação com a subtração espectral, embora ainda existam picos senoidais nas freqüências mais altas.

espectral de potência em (2.11) por uma constante α positiva e maior que zero. Esta nova estimativa $\alpha S_N(m)$ elevaria o nível geral de ruído, diminuindo a amplitude das componentes processadas que ainda apresentassem SNR maior que zero.

A outra forma é deixar um chão de ruído, alterando a equação (2.15). Com $f(Y(m)) = \beta \sqrt{S_N(m)}$ para $\rho(m) < 0$, o processamento deixa um resquício do ruído original ponderado pela constante β.

Das duas formas, o ruído musical é mascarado, sendo uma tarefa extra encontrar valores de α e β para que o mascaramento ocorra sem piorar o sinal que está sendo processado.

2.5 Filtragem adaptativa

Se pensarmos no filtro de Wiener da seção anterior com ganhos individuais para as componentes senoidais variantes no tempo, temos um filtro capaz de se adaptar a não-estacionaridades do ruído. A filtragem adaptativa estuda o projeto deste tipo de filtro.

Na figura (12) são identificadas as variáveis de um filtro adaptativo. A com-

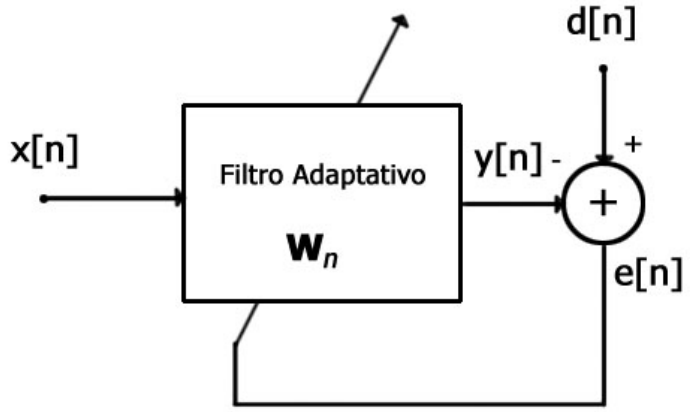

Figura 12: Diagrama de blocos do filtro adaptativo.

paração da saída do filtro $y(n)$ com um sinal desejado $d(n)$ gera um sinal de erro, que é minimizado em função dos coeficientes do filtro.

A modelagem de supressão de ruído pode ser vista na figura (13). O filtro adaptativo encontra a função de transferência que transforma $r'(n)$ no ruído aditivo $r(n)$ que está contaminando o sinal de interesse $x(n)$. O sinal limpo é encontrado na diferença entre a saida do filtro e o sinal desejado, ou seja, o próprio sinal de erro $e(n)$. Num carro de fórmula 1, por exemplo, uma entrada do filtro ($d(n)$) seria a voz do piloto adicionada ao ruído do motor, enquanto a outra entrada seria o ruído do motor puro.

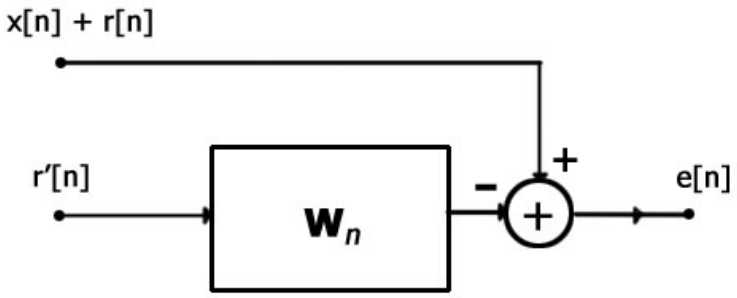

Figura 13: Diagrama de blocos do filtro adaptativo - Supressão de ruído.

Para os coeficientes do filtro, temos uma solução ótima chamada de solução de Wiener [4]:

$$w_0 = R^{-1}p \tag{2.16}$$

Onde R é a matriz de auto-correlação do sinal de entrada, e p a matriz de correlação cruzada entre a entrada do filtro e o sinal desejado.

Para calcular R^{-1} é necessária alta complexidade computacional, inviabilizando a utilização de filtros adaptativos para problemas reais. Por isso, os algoritmos que viabilizam a utilização dos filtros tentam chegar ao valor ótimo utilizando várias interações. O processo é feito a partir de uma função de custo relacionada ao sinal de erro $F(e(n)) = \xi(n)$ sendo minimizada por um determinado algoritmo.

O LMS (*Least-Mean Square*) é provavelmente o algoritmo adaptativo mais utilizado, pela sua simplicidade e pelo baixo custo computacional inerente ao método implementado. Ele utiliza uma aproximação simples para as matrizes R e p (eq. 2.18 e 2.19), o que o torna leve computacionalmente e utiliza como função-custo o erro médio quadrático - MSE (*Mean-square Error*), e a função de aproximação conhecida como *Stepeest-Descent* [4]. A equação 2.17 mostra o algoritmo baseado no *stepeest-descent* utilizado para buscar a solução de Wiener w_0. A equação 2.18 mostra a aproximação utilizada para $\hat{R}(k)$ e a equação 2.19 mostra a aproximação para $\hat{p}(k)$.

$$w(k+1) = w(k) - 2\mu(-\hat{p}(k) + \hat{R}(k)w(k)) \tag{2.17}$$

$$\hat{R}(k) = x(k)x^T(k) \tag{2.18}$$

$$\hat{p}(k) = d(k)x(k) \tag{2.19}$$

Substituindo as equações 2.18 e 2.19 em 2.17 chegamos a equação final que determina o LMS:

$$\begin{aligned} w(k+1) &= w(k) + \mu(-2d(k)x(k) + 2x(k)x^T(k)w(k)) && (2.20)\\ &= w(k) + \mu(2x(k)[-d(k) + x^T(k)w(k)]) && (2.21)\\ &= w(k) + 2\mu\ e(k)x(k) && (2.22) \end{aligned}$$

a constante μ é chamada de passo de convergência, e define quão bruscamente o algoritmo movimenta-se para o ótimo. Um passo pequeno resulta numa convergência lenta e precisa, e o oposto, uma convergência rápida e instável. O

algoritmo NLMS (*Normalized LMS*) torna o passo também adaptativo, com o objetivo de acelerar a aproximação para o ótimo (μ grande) e diminuindo a constante de forma a aumentar a precisão.

Outro algoritmo popular é o RLS (*Recursive least-squares*), que utiliza como função-custo os mínimos quadrados. Ele parte de outra aproximação para R e d, gerando duas matrizes conhecidas como auto-correlação e correlação cruzada determinísticas [4]. A partir delas, o vetor de pesos é calculado em cada interação.

$$w(k+1) = S_D(k)p_D(k) \tag{2.23}$$

No calculo de S_D definimos uma constante λ que é conhecida como fator de esquecimento, e define quantas amostras passadas o modelo considera na hora de atualizar os pesos em k.

Os dois algoritmos apresentados acima definem duas abordagens diferentes para lidar com filtros adaptativos. o LMS é leve, mas de convergência lenta, e pior estabilidade sobre o ótimo que o RLS, que por sua vez exige mais complexidade computacional. Optar por um dos dois é relativo ao problema em questão, da mesma forma que todos os métodos apresentados neste capítulo.

2.6 Transformada Wavelet

Como descrito na primeira seção deste capítulo, a transformada de Fourier decompõe um sinal numa soma de senóides complexas. Para um sinal no domínio do tempo, são calculados os coeficientes que definem módulo e fase das componentes senoidais que o compõem. Este processo é similar a uma troca de bases, na álgebra linear, quando um vetor é projetado sobre o outro a partir do produto escalar entre os dois.

O módulo do vetor ao ser projetado sobre as componentes de uma base representa quão bem aquele componente consegue concentrar a informação do vetor original. Sendo as infinitas senóides complexas de diferentes freqüências as compo-

nentes da base da transformada de Fourier, seus coeficientes seriam a projeção do sinal processado em cada vetor desta base.

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j\omega t}dt \tag{2.24}$$

A transformada Wavelet pode ser vista como uma generalização do processo acima. A função-mãe $\psi(t)$, conhecida como Wavelet, define as bases da transformada que serão utilizadas para decompor o sinal. Enquanto a função que define as bases da transformada de Fourier permite variações na freqüência, a transformada Wavelet permite expansão/contração no tempo e translação.

$$\check{F}(\omega) = \int_{+\infty}^{-\infty} f(t)\psi(t)dt \tag{2.25}$$

É possível construir famílias de Wavelets discretas através do conceito de análise em multiresolução. Esta análise parte da existência de duas funções, a Wavelet mãe $\psi(t)$ e a função de escala mãe $\varphi(t)$, ortogonais entre si, que se relacionam com as funções de escala em cada nível de resolução de acordo com as equações:

$$\varphi_{m,n}(t) = 2^{m/2}\psi(2^m t - n) \tag{2.26}$$

$$\psi_{m,n}(t) = 2^{m/2}\varphi(2^m t - n) \tag{2.27}$$

Isso mostra que as bases podem ser contraídas ou expandidas por um fator de 2 em cada nível de escala, e podem sofrer deslocamentos unitários no tempo. Esse poder de resolução dá à Wavelet capacidade de lidar melhor com sinais finitos no tempo, ao contrário da DFT.

A Wavelet pode ser implementada utilizando um banco de filtros, onde os canais passa-baixa são decompostos recursivamente. Essa estrutura é conhecida como decomposição em oitavas. Nesta estrutura, a envoltória da resposta ao impulso dos filtros da equação tem o mesmo formato para todos os estágios de decomposição do banco [5]. Portanto, esta envoltória pode ser calculada a partir da wavelet mãe e a filtragem pode ser encarada como o cálculo da correlação cruzada entre o sinal e a Wavelet, para cada nível de detalhe, ou seja, para cada versão escalada e deslocada

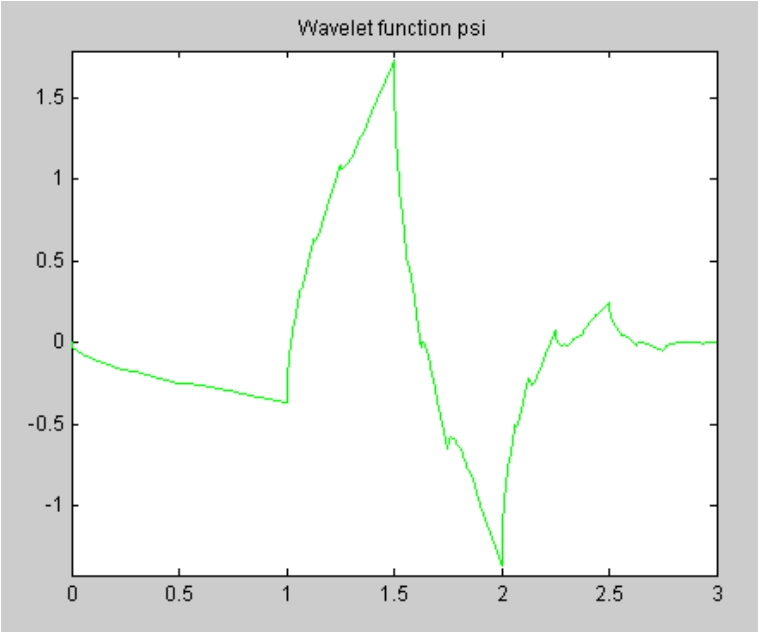

Figura 14: Wavelet Mãe da família Daubechies 2.

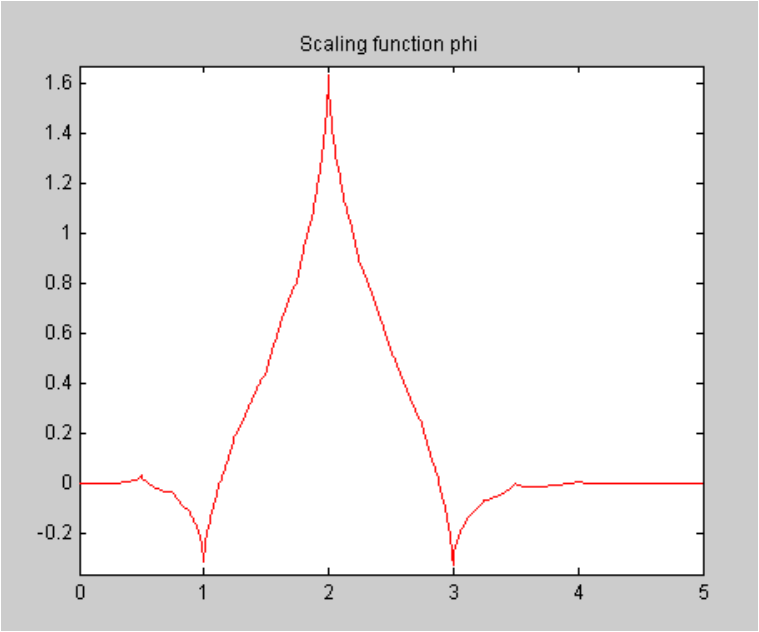

Figura 15: Wavelet Mãe da família Coiflets 1.

da Wavelet mãe, representado por cada nível do banco de filtros. O valor de cada correlação cruzada é um coeficiente da transformada Wavelet [3].

Portanto, escolher a Wavelet mãe apropriadamente permite concentrar o sinal em poucos coeficientes. Isto pode ser utilizado para supressão de ruído de duas maneiras: caso o sinal ruidoso seja concentrado em poucos coeficientes, esses podem ser descartados e assim descarta-se o ruído. Caso o sinal de interesse se concentre em poucos coeficientes, todos os outros podem ser eliminados.

O ato de preservar ou atenuar um coeficiente é relacionado a aplicação de um *threshold*, um limiar. No processamento utilizando Wavelets, *hard threshold* é o termo dado quando todo sinal abaixo do limiar é descartado, enquanto *soft threshold* é a atenuação por uma constante. Estes detalhes são relevantes para o entendimento do funcionamento do toolbox de Wavelets no capítulo 4.

2.7 Conclusão

Este capítulo expôs as técnicas que compõem os módulos. A primeira técnica vista foi a transformada discreta de Fourier, seguido da seguida média móvel, filtros FIR e IIR. Utilizando o domínio da estatística, subtração espectral e filtragem de Wiener. Por fim, filtragem adaptativa e transformada Wavelet.

Os fundamentos descritos nesse capítulo foram aplicados no desenvolvimento dos módulos, assim como na criação e aprimoramento dos pacotes GUIs que fazem parte do projeto. Os exemplos escolhidos foram desenvolvidos utilizando as ferramentas aqui vistas para justificar a utilização dos modelos no escopo da supressão de ruído, e serão descritos no próximo capítulo.

3 Descrição das apresentações

Este capítulo mostra o que foi abordado nas apresentações criadas para os módulos, assim como os exemplos que foram utilizados. A regra de composição, como dito na primeira parte, visa encadear as ferramentas matemáticas levando a compreensão de uma técnica posterior a partir de uma técnica apresentada previamente.

A primeira apresentação ilustra a transformada de Fourier. A segunda inicia o estudo de filtros digitais, com filtros FIR, encerrando na terceira apresentação que exibe os filtros IIR. A quarta apresentação aborda subtração espectral e filtro de Wiener, e a quinta e última trata de filtros adaptativos.

Em cada parte haverá uma descrição dos exemplos e uma sinopse de cada apresentação, resumida em uma tabela. Os *slides* de título e índice serão ignorados na sinopse por se repetirem em todas as apresentações.

3.1 Apresentação 1 - Transformada de Fourier

A transformada de Fourier é a base da maior parte das técnicas de supressão de ruído e portanto é o tema da primeira apresentação.

Inicia-se com séries de Fourier, e como um sinal periódico pode ser decomposto numa soma de senóides. Para explicar tal tema de forma simples e didática na apresentação, é feito uma analogia com a culinária: conhecendo os "ingredientes" senoidais que compõem dado sinal, pode-se reconstruí-lo conhecendo-se a quantidade de cada componente - lembrando que o objetivo principal dos módulos é a didática.

Determinados sinais (por exemplo, ruído branco) apresentam componentes freqüenciais espalhadas por todo o domínio, enquanto a maior parte dos sinais de interesse concentram sua energia em alguns intervalos específicos. Portanto, uma forma primordial de eliminação de ruído, é zerar as componentes espectrais nas quais o sinal de interesse não apresenta energia signficativa.

Os exemplos envolvem aplicações da transformada de Fourier para eliminação de ruídos convenientes a esta análise simplista. Ruídos cujas componentes concentram-se sobre determinada freqüência são eliminados eliminando apenas o "ingrediente", ou seja, zerando as mesmas. Caso o sinal concentrado seja o de interesse, aplica-se processo análogo, eliminando todo o resto.

Exemplo 1

O primeiro exemplo aborda um radar simplificado: o sinal de interesse é uma emissão numa determinada freqüência conhecida, que refletida, é captada novamente misturada ao ruído do ambiente ao redor. Portanto, preservando apenas esta componente, elimina-se todo o ruído ambiente. Para este exemplo foi criado um arquivo de áudio, a soma de um sinal senoidal em 2 kHz com ruído branco gerado utilizando a função $randn()$ do MATLAB. Calculando a FFT deste sinal, zerando todas as componentes exceto a freqüência da senóide e encontrando a IFFT, gerou-se o sinal limpo utilizado no exemplo, que é simplesmente o som da senóide de 2 kHz.

Exemplo 2

No segundo exemplo, o ruído é uma componente senoidal sobre um sinal musical. O sinal contaminado foi gerado importando-se um trecho de música no MATLAB, e adicionando uma senóide de 500 Hz a esse. Neste exemplo a única componente eliminada foi a componente do sinal senoidal, e o sinal obtido pela IFFT foi a música limpa.

Exemplo 3

O terceiro exemplo apresenta uma imagem corrompida por ruído senoidal. Desta vez a transformada de Fourier é bidimensional, e o processo aplicado é o

mesmo: zerando a componente, zera-se o ruído. A imagem corrompida foi gerada a partir da soma dos valores dos pixels com um sinal senoidal, para cada linha. Como no exemplo 2, a partir do calculo da FFT, desta vez bidimensional, e da eliminação do ruído apenas zerando o valor da componente foi suficiente para atenuar o efeito da interferência.

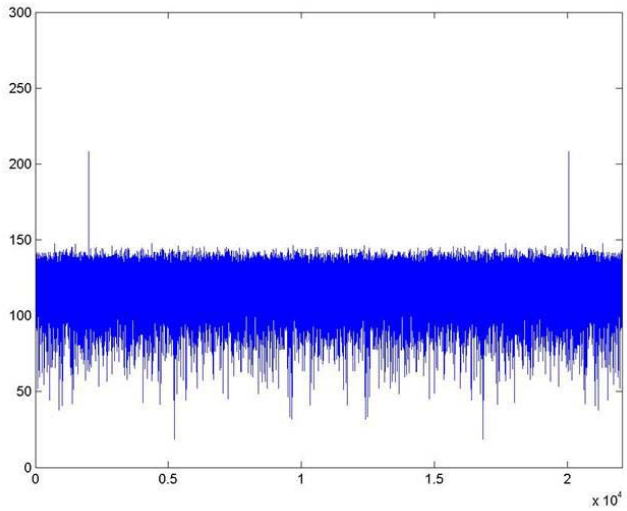

Figura 16: Exemplo 1.1 - Sinal senoidal visivelmente destacado em meio ao ruído branco no domínio da freqüência, no caso do radar.

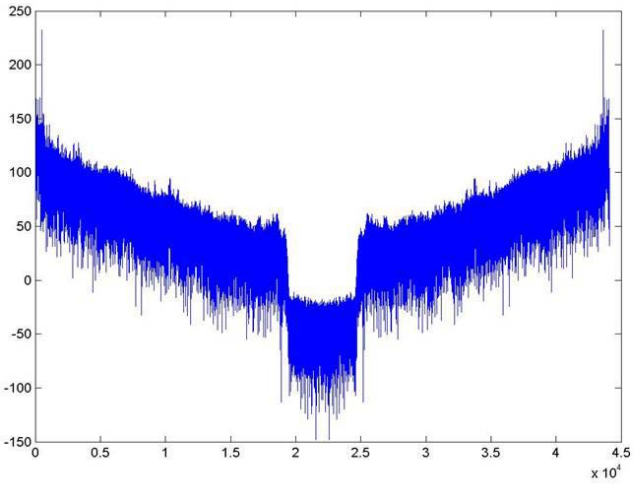

Figura 17: Exemplo 1.2 - Espectro de um sinal musical corrompido por um sinal senoidal.

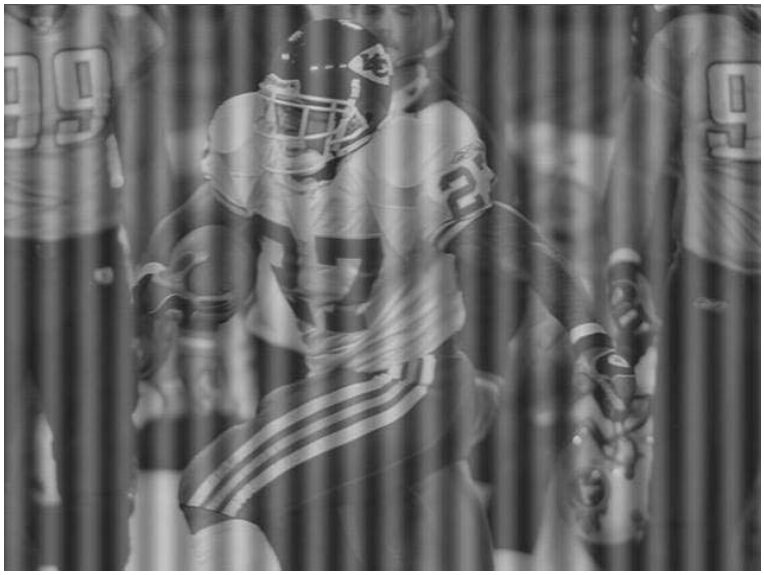

Figura 18: Exemplo 1.3 - Imagem corrompida por ruído senoidal.

Figura 19: Exemplo 1.3 - Apenas removendo os picos senoidais da FFT, obtemos o resultado acima.

Tabela 1: Sinopse da Apresentação 1

Slides	Descrição
3 a 5	Abordagem do tema - séries de Fourier
6 e 7	Abordagem do tema - transformada de Fourier
8	Exemplo 1.1
9	Exemplo 1.2
10 a 14	exemplo 1.3

3.2 Apresentação 2 - Filtros FIR

A segunda apresentação introduz o conceito de filtragem. Enquanto os exemplos da primeira apresentação utilizam sinais inteiros e o processamento é todo *offline*, na segunda apresentação o processamento em tempo real é exposto. O desenvolvimento e solidificação do conceito de filtragem toma caminho idêntico ao visto no capítulo anterior: começa por média móvel e o efeito passa-baixas, depois visualiza um filtro ideal na freqüência que preserva todas as componentes de interesse, e por fim discute maneiras de projetar estes filtros.

Os tipos de filtro são apresentados: Passa-baixas, Passa-alta, etc. O truncamento da função $sinc(n)$ e o efeito de Gibbs também são ilustrados, e como o janelamento é utilizado para projeto de filtros, encerrando com exemplos. Todas as figuras utilizadas no capítulo 2, sobre filtros FIR, foram retiradas desta apresentação.

Exemplo 1

O primeiro exemplo apresenta comandos de MATLAB para projetar filtros FIR, utilizando janelas de Blackman, em diferentes bandas, e aplica os filtros para isolar três sinais senoidais que encontram-se misturados.

As três senóides misturadas tem freqüência igual a 800, 4000 e 10.000 Hz. O filtro passa-baixas foi projetado com freqüência de corte igual a 2200 Hz. O passa-faixa, com freqüência de corte inferior igual a 2200 Hz e superior igual a 5500 Hz. O passa-altas com corte em 7000 Hz. Os três foram projetados utilizando ordem igual a 30.

Além da figura ilustrativa, foram gerados quatro sinais de áudio, um sendo a soma das senóides e três sinais de saída do processamento de cada filtro. A freqüência de amostragem foi de 22050 Hz.

Exemplo 2

O segundo exemplo compara o desempenho da média móvel com o de um filtro projetado utilizando a janela de Hamming. Para este exemplo não foram gerados sinais de áudio. Foram comparados o filtro projetado usando a janela de Hamming com 400 coeficientes, média móvel de 500 e 750 coeficientes.

A primeira ilustração do exemplo apresenta uma senóide contaminada por ruído branco. A segunda e a terceira apresentam a resposta em freqüência do filtro e das médias móveis, em duas diferentes aproximações, para salientar a diferente atenuação na banda de rejeição dos três. A quarta apresenta o resultado do processamento pelos três modelos.

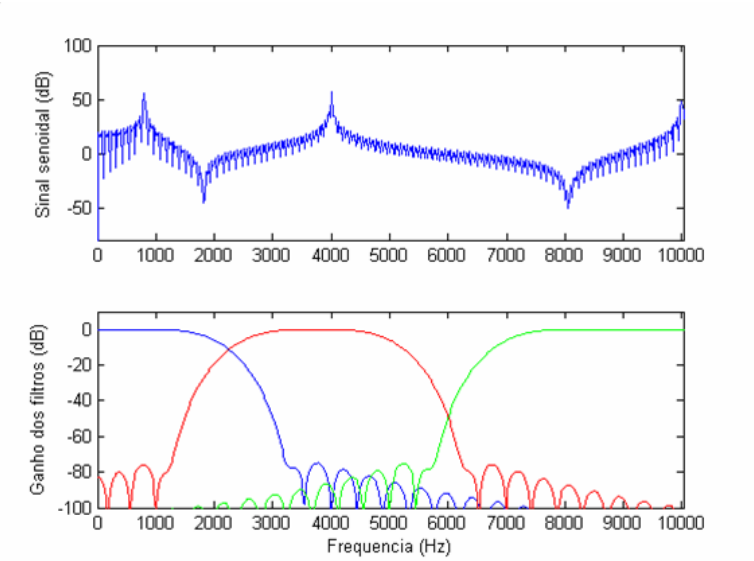

Figura 20: Exemplo 2.1 - Filtros em diferentes bandas projetados utilizando a janela de Blackman para processamento de sinal composto por três componentes senoidais.

3.3 Apresentação 3 - Filtros IIR

A segunda apresentação de filtros digitais - terceira do trabalho - aborda os filtros IIR, discute os algoritmos para calcular os coeficientes de um filtro IIR, dado

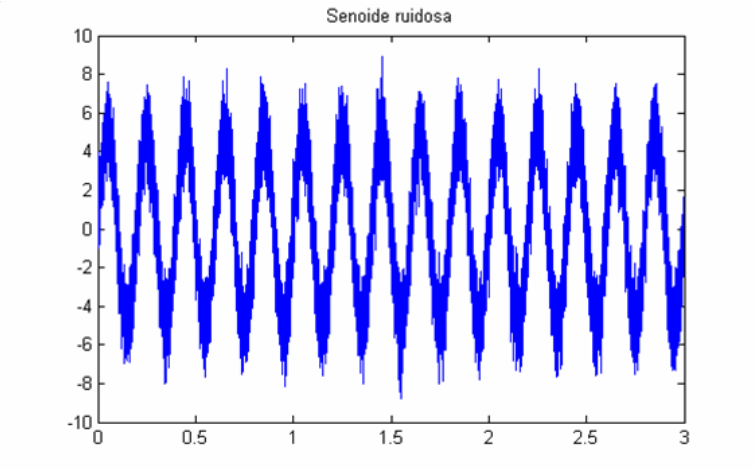

Figura 21: Exemplo 2.2 - Senóide ruidosa antes do processamento.

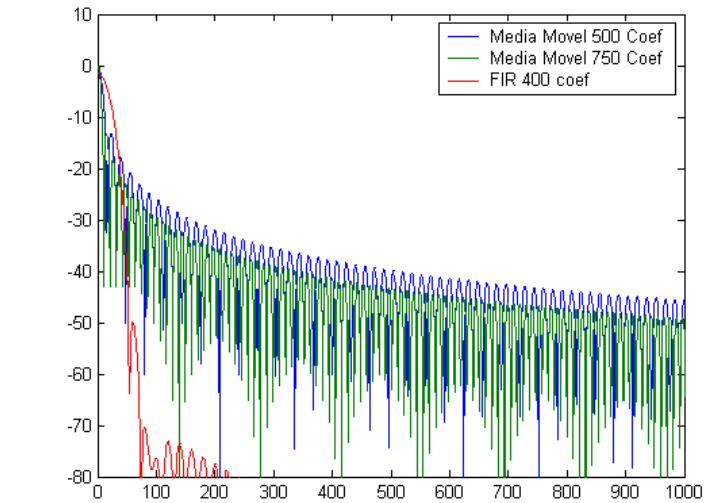

Figura 22: Exemplo 2.2 - Comparação de desempenho entre a média móvel e um filtro FIR projetado usando a janela de Hamming. A atenuação na banda de rejeição do filtro projetado usando a janela é maior do que a da média móvel, utilizando menos coeficientes.

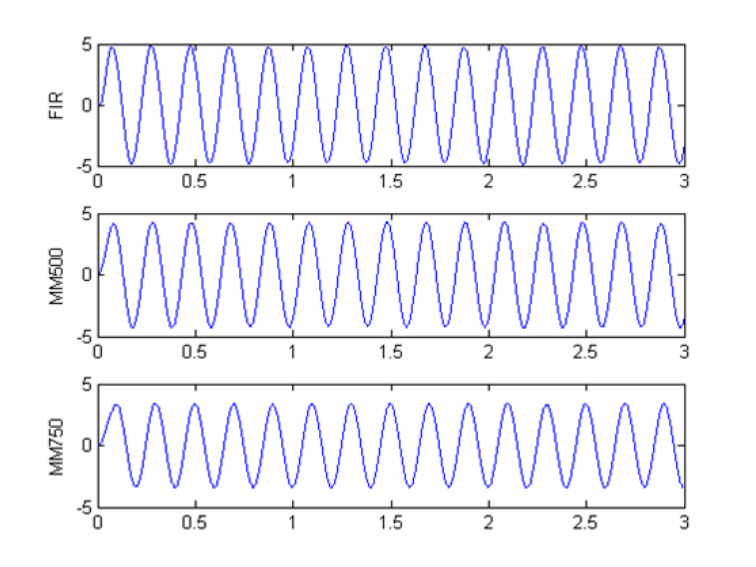

Figura 23: Exemplo 2.2 - Resultado final do processamento. O ruído é pouco visível nos três casos, mas nota-se atenuação na senóide para os casos da média móvel.

Tabela 2: Sinopse da Apresentação 2

Slides	**Descrição**
3	Introdução ao processamento em tempo real e filtragem
4 a 10	Abordagem do tema - Média Móvel
11 e 12	Introdução ao tema - Filtros digitais
13 a 16	Abordagem do tema - Projeto de filtros
17 a 19	Abordagem do tema - Janelamento
20 a 22	Exemplo 2.1
23 a 26	Exemplo 2.2

as condições de banda de passagem e rejeição e compara os prós e contra deste tipo de filtro versus filtros FIR.

Os três métodos de projeto de filtros IIR apresentados são: Butterworth, Chebyshev e Elíptico. O projeto do filtro é ilustrado através dos comandos de MATLAB utilizados de acordo com o método, bem como um gráfico da resposta em freqüência e da constelação (diagrama de distribuição de pólos e zeros).

Exemplo 1

No primeiro exemplo desta apresentação um trecho musical é decomposto utilizando filtros passa-faixa em diferentes faixas, para ilustrar a distribuição das componentes harmônicas dos instrumentos de uma orquestra ao longo do espectro

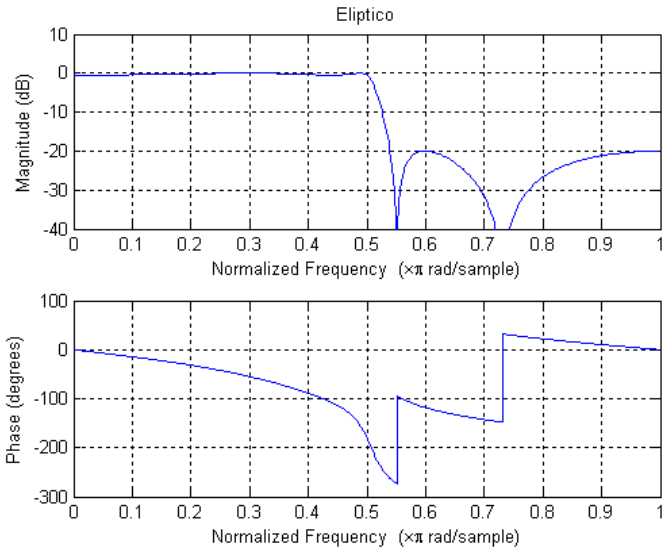

Figura 24: Resposta em freqüência de um filtro IIR Elíptico. Um dos gráficos da apresentação, ilustrando o projeto de filtros IIR.

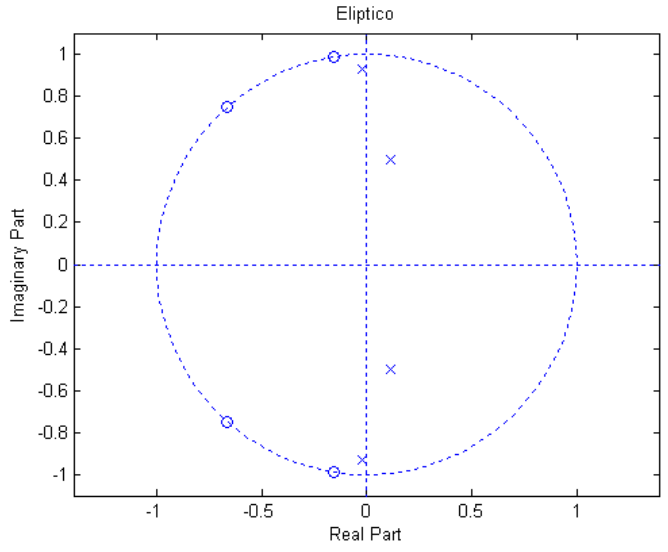

Figura 25: Constelação do filtro IIR Elíptico - Diagrama de pólos e zeros.

de freqüência, e como isto pode ser utilizado para equalização, amplificando determinadas bandas desejadas.

A freqüência de amostragem deste exemplo foi 44100 Hz. A freqüência de corte do passa-baixas foi 1100 Hz. Dos quatro passa-faixas, as freqüências de corte inferior e superior foram, respectivamente, 1100 e 3307, 3307.5 e 5512,5, 5512,5 e 11025, 11025 e 16537,5 Hz. A freqüência de corte do passa-altas foi 16537,5 Hz. Os coeficientes do filtro foram gerados utilizando a função butterworth do MATLAB.

Para cada um dos filtros foi obtido um sinal processado. Este exemplo visa mostrar como as componentes da maioria dos instrumentos ficam concentradas nas baixas freqüências. A importância disto é vista no exemplo seguinte.

Exemplo 2

O segundo exemplo demonstra o efeito destrutivo de uma filtragem passa-baixas abrupta sobre um sinal de voz, ilustrando o compromisso entre a redução do ruído de fundo e a queda drástica de qualidade no sinal. Isso é feito comparando dois sinais de voz, no exemplo, a pronúncia das palavras “faca” e “saca”. Como as primeiras sílabas são fricativas, apresentam componentes espalhadas no espectro. Ao eliminar estas freqüências utilizando um filtro passa-baixas, a diferenciação das duas palavras torna-se difícil.

Foram utilizados quatro filtros passa-baixas neste exemplo, todos projetados usando a função butterworth do MATLAB. A freqüência de corte foi: 300, 500, 700, 1100 Hz. Os sinais de áudio da pronúncia das palavras faca e saca foram pré-processados para retirar a média, normalizar a energia, de forma a não tornar um dos dois sinais mais perceptível pela intensidade.

O total de dez amostras de áudio, as originais e as processadas, fazem parte da apresentação.

Exemplo 3

O terceiro exemplo procura validar o exemplificado pelo segundo, o compromisso entre redução do ruído de fundo e a preservação da inteligibilidade dos

sinais de áudio. O exemplo consiste em diversas filtragens sobre um sinal de voz contaminado por ruído branco utilizando diferentes filtros passa-baixas. Dado que as componentes da voz concentram-se nas baixas freqüências, eliminando as altas, remove-se boa parte do ruído. Isto vale também para sinais musicais, como visto no primeiro exemplo desta apresentação. Quanto mais baixa é a freqüência de corte, mais sinal de interesse é descartado, piorando o resultado final da filtragem. Para um sinal de voz onde o importante é compreender a mensagem, este problema não é tão crítico quando comparado com um sinal de áudio, onde a riqueza espectral do som dos instrumentos é importante.

Para o sinal musical, a freqüência de amostragem utilizada foi de 44100 Hz e o corte dos três passa-baixas foi em 1100, 3000 e 5000 Hz.

Para o sinal de voz, foi utlizada a freqüência de amostragem de 8000 Hz, e corte em 300, 500 e 700 Hz.

Tabela 3: Sinopse da Apresentação 3

Slides	**Descrição**
3 a 6	Introdução aos filtros IIR
7 a 9	Coeficientes, constelação e resposta em freqüência do filtro IIR de Butterworth
10 a 12	Coeficientes, constelação e resposta em freqüência do filtro IIR de Chebyshev
13 a 15	Coeficientes, constelação e resposta em freqüência do filtro IIR Elíptico
16 e 17	Exemplo 3.1
18 e 19	Exemplo 3.2
20 e 21	Exemplo 3.3

3.4 Apresentação 4 - Subtração Espectral e Filtragem de Wiener

A quarta apresentação reúne subtração espectral e filtragem de Wiener. Iniciando por subtração espectral, são expostas as fórmulas de dedução da aproximação das densidades espectrais de potência e como é feito o cálculo para remover o ruído do sinal.

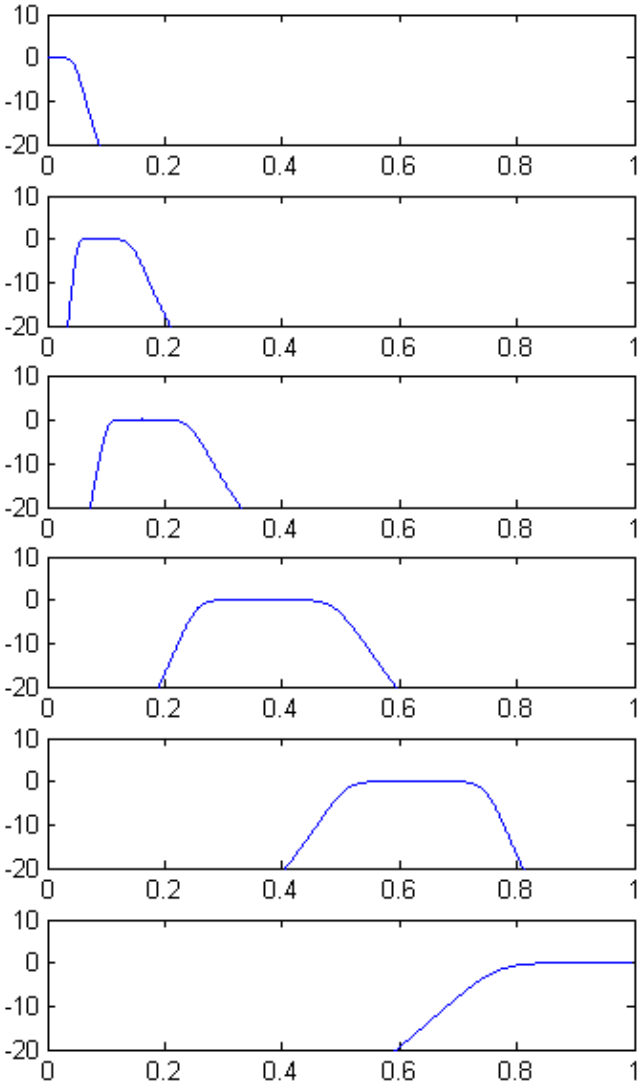

Figura 26: Exemplo 3.1 - Divisão de um sinal de áudio em diferentes bandas. Nesta figura vê-se a distribuição dos filtros ao longo do espectro.

Para o filtro de Wiener é apresentada a fórmula do cálculo dos coeficientes do filtro para cada componente espectral do sinal. Ela é deduzida a partir da relação apresentada no capítulo 2, de minimização do MSE do sinal.

A terceira parte da apresentação descreve os dois métodos utilizados para reduzir o ruído musical, que foram utilizados na geração dos exemplos da apresentação e que foi um aprimoramento realizado no toolbox de supressão de ruído do projeto. O método da superestimação do ruído e o método do "chão" de ruído deixado para mascarar o sinal são apresentados.

Exemplo 1

O único exemplo desta apresentação compara o desempenho entre os dois métodos, bem como compara um sinal de alta SNR e muito ruído musical, com um de SNR menor, devido ao ruído de fundo deixado para mascarar o ruído musical. O objetivo é mostrar que nem sempre o sinal com maior SNR vai apresentar melhor qualidade. O "chão" de ruído deixado para mascarar o ruído musical, embora reduza a SNR, soa melhor do que o efeito mascarado.

O sinal utilizado foi um sinal musical amostrado a 44100 Hz. Foi efetuada a subtração espectral, filtragem de Wiener e filtragem de Wiener utilizando o método de supressão de ruído musical. Na tabela 4 vemos uma comparação da SNR vista na apresentação: embora o sinal processado sem tratamento de ruído musical apresente maior SNR, o outro soa mais agradável. O exemplo serve justamente para mostrar que nem sempre uma medida como a SNR pode ser considerado o principal parâmetro de qualidade quando estamos tratando do que pode soar "melhor" para a audição humana.

Os gráficos de comparação entre os processos foram utilizados no capítulo anterior para exemplificar os métodos, e fazem parte da apresentação, são as figuras 10 e 11.

Para o exemplo desta aplicação, foi utilizado o toolbox desenvolvido em [2]. Para o tratamento do ruído musical foi desenvolvido algoritmo que foi acrescentado

Tabela 4: Comparação entre a subtração espectral e a filtragem de Wiener

Sinal	**SNR**
Corrompido	09,5507
Subtração Espectral	03,9903
Wiener	15,5591
Wiener s/ ruído musical	13,7043

ao toolbox, como será descrito melhor no próximo capítulo.

Tabela 5: Sinopse da Apresentação 4

Slides	**Descrição**
3 a 8	Abordagem do tema - Subtração espectral
9 a 11	Abordagem do tema - Filtro de Wiener
12 a 17	Abordagem do tema - Redução do ruído musical
18	Exemplo 4.1

3.5 Apresentação 5 - Filtragem Adaptativa

A ultima apresentação discute filtragem adaptativa e suas aplicações. A abordagem do tema segue o mesmo caminho das outras apresentações: inicia discutindo os casos onde os filtros adaptativos podem ser utilizados, devido a sua necessidade de utilizar dois sinais distintos: o ruído puro $r'(n)$, e o sinal de interesse contaminado $x(n) + r(n)$, como exposto no capítulo 2. A partir daí, apresenta os algoritmos, discute fatores específicos de cada algoritmo como por exemplo, o passo de convergência do LMS e constante de esquecimento no RLS, e encerra nos exemplos. Para esta apresentação foram implementados algoritmos referentes ao LMS, NLMS e RLS, sendo os dois últimos utilizados para gerar os exemplos. Todos foram desenvolvidos utilizando a linguagem do MATLAB.

A apresentação começa incentivando com situações cotidianas onde a filtragem adaptativa pode ser aplicada: Um repórter falando ao microfone num helicóptero, um operário trabalhando com a britadeira, são alguns exemplos. A partir disso, discute o modelo do filtro e como, diferente de todos os outros casos vis-

tos neste trabalho, para a supressão de ruído utilizando filtragem adaptativa são necessários dois canais.

A partir daí a apresentação toma o rumo das outras, exibindo o conteúdo matemático do modelo assim como vantagens e desvantagens em relação aos outros métodos e por fim, os exemplos. Nesta apresentação, todos os exemplos são sinais de áudio contaminado por algum tipo característico de ruído, e o objetivo é comparar a convergência dos algoritmos, bem como o efeito gerado pela variação do número de coeficientes do filtro.

Exemplo 1

O primeiro exemplo é dividido em dois conjuntos. Este exemplo visa comparar o desempenho dos algoritmos, quando a característica do ruído é de estacionaridade no domínio da estatística.

No primeiro conjunto de amostras, o sinal alvo é um trecho musical contaminado por ruído branco, gerado pelo MATLAB. A freqüência de amostragem utilizada foi de 44100 Hz. Três amostras processadas foram geradas, utilizando o LMS com 100 coeficientes, 500 coeficientes, e o RLS com 10 coeficientes. Nos exemplos gerados pelo LMS, é perceptível um breve intervalo de convergência no início do sinal processado, refletindo um tempo de convergência do algoritmo. No RLS, o intervalo é imperceptível.

No segundo conjunto de amostras, o sinal alvo é um trecho de voz amostrado a 8000 Hz. O ruído é inerente ao próprio sistema de gravação, uma espécie de chiado do microfone. Este sinal foi processado utilizando o LMS com 10 e 100 coeficientes, e o RLS com 10. Não é perceptível nenhum intervalo de convergência neste exemplo.

Este exemplo visa ilustrar que para sinais de característica estacionária, o LMS consegue suprimir o ruído com mesmo desempenho que o RLS, sendo um algoritmo mais leve computacionalmente.

Exemplo 2

O segundo exemplo apresenta sinais contaminados com ruído não-estacionário.

No primeiro conjunto de amostras, é utilizado um sinal musical contaminado por uma série de sinais artificiais gerados no MATLAB (utilizando a função, *square*(), *chirp*(),*randn*(*n*)). No segundo conjunto é utilizado um sinal de voz, contaminado com um sinal musical. Ambos os exemplos foram amostrados a 44100 Hz.

No primeiro conjunto foi utilizado o LMS com 100 e 500 coeficientes, e o RLS com 10 e 50 coeficientes. Mesmo com o aumento no número de coeficientes, a convergência lenta do LMS impossibilitou completa eliminação do sinal ruidoso nos dois casos. O ruído é atenuado, mas continua perceptível. No RLS com 50 coeficientes, o ruído é eliminado.

No segundo conjunto foram utilizadas as mesmas combinações do primeiro conjunto. Novamente o LMS não conseguiu eliminar o ruído e o RLS conseguiu.

Neste exemplo, o tempo de processamento do LMS e do RLS passou a ser relevante. Um sinal de 10 segundos amostrado a 44100 Hz apresenta 441000 amostras, e o tempo necessário para o RLS processar o arquivo passava de 1 minuto. O LMS demorava cerca de 20 segundos para realizar o mesmo processamento, embora não eliminasse completamente o ruído.

Exemplo 3

O terceiro exemplo apresenta cancelamento de eco. A modelagem do sistema é diferente do utilizado nos exemplos acima, mas como o eco pode ser considerado ruído para diversas aplicações, o exemplo foi incluído. O desafio neste caso é o tempo de eco, que faz com que o filtro necessite de muitos coeficientes (um segundo de eco equivale a 8000 amostras, utilizando amostragem a 8000 Hz).

Para o exemplo foi utilizado um arquivos de voz amostrado a 8000 Hz, como um eco de aproximadamente 0.1 segundos de *delay* sobre outro arquivo de voz. Isto equivale a aproximadamente 800 amostras do sinal. Neste exemplo, a utilização do RLS foi inviável, já que o tempo de processamento para 200 coeficientes era da ordem de alguns minutos. O LMS com 2000 e 3000 coeficientes conseguiu atenuar parte do eco, como no exemplo 2, demorando menos tempo.

Este exemplo encerrou a apresentação e a comparação entre os dois algoritmos, ponderando o efeito atenuante, o tempo de processamento, o número de coeficientes e discutindo a viabilidade de utilização de cada um deles para cada problema.

Exemplo 4

O ultimo *slide* mostra um exemplo comercial de aplicação da supressão de ruído com dois canais. Fones de ouvidos que são capazes de eliminar a interferência do ambiente, utilizando uma tecnologia que é chamada de ANR (*Active Noise Reduction*), implementada comercialmente utilizando vários métodos diferentes. Este exemplo encerra a última apresentação, bem como o capítulo.

Tabela 6: Sinopse da Apresentação 5

Slides	**Descrição**
3 e 4	Exemplos cotidianos de ruído
5 a 8	Abordagem do tema - Filtro Adaptativo
9 a 12	Apresentação dos Algoritmos - LMS e RLS
13	Exemplo 5.1
14	Exemplo 5.2
15	Exemplo 5.3
16	Exemplo 5.4

3.6 Conclusão

O objetivo desta parte foi exibir todo o material gerado para acompanhar as apresentações - os exemplos - que são peça chave para a didática de cada módulo e para exemplificação de cada técnica. As apresentações tem papel fundamental na introdução do tema, e os exemplos trazem aplicações de engenharia para as ferramentas, ou seja, dão embasamento prático à teoria. O estudo de cada técnica, o desenvolvimento dos algoritmos e aprendizado das funções do MATLAB para aplicação dos método, a modelagem dos exemplos, torna este capítulo o centro do trabalho desenvolvido neste projeto.

O próximo capítulo descreve a ferramenta que foi desenvolvida para acres-

centar mais um pacote GUI de MATLAB ao conjunto já existente.

4 Toolboxes de supressão de ruído

Este capítulo descreve os *toolboxes* desenvolvidos no MATLAB para acompanhar as apresentações vistas no capítulo 3. Cada um dos toolboxes aborda uma das técnicas vistas, e são a principal ferramenta de análise e tratamento de sinais ruidosos para o projeto. Revisando o objetivo: para cada técnica de supressão de ruído há uma apresentação com exemplos ilustrativos, de forma a exemplificar a aplicabilidade de cada uma para determinado tipo de problema. Junto das apresentações, os pacotes servem de interface para o desenvolvimento de novos exemplos, bem como ferramenta para análise e processamento. Os *toolboxes* são o centro do trabalho e principal plataforma de execução das técnicas.

O primeiro *toolbox* abrange a transformada Wavelet, o segundo, subtração espectral e filtragem de Wiener, o terceiro implementa filtros adaptativos, e o quarto, filtragem digital e projeto de filtros.

Será visto a seguir a funcionalidade de cada *toolbox*, a aplicabilidade de cada um para determinado tipo de supressão de ruído e as implementações feitas em cada um neste trabalho. Em seguida, será descrito o exemplo que foi desenvolvido combinando a utilização das ferramentas de forma a exemplificar a capacidade de processamento do conjunto de técnicas abrangidas pelo projeto.

4.1 Toolbox de Transformada Wavelet

O primeiro *toolbox* implementa a transformada Wavelet. Este toolbox não foi desenvolvido e nem aprimorado neste trabalho, mas foi utilizado na composição do exemplo visto neste capítulo, para a eliminação de ruído impulsivo. Como explicado

no capítulo de técnicas, a Transformada Wavelet tem como base a Wavelet-mãe. Caso esta seja escolhida apropriadamente de modo a concentrar em alguns coeficientes a energia do ruído, estes podem ser descartados de forma a ”limpar“ o sinal.

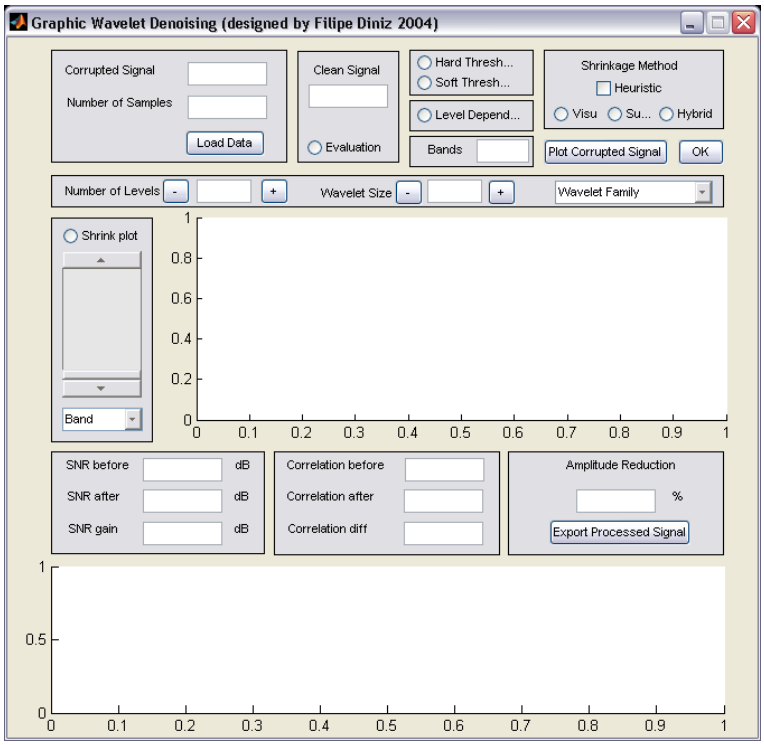

Figura 27: Toolbox desenvolvido para MATLAB - Transformada Wavelet.

As funcionalidades do toolbox são as seguintes:

1. Campos para a inserção do sinal corrompido, para o processamento, e do sinal original, para construção das métricas de avaliação do processamento: SNR e correlação cruzada.

2. Campo para escolha do tipo de *Treshold* que será aplicado: *Hard Threshold* elimina completamente as bandas escolhidas, enquanto *Soft Treshold* atenua.

3. *Shrikage Method*: Seleção do método que calcula quais coeficientes da transformada serão zerados ou atenuados: *Visu Shrink*, *Sure Shrink* e Híbrido.

4. Caixa de Texto *Bands*: Serve para selecionar quais bandas do banco de filtros que implementa a transformada que serão consideradas.

5. *Number of Levels*: Número de níveis de detalhe utilizado.

6. *Wavelet Size* e *Wavelet Family*: Define a Familia de Wavelets que será utilizada.

7. *Slider* Lateral: Selecionando a banda desejada no menu abaixo, determina o valor do *threshold* aplicado. Quando acionado o botão *Shrink Plot*, atualiza o gráfico com o valor do *threshold*.

8. Campos de *Evaluation*: SNR e correlação antes, após, o ganho e redução de amplitude. Medidas utilizadas para avaliar o desempenho do método: necessitam que o botão *Evaluation* seja marcado antes do processamento.

O *toolbox* possui duas saídas gráficas: a primeira traça a transformada correspondente à decomposição do sinal para cada nível de detalhe, e permite a utilização do *Slider* para definir o limiar aplicado em cada uma das bandas. A segunda apresenta o sinal processado, e o compara com sinal desejado caso o botão *Evaluation* tenha sido marcado.

O ganho da SNR, a diferença entre a SNR antes e depois do processamento, é utilizada como método de validação em todos os *toolboxes* e posteriormente nos exemplos. Ela é calculada a partir do sinal limpo, para fins de validação dos algoritmos. Em problemas reais este ganho não poderia ser medido, já que o sinal limpo não existiria.

4.2 Toolbox - Subtração espectral e Filtro de Wiener

O segundo toolbox implementa a subtração espectral e a filtragem de Wiener. A ferramenta é capaz de processar um sinal corrompido utilizando filtragem de Wiener ou subtração espectral, calculando a aproximação da densidade espectral de potência do ruído no próprio arquivo do sinal corrompido, num trecho onde não esteja sendo reproduzido o sinal de interesse. As funcionalidades são as seguintes:

1. Campos para a inserção do sinal corrompido, para o processamento, e do sinal

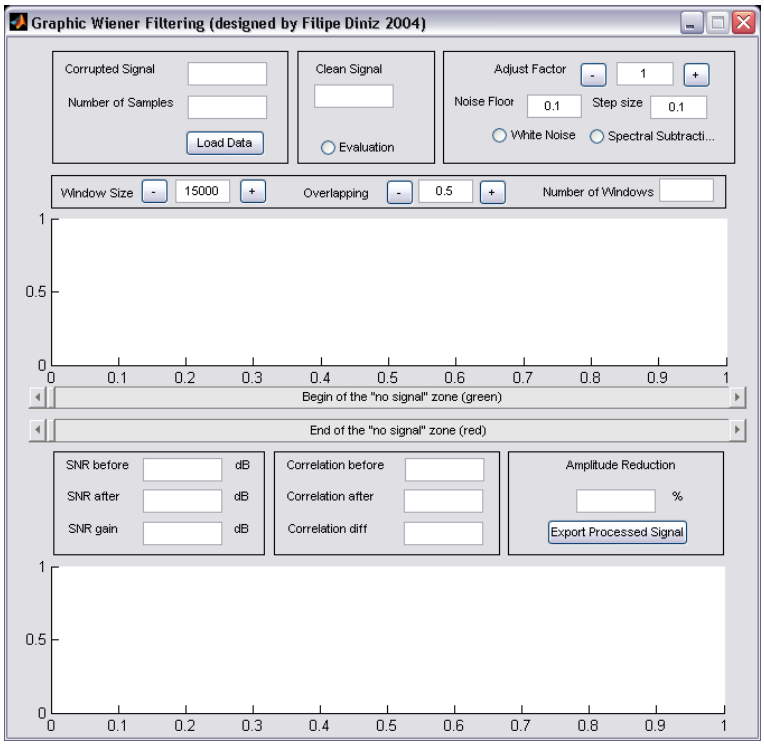

Figura 28: Toolbox desenvolvido para MATLAB - Subtração Espectral e Filtragem de Wiener.

original, para construção das métricas de avaliação do processamento: SNR e correlação cruzada.

2. Campos *Window Size*, *Overlapping* e *Number of Windows*: Controlam o tamanho das janelas nas quais o sinal é dividido para realizar o processamento. *Overlapping* controla a porcentagem de superposição entre as janelas.

3. *Sliders* para o ajuste do intervalo de ruído: Visualizando o sinal no gráfico, é possivel determinar o intervalo onde há apenas ruído. Este intervalo é utilizado no calculo da estimativa da PSD do ruído que será utilizada no processamento.

4. Campos de *Evaluation*: SNR e correlação antes, após, e o ganho e redução de amplitude. Medidas utilizadas para avaliar o desempenho do método: necessitam que o botão *Evaluation* seja marcado antes do processamento.

5. Botão *White Noise*: Quando marcado, o Toolbox utiliza uma PSD constante para processamento, de valor definido pelo campo *Step Size*.

6. *Adjust Factor*: Serve como ajuste para a PSD estimada do trecho de ruído. É

a constante multiplicadora α descrita no capítulo de técnicas, quando descrito tratamento de ruído musical, e pode ser utilizada para a superestimação da PSD.

7. *Noise Floor*: Define a constante β utilizada para deixar um resíduo ruidoso, para mascarar o ruído musical gerado no processamento.

O toolbox possui duas saídas gráficas. A primeira apresenta o sinal corrompido antes do processamento, e sua visualização é utilizada para ajuste dos *sliders* que determinam início e fim do intervalo de onde o ruído será estimado. A segunda compara o sinal original com o sinal processado. As dois gráficos são atualizados instantaneamente após cada processamento realizado.

A contribuição para este pacote foi a adição de um método de tratamento de ruído musical descrito no capítulo de técnicas e um exemplo para ilustrar o efeito deste método no desempenho do toolbox. O pacote já continha a constante de ajuste α, portanto, foi acrescentado apenas o fator β. O valor dessa constante é customizável no campo *Noise Floor*, e os efeitos são vistos no exemplo deste capítulo.

4.3 Toolbox de Filtragem Adaptativa

O terceiro *toolbox* implementa os filtros adaptativos discutidos no capítulo de técnicas. Para este *toolbox* não foram realizados aprimoramentos, mas todos os algoritmos disponíveis neste *toolbox* foram implementados para a geração dos exemplos do capítulo de apresentações. Abaixo, a descrição de suas funcionalidades.

1. Campos para a inserção do sinal corrompido, para o processamento, e do sinal original, para construção das métricas de avaliação do processamento: SNR e correlação cruzada.

2. Campo *Iteractions*: definição do número de iterações.

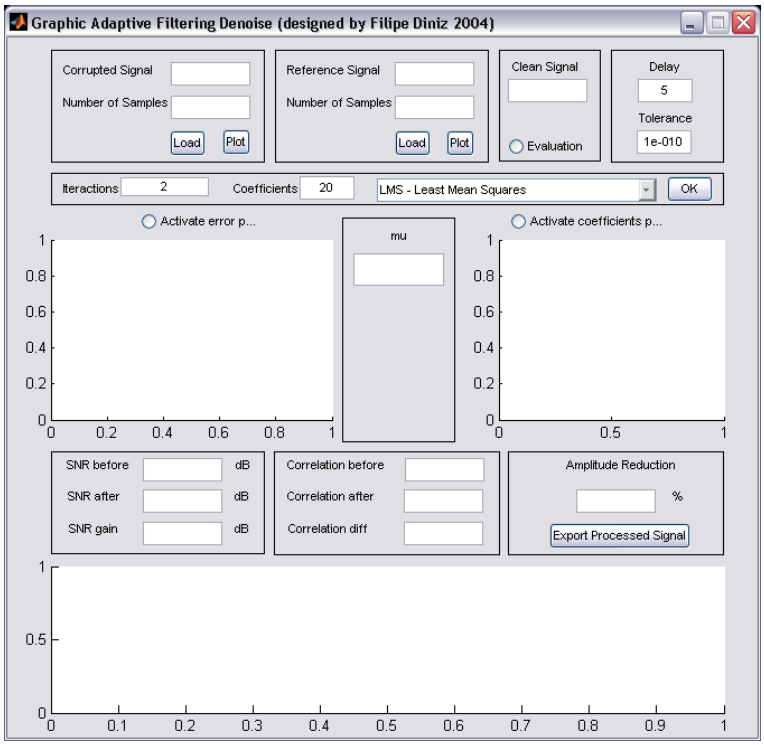

Figura 29: Toolbox desenvolvido para MATLAB - Filtragem Adaptativa.

3. Características do filtro: Algoritmo, campo *Coefficients* para definição do número de coeficientes e *Delay* para definição do atraso do sinal desejado em relação ao sinal de entrada.

4. Campo para definição de variáveis particulares de cada algoritmo.

O *toolbox* possui três gráficos: o primeiro, controlado pelo botão *Activate Error Plot*, exibe o gráfico do erro médio quadrático ao longo do tempo, em dB. O segundo, controlado pelo botão *Activate Coefficients Plot*, exibe gráfico com os valores dos coeficientes. Ambos quando ativados, são atualizados em tempo real de processamento, permitindo acompanhar a convergência do erro e a estabilização no valor dos coeficientes. O terceiro gráfico compara o sinal obtido no processamento com o sinal ideal, utilizado para validação.

Entre os algoritmos implementados estão os discutidos no capítulo de técnicas: LMS, NLMS e RLS. Para cada um deles é possível selecionar o valor das constantes relevantes: o passo de convergência μ do LMS, o passo inicial μ_n e o coeficiente γ que controla o valor máximo do passo no NLMS, e a constante de

esquecimento λ do RLS.

4.4 Toolbox - Filtros Digitais

O MATLAB contém dois *toolboxes* na área de filtros e processamento de sinais, que apresentam excessivas funcionalidades, contrárias a proposta de didática deste projeto. Portanto, este pacote foi desenvolvido completamente para o projeto, de forma a reunir e resumir os principais métodos de projeto de filtro numa só ferramenta. A seguir, descrição dos componentes e funcionalidades do pacote.

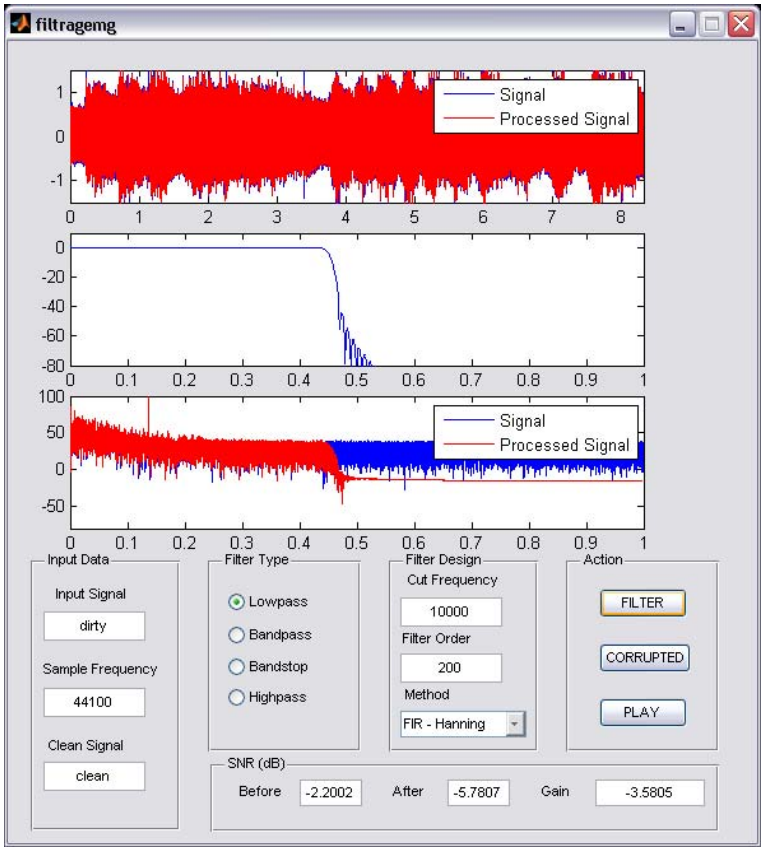

Figura 30: Toolbox desenvolvido para MATLAB - Projeto de filtros e processamento de sinais de áudio.

1. *Input Data* - Dados de entrada.

   - Caixa de texto para inserção do sinal de entrada (*Input Signal*): Carrega um sinal que esteja armazenado numa variável na área de trabalho do MATLAB.

- Caixa de texto para inserção da freqüência de amostragem (*Sample Frequency*): Determina a freqüência de amostragem do sinal de entrada.

2. *Filter Type* - Tipo de Filtro. Determina o tipo de filtro a ser utilizado no processamento, que pode ser um passa-baixas, faixas, altas ou rejeita-faixas.

3. *Filter Design* - Características do filtro.

   - Caixa de texto para inserção da(s) freqüência(s) de corte (*Cutoff Frequency*): Determina a freqüência de corte no caso dos filtros passa-baixas e passa-altas, as freqüências relativas ao intervalo de passagem no filtro passa-faixas e ao intervalo de rejeição no filtro rejeita-faixas. Para os rejeita-faixa e passa-faixas, só é possível uma banda de rejeição/passagem respectivamente.

   - Caixa de texto para inserção da ordem desejada para o filtro. O padrão é $N = 200$ para filtros FIR e $N = 10$ para filtros IIR.

   - Menu para a escolha do método de geração do filtro (*Method*): Determina o método a ser utilizado para o projeto do filtro.

4. *Action* - Opções de processamento.

   - Botão FILTER: Realiza o processamento do sinal de entrada, utilizando o filtro projetado. Plota três gráficos na janela principal do toolbox: o sinal antes e depois do processamento, no domínio do tempo; a resposta em freqüência do filtro, plotada de 0 a 1 (freqüência normalizada em π rad/amostras); por último, o espectro do sinal antes e depois do processamento.

   - Botão CORRUPTED: Reproduz o sinal antes do processamento.

   - Botão PLAY: Reproduz o sinal após o processamento.

Abaixo, a descrição das características de cada um dos métodos de geração dos coeficientes utilizado no toolbox:

1. FIR - Hanning: fir1(ordem,Wn,'tipo',hanning(ordem+1)) - Gera um filtro FIR utilizando a janela de Hanning.

2. FIR - Hamming: fir1(ordem,Wn,'tipo',hamming(ordem+1)) - Gera um filtro FIR utilizando a janela de Hamming.

3. FIR - Blackmann: fir1(ordem,Wn,'tipo',blackman(ordem+1) - Gera um filtro FIR utilizando a janela de Blackmann.

4. IIR - Butterworth: butter(ordem,Wn,'tipo') - Gera um filtro IIR utilizando o método de Butterworth.

5. IIR - Eliptico: ellip(ordem,Rp,Rs,Wn,'tipo') - Gera um filtro IIR utilizando o método Elíptico, com ripple na banda de passagem $R_p = 0.5$ e ripple na banda de rejeição $R_s = 20$.

6. IIR - Chebyshev: cheby1(ordem,Rp,Wn,'tipo') - Gera um filtro IIR utilizando o método de Chebyshev, com ripple na banda de passagem $R_p = 0.5$.

7. IIR - Notch: iirnotch(Wn,BW) - Gera um filtro Notch de ordem 2.

O funcionamento do Toolbox é simples: a partir do sinal importado da área de trabalho do MATLAB e da freqüência de amostragem digitada na caixa de texto correspondente, o usuário seleciona o tipo de filtro desejado. Após isso, ele deve inserir no campo *Cutoff Frequency* a(s) freqüência(s) de corte do filtro a ser projetado. O método então é escolhido no menu *Method* e o processamento é realizado com um clique no botão *FILTER*. Este comando preenche os três gráficos com os sinais no domínio do tempo, a resposta em magnitude do filtro e a transformada de Fourier dos dois sinais. Após isso, o usuário pode utilizar o botão *CORRUPTED* ou *PLAY* para escutar o sinal antes e depois do processamento, para comparação do resultado.

Este *toolbox* encerra o conjunto de ferramentas para o projeto. A próxima seção descreve o exemplo que foi desenvolvido para ilustrar a utilização conjunta dos *toolboxes* num problema de supressão de ruído.

4.5 Exemplo

Ao contrário dos exemplos que acompanham as apresentações, que visavam justificar a utilização de determinada técnica, o objetivo deste exemplo é mostrar como as ferramentas do projeto podem ser utilizadas em conjunto de forma a extrair múltiplos tipos de ruídos do sinal corrompido. O objetivo é justificar a capacidade do conjunto de dar ao usuário suporte necessário para lidar com situações onde a aplicação de apenas uma técnica não é suficiente para a melhor performance.

Neste exemplo apenas não foi utilizado o pacote de filtragem adaptativa. Por utilizar dois sinais no processamento, o seu escopo é diferente dos outros pacotes e por isto entendemos que não foi conveniente forçar sua utilização aqui.

O Sinal

O sinal utilizado neste exemplo foi extraído do trecho de encerramento da música "Rhapsody in Blue". Por ser tocada por uma orquestra, o espectro é rico, contendo componentes harmônicas em quase todo o domínio da freqüência. Portanto, cada método utilizado para remover ruído acaba removendo parte do sinal, e o teste comparativo ao final do processamento serve para validar subjetivamente a qualidade dos métodos combinados. A freqüência de amostragem foi utilizada de 44100 Hz.

O ruído

O ruído é composto de três sinais distintos, sendo cada um alvo de um toolbox particular. O primeiro é um sinal senoidal de 3 kHz. O segundo é ruído branco, criado com a função *randn*() do MATLAB. O terceiro são diversos "clicks", ruído impulsivo, espalhados pelo sinal. O objetivo é utilizar o toolbox de filtragem para remover o sinal senoidal, o toolbox de Wavelets para remover os clicks e o toolbox de filtragem de Wiener para remover o ruído branco.

Primeira Parte - Removendo o sinal senoidal

A adição do sinal senoidal no exemplo serve para justificar o uso de filtragem

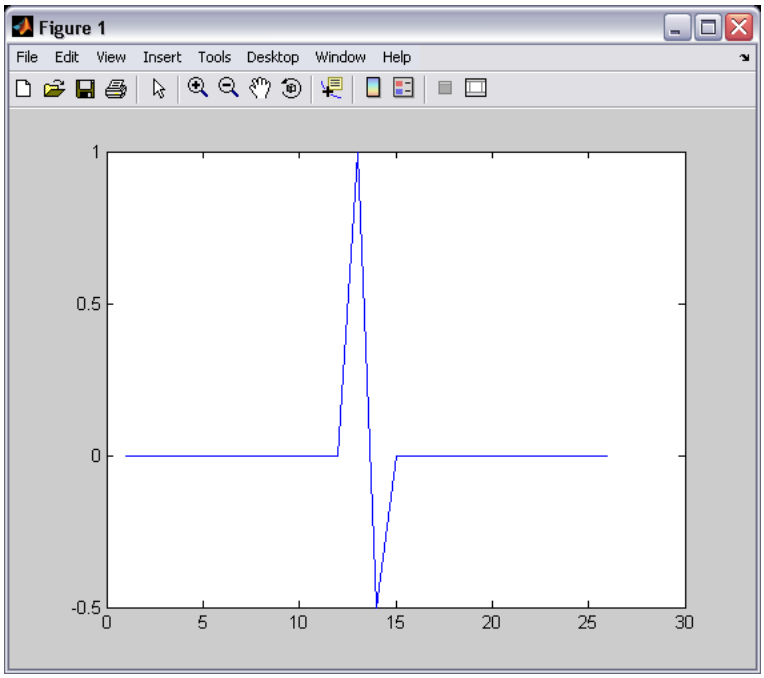

Figura 31: Detalhe do "Click" utilizado como ruído impulsivo no exemplo. O sinal corresponde a duas amostras no domínio do tempo.

para remoção de sinal "sintonizado", que ocupe apenas uma freqüência, ou uma banda estreita centrada em uma. Poderia ter sido utilizado um filtro rejeita-faixa para remover o ruído neste exemplo, mas como sua freqüência era conhecida, foi utilizado um filtro Notch em 3 kHz. Este filtro foi suficiente para remover totalmente o sinal senoidal do áudio. O ganho da SNR foi de 16.4771 dB.

Segunda Parte - Removendo os Clicks

O click é um tipo de ruído de curta duração no tempo, cuja energia se espalha pelo espectro na transformada de Fourier. Escolhendo uma Wavelet que seja capaz de concentrar esta energia em alguns coeficientes, ficando o resto do sinal espalhado por todos, basta eliminar aqueles onde o ruído é relevante para eliminar os clicks.

O sinal foi decomposto utilizando a Wavelet *Daubechies* de tamanho 2, que é a melhor para este tipo de click [3]. Foram utilizados cinco bandas, e na figura 33 é possível ver como o ruído ficou concentrado na primeira banda. Portanto, apenas descartando a informação da mesma, foi possível remover o ruído impulsivo. O ganho da SNR foi de 2.6312 dB.

Terceira Parte - Removendo o ruído branco

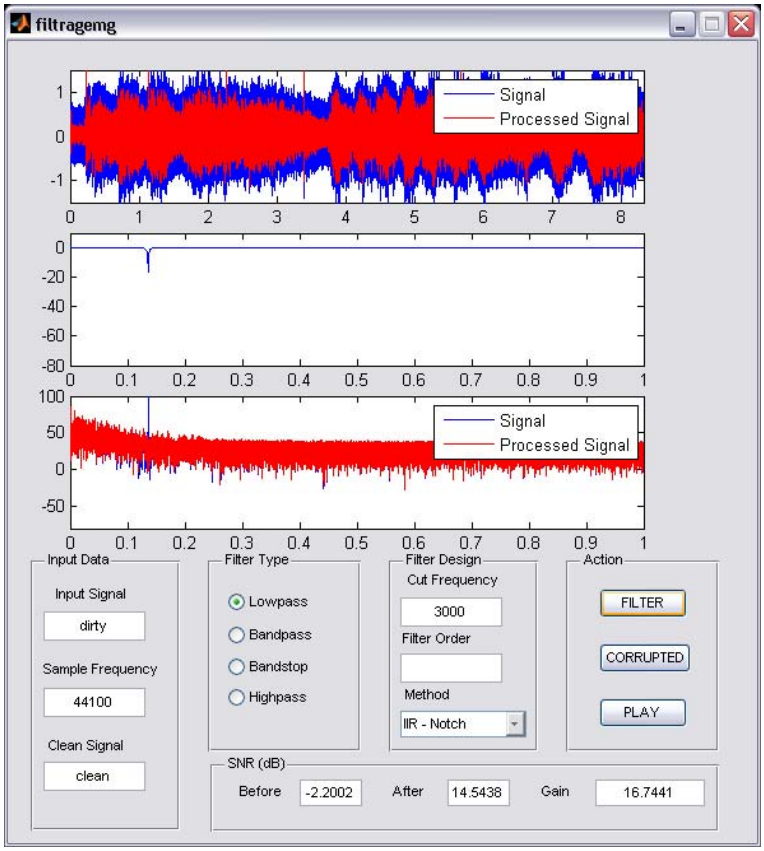

Figura 32: Utilização do Toolbox de Filtragem para remoção do ruído senoidal. Os gráficos comparam os sinais antes e depois do processamento: o primeiro mostra o domínio do tempo, o segundo mostra a resposta em freqüência do filtro, e o terceiro as transformadas de Fourier.

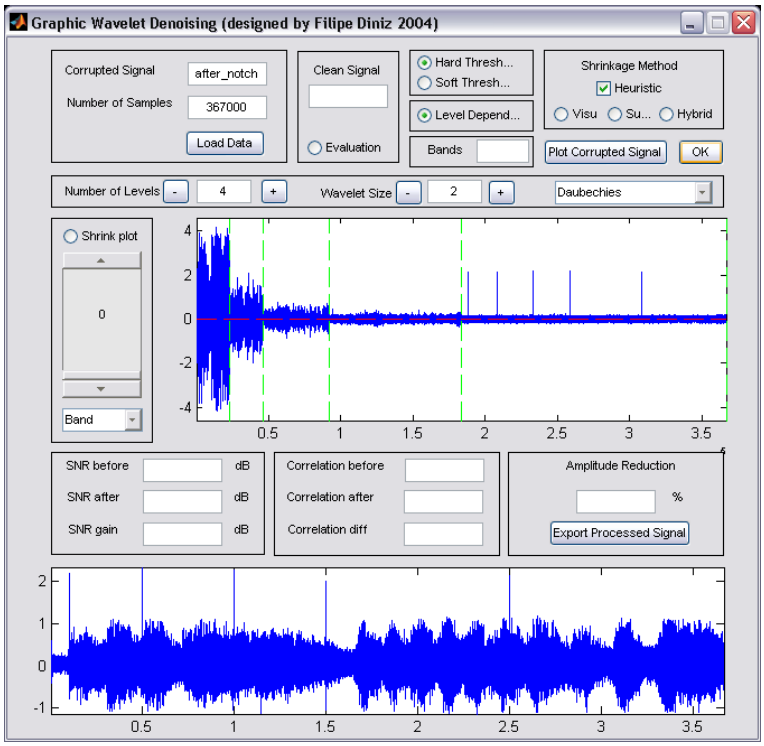

Figura 33: Utilização do Toolbox de Wavelets para remoção dos clicks: Decompondo o sinal utilizando a Wavelet Daubechies, é possível ver os clicks concentrados no maior nível de detalhe.

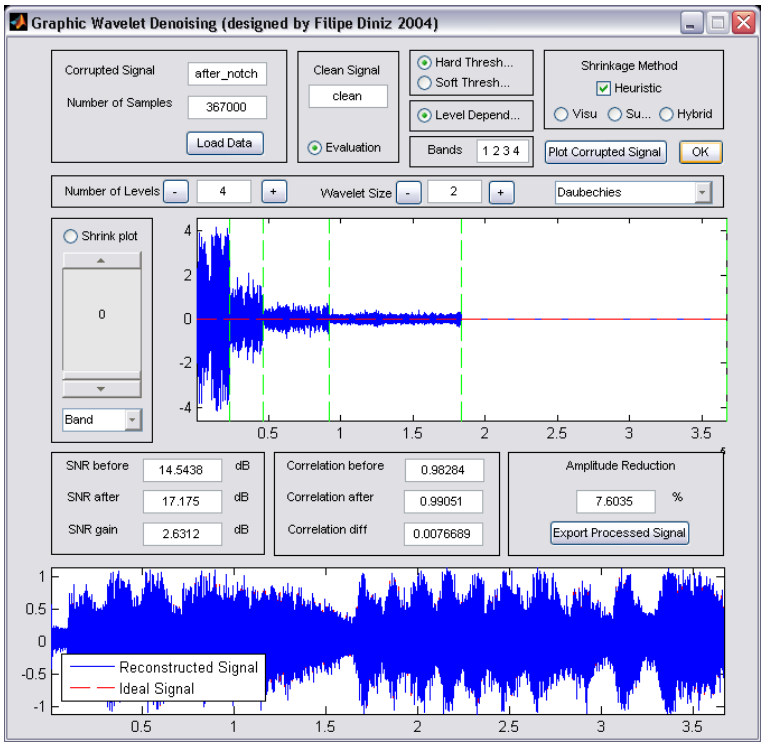

Figura 34: Utilização do Toolbox de Wavelets para remoção dos clicks: Zerando a banda onde o ruído impulsivo estava concentrado, foi possível removê-lo do sinal.

O último processamento corresponde à separação do ruído branco da música. Para isto, foi utilizado o filtro de Wiener. Nesta parte podemos ver a contribuição das constantes α e β utilizadas para tratar o ruído musical. Para isso, o sinal foi processado de formas distintas, variando essas duas constantes e observando os resultados. Para todas as combinações das variáveis, foram utilizadas janelas de 20 ms (882 amostras) com *overlapping* de 50%. A superestimação do ruído (aumentando α) acarreta numa maior remoção, porém aumenta o ruído musical residual do processamento. A variação do β visou contrabalancear este efeito, mascarando o ruído musical com um chão de ruído de fundo. O valor 5 para α foi encontrado realizando testes para uma gama de valores, sendo este suficiente para suprimir o ruído branco sem distorcer demais a música. O valor de β também foi variado de forma a encontrar um ponto satisfatório.

Tabela 7: Combinações das constantes de tratamento de ruído musical

Nome	α	β
WienerA1B0	1	0
WienerA3B1	3	0.1
WienerA5B0	5	0
WienerA5B1	5	0.1
WienerA5B3	5	0.3
WienerWN	1	0

Tabela 8: Ganho para cada Combinação

Filtro	Ganho SNR (dB)
WienerA1B0	-2.5579
WienerA3B1	-4.8313
WienerA5B0	-6.6686
WienerA5B1	-6.2364
WienerA5B3	-5.7297
WienerWN	-0.35553

A combinação chamada de WienerWN (*Wiener White Noise*)utilizou ruído branco para estimar o espectro, sem retirar a amostra do trecho de sinal. Portanto, esta foi a combinação que obteve o melhor resultado subjetivo. Em seguida a com-

binação WienerA3B1 conseguiu remover um pouco mais do que a WienerA1B0, que era o padrão de comparação, e o chão de ruído deixado conseguiu mascarar o efeito do ruído musical. As combinações usando $\alpha = 0.5$ resultaram em muito ruído musical de fundo, e mesmo fazendo $\beta = 0.3$ não foi suficiente para eliminar o efeito desagradável.

Este exemplo serviu para demonstrar a capacidade do conjunto de *toolboxes* para resolver problemas mais complexos de supressão de ruído, onde apenas a aplicação de uma técnica não seria suficiente. Serve para ilustrar as funcionalidades das ferramentas e seu poder de processamento e por fim, para justificar o objetivo do trabalho, que é apresentar, explicar e implementar diversos métodos de processamento de sinais, e suas aplicações no escopo do tratamento de ruído.

4.6 Conclusão

Neste capítulo foram apresentados os *toolboxes* que acompanham os módulos e exemplificam as técnicas descritas em cada apresentação. Para o toolbox de filtragem de Wiener, foi explicada a alteração feita, a inclusão do tratamento de ruído musical. para o toolbox de filtros, apresentada a proposta do pacote e como foi desenvolvido o projeto.

O exemplo serviu para demonstrar a utilização dos *toolboxes* e para exibir como as técnicas podem ser combinadas de forma a tratar combinações complexas de sinais ruidosos. Mostrou o funcionamento individual de cada um aplicado a um tipo particular de ruído, e o resultado do processamento após cada trecho. O capítulo a seguir apresenta a conclusão do trabalho quais são os futuros caminhos para o projeto.

5 Conclusão

Este capítulo conclui este trabalho, resumindo o que foi feito ao longo do mesmo, os resultados obtidos e quais as possíveis extensões e trabalhos futuros para o projeto. Relembrando, o primeiro capítulo expôs o objetivo do trabalho, vista a necessidade crescente do mundo moderno em tratar ruído. O segundo capítulo mostra quais técnicas foram alvo deste trabalho que figuraram nas apresentações e nos algoritmos. O terceiro capítulo descreveu as técnicas e seus exemplos, e o quarto os *toolboxes* do MATLAB.

5.1 Resultados Alcançados

A proposta deste trabalho foi acrescentar material ao projeto que visava apresentar, explicar e implementar diversas técnicas de supressão de ruído. Dentro do escopo do projeto, foram alcançados por este trabalho os seguintes resultados:

- No segundo capítulo foram resumidas algumas técnicas de processamento de sinais vistas em diversas disciplinas do curso de engenharia eletrônica. Estas técnicas, que normalmente são vistas em diversas áreas da engenharia, descorrelacionadas, foram reunidas e exemplificadas no escopo da supressão de ruído. O trabalho neste capítulo consistiu no estudo de cada uma e na uniformização das mesmas no âmbito do projeto. O resultado alcançado foi uma seqüência encadeada de métodos, de forma a tornar dinâmicos e intuitivos a explanação e o ensino da supressão de ruído.

- No terceiro capítulo foram exibidas as apresentações, que têm a função de

introduzir e exemplificar cada uma das técnicas. Para o primeiro método, transformada de Fourier, foram desenvolvidos três exemplos. Para o segundo, Filtragem Digital, foram geradas duas apresentações, uma para filtros FIR e uma para filtros IIR, e foram desenvolvidos cinco exemplos conjuntamente. Para filtragem de Wiener e subtração espectral foi criada uma apresentação e um exemplo comparando os dois métodos e o método de redução de ruído musical. Para a última apresentação desenvolvida, filtragem adaptativa, foram gerados quatro exemplos. Todos os exemplos serviram para demonstrar a aplicação da técnica sobre um tipo determinado de problema de supressão de ruído. Para cada exemplo foram desenvolvidos scripts no MATLAB, implementando as técnicas e todos os algoritmos de pré-processamento.

- No quarto capítulo foram descritos os toolboxes, as alterações efetuadas e a descrição do *toolbox* que foi desenvolvido completamente para o trabalho. Para o *toolbox* de Filtragem de Wiener, a adição do algoritmo de redução de ruído musical visto no terceiro capítulo. Foi descrito também o *toolbox* de Filtragem Digital, que foi completamente desenvolvido neste trabalho. Além destas implementações, foi gerado um exemplo para justificar a capacidade de utilização conjunta de técnicas para a resolução de problemas na qual um método apenas não obteria resultados satisfatórios. Este exemplo completa os exemplos desenvolvidos no terceiro capítulo que visaram demonstrar individualmente as técnicas.

5.2 Trabalhos futuros e possíveis extensões

O tema supressão de ruído é vasto, e os resultados alcançados neste trabalho, embora tenham abrangido algumas parcelas de tudo que é visto na engenharia nesta área, pode ser expandido. Para o projeto proposto por este trabalho, podemos citar dois pontos em que tal expansão possa ser realizada:

- A única parte do projeto que não fez parte do escopo deste trabalho foram as apostilas com o conteúdo matemático das técnicas. Embora elas tenham sido

resumidas e apresentadas ao longo do segundo capítulo e em cada apresentação, as fórmulas e teorias que determinam cada técnica não foram completamente apresentadas. Portanto, o próximo passo dentro do projeto desenvolveria as apostilas explicando a fundo a matemática por trás de cada método de processamento de sinais visto.

- Outra extensão interessante seria o acréscimo de exemplos de supressão de ruído em outras áreas da engenharia. Neste trabalho foram desenvolvidos exemplos dentro do ramo de processamento de sinais digitais. Para justificar a universalidade do problema e da aplicação das técnicas vistas neste trabalho, exemplos desenvolvidos utilizando os *toolboxes* em outras áreas seria de muito valor.

Este trabalho reuniu um conjunto de técnicas que também pode ser expandido. O objetivo foi abrangir os principais métodos utilizados. Junto com o crescente desenvolvimento e evolução da ciência da supressão de ruído podem ser acrescentadas novas técnicas, novos exemplos e implementados novos *toolboxes*.

Referências

[1] L. W. P. Biscainho, "Restauração digital de sinais de áudio provenientes de gravações musicais degradadas," Dezembro 2000.

[2] F. C. da Costa Beltrão Diniz e S. L. Netto, "A package tool for general-purpose signal denoising," *Proceedings IEEE*, vol. 86, pp. 573–576, Junho 2005.

[3] F. C. da Costa Beltrão Diniz, "Supressão de ruído, detecção e classificação de sinais de descargas parciais em transformadores de potência," Fevereiro 2005.

[4] P. S. R. Diniz, *Adaptive Filtering*, 1st ed. Kluwer Academic Publishers, 2001.

[5] P. S. R. Diniz, E. A. de Barros da Silva, e S. L. Netto, *Processamento Digital de Sinais*, 1st ed. Editora Bookman, 2004.

[6] S. Godsill, *Digital Audio Restoration*, 1st ed. Editora Springer-Verlag London, 2004.