Uma Análise Espectral de Sinais Sonoros: Identificação de Notas Musicais e Implementação de Afinadores Digitais com Aplicação em Filtros de Butterworth e Filtros Sinc Janelados

Mateus S. Araújo[1], **Robert de A. Cabral**[1], **Paulo Armando C. Aguilar**[1]

[1]Universidade Federal do Ceará (UFC) – Campus Quixadá
Caixa Postal 63902.580 – Quixadá – CE – Brasil

{mateuseng_ec, robertcabral}@alu.ufc.br

cavalcante.aguilar@gmail.com

Abstract. *This article is intended for the construction of FIR and IIR digital filters using mathematical manipulation tools such as Matlab and Octave for analysis and comparison of algorithms at the processing level. To do this, it is necessary to understand some basic concepts of filtering in the frequency domain such as Tustin discretization and windowed filtering. A function has also been created that allows the spectral analysis of sound signals in the frequency domain with operation exemplified in the identification of musical notes by various instruments. Because they are widely used filters in the signal processing field, the Hamming windowed filters and the Butterworth filters allow audio systems to be built with greater fault tolerance control, thereby allowing noise or undesirable features to be omitted in the end result of the processed signal. The analysis in the frequency domain of the filtered audio will use the same function used to identify the musical notes. At the end of the project a digital guitar tuner was also implemented.*

Resumo. *Este artigo destina-se na construção de filtros digitais FIR e IIR utilizando ferramentas de manipulação matemática como o Matlab e Octave para análise e comparação dos algoritmos a nível de processamento. Para tal, é necessário compreender alguns conceitos básicos de filtragem no domínio da frequência como discretização de Tustin e filtragem por janelamento. Também foi criado uma função que permite a análise espectral de sinais sonoros no domínio da frequência com funcionamento exemplificado na identificação de notas musicais por instrumentos diversos. Por serem filtros de vasta utilização no campo de processamento de sinais, os filtros janelados de Hamming e os filtros de Butterworth permitem que sistemas de áudio sejam construídos com maior controle de tolerância a falhas, permitindo dessa forma, que ruídos ou características indesejáveis estejam omissas no resultado final do sinal processado. A análise no domínio da frequência dos áudios filtrados se utilizará da mesma função utilizada para identificação das notas musicais. No final do projeto também foi implementado um afinador digital para violão.*

1. Introdução

Para extrair informações desejadas de um sinal é necessário um dispositivo que selecione frequências de interesse que compõem o mesmo. Tal dispositivo é denominado de filtro,

cuja resposta em frequência é caracterizada por uma faixa de passagem, uma faixa de rejeição e uma faixa de transição. Sinais que possuem frequência dentro da faixa de passagem são recuperados com pouca ou nenhuma distorção, visto que aqueles sinais que possuem frequências dentro da faixa de rejeição são completamente atenuados. Dessa forma, os filtros podem ser classificados como passa alta, passa baixa, passa faixa ou rejeita faixa de acordo com a característica de interesse do sinal que se deseja remover [Welch et al. 2016].

2. Condições para Transmissão sem distorções

Um filtro passa baixas ideal é aquele no qual preserva-se todas as frequências dentro da faixa de passagem e atenua por completo todas as frequências na faixa de transição. Tal filtro será construído tomando como base o seguinte sistema linear invariante no tempo:

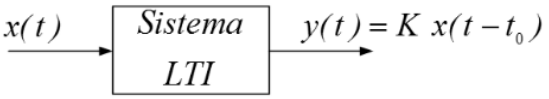

Figura 1. Diagrama de blocos do filtro

Aplicando uma Transformada de Fourier nos sinais de entrada $x(t)$ e saída $y(t)$ da equação da figura 1 temos a seguinte função de transferência do sistema:

$$H(j\omega) = \frac{Y(j\omega)}{X(j\omega)} = Ke^{-j\omega t_0} \tag{1}$$

Com base na função da resposta em frequência acima, determina-se que para uma transmissão sem distorções desse sistema o módulo de $H(j\omega)$ seja constante e que o argumento de $H(j\omega)$ seja linear [Stearns and Hush 2016]. O que irá caracterizar o filtro como um passa baixas será a definição de uma frequência de corte ω_c tal que sua função resposta em frequência seja da seguinte forma:

$$H(j\omega) = \begin{cases} Ke^{-j\omega t_0},\ -\omega_c \leq \omega \leq \omega_c \\ 0,\ |\omega| > \omega_c \end{cases} \tag{2}$$

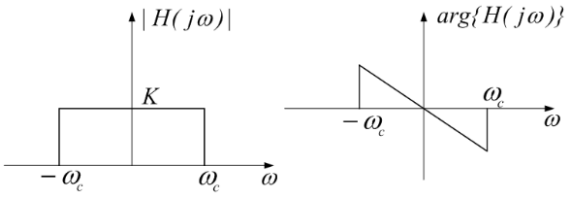

Figura 2. Resposta em fase e magnitude do sistema

A resposta ao impulso deste filtro pode ser obtida fazendo uma Transformada de Fourier inversa de $H(j\omega)$, resultando em:

$$h(t) = \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi}(t - t_0)\right) \tag{3}$$

A equação 2 caracteriza uma resposta do tipo não causal. Sistemas não causais dependem de valores de entrada futuros, o que acarreta da sua não existência real para a construção de filtros ideais [Acharya et al. 2014].

3. Projeto de Filtros

Diferente dos filtros ideais, não realizáveis na prática, na construção de filtros consideram-se alguns desvios e especificações que são utilizados como parâmetros de projeto. São eles:

- Dentro da faixa de passagem a resposta do filtro deve situar-se entre 1 e 1 - ε, tal que:
$$1 - \varepsilon \leq |H(j\omega)| \leq 1, \text{ para } \omega_{p1} \leq \omega \leq \omega_{p2} \tag{4}$$
Onde $\omega_{p1} - \omega_{p2}$ definem a faixa de passagem do filtro e ε é um parâmetro de tolerância.
- Dentro da faixa de rejeição, a resposta em módulo do filtro não deve ultrapassar δ:
$$|H(j\omega)| \leq \delta \tag{5}$$
Onde ω está contido na zona de rejeição e δ é um outro parâmetro de tolerância.
- A faixa de transição (FT) é composta pelas frequências ω_p e ω_s contidas entre a faixa de passagem (FP) e a faixa de rejeição (FR).

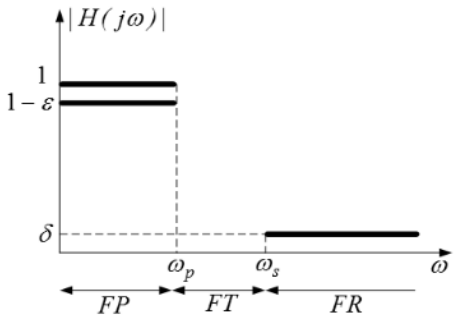

Figura 3. Especificações para resposta em frequência de um filtro passa baixas

4. Implementação dos Filtros de Butterworth

A construção dos filtros de Butterworth são realizadas com base nas frequências na faixa de passagem e rejeição, nos parâmetros ε e δ e a função que representa a resposta em frequência para este tipo de filtro [Podder et al. 2014]. Cuja função pode ser escrita como:

$$H(s) = \frac{\omega_c^N}{Q(s)} \tag{6}$$

Onde ω_c é a frequência de corte do filtro de Butterworth, N é a ordem e $Q(s)$ é um polinômio característico já tabelado que depende de N. Para a construção do filtro no Matlab/Octave é necessário seguir alguns passos:

(I) Fazer as especificações do Filtro.
(II) Descobrir a ordem N.
(III) Descobrir a frequência de corte ω_c.
(IV) Montar a função de transferência $H(s)$.
(V) Implementar o algoritmo.

Vamos especificar os valores de ε como sendo $0,1$ na tolerância na faixa de aceitação e $0,7$ para o δ na tolerância na faixa de rejeição. A frequência de passagem ω_p será 3 kHz e a de rejeição ω_s será $8,5$ kHz. Para encontrar a ordem N do filtro, devemos considerar a seguinte relação:

$$N = \frac{\log\left(\frac{(2\varepsilon - \varepsilon^2)\delta^2}{(1-\varepsilon)^2(1-\varepsilon^2)}\right)}{2\log\left(\frac{\omega_p}{\omega_s}\right)} \tag{7}$$

Usando as especificações e a equação 7, a ordem N é igual a $3,35$. Por padrão de projetos de filtros arredondamos o resultado para o inteiro mais próximo, nesse caso, $N = 4$. Para encontrar a frequência de corte, é necessário fazer uma média simples da frequência de corte de rejeição e a frequência de corte de passagem, ambas respectivamente, calculadas da seguinte forma:

$$\omega_c = \omega_s \left(\frac{1-\delta^2}{\delta^2}\right)^{-1/2N} \tag{8}$$

$$\omega_c = \omega_p \left(\frac{2\varepsilon - \varepsilon^2}{(1-\varepsilon)^2}\right)^{-1/2N} \tag{9}$$

Utilizando as especificações e fazendo a média aritmética simples de 8 e 9, temos que a frequência de corte do filtro passa baixas é 25028 rad/s ou 25.10^3 rad/s. Podemos consultar o valor de $Q(s)$ com ordem 4 e substituir na equação característica 6, resultando na seguinte função de transferência:

$$H(s) = \frac{3,9.10^{17}}{s^4 + 6,5.10^4 s^3 + 2,1.10^9 s^2 + 4.10^{13} s + 3,9.10^{17}} \tag{10}$$

A equação 10 pode não significar tanta coisa em um primeiro momento já que a mesma se demonstra simples e é expressa apenas em uma fração [Shively 1975]. No entanto, se construirmos um algoritmo que aplique essa função em qualquer sinal de áudio podemos ver claramente a passagem de sons mais graves pelo filtro. Para isso, o Matlab será utilizado para tal objetivo.

5. Método de discretização de Tustin ou Transformada Z Bilateral

A transformada Z é um caso mais amplo da Transformada de Fourier de tempo discreto (DTFT). Tal transformada é um método matemático simples que permite construir equivalentes digitais a partir de uma função de transferência de componentes analógicos. O método de Tustin é umas das técnicas mais conhecidas para mapeamento de pólos e zeros

de uma função de tempo contínuo para um sistema invariante no tempo de domínio discreto. De forma mais simples, podemos escrever uma função de um filtro digital $H(z)$ a partir de um um filtro analógico $H(s)$ [Mettke et al. 1994]. A transformada bilinear pode ser descrita como $H(z) = H(s)$ com s sendo:

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \tag{11}$$

Basta tomar toda função em $H(s)$ e todas as variáveis acompanhadas por s e substituir pelo termo equivalente em z, e assim simplificar a expressão. Com isso, obteremos a função de transferência em tempo discreto do protótipo analógico.

5.1. Exemplo de um circuito RC

Podemos considerar um circuito RC simples em série e montar a sua função de transferência:

$$H(s) = \frac{1}{sRC + 1} \tag{12}$$

Podemos aplicar, em seguida, a transformada bilinear. Dessa forma, obtemos:

$$H(s) = \frac{1}{\frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) RC + 1} \tag{13}$$

Podemos simplificar a equação acima até obter a função de transferência final na forma geral de um filtro IIR:

$$H(s) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} \tag{14}$$

Os coeficientes são definidos da seguinte forma:

$$a_1 = \frac{-2RC + T}{2RC + T} \tag{15}$$

$$b_0 = \frac{T}{2RC} \tag{16}$$

$$b_1 = \frac{T}{2RC} \tag{17}$$

Chegado na função de transferência, basta agora tomar a transformada Z inversa e chegar na equação de diferenças para ser programada no Octave. Assim, obtemos a função de diferenças abaixo:

$$u[n] = b_0 e[n] + b_1 e[n-1] + a_1 u[n-1] \tag{18}$$

Definindo os valores de $R = 10k\Omega$ e $C = 1uF$, nossa função de transferência $H(s)$ será:

$$H(s) = \frac{1}{0.01s + 1} \tag{19}$$

A plotagem desse sistema foi feito no Octave a fim de escolher uma taxa de amostragem. O código abaixo [Araújo 2018a] exemplifica o que foi feito:

```
%Cria funcao de transferencia analogica

Hs = tf([1] ,[0.01 1]);

%Plota a resposta ao degrau

step(Hs);
```

Como resultado, temos a resposta ao degrau do sistema como mostrado na figura 4:

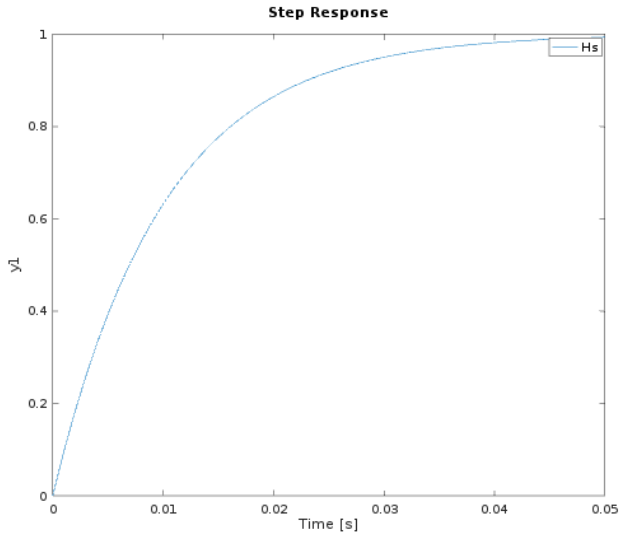

Figura 4. Plot da resposta do filtro exemplo

Podemos simplificar todo o cálculo feito na mão usando a função **c2d** do Octave, responsável por entregar a função de transferência discreta [Storn 1996]. Vamos adicionar ao nosso script [Araújo 2018a] o método de **tustin** como um dos parâmetros. Portanto, o novo script será:

```
%Cria funcao de transferencia analogica

Hs = tf([1], [0.01 1]);

%Discretiza a funcao de transferencia com amostragem de 1000Hz

Hz = c2d(Hs, 0.001, 'tustin');

```

```
9  %Plota a resposta ao degrau
10
11 step(Hs, '-', Hz, '--');
```

Após isso, temos o gráfico do sistema discretizado no tempo como mostrado na figura 5. Tal método ajuda muito na velocidade com que filtros são executados a nível de processamento [Sühling et al. 2004]. Apenas uma Transformada de Fourier é necessária para tal realização.

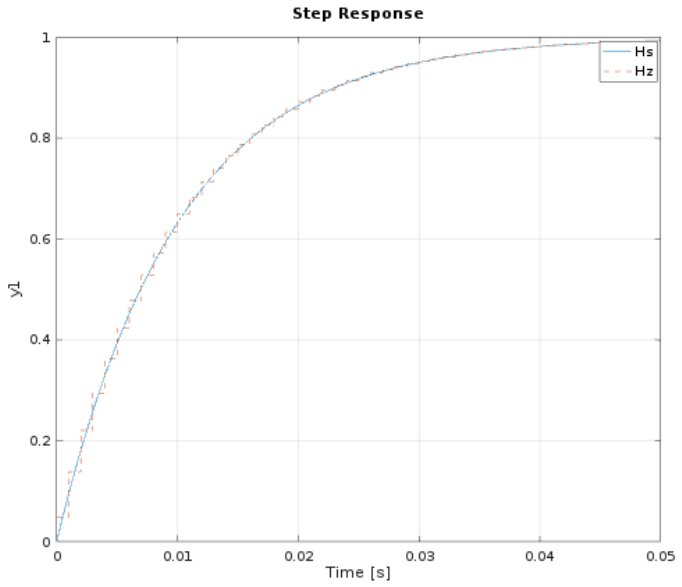

Figura 5. Resposta ao degrau do sistema discreto e contínuo

6. Algoritmo do Filtro Passa Baixas no Matlab/Octave

O algoritmo no Matlab/Octave consiste basicamente na construção da função $H(s)$ e na digitalização dessa função de transferência pelo método de Tustin [Jackson 2013]. Também deve ser levado em conta a criação de todas as especificações do filtro. A discretização pelo método de Tustin utiliza um filtro digital de Resposta ao Impulso Infinito (IIR) que se caracteriza em transformar o sinal de áudio no domínio do tempo para o domínio discreto na frequência [Stearns and Hush 2016]. Isso permite um poder de processamento muito mais rápido se fosse utilizado outras técnicas de processamento de áudio por filtros analógicos [Ristic et al. 2013]. O tamanho do arquivo de áudio nesse caso não é levado tanto em consideração, uma vez que o mesmo é quebrado em diversas partes e é feito automaticamente uma operação de convolução entre a função de transferência e o áudio já discretizado [Bose 1985]. Dessa forma, as frequências mais altas inclusas na faixa de rejeição são completamente atenuadas, deixando apenas as frequências de tonalidade mais graves serem perceptíveis ao ouvido humano [Podder et al. 2014]. Após a filtragem, um arquivo WAV pode ser obtido com tamanho bastante reduzido do original, já que características que antes estavam no áudio original foram filtradas de acordo com nossas especificações.

6.1. Criação da Função my_fft

Sinais no domínio do tempo na maioria das vezes não são de grande interesse para análise espectral. Dessa forma, podemos achar ou criar alternativas que nos forneça a plotagem de gráficos no domínio tempo para o domínio frequência, nos fornecendo assim, uma melhor visualização do comportamento de sinais de naturezas diversas [Brandenstein and Unbehauen 1998]. Com essa ideia, criamos uma função no Octave que recebe dois parâmetros iniciais para a plotagem de gráficos no domínio frequência usando a Transformada de Fourier, sendo o primeiro deles o sinal em si e o segundo parâmetro a amostragem desejada. O código desenvolvido se encontra abaixo [Araújo 2018b].

```
function [X,freq] = my_fft(x,Fs)

N = length(x);          % variavel N recebe o tamanho do vetor x
k = 0:N-1;              % k e um vetor que vai de zero ate N-1

% Vetor de tempo N dividido pela frequencia de amostragem
T = N/Fs;
freq = k/T;

% X recebe a FFT normalizada do vetor x sobre N
X = fftn(x)/N;
cutOff = ceil(N/2);     % cutOff ajusta o eixo X
X = X(1:cutOff);
figure();

% Plota a transformada de Fourier e o valor de X em modulo
plot(freq(1:cutOff),abs(X));
title('Fast Fourier Transform');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

end
```

O código acima demonstra a normalização dos eixos para uma melhor visualização dos coeficientes da Transformada de Fourier.

6.2. Funcionamento da my_fft para Funções Senoidais

Se assumirmos a criação de cinco sinais senoidais de amplitudes e frequências diferentes e somarmos, então teremos um sinal resultante muito complicado de ser analisado se plotado no domínio do tempo [Haykin 1985]. O código abaixo [Araújo 2018a] exemplifica esse processo:

```
fs = 10000;                 %freq. de amostragem
t = [0:1/fs:10];
s1 = sin(2*pi*50*t);        % sinal 1
s2 = 15*sin(2*pi*150*t);    % sinal 2
s3 = 40*sin(2*pi*80*t);     % sinal 3
s4 = 30*sin(2*pi*350*t);    % sinal 4
s5 = 65*sin(2*pi*200*t);    % sinal 5
```

```
8
9  s = s1+s2+s3+s4+s5;          % soma dos 5 sinais
10
11 plot(t,s);                   % plota o grafico (time)
12
13 % my_fft(s, fs);             % plota o grafico (frequency)
```

Como é perceptível na figura 6 o sinal resultante não nos fornece muitas informações sobre o comportamento final. Porém, se retirarmos o comentário da linha 13 do código, temos a resposta em frequência do sinal gerado como mostrado na figura 7.

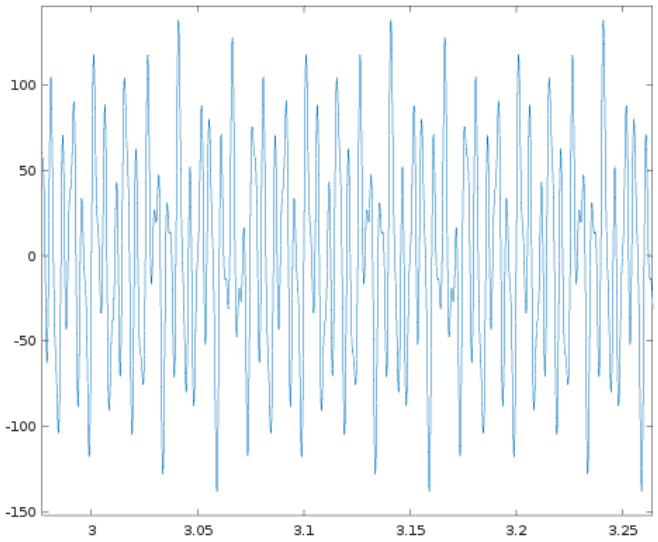

Figura 6. Soma dos sinais no domínio do tempo

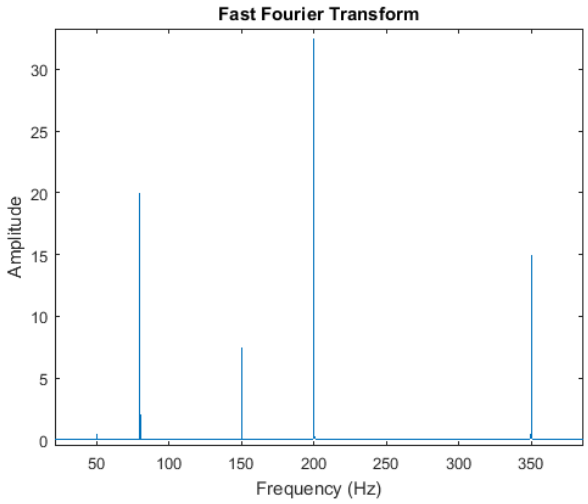

Figura 7. Soma dos sinais no domínio da frequência

O gráfico da figura 7 é bem mais informativo e nos informa os picos de amplitude para cada frequência no espectro [Lutovac et al. 2001]. Tal função será utilizada para a plotagem de arquivos de áudio no domínio da frequência para nos ajudar a identificar

o comportamento final do filtro, se o mesmo foi eficiente ou não nas especificações de construção [Litwin 2000]. A desvantagem de sinais de áudio é que os mesmos são compostos por diversos harmônicos e sua análise no domínio temporal fica praticamente impossível [Mertins and Mertins 1999]. A única diferença é que a função irá receber como primeiro parâmetro um sinal em formato **wav**.

6.3. Funcionamento da my_fft para Sinais Sonoros

Também fizemos um exemplo que identifica frequências de qualquer sinal sonoro gravado no Matlab usando a função my_fft. Dessa forma, testamos sons diversos com instrumentos musicais usando um violão e uma flauta. O programa [Araújo 2018d] capta o áudio gravado, plota no domínio do tempo e mostra na tela a frequência da nota tocada, a nota e sua oitava. É sabido que a frequência pode variar de cada instrumento para uma mesma nota devidos as oitavas acima ou abaixo da afinação. A tabela 1, mostra as frequências que servem para auxílio na identificação das notas e foi utilizado um violão para testes. Foi tocado uma nota Sol (G) a 194.5 Hz, uma nota Lá (A) a 220 Hz e uma nota Si (B) a 247 Hz na terceira oitava para mostrar o funcionamento da my_fft. Foi plotado logo em seguida o gráfico no domínio tempo e em frequência. O domínio tempo não nos dá tanta informação como mostrado nas figuras 8, 10 e 12. No entanto, os gráficos 9, 11 e 13 nos fornecem as frequências exatas das mesmas. O programa feito funciona para qualquer instrumento em qualquer nota e a verificação pode ser feita usando a tabela de frequências. No final do projeto também foi implementado um afinador digital para violão [Araújo 2018c] que funciona baseado na função criada que, por sua vez, será usada para análise dos filtros digitais.

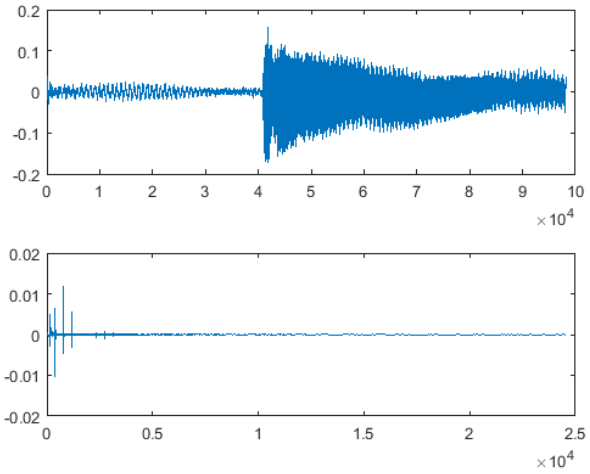

Figura 8. Nota sol a 194.5 Hz no domínio tempo

7. Especificações do Filtro de Butterworth

(I) Frequência de passagem wp = 300 Hz
(II) Frequência de rejeição ws = 750 Hz
(III) Perda de amplitude na faixa de passagem igual a -0.2228 dB
(IV) Perda de amplitude na faixa de rejeição igual a -13.01 dB

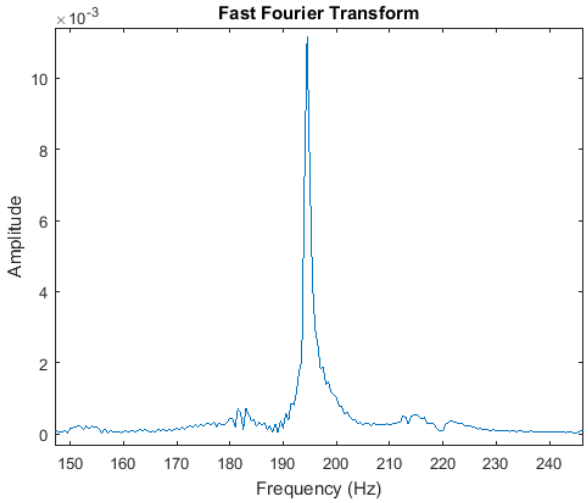

Figura 9. Nota sol (G) a 194.5 Hz no domínio frequência

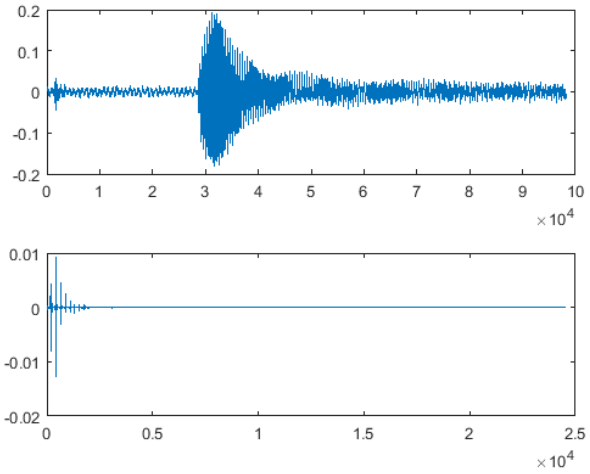

Figura 10. Nota lá (A) a 220 Hz no domínio tempo

O algoritmo feito no Matlab foi feito de acordo com as especificações acima para a construção do filtro. A técnica da discretização de Tustin também foi utilizada para um maior desempenho no algoritmo [Rabiner and Gold 1975]. O algoritmo abaixo [Araújo 2018a] exemplifica o que foi feito:

```
%|H(jw)| - Achar funcao de transferencia do filtro

% obtendo as especificacoes

% Hz para ciclos/seg = 2*pi/s

wp = 300*2*pi;                  % frequencia de passagem [rad/s]
ws = 750*2*pi;                  % frequencia de rejeicao [rad/s]

epsilon = 1-10^(-0.2228/20);    % 20log(1-eps) = -0.2228 dB
delta = 10^(-13.01/20);         % 20log(del) = -13.01 dB
```

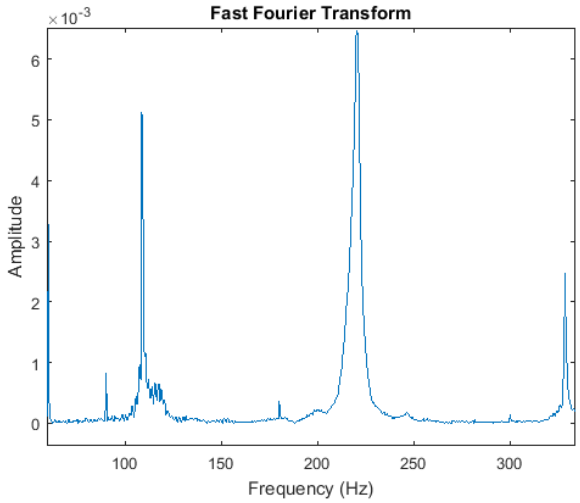

Figura 11. Nota lá (A) a 220 Hz no domínio frequência

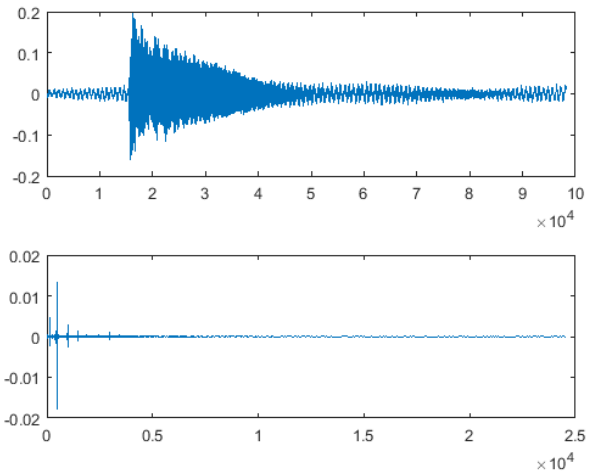

Figura 12. Nota Si (B) a 247 Hz no domínio tempo

```
12
13 % obtencao da ordem do filtro:
14 N = log10 (((2*epsilon - epsilon^2)*delta^2)/((1-epsilon)^2*(1-
       delta^2)))/(2*log10(wp/ws));
15 N = ceil(N);          % arredondamento p/ o proximo inteiro
16
17 % obtendo a frequencia de corte
18
19 wcp = wp/((2*epsilon - epsilon^2)/(1-epsilon)^2)^(1/(2*N));
20 wcs = ws/((1-delta^2)/delta^2)^(1/(2*N));
21 wc = (wcp + wcs)/2;              % media entre os dois
22
23 % funcao de transferencia do filtro de Butterworth ordem N
24
25 num = wc^N;                      % criacao de um numerador
```

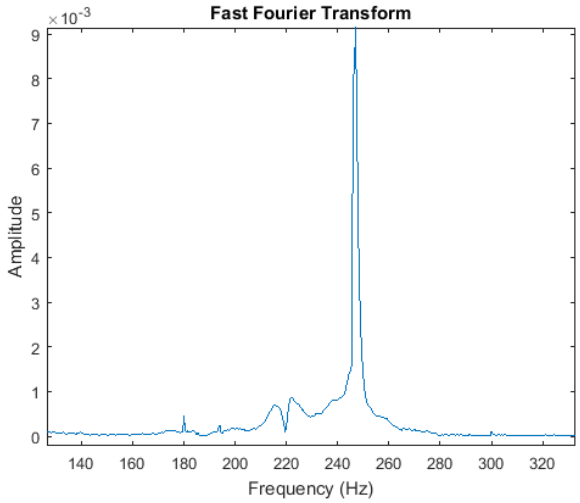

Figura 13. Nota Si (B) a 247 Hz no domínio frequência

Notas		1	2	3	4	5	6	7
1	**C**	32,70	65,40	130,81	261,62	523,25	1046,50	2093,00
2	**C#**	34,64	69,29	138,59	277,18	554,36	1108,73	2217,46
3	**D**	38,70	73,41	146,83	293,66	587,32	1174,65	2349,31
4	**D#**	38,89	77,78	155,56	311,12	622,25	1244,50	2489,01
5	**E**	41,20	82,40	164,81	329,62	659,25	1318,51	2637,02
6	**F**	43,65	87,30	174,61	349,22	698,45	1396,91	2793,82
7	**F#**	46,24	92,49	184,99	369,99	739,98	1479,97	2959,95
8	**G**	48,99	97,99	195,99	391,99	783,99	1567,98	3135,96
9	**G#**	51,91	103,82	207,65	415,30	830,60	1661,21	3322,43
10	**A**	55	110	220	440	880	1760	3520
11	**A#**	58,27	116,54	223,08	466,16	932,32	1864,65	3729,31
12	**B**	61,73	123,47	246,94	493,88	987,76	1975,53	3951,06

Tabela 1. Tabela de frequências sonoras com 7 oitavas

```
% criacao de um denominador
den = [1 2.6131*wc 3.4142*wc^2 2.6131*wc^3 wc^4];
Hs = tf(num, den);    % transformacao (analogico para digital)

% discretizacao de Tustin (IIR)

filename = 'big-ben-strikes-12-good-quality-sound.wav';

% obtem arquivo e freq. do audio
[som, freq] = audioread (filename);

Ts = 1/freq;                    % definicao do periodo

IIR = c2d(Hs, Ts, 'tustin');    % aproximacao de Tustin
[IIRnum, IIRden] = tfdata(IIR, 'v');
```

```
41
42 % Aplicacao do filtro digital
43
44 somlowpass = filter(IIRnum, IIRden, som); % aplica o filtro
45
46 % gera o audio filtrado
47 audiowrite('result.wav', somlowpass, freq);
48
49 filter_signal = audioread('impalalow.wav');
50
51 f_s = 11025;                    % frequencia de amostragem
52
53 [wv,f_s]=audioread('big-ben-strikes-12-good-quality-sound.wav');
54
55 sig1 = audioread('big-ben-strikes-12-good-quality-sound.wav');
56
57 sig2 = audioread('result.wav');
58
59 %sound(wv,fs);
60
61 t=0:1/f_s:(length(wv)-1)/f_s;
62
63 plot(t,wv);                     % plot do sinal original (time)
64
65 my_fft(sig1, f_s);              % plot do sinal original (freq)
66
67 my_fft(sig2, f_s);              % plot do sinal filtrado (freq)
```

Foi escolhido um áudio das badaladas dos sinos do Big Ben de Londres. As figuras 14, 15 e 16 nos mostram os sinais nos mais diversos domínios [Poularikas and Ramadan 2006]. Para os gráficos no domínio da frequência foram utilizados a função my_fft para a plotagem dos coeficientes da Série de Fourier. O algoritmo acima foi bastante eficiente a nível de processamento, uma vez que foi utilizado o Método de Tustin para discretização do sinal [Mitra and Kuo 2006].

8. Filtragem por Janelamento de Hamming

A intuição nos leva a crer que para um filtro ideal uma janela retangular se adequaria muito bem as especificações representadas [Brown et al. 1992]. Podemos tomar um pulso retangular e em seguida aplicar uma Transformada de Fourier deste pulso. A função que será gerada possui um comportamento muito especial quando lidamos com filtros [Astola et al. 1990]. O sinal retangular é mostrado na figura 17 e a transformada de Fourier deste pulso é mostrado na figura 18. O código abaixo [Araújo 2018b] exemplifica o que foi feito para a plotagem dos dois gráficos:

```
1 fs = 500;
2 T = 0.2;
3
4 t = -0.5:1/fs:0.5;
5
```

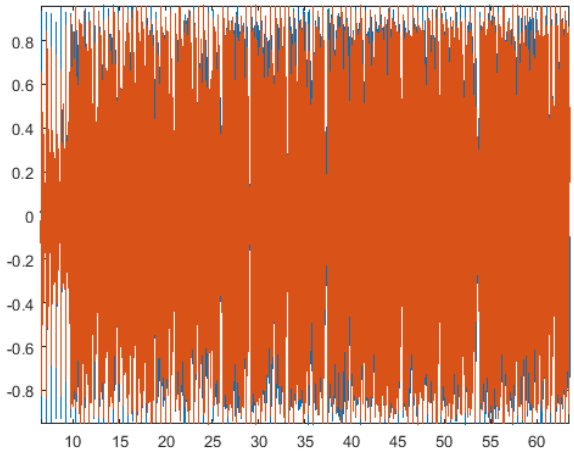

Figura 14. Sinal original no domínio do tempo

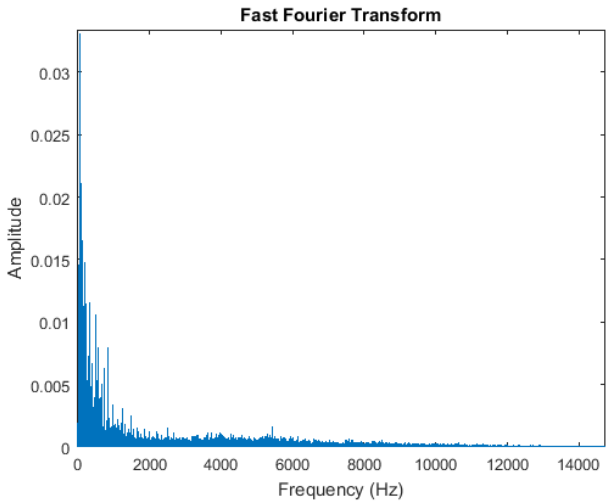

Figura 15. Sinal original no domínio da frequência

```
x=rectpuls(t,T);

plot(t,x,'k');
axis([-0.5 0.5 0 3]);
title('Rectangular Pulse');
xlabel('Times(s)');
ylabel('Amplitude');

L=length(x);
NFFT = 1024;

%FFT with FFTshift for both negative & positive frequencies
X = fftshift(fft(x,NFFT));
f = fs*(-NFFT/2:NFFT/2-1)/NFFT; %Frequency Vector

```

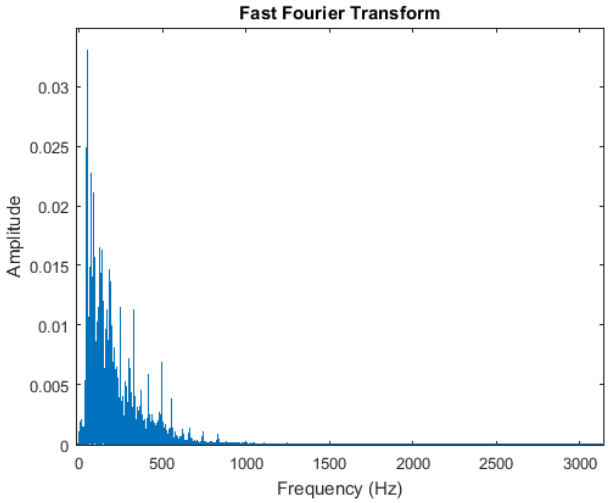

Figura 16. Sinal filtrado no domínio da frequência

```
figure;
plot(f,abs(X)/(L),'r');
title('Magnitude of FFT');
xlabel('Frequency (Hz)')
ylabel('Magnitude |X(f)|');
```

Se tomarmos a DTFT inversa de um pulso retangular no domínio da frequência de um filtro ideal, então teremos a resposta ao impulso correspondente **sinc**. A função **sinc** é definida de $-\infty$ a $+\infty$ [Wanhammar and Johansson 2011]. Desta forma, para representá-la em um computador, temos que truncá-la. O problema é que isto leva a ondulações na banda passante e na banda de bloqueio, como visto na figura 19. Para diminuir este problema, basta levarmos em consideração a multiplicação do sinal por uma janela suave, uma vez que uma janela retangular causaria mais ondas (ripples) na janela de filtragem [Acharya et al. 2014]. Existem diversos tipos de janelas para amenizar os efeitos de ripple, porém escolhemos pela janela de Hamming, como mostrado na figura 20. O código abaixo [Araújo 2018a] demonstra uma convolução dos sinais para a atenuação das ondas na faixa de passagem e rejeição com a função **fvtool**. O resultado é mostrado na figura 21.

```
b = 0.4*sinc(0.4*(-25:25));
b = b.*hamming(51)';
fvtool(b,1)
```

8.1. Criação da Função my_low_pass_ideal

```
function hd = my_lowpass_ideal(wc,M)

alpha = (M-1)/2;
n = [0:M-1];
m = n - alpha + eps;
hd = sin(wc*m)./(pi*m);

end
```

A função acima retorna a função **sinc** e recebe de entrada a frequência de corte do filtro e o comprimento do mesmo que já se encontra tabelado [Brown et al. 1992]. Essa função será multiplicada com a janela de Hamming para a preservação dos lóbulos principais e rejeição dos lóbulos secundários da **sinc**.

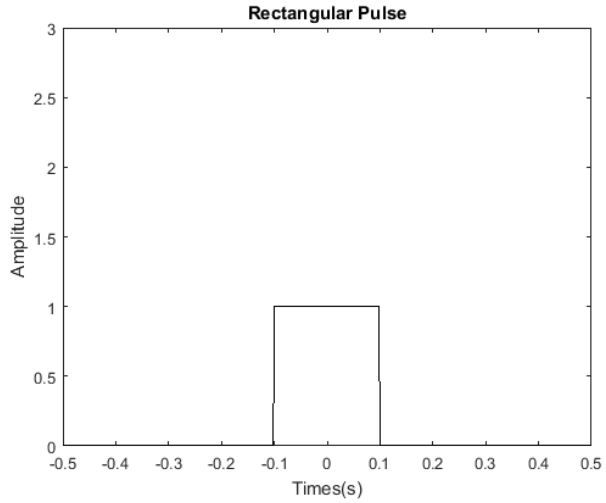

Figura 17. Pulso retangular no domínio tempo

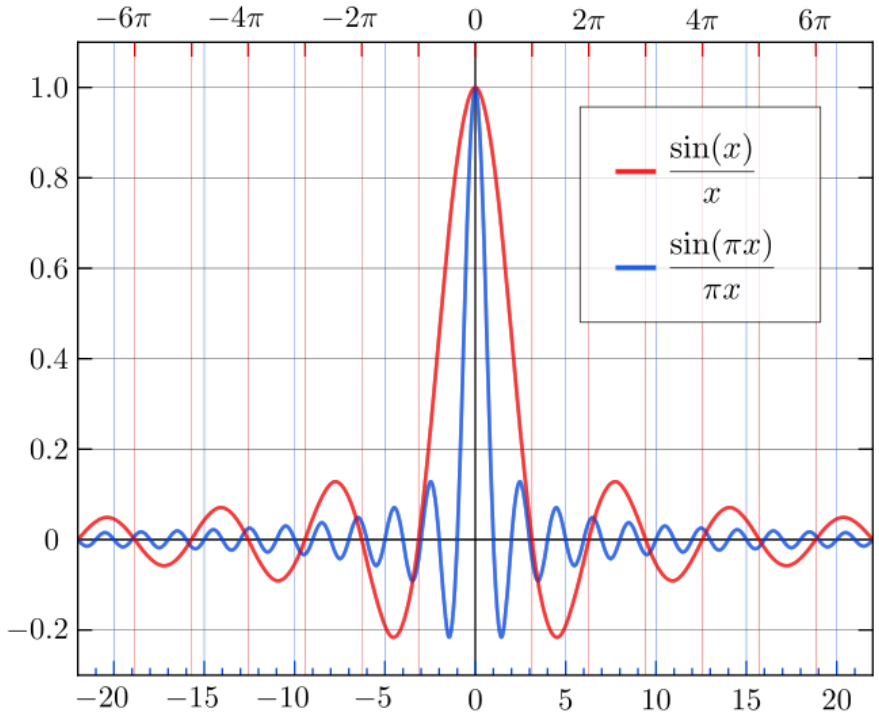

Figura 18. Função sinc

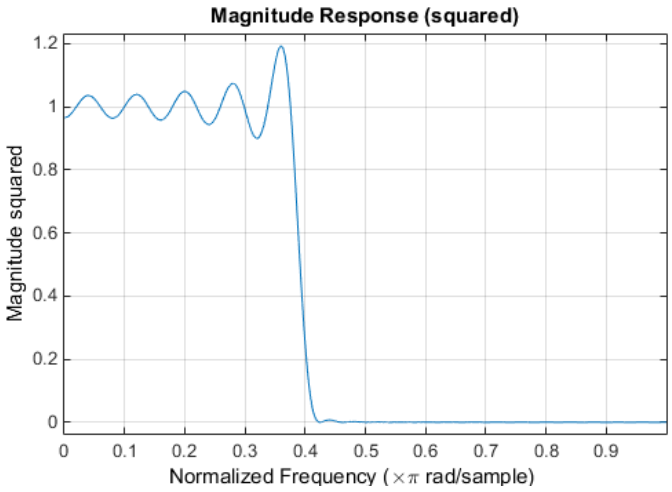

Figura 19. Resposta em frequência da sinc truncada

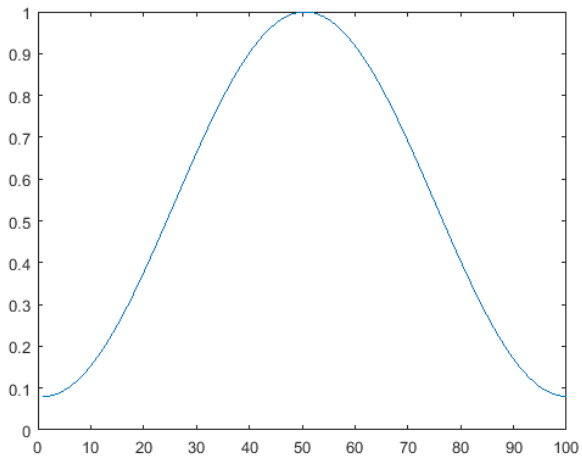

Figura 20. Janela de Hamming com frequência de amostragem 100

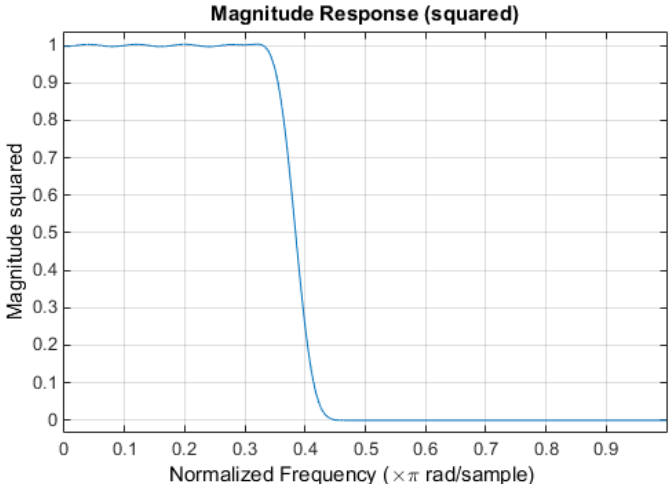

Figura 21. Redução de ruídos usando a Janela de Hamming

9. Especificações do Filtro por Janelamento

(I) Frequência de passagem fp = 1000 Hz
(II) Frequência de rejeição fs = 1500 Hz
(III) Frequência de transição $f7$ = 1500 - 1000 = 500 Hz
(IV) Frequência de corte $(1500 + 1000)/2$ = 1250 Hz
(V) f_amostragem $>=$ 2*f_sinal (Teorema de Nyquist)

O código abaixo [Araújo 2018a] exemplifica o que foi feito. No final foi plotado no domínio da frequência usando a função **my_fft** os gráficos do sinal original e do gráfico filtrado como mostram as figuras 22 e 23.

```
% Filtro passa-baixas FIR
fa = 20000;

f_samp = fa;

[s1, fa] = audioread ('Voice 5.wav');

fsamp = fa;

fp = 1000; % frequencia de passagem
fs = 2000; % frequencia de corte

% normalizacao das frequencias
wp = (fp/(fsamp/2))*pi;
ws = (fs/(fsamp/2))*pi;

wt = ws - wp;     %frequencia de transicao

M = ceil((6.6*pi/wt)) + 1;

wc = (ws + wp)/2; %frequencia de corte intermediaria

%funcao sinc para passa baixas ideal
hd = my_lowpass_ideal(wc,M);

w_hamm = hamming(M)';  %calcula a janela de hamming

h = hd.*w_hamm; %faz a multiplicacao entre os vetores

s1_filtrado = conv(h,s1(:,1)); %convolucao entre os sinais

% plota o som original no dominio tempo

[wave,fa]=audioread('Voice 5.wav');

t=0:1/fa:(length(wave)-1)/fa;

my_fft(s1,fsamp);

```

```
40 my_fft(s1_filtrado,fsamp);
```

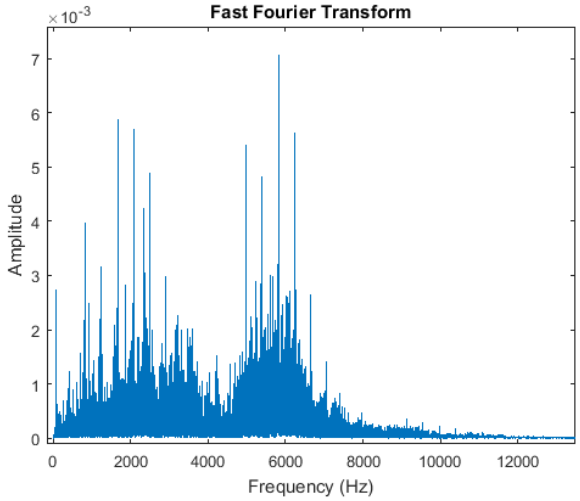

Figura 22. Sinal original no domínio da frequência

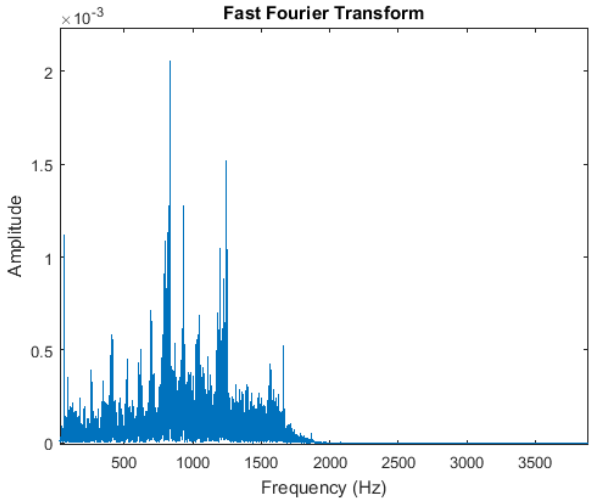

Figura 23. Sinal filtrado no domínio da frequência

10. Conclusões

O uso de ferramentas matemáticas como o Matlab e o Octave no ensino de Sinais e Sistemas e Eletrônica podem agregar muito ao conhecimento de quem esteja estudando os conceitos de filtragem na frequência. A construção de algoritmos que utilizam especificações de filtros são fáceis de serem construídos e implementados, podendo dessa forma, incentivar a criação de algoritmos mais complexos para sistemas mais específicos como em sistemas embarcados ou o uso de microcontroladores para filtragem de sinais digitais. Tais ferramentas ainda ajudam no desenvolvimento de conceitos mais complexos como Transformadas de Fourier e Laplace para a viabilização das técnicas de filtragem na frequência. Além disso, ao longo do trabalho foram tratados de diversas aplicações das séries e transformadas de Fourier na criação de funções, análise espectral de filtros,

identificação de notas musicais, implementação de afinadores de instrumentos e plotagem de gráficos em dois tipos de filtros simples FIR e IIR.

Referências

Acharya, A., Das, S., Pan, I., and Das, S. (2014). Extending the concept of analog butterworth filter for fractional order systems. *Signal processing*, 94:409–420.

Araújo, M. S. (2018a). Analisador de espectro. `https://github.com/mattsousaa/Spectrum_Analyzer`.

Araújo, M. S. (2018b). Filtros de butterworth e janelas de hamming. `https://github.com/mattsousaa/ButterworthFilter_HammingWindow`.

Araújo, M. S. (2018c). Guitar tuner. `https://github.com/mattsousaa/Guitar_Tuner_Digital`.

Araújo, M. S. (2018d). Reconhecimento de frequências sonoras. `https://github.com/mattsousaa/Recognition_SoundFrequency`.

Astola, J., Haavisto, P., and Neuvo, Y. (1990). Vector median filters. *Proceedings of the IEEE*, 78(4):678–689.

Bose, N. K. (1985). *Digital filters: theory and applications*. North-Holland.

Brandenstein, H. and Unbehauen, R. (1998). Least-squares approximation of fir by iir digital filters. *IEEE Transactions on Signal Processing*, 46(1):21–30.

Brown, R. G., Hwang, P. Y., et al. (1992). *Introduction to random signals and applied Kalman filtering*, volume 3. Wiley New York.

Haykin, S. (1985). Array signal processing. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1985, 493 p. For individual items see A85-43961 to A85-43963.*

Jackson, L. B. (2013). *Digital Filters and Signal Processing: With MATLAB® Exercises*. Springer Science & Business Media.

Litwin, L. (2000). Fir and iir digital filters. *IEEE potentials*, 19(4):28–31.

Lutovac, M. D., Tošić, D. V., and Evans, B. L. (2001). *Filter design for signal processing using MATLAB and Mathematica*. Miroslav Lutovac.

Mertins, A. and Mertins, D. A. (1999). *Signal analysis: wavelets, filter banks, time-frequency transforms and applications*. John Wiley & Sons, Inc.

Mettke, M., Medley, M. J., Saulnier, G. J., and Das, P. (1994). Wavelet transform excision using iir filters in spread spectrum communication systems. In *Global Telecommunications Conference, 1994. GLOBECOM'94. Communications: The Global Bridge., IEEE*, volume 3, pages 1627–1631. IEEE.

Mitra, S. K. and Kuo, Y. (2006). *Digital signal processing: a computer-based approach*, volume 2. McGraw-Hill New York.

Podder, P., Hasan, M. M., Islam, M. R., and Sayeed, M. (2014). Design and implementation of butterworth, chebyshev-i and elliptic filter for speech signal analysis. *International Journal of Computer Applications*, 98(7).

Poularikas, A. D. and Ramadan, Z. M. (2006). *Adaptive filtering primer with MATLAB*. CRC Press.

Rabiner, L. R. and Gold, B. (1975). Theory and application of digital signal processing. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.*

Ristic, B., Vo, B.-T., Vo, B.-N., and Farina, A. (2013). A tutorial on bernoulli filters: theory, implementation and applications. *IEEE Transactions on Signal Processing*, 61(13):3406–3430.

Shively, R. (1975). On multistage finite impulse response (fir) filters with decimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(4):353–357.

Stearns, S. D. and Hush, D. R. (2016). *Digital signal processing with examples in MATLAB®*. CRC Press.

Storn, R. (1996). Differential evolution design of an iir-filter. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 268–273. IEEE.

Sühling, M., Arigovindan, M., Hunziker, P., and Unser, M. (2004). Multiresolution moment filters: Theory and applications. *IEEE Transactions on Image Processing*, 13(LIB-ARTICLE-2004-013):484–495.

Wanhammar, L. and Johansson, H. (2011). *Digital filters using matlab*. Department of Electrical Engineering, Linköping University.

Welch, T. B., Wright, C. H., and Morrow, M. G. (2016). *Real-time digital signal processing from MATLAB to C with the TMS320C6x DSPs*. CRC Press.