



**INSTITUTO
FEDERAL**

Santa Catarina

Câmpus
São José

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	3
2. Questão 1	3
2.1. Projeto do Filtro IIR no Simulink:	3
2.1.1. Pré-Distorção do Sinal:	3
2.1.2. Normalização do Filtro:	4
2.1.3. Calculo das atenuações:	4
2.1.4. Calculo dos parâmetros do filtro:	4
2.1.5. Calculo das Raízes do Filtro:	5
2.1.6. Calculo dos Coeficientes do Filtro:	5
2.1.7. Aplicando a função bilinear:	6
2.2. Script Matlab:	7
2.3. Resultados obtidos:	8
2.3.1. Resposta em Frequência (Magnitude):	9
2.3.2. Resposta em Frequência (Fase):	9
3. Questão 2	10
3.1. Montagem do Filtro IIR no Simulink:	10
3.1.1. Resposta em Frequência (Magnitude):	10
3.1.2. Resposta em Frequência (Fase):	11
3.1.3. Magnitude e Fase do Filtro:	11
3.1.4. Resposta ao impulso:	12
3.1.5. Polos e Zeros do Filtro:	12
3.1.6. Espalhamento de potência do filtro:	13
3.2. Comparativo entre outros Desings:	14
3.3. Aplicando o filtro no sinal de entrada:	14
3.3.1. Script Matlab:	14
3.3.2. Resultados Obtidos:	18
3.3.2.1. Sinal de Entrada:	18
3.3.2.2. Sinal de 770Hz Filtrado:	18
3.3.2.3. Sinal de 852Hz Filtrado:	19
3.3.2.4. Sinal de 941Hz Filtrado:	20
4. Conclusão	21

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:

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

$$0 \leq \omega \leq 0.2\pi \quad (2)$$

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

$$0.3\pi \leq \omega \leq \pi \quad (4)$$

Considere também que:

- T_s (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:

$$\omega = \frac{2}{T_s} \tan\left(\frac{\omega}{2}\right) \quad (5)$$

Desta forma, temos que o script matlab correspondente é:

```
1 wp = 0.2*pi;  
2 wr = 0.3*pi;  
3 ts = 2;  
4  
5 omega_ap = (2/ts)*tan(wp/2)  
6 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:

$$\Omega'_p = \frac{1}{a} \quad (6)$$

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

Desta forma, temos que o script matlab correspondente é:

```
1 a = 1;  
2 omega_p_linha = 1/a  
3 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:

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

- Para a banda de rejeição:

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

Desta forma, temos que o script matlab correspondente é:

```
1 sigma_p = 0.9;  
2 sigma_r = 0.2;  
3  
4 atenuacao_p = -1 * (20*log10(sigma_p))  
5 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 expressões:

- Calculo do fator de normalização:

$$\epsilon = \sqrt{10^{0,1,A_p} - 1} \quad (10)$$

- Cálculo da ordem do filtro:

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

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

```
1 eps = sqrt((10^(0.1*atenuacao_p))-1)
2
3 numerador = log10((10^(0.1*atenuacao_r)-1) / eps^2);
4 denominador = 2*log10(omega_r_linha);
5
6 n = ceil(numerador/denominador)
```

2.1.5. Cálculo das Raízes do Filtro:

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

$$1 + \epsilon^2 (-s'^2)^n = 0 \quad (12)$$

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

```
1 roots([eps^2 0 0 0 0 0 0 0 0 0 0 0 1])
2
3 % Raízes encontradas:
4 % -1.0900 + 0.2921i
5 % -1.0900 - 0.2921i
6 % -0.7979 + 0.7979i
7 % -0.7979 - 0.7979i
8 % -0.2921 + 1.0900i
9 % -0.2921 - 1.0900i
```

2.1.6. Cálculo dos Coeficientes do Filtro:

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

```
1 poly([
2 -1.0900 + 0.2921i
3 -1.0900 - 0.2921i
4 -0.7979 + 0.7979i
5 -0.7979 - 0.7979i
6 -0.2921 + 1.0900i
7 -0.2921 - 1.0900i
8 ])
```

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

```
1 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:

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

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

```
1 %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:

```
1 a = [1 1.4166 1.0034 0.4506 0.1349 0.0256 0.0024]
2 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:

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

```
1 Numerator:
2 0.000602340190409411798885581924473101025;
3 0.009035102856141176766446854173864267068;
4 0.012046803808188236845078122883023752365;
5 0.009035102856141176766446854173864267068;
6 0.003614041142456470793314915468386061548;
7 0.000602340190409411798885581924473101025;
8
9 Denominator:
10 1;
11 -3.293990202908562814343440550146624445915;
12 4.898232666546461722134608862688764929771;
13 -4.084996050238189013725786935538053512573;
14 1.992931744078747957615860286750830709934;
15 -0.535091724334459173384459518274525180459;
16 0.061463339042203794793106652605274575762;
```

2.2. Script Matlab:

O script matlab completo para o projeto do filtro IIR é apresentado abaixo:

```
1 clear all; close all; clc;
2 pkg load signal;
3
4 % Especificações do filtro
5 wp = 0.2*pi;
6 wr = 0.3*pi;
7 ts = 2;
8
9 % Calculo das frequências de passagem e rejeição
10 omega_ap = (2/ts)*tan(wp/2)
11 omega_ar = (2/ts)*tan(wr/2)
12
13
14 % Normalização do filtro
15 a = 1;
16 omega_p_linha = 1/a
17 omega_r_linha = omega_p_linha * (omega_ar/omega_ap)
18
19
20 % Calculo das atenuações
21 sigma_p = 0.9;
22 sigma_r = 0.2;
23 atenuacao_p = -1 * (20*log10(sigma_p))
24 atenuacao_r = -1 * (20*log10(sigma_r))
25
26
27 % Calculo dos parâmetros do filtro
28 eps = sqrt((10^(0.1*atenuacao_p))-1)
29
30 % Calculo da ordem do filtro
31
32 numerador = log10((10^(0.1*atenuacao_r)-1) / eps^2);
33 denominador = 2*log10(omega_r_linha);
34 n = ceil(numerador/denominador)
35
36 % Calculo das raízes do filtro
37 roots([eps^2 0 0 0 0 0 0 0 0 0 0 1])
38
39 poly([-1.0900 + 0.2921i
40 -1.0900 - 0.2921i
41 -0.7979 + 0.7979i
42 -0.7979 - 0.7979i
43 -0.2921 + 1.0900i
44 -0.2921 - 1.0900i])
45
46 % Montando o vetor "a" e "b" a partir dos resultados obtidos
47 b = [0.0024];
48 a = [1 1.4166 1.0034 0.4506 0.1349 0.0256 0.0024];
49 fs = 1/ts;
50
51 % Aplicando a função bilinear
52 [numerador, denominador] = bilinear(b,a,fs)
53
54 % Aplicando a função freqz para plotar a resposta em frequência
55 freqz(numerador, denominador)
```

2.3. Resultados obtidos:

A partir do script acima, obtemos os seguintes resultados para o filtro IIR projetado, conforme apresentado anteriormente:

```
1  omega_ap = 0.3249
2  omega_ar = 0.5095
3  omega_p_linha = 1
4  omega_r_linha = 1.5682
5  atenuacao_p = 0.9151
6  atenuacao_r = 13.979
7  eps = 0.4843
8  n = 6
9
10 % Raizes encontradas:
11 1 -1.08999
12   0.292061i
13
14 2 -1.08999
15   -0.292061i
16
17 3 -0.797926
18   0.797926i
19
20 4 -0.797926
21   -0.797926i
22
23 5 -0.292061
24   1.08999i
25
26 6 -0.292061
27   -1.08999i
28
29 7  0.292061
30   1.08999i
31
32 8  0.292061
33   -1.08999i
34
35 9  0.797926
36   0.797926i
37
38 10 0.797926
39   -0.797926i
40
41 11 1.08999
42   0.292061i
43
44 12 1.08999
45   -0.292061i
46
47 % Calculo dos coeficientes do filtro
48 1   2       3       4       5       6       7
49 1 4.36  9.5048 13.1362 12.1033 7.06973 2.06477
50
51
52 % Aplicando a função bilinear:
53
54 % Numerador:
```



```

55 1 4.11341e-07
56 2 2.46805e-06
57 3 6.17012e-06
58 4 8.22683e-06
59 5 6.17012e-06
60 6 2.46805e-06
61 7 4.11341e-07
62
63 % Denominador:
64 1 1
65 2 -5.29386
66 3 11.71430
67 4 -13.86580
68 5 9.25758
69 6 -3.30503
70 7 0.49283

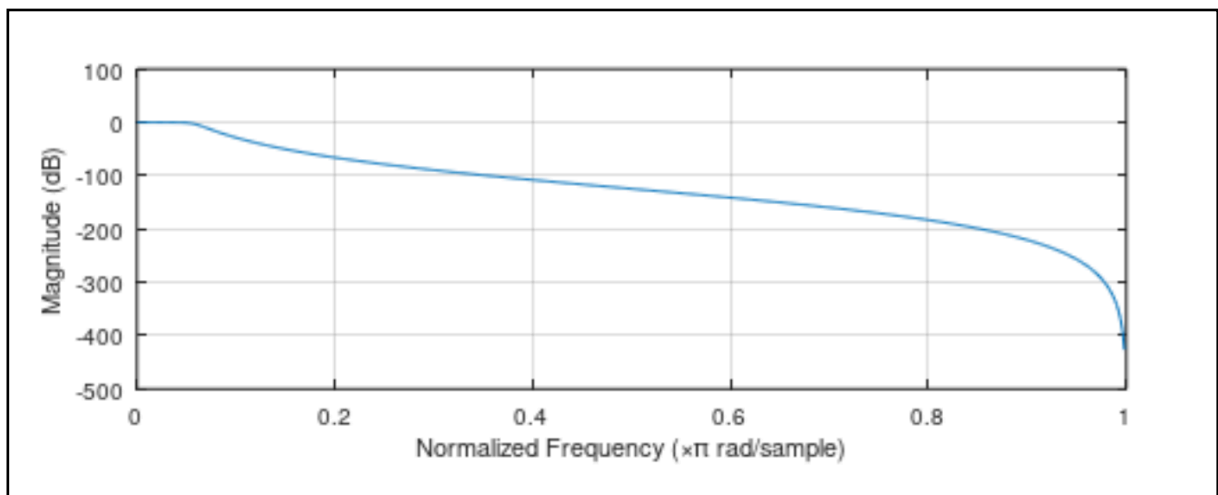
```

Aplicando a função `freqz` para plotar a resposta em frequência, obtemos os seguintes gráficos:

2.3.1. Resposta em Frequência (Magnitude):

No gráfico abaixo é apresentada a resposta em frequência do filtro projetado. Note que o filtro atende as especificações de banda de passagem e de rejeição, com uma atenuação de aproximadamente 0.2 na banda de rejeição e uma atenuação de 0.9 na banda de passagem.

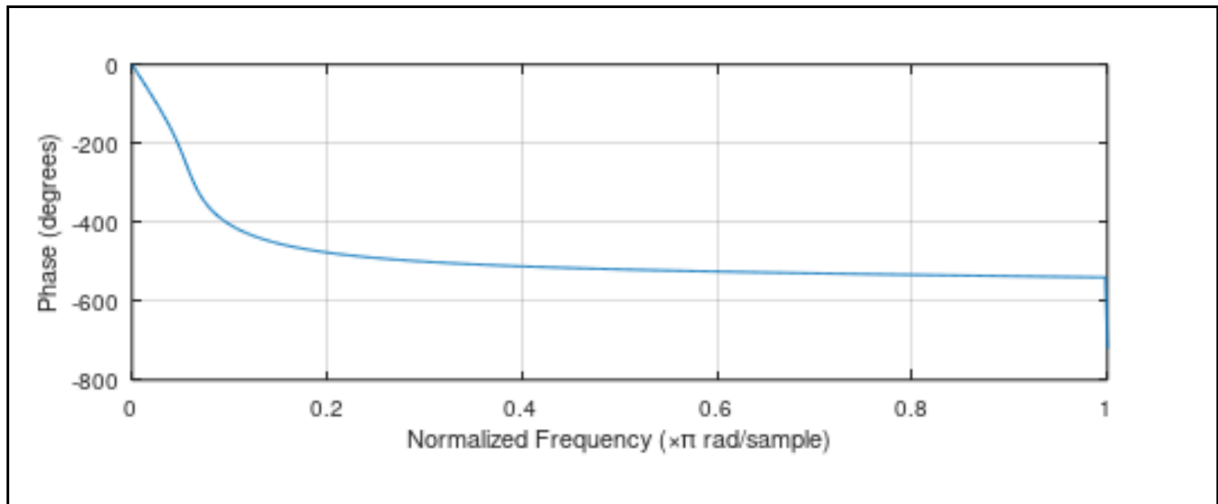
Figura 1: Elaborada pelo Autor



2.3.2. Resposta em Frequência (Fase):

A fase do filtro é apresentada na figura abaixo. Note que a fase do filtro é linear e apresenta uma variação pequena na banda de passagem:

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 $\Omega_s = 8$ 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.

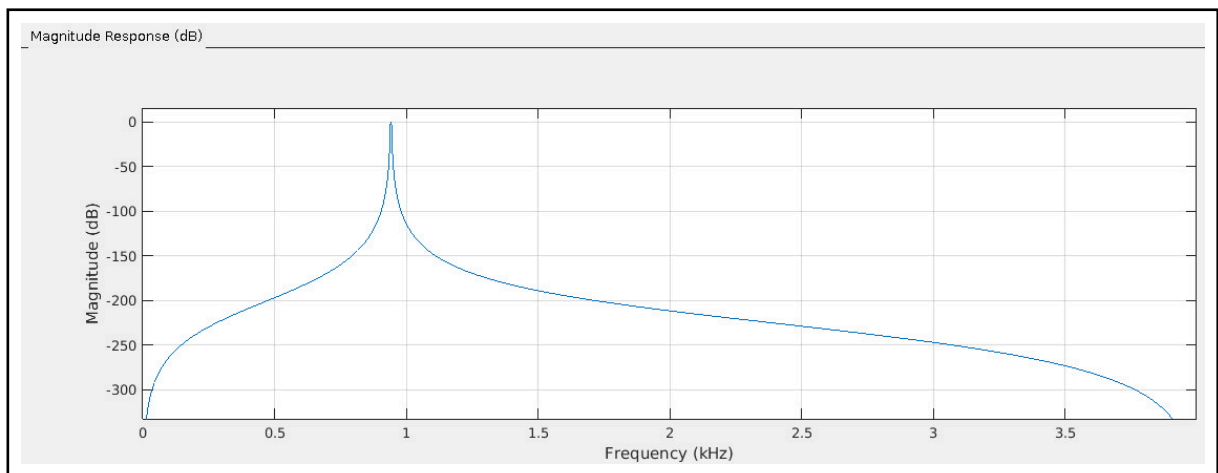
Os valores apresentados abaixo foram obtidos para o filtro de 941Hz, porem, foram realizados mais duas baterias de testes exatamente como as apresentadas abaixo, para as frequências de 770Hz e 852Hz.

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.

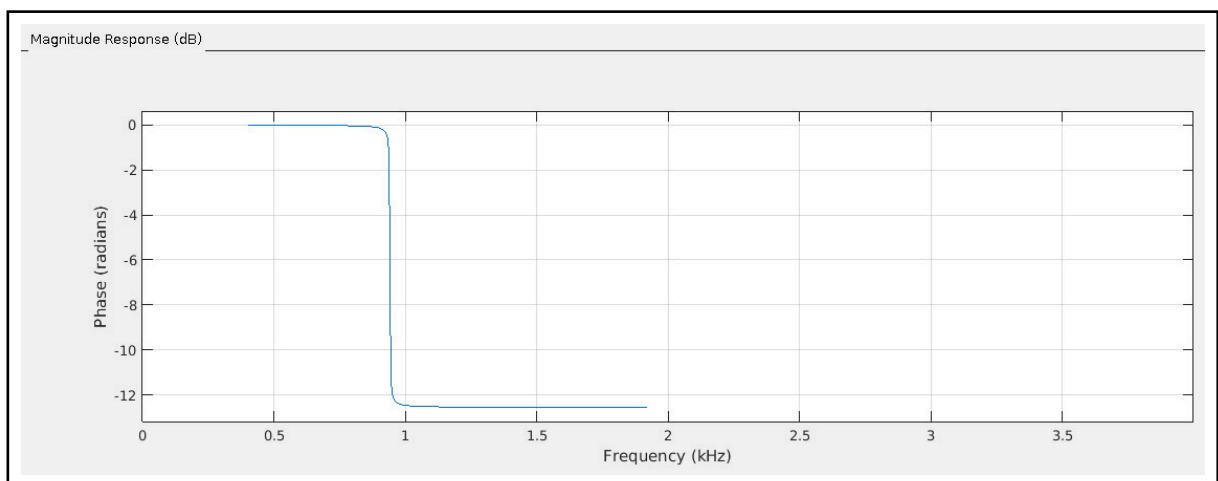
Figura 3: Elaborada pelo Autor



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.

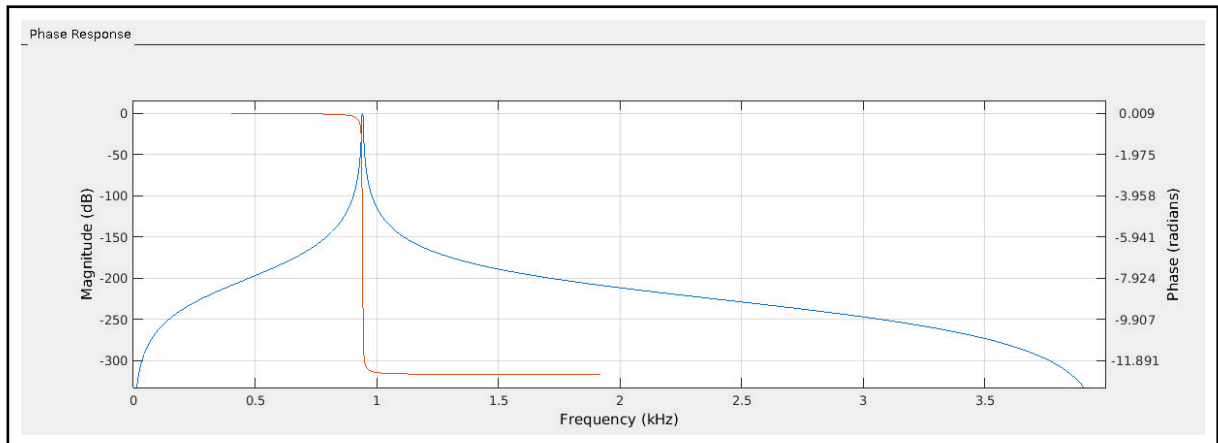
Figura 4: Elaborada pelo Autor



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

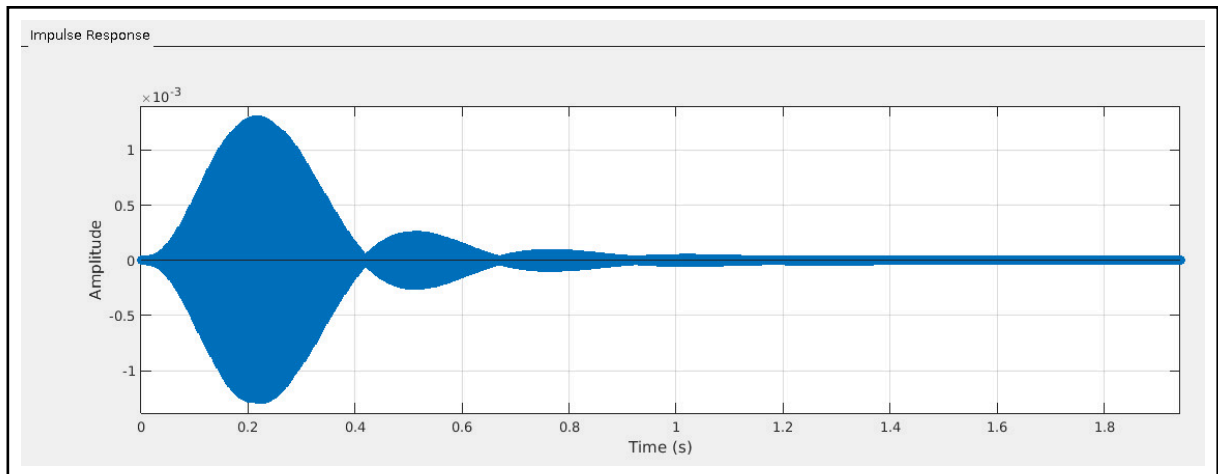


3.1.4. Resposta ao impulso:

Abaixo está apresentada a resposta ao impulso do filtro. Note que a resposta ao impulso é finita, o que 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, o que é esperado para um filtro passa-faixa.

Figura 6: Elaborada pelo Autor

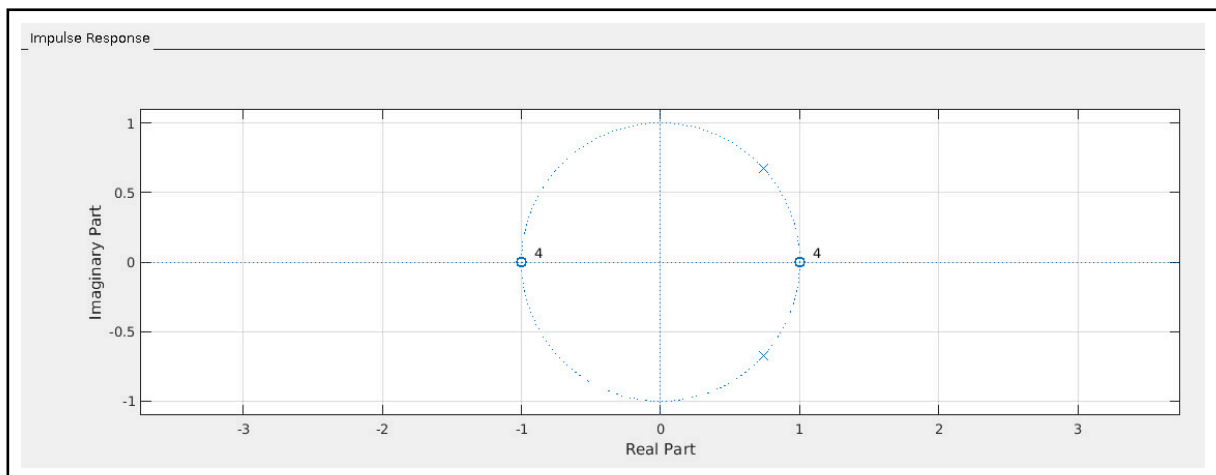


3.1.5. Polos e Zeros do Filtro:

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 o que é um requisito para o desenvolvimento deste projeto por se tratar de um filtro para sinais digitais.

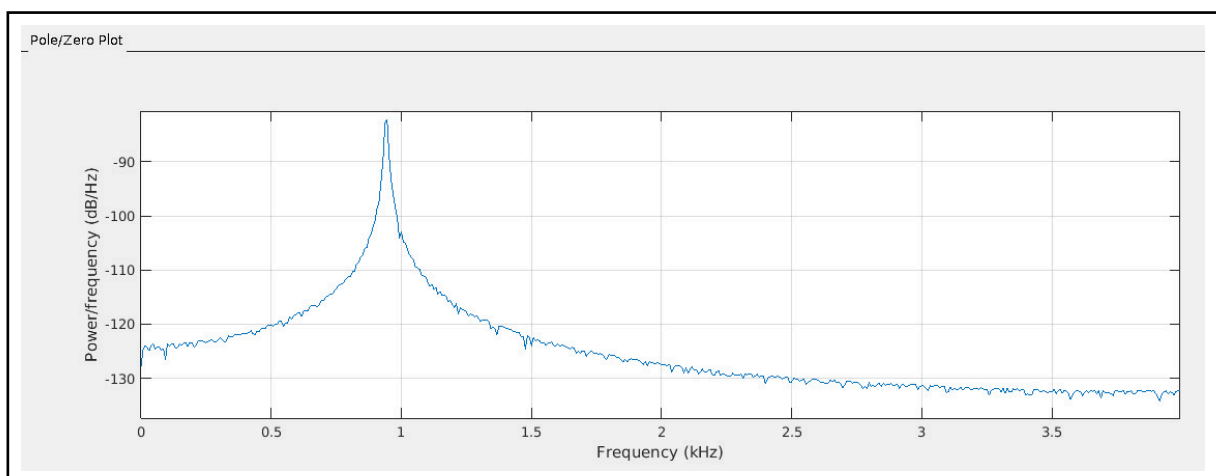
Figura 7: Elaborada pelo Autor



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 possível verificar que grande parte da potência do sinal está concentrada na banda de passagem como esperado para as características do filtro.

Figura 8: Elaborada pelo Autor



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

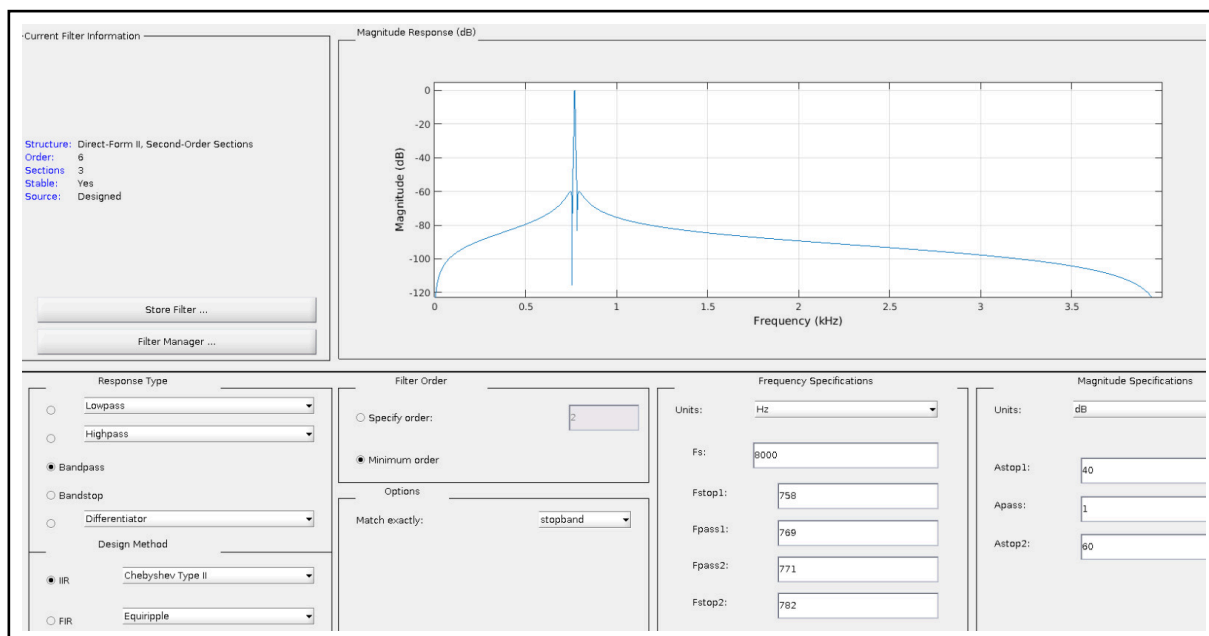
- Ordem do filtro: 8
- Banda de passagem: 941Hz - 943Hz
- Banda de transição Baixa: 930Hz - 941Hz
- Banda de transição Alta: 943Hz - 954Hz
- Frequência de amostragem: 8kHz
- Astop1 = 40dB
- Astop2 = 60dB
- Método de Desing: IIR-Butterworth

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 método 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 possível 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 próxima de zero e queda abrupta no início 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 mantém 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 possível 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:

```

1  clc; close all; clear all;
2
3  % Parâmetros do sinal
4  tmin = 0;
5  tmax = 2;
6  Omega_s = 8000;
7  Fs = Omega_s;
8  Ts = 1/Fs;
9  L = (tmax - tmin)/Ts;
10 t = 0:Ts:tmax-Ts;
11
12 % Componentes do sinal de entrada
13 f1 = 770;
14 f2 = 852;
15 f3 = 941;
16
17 % Sinal de entrada composto:
18 s_t = sin(2*pi*f1*t) + sin(2*pi*f2*t) + sin(2*pi*f3*t);
19
20 % Realizando a FFT do sinal composto:
21 S_f = fft(s_t);
22 S_f = abs(2*S_f/L);
23 S_f = fftshift(S_f);
24 freq = Fs*(-(L/2):(L/2)-1)/L;
25
26 % Realizando o plot do sinal composto no dominio do tempo:
27 figure;
28 subplot(2,1,1);
29 plot(t,s_t);
30 ylabel('Amplitude');
31 xlabel('Tempo (s)');
32 title('Sinal de entrada');
33
34 % Plotando a FFT do sinal composto
35 subplot(2,1,2);
36 plot(freq,S_f);
37 title('Espectro do sinal composto de três componentes senoidais');
38 xlabel('Frequência (Hz)');
39 ylabel('Amplitude');
40 xlim([-1000 1000]);
41 ylim([-0.1 1.1]);
42
43 % =====
44 % Aplicando as componentes de filtragem
45 % Filtro de 770Hz Butterworth
46 Num = [
47     0.000000000007648974625344080593812088956; 0;
48     -0.000000000030595898501376322375248355826; 0;
49     0.000000000045893847752064483562872533739; 0;
50     -0.000000000030595898501376322375248355826; 0;
51     0.000000000007648974625344080593812088956
52 ];
53
54 Den = [1;
55     -6.573968478516325930627317575272172689438;
56     20.197703598146055981032986892387270927429;
57     -37.435767531546105146844638511538505554199;
58     45.612066895335409810741111868992447853088;

```

```

59     -37.354424964373507123127637896686792373657;
60     20.110025517173959030969854211434721946716;
61     -6.5312086462963758748401232878677546978;
62     0.991336860368820738109718604391673579812
63 ];
64
65 % Realizando a filtragem do sinal composto:
66 s_770hz = filter(Num, Den, s_t);
67 S_770hz = fft(s_770hz);
68 S_770hz = abs(2*S_770hz/L);
69 S_770hz = fftshift(S_770hz);
70
71 % Realizando o plot do sinal composto no dominio do tempo:
72 figure;
73 subplot(2,1,1);
74 plot(freq,S_f);
75 title('Espectro do sinal composto de três componentes senoidais');
76 xlabel('Frequência (Hz)');
77 ylabel('Amplitude');
78 xlim([-1000 1000]);
79 ylim([-0.1 1.1]);
80 subplot(2,1,2);
81 plot(freq, S_770hz);
82 title('Componente do sinal de 770 Hz filtrado com passa-faixa
83 Butterworth. ');
84 xlabel('Frequência (Hz)');
85 ylabel('Amplitude');
86 xlim([-1000 1000]);
87 ylim([-0.1 1.1]);
88 % =====
89 % Aplicando as componentes de filtragem
90 % Filtro de 852Hz Butterworth
91 Num = [
92     0.000000000007674623107715503777944271234; 0;
93     -0.0000000000030698492430862015111777084937; 0;
94     0.0000000000046047738646293025898839895191; 0;
95     -0.0000000000030698492430862015111777084937; 0;
96     0.000000000007674623107715503777944271234
97 ];
98
99 Den = [1;
100     18.723854036105780807019982603378593921661;
101     -34.153268845102040529582154704257845878601;
102     41.404068891609767888439819216728210449219;
103     -34.07899654791336274683999363332986831665 ;
104     18.642505949254868369280302431434392929077;
105     -6.22711645042142070138879716978408396244;
106     0.991329630860964927663303569715935736895
107 ];
108
109 % Realizando a filtragem do sinal composto:
110 s_852hz = filter(Num, Den, s_t);
111 S_852hz = fft(s_852hz);
112 S_852hz = abs(2*S_852hz /L);
113 S_852hz = fftshift(S_852hz );
114
115 % Realizando o plot do sinal composto no dominio do tempo:

```



```

116 figure;
117 subplot(2,1,1);
118 plot(freq,S_f);
119 title('Espectro do sinal composto de três componentes senoidais');
120 xlabel('Frequência (Hz)');
121 ylabel('Amplitude');
122 xlim([-1000 1000]);
123 ylim([-0.1 1.1]);
124 subplot(2,1,2);
125 plot(freq, S_852hz );
126 title('Componente do sinal de 852 Hz filtrado com passa-faixa
Butterworth. ');
127 xlabel('Frequência (Hz)');
128 ylabel('Amplitude');
129 xlim([-1000 1000]);
130 ylim([-0.1 1.1]);
131
132 % =====
133 % Aplicando as componentes de filtragem
134 % Filtro de 940Hz Butterworth
135 Num = [0.000000000007698496068837793746781982269; 0;
136        -0.000000000030793984275351174987127929078; 0;
137         0.000000000046190976413026762480691893617; 0;
138        -0.000000000030793984275351174987127929078; 0;
139         0.000000000007698496068837793746781982269
140 ];
141 Den = [1;
142        -5.902149339413338857696089689852669835091;
143         17.054552786823688137474164250306785106659;
144        -30.518120678802585388211809913627803325653;
145         36.783803548311759357147820992395281791687;
146        -30.451702102949198547321429941803216934204;;
147         16.980399695100125256885803537443280220032;
148        -5.863697447546361019021787797100841999054;
149         0.991322918116968265778155000589322298765
150 ];
151
152 % Realizando a filtragem do sinal composto:
153 s_940hz = filter(Num, Den, s_t);
154 S_940hz = fft(s_940hz);
155 S_940hz = abs(2*S_940hz/L);
156 S_940hz = fftshift(S_940hz);
157
158 % Realizando o plot do sinal composto no dominio do tempo:
159 figure;
160 subplot(2,1,1);
161 plot(freq,S_f);
162 title('Espectro do sinal composto de três componentes senoidais');
163 xlabel('Frequência (Hz)');
164 ylabel('Amplitude');
165 xlim([-1000 1000]);
166 ylim([-0.1 1.1]);
167 subplot(2,1,2);
168 plot(freq, S_940hz);
169 title('Componente do sinal de 940 Hz filtrado com passa-faixa
Butterworth. ');
170 xlabel('Frequência (Hz)');

```

```

171 ylabel('Amplitude');
172 xlim([-1000 1000]);
173 ylim([-0.1 1.1]);

```

3.3.2. Resultados Obtidos:

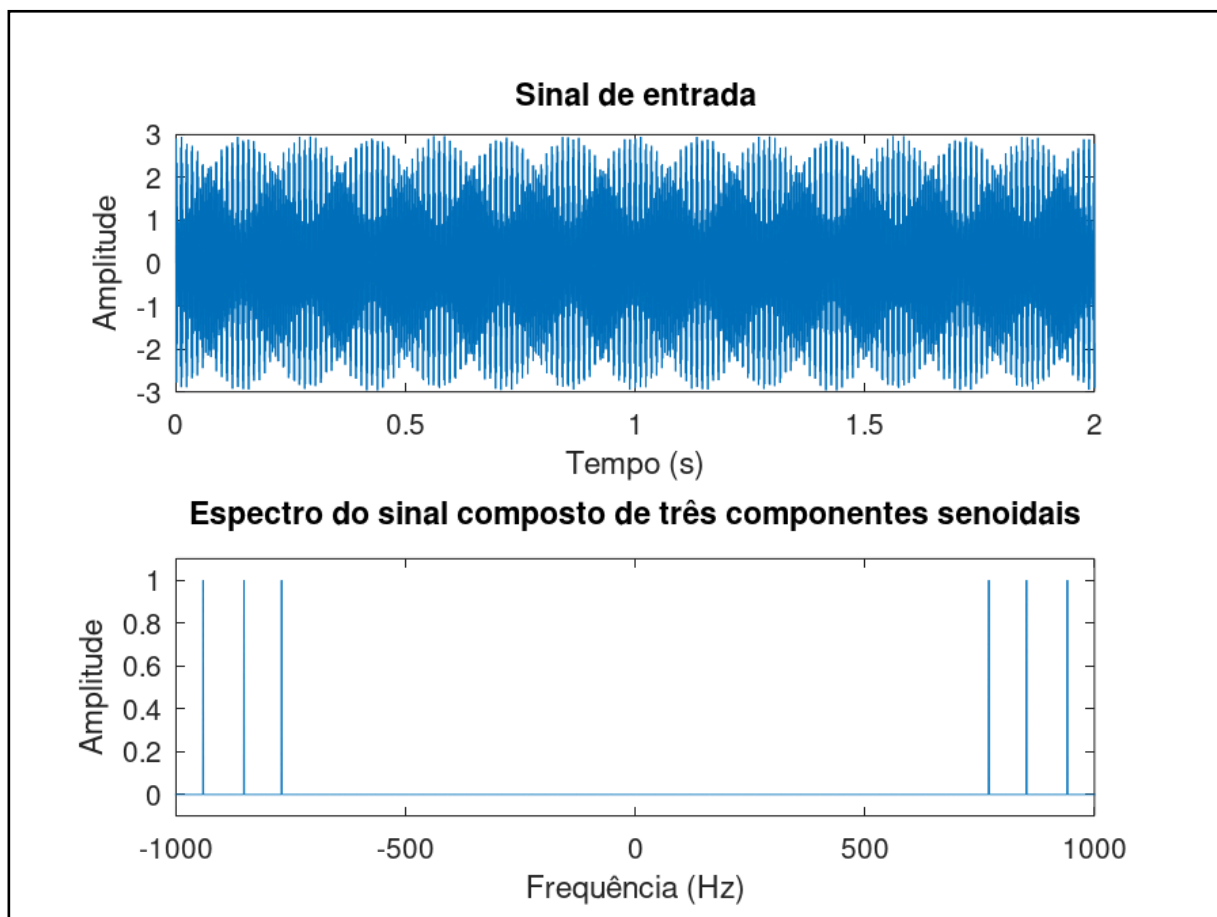
Os resultados obtidos para a filtragem das componentes cossenoidais são apresentados abaixo.

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 domínio do tempo quanto no domínio da frequência, note que no domínio da frequência aparece as três componentes cossenoidas separadamente permitindo visualizar a soma completa dos sinais.

Figura 10: Elaborada pelo Autor



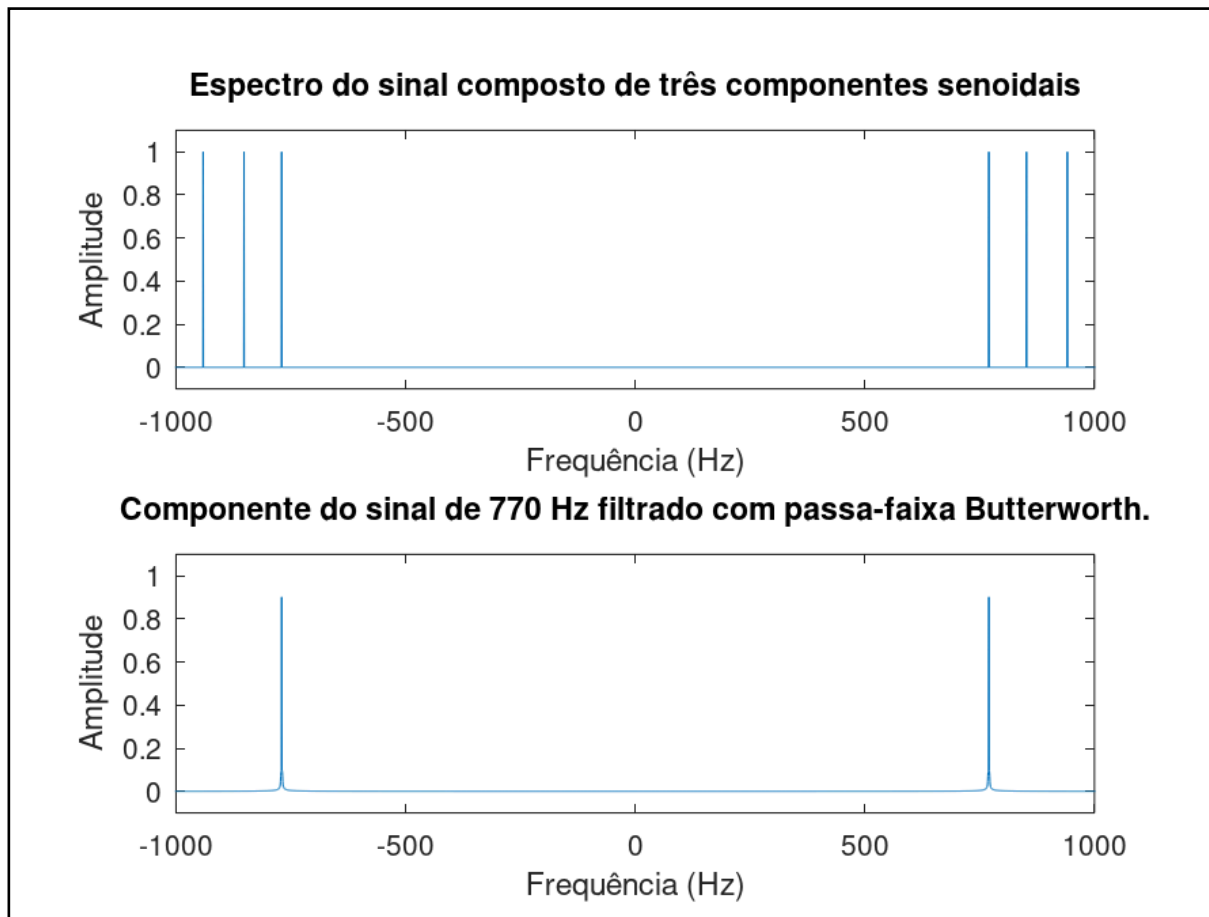
Filtro de 941Hz Utilizando Chebyshev

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, porém, a atenuação é suficiente para que a componente de 770Hz seja isolada das demais.

Figura 11: Elaborada pelo Autor



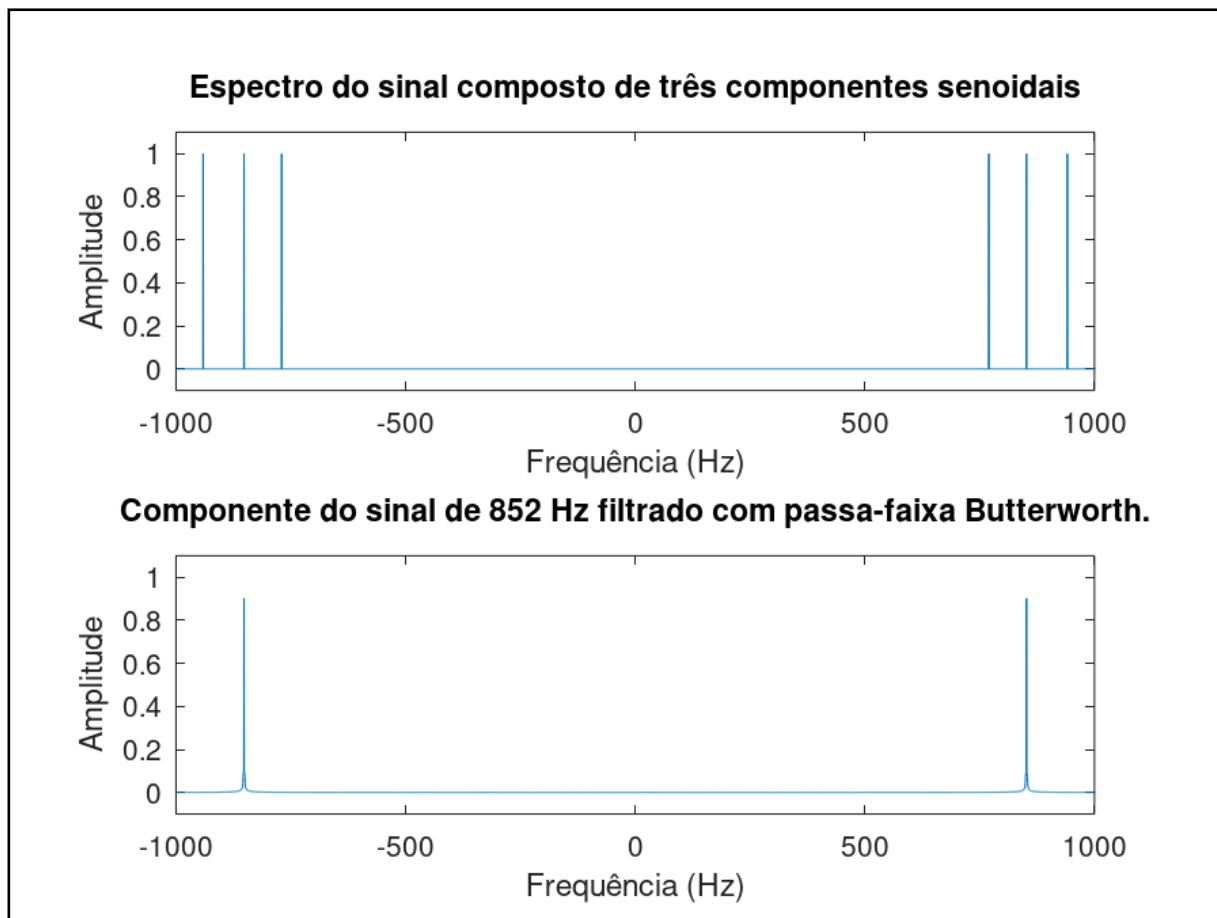
Filtro de 941Hz Utilizando Chebyshev

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, porém, a atenuação é suficiente para que a componente de 852Hz seja isolada das demais.

Figura 12: Elaborada pelo Autor



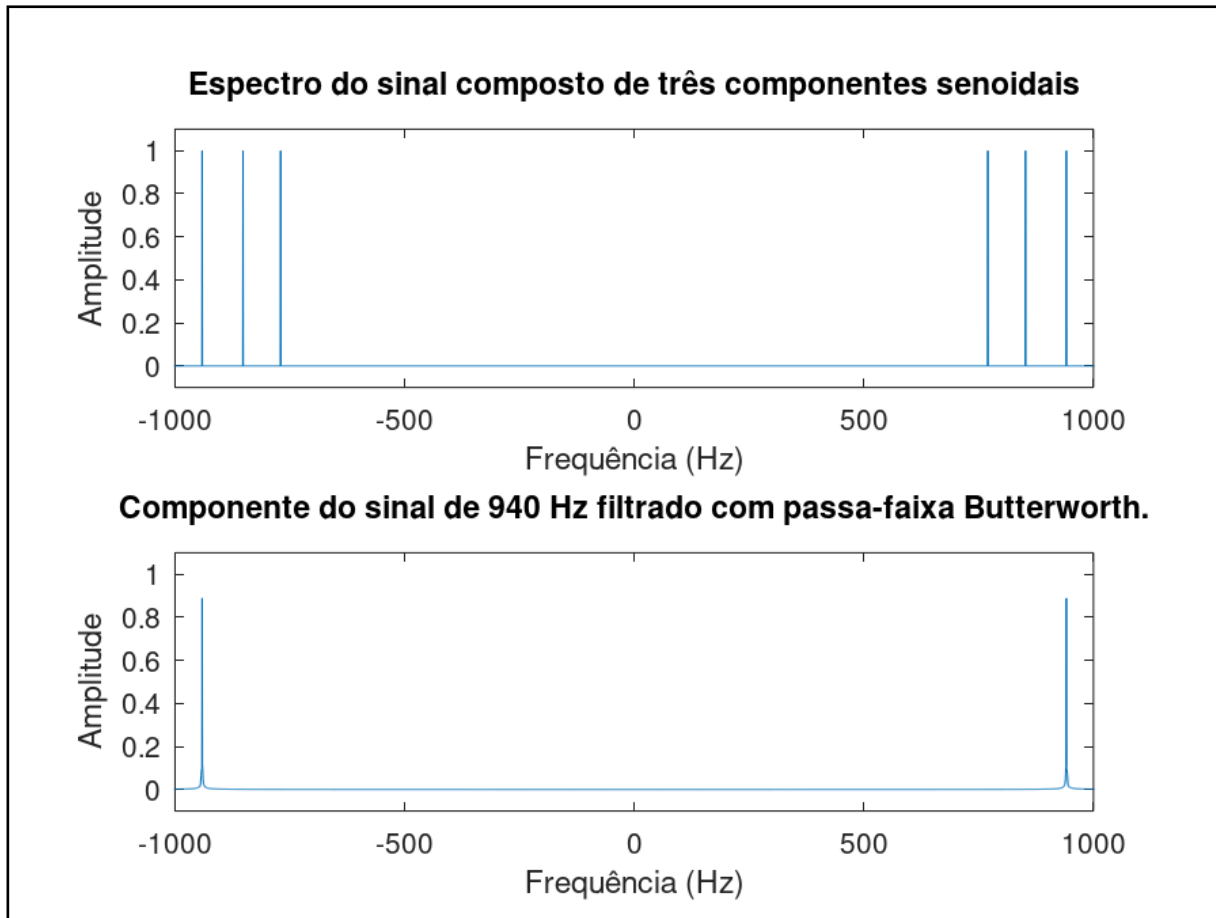
Filtro de 941Hz Utilizando Chebyshev

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 possível 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.