Projeto de Filtros IIR

Processamento de Sinais Digitais

Arthur Cadore Matuella Barcella

12 de Agosto de 2024

Engenharia de Telecomunicações - IFSC-SJ

Sumário

1. Introdução

O objetivo deste relatório é apresentar o desenvolvimento do projeto de filtros IIR para o processamento de sinais digitais. O projeto foi desenvolvido utilizando o MATLAB e o Simulink, e tem como objetivo a filtragem de sinais digitais para a separação de componentes cossenoidais em um sinal composto.

2. Questão 1

Usando a transformação bilinear, projete um filtro passa-baixas Butterworth que atenda as seguintes especificações:

$$0.9 \leq |H(e^{\{j\omega\}})| \leq 1 \tag{1}$$

$$0 \leq \omega \leq 0.2\pi \tag{2}$$

$$|H(e^{\{j\omega\}})| \leq 0.2 \tag{3}$$

$$0.3\pi \leq \omega \leq \pi \tag{4}$$

Considere também que:

- Ts (tempo de amostragem) = 2
- Faça o mesmo projeto usando o MATLAB ou simulink. Plote a resposta em frequência

2.1. Projeto do Filtro IIR no Simulink:

2.1.1. Pré-Distorção do Sinal:

O primeiro passo no desenvolvimento do projeto foi a montagem do filtro IIR no Simulink. Utilizamos a ferramenta para alterar os parâmetros do filtro procurando pelos valores que atendessem as especificações de banda de passagem e de rejeição.

Seguindo a formula para transformação bilinear para transformar a frequência analógica em frequência digital:

$$\omega = \frac{2}{\mathrm{Ts}} \tan\left(\frac{\omega}{2}\right) \tag{5}$$

Desta forma, temos que o script matlab corresponte é:

```
wp = 0.2*pi;
wr = 0.3*pi;
ts = 2;

omega_ap = (2/ts)*tan(wp/2)
omega_ar = (2/ts)*tan(wr/2)
```

2.1.2. Normalização do Filtro:

A normalização do filtro é realizada para que o filtro projetado atenda as especificações de banda de passagem e de rejeição.

Para o processo de normalização, seguimos a tabela apresentada em aula que contém a seguinte transformada:

$$\Omega'_p = \frac{1}{a} \tag{6}$$

$$\Omega'_r = \frac{1}{a}\left(\frac{\Omega_r}{\Omega_p}\right) \tag{7}$$

Desta forma, temos que o script matlab corresponte é:

```
a = 1;
omega_p_linha = 1/a
omega_r_linha = omega_p_linha * (omega_ar/omega_ap)
```

2.1.3. Calculo das atenuações:

Para o calculo das atenuações, utilizamos a formula apresentada em aula para obter o valor de ganho em dB para a banda de passagem e de rejeição:

- Para a banda de passagem:

$$G_p = 20\log_{10}(1-\sigma_p) \tag{8}$$

- Para a banda de rejeição:

$$G_r = 20\log_{10}(\sigma_r) \tag{9}$$

Desta forma, temos que o script matlab corresponte é:

```
sigma_p = 0.9;
sigma_r = 0.2;

atenuacao_p = -1 * (20*log10(sigma_p))
atenuacao_r = -1 * (20*log10(sigma_r))
```

2.1.4. Calculo dos parâmetros do filtro:

Em seguida podemos calcular os demais parâmetros do filtro, que são dados através das seguintes espressões:

- Calculo do fator de normalização:

$$\epsilon = \sqrt{10^{0,1,A_p} - 1} \tag{10}$$

- Calculo da ordem do filtro:

$$n \geq \frac{\log_{10}\left(\frac{10^{0,1,A_{r-1}}}{\epsilon^2}\right)}{2\log_{10}\Omega'_r} \tag{11}$$

Desta forma, aplicando na sintaxe do matlab, temos que o script corresponte é:

```
eps = sqrt((10^(0.1*atenuacao_p))-1)

numerador = log10((10^(0.1*atenuacao_r)-1) / eps^2);
denominador = 2*log10(omega_r_linha);

n = ceil(numerador/denominador)
```

2.1.5. Calculo das Raízes do Filtro:

Em seguida, podemos calcular as raízes do filtro utilizando a função roots do matlab, para isso, utilizamos a seguinte formula:

$$1 + \epsilon^2\left(-s'^2\right)^n = 0 \tag{12}$$

Desta, forma, como possuimos um filtro de ordem 12, temos que o script matlab corresponte é:

```
roots([eps^2 0 0 0 0 0 0 0 0 0 0 0 1])

% Raizes encontradas:
% -1.0900 + 0.2921i
% -1.0900 - 0.2921i
% -0.7979 + 0.7979i
% -0.7979 - 0.7979i
% -0.2921 + 1.0900i
% -0.2921 - 1.0900i
```

2.1.6. Calculo dos Coeficientes do Filtro:

Em seguida, podemos calcular os coeficientes do filtro utilizando a função poly do matlab aplicando as raizes encontradas no calculo anterior.

```
poly([
-1.0900 + 0.2921i
-1.0900 - 0.2921i
-0.7979 + 0.7979i
-0.7979 - 0.7979i
-0.2921 + 1.0900i
-0.2921 - 1.0900i
])
```

Com a aplicação da função poly, obtemos os coeficientes do filtro que são:

```
h'(s') = 2.0648 / 1s'6 + 4.3600s'5 + 9.5048s'4 + 13.1362s'3 + 12.1033s'2 +
7.0697s + 2.0648
```

Em seguida, podemos aplicar a desnormalização do filtro para obter os coeficientes do filtro no domínio do tempo, para isso, utilizamos a função bilinear do matlab, a partir da seguinte formula:

$$s' = \left(\frac{1}{a}\right)\left(\frac{s}{\Omega'_p}\right) \tag{13}$$

Desta forma, temos que o resultado da espressão é o seguinte:

```
%h(s) = 0.0024 / 1s^6 1.4166s^5 + 1.0034s^4 + 0.4506s^3 + 0.1349s^2+ 0.0256s
+ 0.0024
```

E desta forma, os valores dos coeficiente podem ser separados em numerador e denominador, como apresentado abaixo, sendo o vetor 'a' o numerador e o vetor 'b' o denominador:

```
a = [1 1.4166 1.0034 0.4506 0.1349 0.0256 0.0024]
b = [0.0024]
```

2.1.7. Aplicando a função bilinear:

Por fim, podemos aplicar a função bilinear do matlab para obter os coeficientes do filtro no domínio do tempo, para isso, utilizamos a função bilinear do matlab, a partir do script abaixo:

```
[numerador, denominador] = bilinear(b,a,fs)
```

```
Numerator:
  0.000602340190409411798885581924473101025;
  0.009035102856141176766446854173864267068;
  0.012046803808188236845078122883023752365;
  0.009035102856141176766446854173864267068;
  0.003614041142456470793314915468386061548;
  0.000602340190409411798885819244731010258;

Denominator:
  1;
 -3.293990202908562814343440550146624445915;
  4.898232666546461722134608862688764929771;
 -4.084996050238189013725786935538053512573;
  1.992931744078747957615860286750830709934;
 -0.535091724334459173384459518274525180459;
  0.061463339042203794793106652605274575762;
```

2.2. Script Matlab:

```
clear all; close all; clc;
 %---------------------------------------pré distorçao ------------------
 wp = 0.2*pi;
 wr = 0.3*pi;
 ts = 2;

 omega_ap = (2/ts)*tan(wp/2)
 omega_ar = (2/ts)*tan(wr/2)

 a = 1;
 omega_p_linha = 1/a

 omega_r_linha = omega_p_linha * (omega_ar/omega_ap)

 sigma_p = 0.9;
 sigma_r = 0.2;

 atenuacao_p = -1 * (20*log10(sigma_p))
 atenuacao_r = -1 * (20*log10(sigma_r))

 eps = sqrt((10^(0.1*atenuacao_p))-1)

 numerador = log10((10^(0.1*atenuacao_r)-1) / eps^2);
 denominador = 2*log10(omega_r_linha);


 n = ceil(numerador/denominador)


 % utilizando a expressâo -> 1 + e^2 (-s^2)^n = 0)

 roots([eps^2 0 0 0 0 0 0 0 0 0 0 1])

 poly([-1.0900 + 0.2921i
 -1.0900 - 0.2921i
 -0.7979 + 0.7979i
 -0.7979 - 0.7979i
 -0.2921 + 1.0900i
 -0.2921 - 1.0900i])


 %h'(s') = 2.0648 / 1s'6 + 4.3600s'5 + 9.5048s'4 + 13.1362s'3 + 12.1033s'2
+ 7.0697s +
2.0648

%para desnormalizar; utilizamos a expressão s = (1/a)*(s/omega_ap) e
%substituimos os valores; achando o mínimo comun obtivemos:

%h(s) = 0.0024 / 1s^6 1.4166s^5 + 1.0034s^4 + 0.4506s^3 + 0.1349s^2+ 0.0256s
+ 0.0024

b = [0.0024]
a = [1 1.4166 1.0034 0.4506 0.1349 0.0256 0.0024]
fs = 1/ts;

[numerador, denominador] = bilinear(b,a,fs)

freqz(numerador, denominador)
```

2.3. Resultados obtidos:

Figura 1: Elaborada pelo Autor

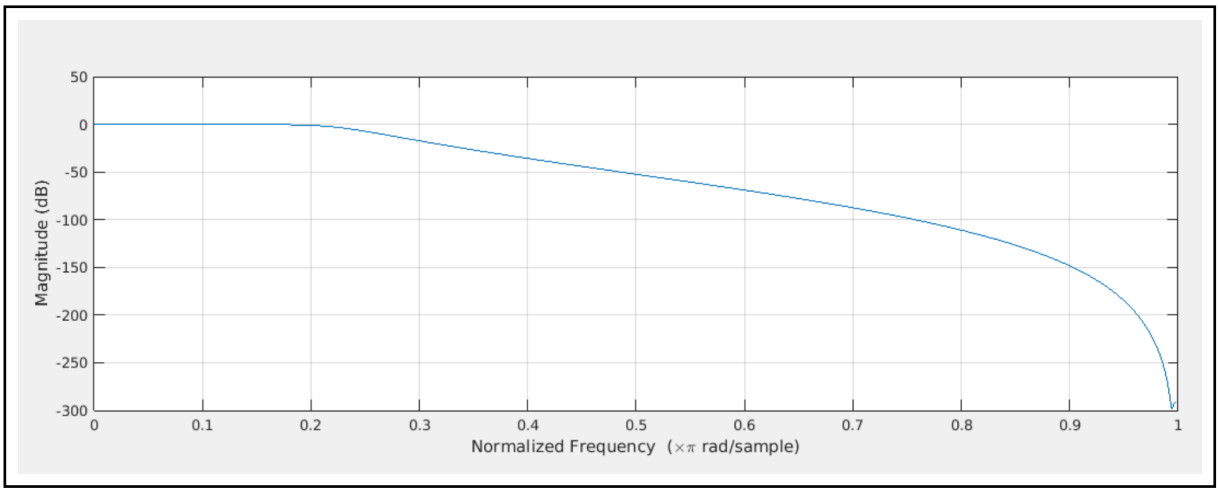
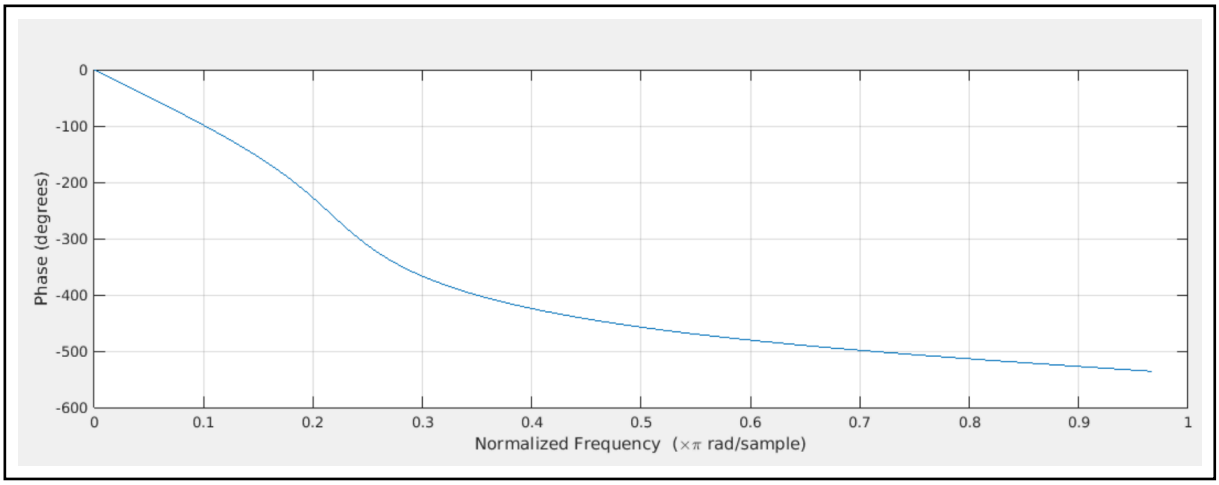


Figura 2: Elaborada pelo Autor



3. Questão 2

Crie, usando MATLAB, um sinal de entrada composto de três componentes senoidais, nas frequências 770Hz, 852Hz e 941Hz, com Ωs = 8 kHz. Projete, usando o simulink ou o MATLAB, um filtro IIR para isolar cada componente. Documente as especificações utilizadas. Faça comentários.

3.1. Montagem do Filtro IIR no Simulink:

O primeiro passo no desenvolvimento da questão foi a montagem do filtro IIR no Simulink seguindo as especificações solicitadas pela questão. Utilizamos a ferramenta para alterar o parâmetros do filtro procurando pelos valores que atendessem as especificações de banda de passagem e de rejeição.

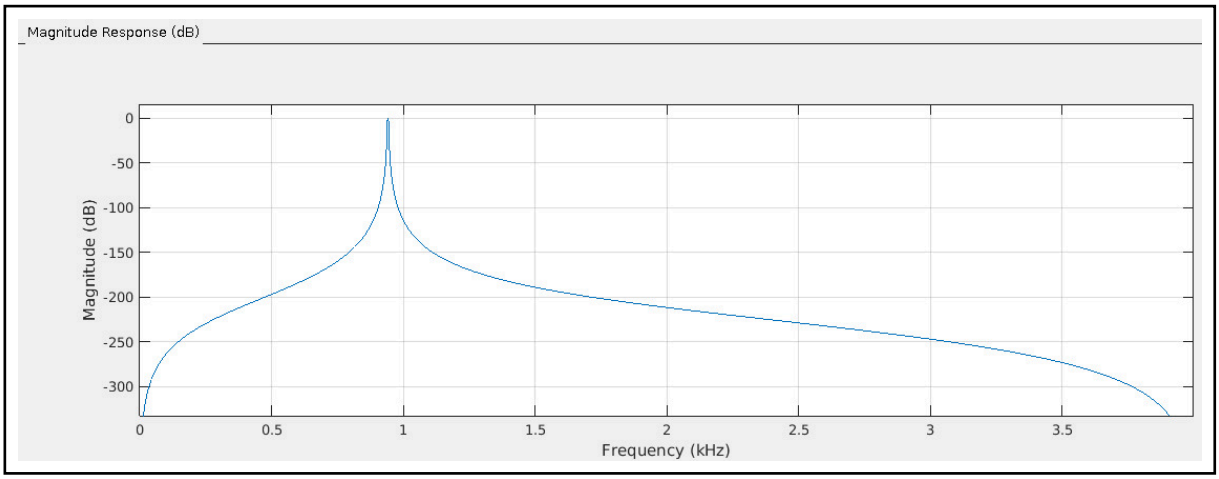
Os valores apresentados abaixo foram obtidos para o filtro de 941Hz, porem, foram realizados mais duas baterias de testes exatamente como as apresentadas abaixo, para as frequências de 770Hz e 852Hz.

3.1.1. Resposta em Frequência (Magnitude):

Inicialmente utilizamos a ferramenta de plot da resposta em frequência do filtro para analisar a magnitude do filtro. A figura abaixo apresenta a resposta em frequência do filtro para a frequência de 941Hz.

O filtro que estamos procurando precisa possuir uma alta atenuação na banda de rejeição e uma queda abrupta na banda de transição para a banda de rejeição. O filtro abaixo apresenta uma atenuação de aproximadamente 80dB em toda a banda de rejeição.

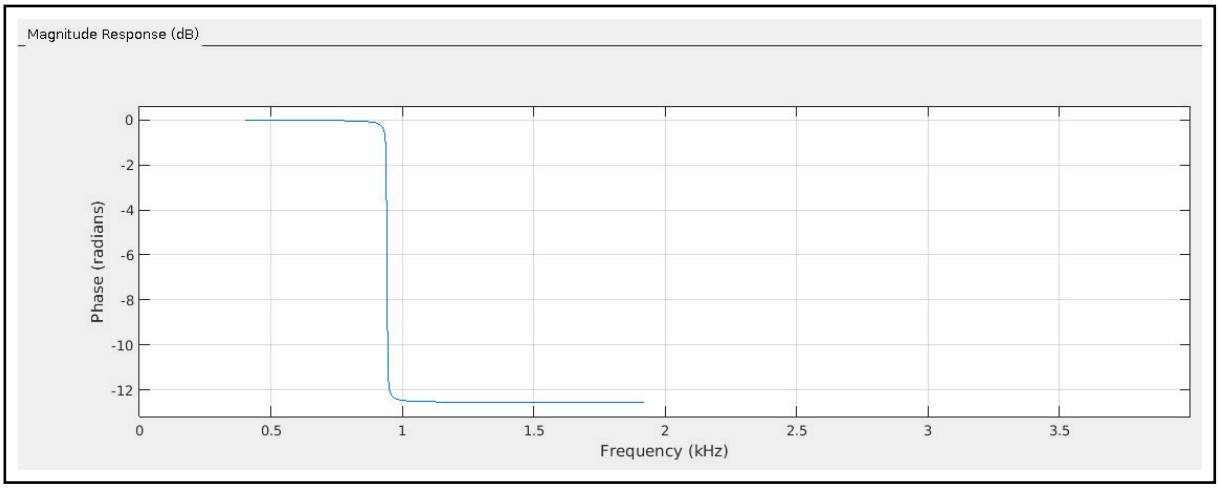
Figura 3: Elaborada pelo Autor



3.1.2. Resposta em Frequência (Fase):

A fase do filtro é apresentada na figura abaixo. A fase do filtro é linear e apresenta apenas uma variação de 11° na banda de passagem.

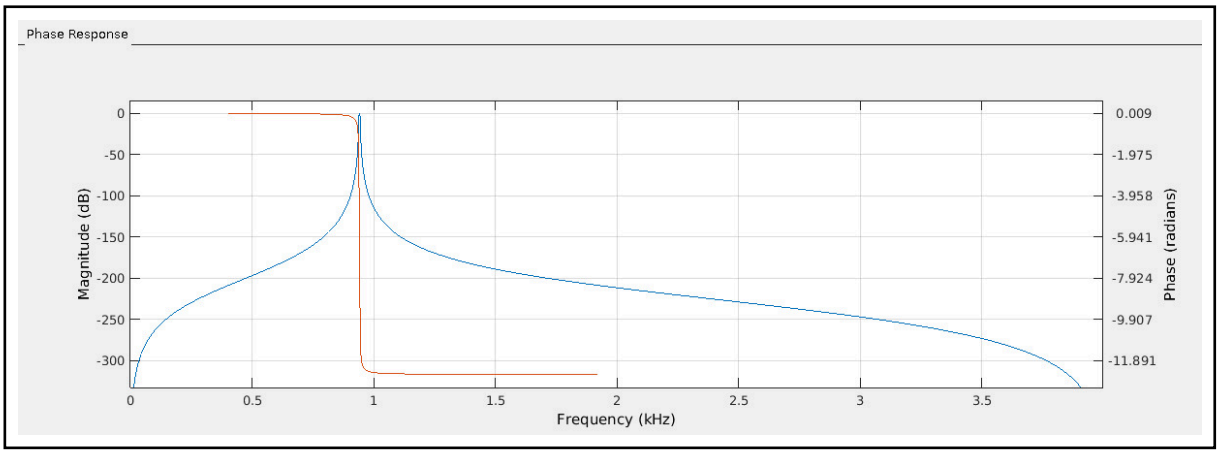
Figura 4: Elaborada pelo Autor



3.1.3. Magnitude e Fase do Filtro:

Abaixo podemos ver a magnitude e a fase do filtro sendo apresentadas juntas, note que a fase é linear dentro da banda de passagem, e a magnitude apresenta uma queda abrupta na banda de transição:

Figura 5: Elaborada pelo Autor

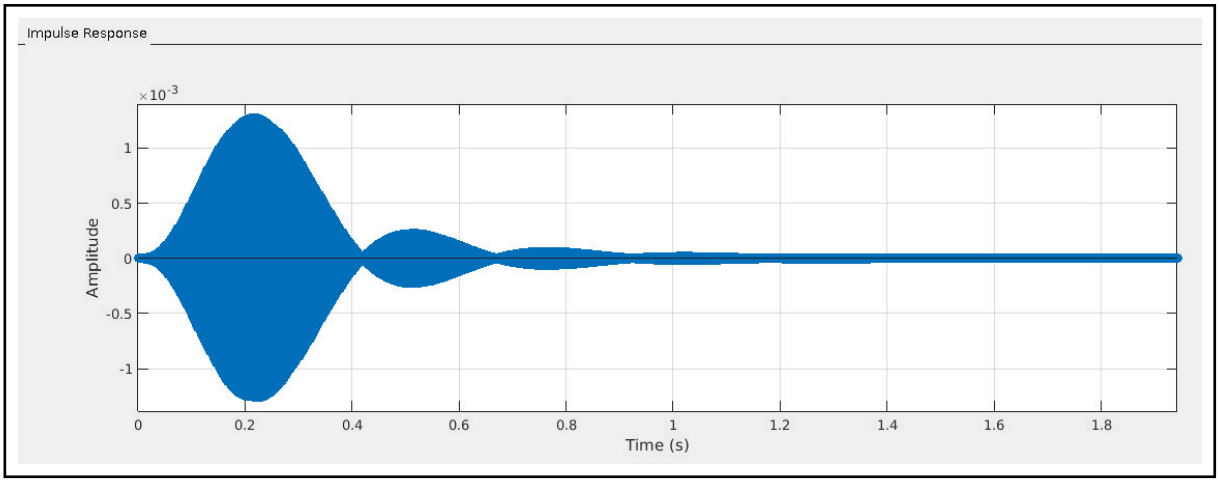


3.1.4. Resposta ao impulso:

Abaixo está apresentada a resposta ao impulso do filtro. Note que a resposta ao impulso é finita, oque é um requisito para o desenvolvimento deste projeto por se tratar de um filtro para sinais digitais.

Note que a resposta ao impulso apresenta um pico inicial e em seguida cai para zero, oque é esperado para um filtro passa-faixa.

Figura 6: Elaborada pelo Autor

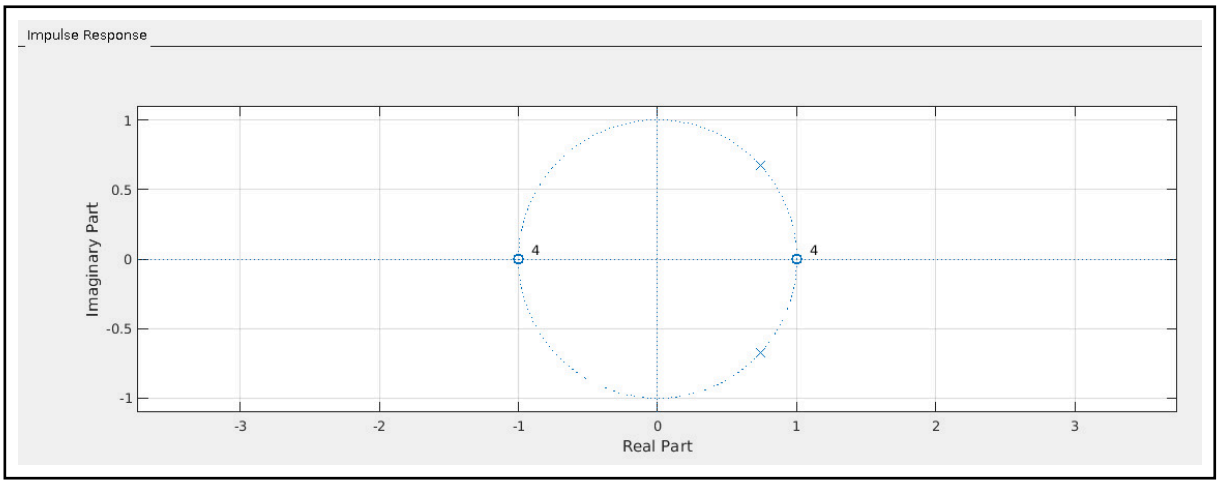


3.1.5. Polos e Zeros do Filtro:

Os polos e zeros do filtro são apresentados na figura abaixo. Note que o filtro possui 4 polos e 4 zeros, e que os polos estão localizados na região de interesse, ou seja, todos os polos estão dentro do círculo unitário.

Desta forma, o filtro é estável e possui uma resposta ao impulso finita oque é um requisito para o desenvolvimento deste projeto por se tratar de um filtro para sinais digitais.

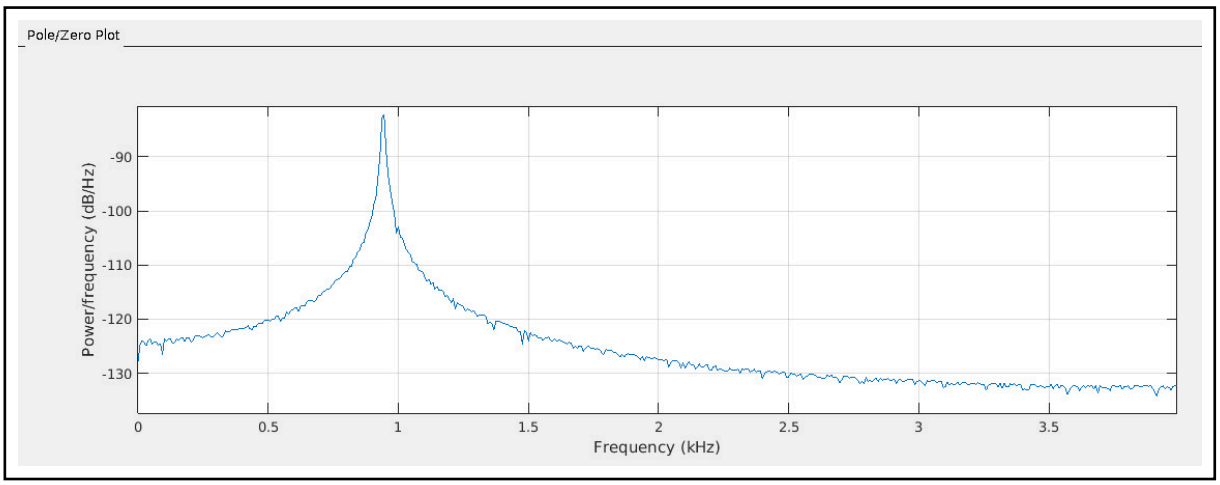
Figura 7: Elaborada pelo Autor



3.1.6. Espalhamento de potência do filtro:

Abaixo também está apresentado o espalhamento de potência do filtro. Note que o filtro apresenta uma atenuação de aproximadamente 80dB na banda de rejeição, também é possivel verificar que grande parte da potência do sinal está concentrada na banda de passagem como esperado para as caracteristicas do filtro.

Figura 8: Elaborada pelo Autor



Com base nestes valores, obtivemos os seguintes parâmetros para o filtro de 942Hz:

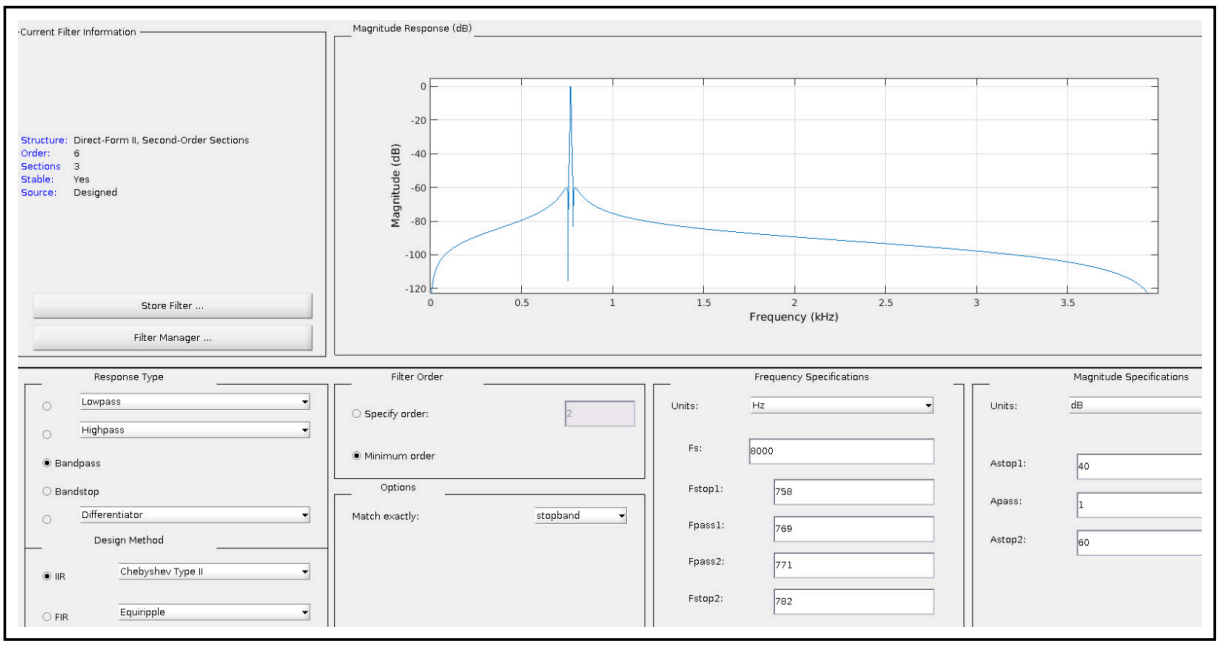
- Ordem do filtro: 8
- Banda de passagem: 941Hz - 943Hz
- Banda de transição Baixa: 930Hz - 941Hz
- Banda de transição Alta: 943Hz - 954Hz
- Frequência de amostagem: 8kHz
- Astop1 = 40dB
- Astop2 = 60dB
- Método de Desing: IIR-Butterworth

Os mesmos parâmetros foram utilizados para os filtros de 1kHz e 1.1kHz.

3.2. Comparativo entre outros Desings:

Neste projeto foi decidido utilizar o méotodo de Butterworth para o desing do filtro, porem, também foram testados outros métodos, como Chebyshev:

Figura 9: Elaborada pelo Autor



Filtro de 941Hz Utilizando Chebyshev

Entretanto, durante os testes com os demais filtros, foi visto que a taxa de atenuação apresentada pelo filtro de Butterworth era superior a dos demais filtros, como é possivel verificar acima para o filtro chebyshev em relação ao filtro Butterworth apresentado abaixo.

Note que na banda de passagem, os filtros se assemelham com atenuação proxima de zero e queda abrupta no inicio da banda de rejeição, porem, o filtro Butterworth apresenta uma atenuação de aproximadamente 80dB na banda de rejeição, enquanto o filtro Chebyshev apresenta uma atenuação de até 60dB e em seguida o filtro se mantem constante neste valor por uma faixa de frequência maior, oque atenua menos as componentes espectrais indesejadas.

Desta forma, o filtro que foi escolhido para o projeto e filtragem das componentes cossenoidais, foi o filtro Butterworth.

3.3. Aplicando o filtro no sinal de entrada:

Uma vez com o filtro projetado e os parâmetros definidos, é possivel gerar um script matlab (octave) para gerar as três componentes de entrada, soma-las de maneira a criar um sinal único, análogo ao que seria transmitido pelo meio de transmissão, e aplicar o filtro projetado “no receptor” para cada uma das componentes, separando-as e verificando a resposta em frequência de cada sinal filtrado.

3.3.1. Script Matlab:

```
clc; close all; clear all;

% Parâmetros do sinal
tmin = 0;
tmax = 2;
Omega_s = 8000;
Fs = Omega_s;
Ts = 1/Fs;
L = (tmax - tmin)/Ts;
t = 0:Ts:tmax-Ts;

% Componentes do sinal de entrada
f1 = 770;
f2 = 852;
f3 = 941;

% Sinal de entrada composto:
s_t = sin(2*pi*f1*t) + sin(2*pi*f2*t) + sin(2*pi*f3*t);

% Realizando a FFT do sinal composto:
S_f = fft(s_t);
S_f = abs(2*S_f/L);
S_f = fftshift(S_f);
freq = Fs*(-(L/2):(L/2)-1)/L;

% Realizando o plot do sinal composto no dominio do tempo:
figure;
subplot(2,1,1);
plot(t,s_t);
ylabel('Amplitude');
xlabel('Tempo (s)');
title('Sinal de entrada');

% Plotando a FFT do sinal composto
subplot(2,1,2);
plot(freq,S_f);
title('Espectro do sinal composto de três componentes senoidais');
xlabel('Frequência (Hz)');
ylabel('Amplitude');
xlim([-1000 1000]);
ylim([-0.1 1.1]);

% ================================================
% Aplicando as componentes de filtragem
% Filtro de 770Hz Butterworth
Num = [
     0.000000000007648974625344080593812088956; 0;
    -0.000000000030595898501376322375248355826; 0;
     0.000000000045893847752064483562872533739; 0;
    -0.000000000030595898501376322375248355826; 0;
     0.000000000007648974625344080593812088956
];

Den = [1;
    -6.573968478516325930627317575272172689438;
    20.197703598146055981032986892387270927429;
   -37.435767531546105146844638511538505554199;
    45.612066895335409810741111868992447853088;
```

```
   -37.354424964373507123127637896686792373657;
    20.110025517173959030969854211434721946716;
    -6.531208646296375874840123287867754697 8;
     0.991336860368820738109718604391673579812
];

% Realizando a filtragem do sinal composto:
s_770hz = filter(Num, Den, s_t);
S_770hz = fft(s_770hz);
S_770hz = abs(2*S_770hz/L);
S_770hz = fftshift(S_770hz);

% Realizando o plot do sinal composto no dominio do tempo:
figure;
subplot(2,1,1);
plot(freq,S_f);
title('Espectro do sinal composto de três componentes senoidais');
xlabel('Frequência (Hz)');
ylabel('Amplitude');
xlim([-1000 1000]);
ylim([-0.1 1.1]);
subplot(2,1,2);
plot(freq, S_770hz);
title('Componente do sinal de 770 Hz filtrado com passa-faixa
Butterworth.');
xlabel('Frequência (Hz)');
ylabel('Amplitude');
xlim([-1000 1000]);
ylim([-0.1 1.1]);

% ================================================
% Aplicando as componentes de filtragem
% Filtro de 852Hz Butterworth
Num = [
     0.000000000007674623107715503777944271234; 0;
    -0.000000000306984924308620151117770849 37; 0;
     0.000000000460477386462930258988398951 91; 0;
    -0.000000000306984924308620151117770849 37; 0;
     0.000000000007674623107715503777944271234
];

Den = [1;
    18.723854036105780807019982603378593921661;
   -34.153268845102040529582154704257845878601;
    41.404068891609767888439819216728210449219;
   -34.078996547913362746839993633329868316 65 ;
    18.642505949254868369280302431434392929077;
    -6.227116450421420701388797169784083962 44;
     0.991329630860964927663303569715935736895
];

% Realizando a filtragem do sinal composto:
s_852hz = filter(Num, Den, s_t);
S_852hz = fft(s_852hz);
S_852hz = abs(2*S_852hz /L);
S_852hz  = fftshift(S_852hz );

% Realizando o plot do sinal composto no dominio do tempo:
```

```
figure;
subplot(2,1,1);
plot(freq,S_f);
title('Espectro do sinal composto de três componentes senoidais');
xlabel('Frequência (Hz)');
ylabel('Amplitude');
xlim([-1000 1000]);
ylim([-0.1 1.1]);
subplot(2,1,2);
plot(freq, S_852hz );
title('Componente do sinal de 852 Hz filtrado com passa-faixa Butterworth.');
xlabel('Frequência (Hz)');
ylabel('Amplitude');
xlim([-1000 1000]);
ylim([-0.1 1.1]);

% ================================================
% Aplicando as componentes de filtragem
% Filtro de 940Hz Butterworth
Num = [0.000000000007698496068837793746781982269; 0;
    -0.000000000030793984275351174987127929078; 0;
     0.000000000046190976413026762480691893617; 0;
    -0.000000000030793984275351174987127929078; 0;
     0.000000000007698496068837793746781982269
];
Den = [1;
     -5.902149339413338857696089689852669835091;
     17.054552786823688137474164250306785106659;
    -30.518120678802585388211809913627803325653;
     36.783803548311759357147820992395281791687;
    -30.451702102949198547321429941803216934204;;
     16.980399695100125256885803537443280220032;
     -5.863697447546361019021787797100841999054;
      0.991322918116968265778155000589322298765
];

% Realizando a filtragem do sinal composto:
s_940hz = filter(Num, Den, s_t);
S_940hz = fft(s_940hz);
S_940hz = abs(2*S_940hz/L);
S_940hz = fftshift(S_940hz);

% Realizando o plot do sinal composto no dominio do tempo:
figure;
subplot(2,1,1);
plot(freq,S_f);
title('Espectro do sinal composto de três componentes senoidais');
xlabel('Frequência (Hz)');
ylabel('Amplitude');
xlim([-1000 1000]);
ylim([-0.1 1.1]);
subplot(2,1,2);
plot(freq, S_940hz);
title('Componente do sinal de 940 Hz filtrado com passa-faixa Butterworth.');
xlabel('Frequência (Hz)');
```

```
ylabel('Amplitude');
xlim([-1000 1000]);
ylim([-0.1 1.1]);
```

3.3.2. Resultados Obtidos:

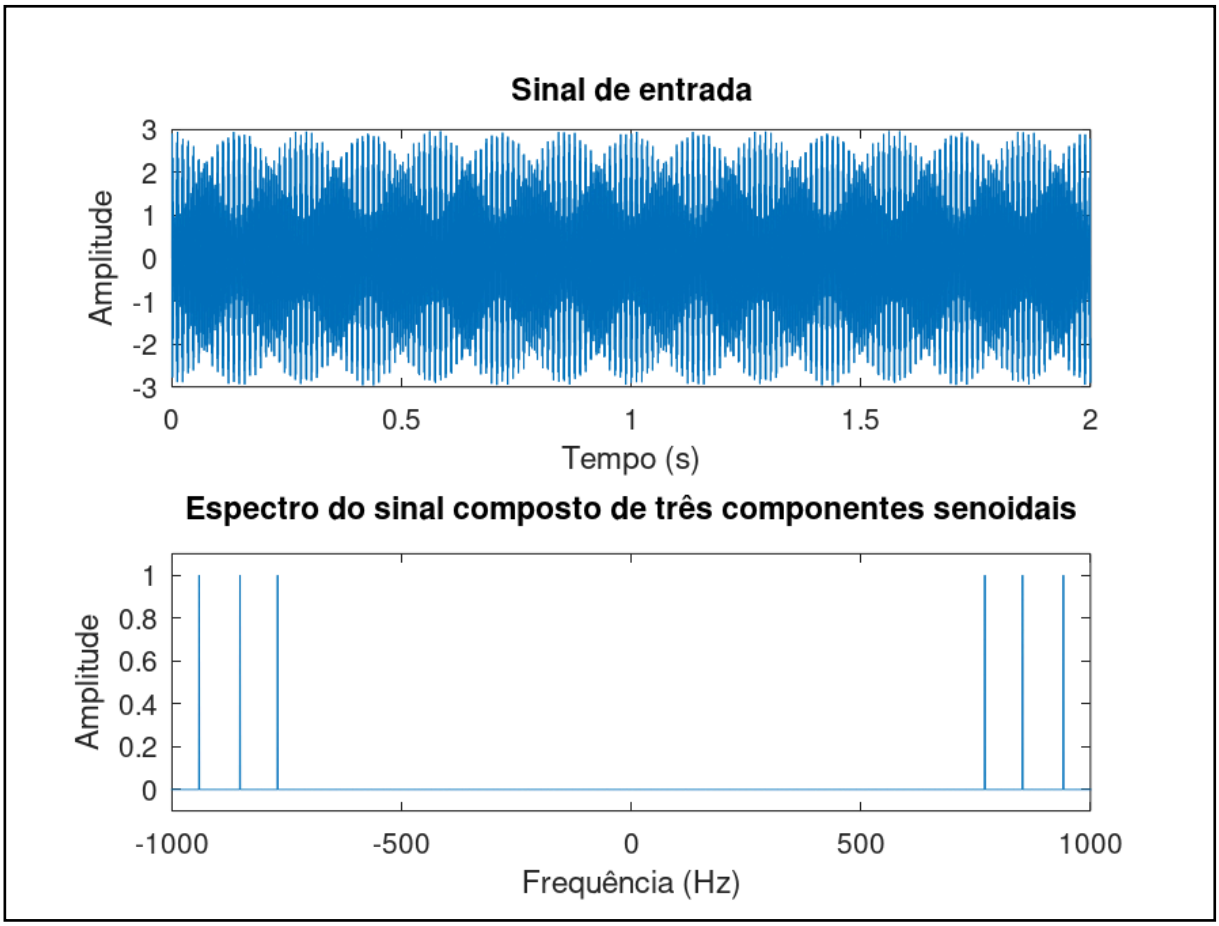
Os resultados obtidos para a filtragem das componentes cossenoidais são apresentados abaixo.

3.3.2.1. Sinal de Entrada:

A partir do script acima, o sinal de entrada foi gerado e apresentado na figura abaixo. Note que o sinal é composto por três componentes senoidais, nas frequências de 770Hz, 852Hz e 941Hz.

As componentes somadas são apresentadas abaixo, tanto no dominio do tempo quanto no dominio da frequência, note que no dominio da frequência aparece as três componentes cossenoidas separadamente permitindo visualizar a soma completa dos sinais.

Figura 10: Elaborada pelo Autor



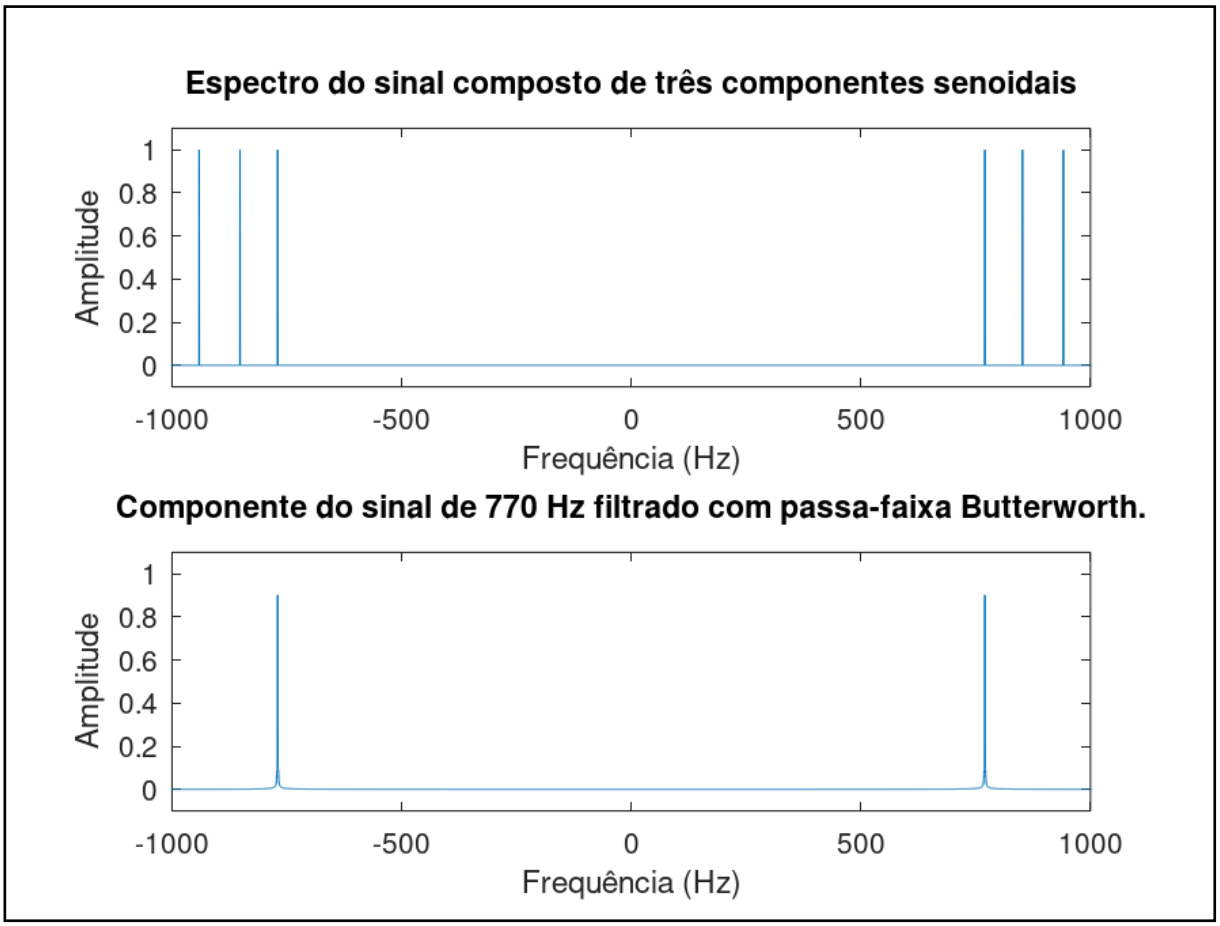
Filtro de 941Hz Utilizando Chebyshev

3.3.2.2. Sinal de 770Hz Filtrado:

Multiplicando o sinal de entrada pelo filtro de 770Hz, obtemos o sinal filtrado para a frequência de 770Hz. Note que o sinal filtrado apresenta apenas a componente de 770Hz, as demais componentes foram atenuadas pelo filtro.

Note que também há uma pequena amplitude associada a lateral do sinal, isso ocorre devido a resposta em frequência do filtro que não é ideal, porem, a atenuação é suficiente para que a componente de 770Hz seja isolada das demais.

Figura 11: Elaborada pelo Autor



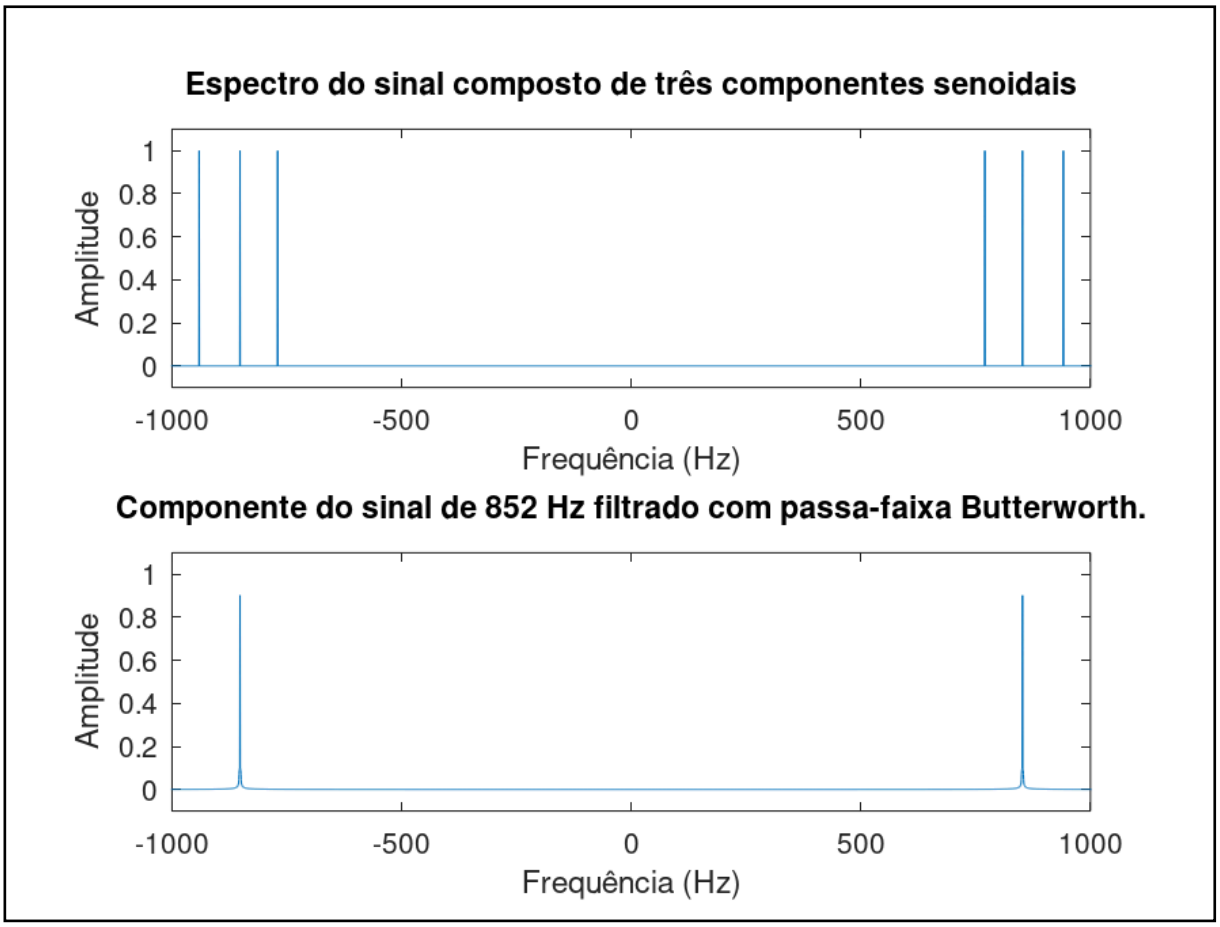
Filtro de 941Hz Utilizando Chebyshev

3.3.2.3. Sinal de 852Hz Filtrado:

Em seguida, podemos ver o sinal filtrado para a frequência de 852Hz. Note que o sinal filtrado apresenta apenas a componente de 852Hz, as demais componentes foram atenuadas pelo filtro.

Note que também há uma pequena amplitude associada a lateral do sinal, isso ocorre devido a resposta em frequência do filtro que não é ideal, porem, a atenuação é suficiente para que a componente de 852Hz seja isolada das demais.

Figura 12: Elaborada pelo Autor



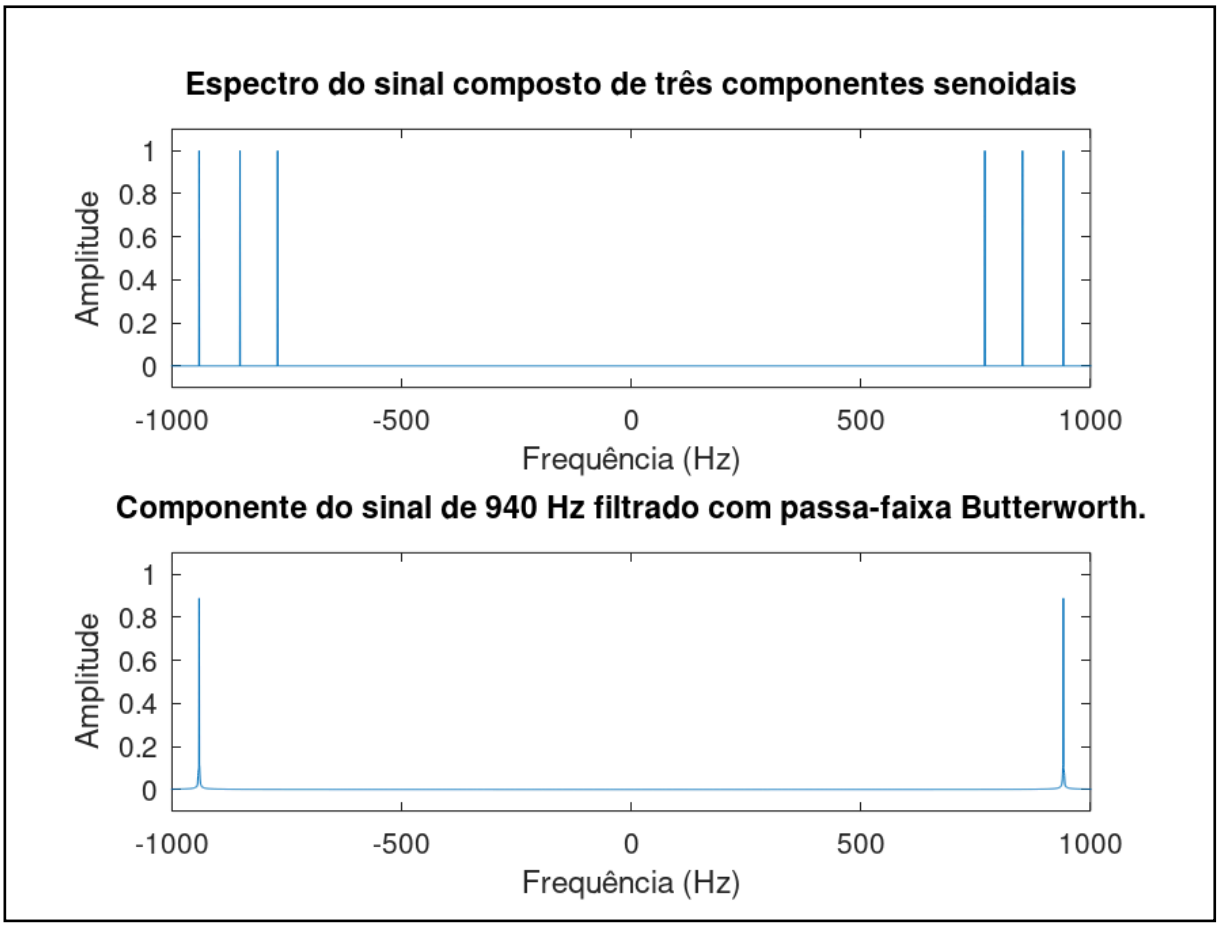
Filtro de 941Hz Utilizando Chebyshev

3.3.2.4. Sinal de 941Hz Filtrado:

Por fim temos o sinal filtrado para a frequência de 941Hz. Note que o sinal filtrado apresenta apenas a componente de 941Hz, as demais componentes foram atenuadas pelo filtro.

Note que também há uma pequena amplitude associada a lateral do sinal, isso ocorre devido a resposta em frequência do filtro que não é ideal, porem, a atenuação é suficiente para que a componente de 941Hz seja isolada das demais.

Figura 13: Elaborada pelo Autor



Filtro de 941Hz Utilizando Chebyshev

4. Conclusão

A partir dos conceitos vistos, desenvolvimento e resultados obtidos, podemos concluir que o projeto de filtros IIR é uma ferramenta poderosa para o processamento de sinais digitais. Através do projeto de filtros IIR é possivel separar componentes cossenoidais de um sinal composto, permitindo a análise de cada componente individualmente.

Além de poder ser utilizado em diversas aplicações, como a separação de componentes em sinais de comunicação, processamento de sinais de áudio e vídeo, entre outras aplicações.