INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Projeto de Filtros IIR

Processamento de Sinais Digitais

Curso: Engenharia de Telecomunicações
Disciplina: PSD129005 - Processamento de Sinais Digitais
Professora: Elen Macedo Lobato

Aluna
Luiza Kuze Gomes

18 de julho de 2025

Sumário

1 Introdução **2**

2 Questão 1 **2**
2.1 Enunciado . . . 2
2.2 Análise Inicial . . . 2
2.3 Resolução . . . 3
2.3.1 Pré-Distorção . . . 3
2.3.2 Normalização . . . 3
2.3.3 Atenuação . . . 4
2.3.4 Fator de Ripple e Ordem do Filtro . . . 4
2.3.5 Cálculo das Raízes . . . 5
2.3.6 Coeficientes do Polinômio . . . 6
2.3.7 Desnormalização . . . 7
2.3.8 Transformação Bilinear . . . 7
2.3.9 Resposta em Frequência . . . 8
2.3.10 Script MATLAB . . . 8

3 Questão 2 **9**
3.1 Enunciado . . . 9
3.2 Análise Inicial . . . 9
3.3 Resolução . . . 12
3.3.1 Magnitude . . . 12
3.3.2 Fase . . . 13
3.3.3 Resposta ao Impulso . . . 13
3.3.4 Polos e Zeros . . . 14
3.3.5 Script MATLAB . . . 15
3.3.6 Aplicação ao Sinal de Entrada . . . 17

4 Conclusão **23**

1 Introdução

Este trabalho tem como objetivo o estudo e a análise do comportamento de filtros digitais do tipo IIR (Infinite Impulse Response). Foi utilizado o filtro Butterworth, conhecido por apresentar uma resposta em frequência suave e sem ondulações na banda passante, proporcionando uma transição eficiente entre as bandas de interesse.

Filtros IIR são caracterizados por sua estrutura recursiva, o que significa que sua saída depende não apenas dos valores atuais e passados da entrada, mas também das saídas anteriores. Essa característica permite que o filtro produza resposta mesmo após o fim do sinal de entrada, resultando em uma resposta ao impulso teoricamente infinita [2].

Ao longo do trabalho, exploramos a resposta dos filtros em frequência, fase, impulso e a extração de componentes espectrais de um sinal composto, buscando compreender e visualizar o funcionamento e desempenho desse tipo de filtro no domínio digital.

2 Questão 1

2.1 Enunciado

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

$$\begin{aligned} 0,9 \leq |H(e^{j\omega})| \leq 1, \quad 0 \leq \omega \leq 0{,}2\pi \\ |H(e^{j\omega})| \leq 0{,}2, \quad 0{,}3\pi \leq \omega \leq \pi \end{aligned} \tag{1}$$

Considere $T_s = 2$. Faça o mesmo projeto usando o MATLAB ou SIMULINK. Plote a resposta em frequência.

2.2 Análise Inicial

A análise inicial consiste em entender como deve ser o filtro que vamos projetar. A partir do enunciado, extraímos as seguintes especificações:

- Um filtro que deixe passar frequências digitais até $\omega = 0{,}2\pi$, com ganho entre 0,9 e 1;
- Um filtro que rejeite frequências digitais a partir de $\omega = 0{,}3\pi$, com ganho no máximo igual a 0,2;
- Utilizar a transformação bilinear;
- O período de amostragem considerado é $T_s = 2$;
- O filtro será do tipo Butterworth passa-baixas.

Foram definidos os passos necessários para o desenvolvimento do filtro, conforme o algoritmo de projeto:

1. Calcular as frequências analógicas Ω_p e Ω_r utilizando a transformação bilinear inversa.
2. Normalizar as frequências, tornando $\Omega'_p = 1$.
3. Calcular as atenuações A_p e A_r a partir dos valores de ripple.
4. Determinar o fator de ripple ε e a ordem n do filtro.
5. Calcular as raízes da equação característica do filtro Butterworth.
6. Determinar os coeficientes do polinômio normalizado.

7. Desnormalizar o filtro.
8. Aplicar a transformação bilinear para obter o filtro digital.
9. Obter e visualizar a resposta em frequência do filtro projetado.

2.3 Resolução

Seguindo o algoritmo da análise inicial, foi implementado o projeto de um filtro digital Butterworth passa-baixas utilizando a transformação bilinear.

2.3.1 Pré-Distorção

A pré-distorção do sinal consiste no processo de conversão das frequências digitais fornecidas (em rad/amostra) para o domínio analógico (em rad/s), utilizando a transformação bilinear inversa. Essa etapa é fundamental, pois o projeto do filtro Butterworth é inicialmente realizado no domínio analógico, sendo posteriormente convertido para o domínio digital. A transformação aplicada é dada por:

$$\Omega = \frac{2}{T_s} \tan\left(\frac{\omega}{2}\right) \tag{2}$$

Com base no enunciado, consideramos $T_s = 2$ e utilizamos as seguintes frequências digitais: $\omega_p = 0{,}2\pi$ (frequência de passagem) e $\omega_r = 0{,}3\pi$ (frequência de rejeição).

O trecho da simulação em MATLAB que representa esse procedimento é o seguinte:

```
% Especificações do filtro
wp = 0.2*pi;                    % Frequência de passagem digital (rad)
wr = 0.3*pi;                    % Frequência de rejeição digital (rad)
ts = 2;                         % Período de amostragem (s)

% Transformação Bilinear Reversa
omega_ap = (2/ts)*tan(wp/2);  % Frequência de passagem analógica (rad/s)
omega_ar = (2/ts)*tan(wr/2);  % Frequência de rejeição analógica (rad/s)
```

A saída desses cálculos é:

$$\Omega_p \approx 0{,}3249, \quad \Omega_r \approx 0{,}5095$$

2.3.2 Normalização

A partir dos valores de Ω_p e Ω_r calculados na etapa de pré-distorção, realizamos o processo de normalização da frequência analógica.

Esse procedimento é importante pois as fórmulas padrão do projeto de filtros Butterworth são construídas assumindo que a frequência de passagem está normalizada em $\Omega'_p = 1$. Com isso, podemos utilizar tabelas, expressões e propriedades matemáticas diretamente, sem adaptação.

No caso do filtro Butterworth, adotamos $a = 1$ como constante de transformação. Assim:

$$\Omega'_p = \frac{1}{a} = 1, \quad \Omega'_r = \frac{1}{a} \cdot \frac{\Omega_r}{\Omega_p} \tag{3}$$

Esse procedimento implica que, ao final do algoritmo, será necessário realizar o processo de desnormalização, isto é, reverter o escalamento para que a função de transferência retorne ao domínio original.

A trecho da simulação correspondente em MATLAB é:

```
% Normalização do filtro analógico
a = 1;                                      % Parâmetro de normalização (Butterworth: a = 1)
omega_p_linha = 1/a;                                % Frequência de passagem normalizada
omega_r_linha = omega_p_linha * (omega_ar/omega_ap);  % Frequência de rejeição normalizada
```

Resultado:

$$\Omega'_r \approx 1{,}5682$$

2.3.3 Atenuação

O objetivo desta etapa é converter os limites de variação da resposta em magnitude do filtro, expressos por δ_p e δ_r, em valores de atenuação em decibéis (dB). Esse procedimento é essencial para a posterior definição do fator de ripple ε e da ordem do filtro.

Na banda de **passagem**, a resposta do filtro deve estar contida dentro de uma faixa de tolerância em torno da unidade. A ondulação máxima permitida é representada por δ_p, e o ganho mínimo permitido é:

$$G_p = 20\log_{10}(1-\delta_p) \tag{4}$$

A atenuação correspondente é:

$$A_p = -G_p = -20\log_{10}(1-\delta_p) \tag{5}$$

Já na banda de **rejeição**, o filtro deve garantir que a magnitude da resposta seja, no máximo, δ_r, e por isso o ganho nessa faixa é dado por:

$$G_r = 20\log_{10}(\delta_r), \quad A_r = -G_r = -20\log_{10}(\delta_r) \tag{6}$$

Substituindo os valores fornecidos no enunciado:

$$\delta_p = 0{,}1, \quad \delta_r = 0{,}2$$

Obtemos os seguintes valores de atenuação por meio da simulação:

$$A_p = -20\log_{10}(1-0{,}1) \approx 0{,}9151 \text{ dB} \tag{7}$$

$$A_r = -20\log_{10}(0{,}2) \approx 13{,}9794 \text{ dB} \tag{8}$$

A trecho da simulação correspondente em MATLAB é:

```
% Cálculo das atenuações
sigma_p = 0.1;                      % Ripple (máxima variação permitida) na passagem
sigma_r = 0.2;                      % Ripple (valor máximo) na rejeição
atenuacao_p = -20 * log10(1 - sigma_p)     % Atenuação em dB na passagem (dB)
atenuacao_r = -20 * log10(sigma_r)         % Atenuação em dB na rejeição (dB)
```

2.3.4 Fator de Ripple e Ordem do Filtro

Com os valores de atenuação calculados nas etapas anteriores, podemos agora determinar dois parâmetros fundamentais para o projeto do filtro Butterworth:

- **O fator de ripple** ε : quantifica a ondulação máxima permitida na banda de passagem (ou a seletividade do filtro). Embora o filtro Butterworth seja monotônico, ε é um parâmetro necessário para os passos seguintes.

- **A ordem do filtro** n: representa o grau do polinômio do denominador e indica quantos polos o filtro terá.

As fórmulas utilizadas são:

$$\varepsilon = \sqrt{10^{0.1A_p} - 1} \tag{9}$$

$$n \geq \frac{\log_{10}\left(\frac{10^{0,1A_r} - 1}{\varepsilon^2}\right)}{2\log_{10}\Omega'_r} \tag{10}$$

A trecho da simulação correspondente em MATLAB é:

```
% Cálculo do parâmetro epsilon
eps = sqrt((10^(0.1*atenuacao_p))-1);

% Cálculo da ordem do filtro
numerador = log10((10^(0.1*atenuacao_r)-1) / eps^2);
denominador = 2*log10(omega_r_linha);
n = ceil(numerador/denominador);   % Ordem mínima do filtro
```

Resultados da simulação: $\varepsilon \approx 0{,}4843$, $n = 6$

2.3.5 Cálculo das Raízes

Adicionalmente, demonstraremos como se chega à equação utilizada para o cálculo das raízes do filtro Butterworth, dada sua simplicidade.

A equação que caracteriza a atenuação de um filtro Butterworth passa-baixas normalizado é dada por:

$$|A(j\Omega')|^2 = 1 + \varepsilon^2(\Omega')^{2n} \tag{11}$$

onde:

- $A(j\Omega')$ é a função de atenuação desejada em frequência analógica normalizada;
- Ω' é a frequência analógica normalizada;
- ε é o fator de ripple;
- n é a ordem do filtro.

A equação (11) pode ser reescrita em termos de sua fatoração como:

$$|A(j\Omega')|^2 = A(-j\Omega')A(j\Omega') = 1 + \varepsilon^2(\Omega')^{2n} \tag{12}$$

Utilizando continuação analítica, substituímos $j\Omega' \to s'$, com $s' \in \mathbb{C}$, obtendo:

$$A(s')A(-s') = 1 + \varepsilon^2(-s'^2)^n \tag{13}$$

Para encontrar os polos do filtro, basta resolver:

$$1 + \varepsilon^2(-s'^2)^n = 0 \tag{14}$$

As soluções dessa equação são $2n$ raízes complexas s_i, distribuídas simetricamente sobre uma circunferência de raio $\varepsilon^{-1/n}$, centrada na origem do plano s'. Essas raízes são igualmente espaçadas em ângulo. No entanto, para garantir a estabilidade do sistema, apenas as n raízes localizadas no

semiplano esquerdo do plano s' (ou seja, com parte real negativa) são utilizadas na construção do polinômio $A(s')$, que define o denominador da função de transferência.

A seguir, apresentamos o cálculo prático dessas raízes com auxílio da ferramenta MATLAB:

```
% Cálculo das raízes do polinômio de Butterworth
disp('>> Raízes do polinômio Butterworth:');
roots([eps^2 0 0 0 0 0 0 0 0 0 0 0 1])
```

O vetor fornecido ao comando `roots()` representa o polinômio:

$$\varepsilon^2 s'^{12} + 1 = 0 \tag{15}$$

Como a ordem calculada do filtro é $n = 6$, esse polinômio possui grau 12 e resulta em 12 raízes complexas (pares conjugados), conforme a saída abaixo:

$$\texttt{ans} = \begin{cases} -1.0900 + 0.2921\text{i}, & -1.0900 - 0.2921\text{i} \\ -0.7979 + 0.7979\text{i}, & -0.7979 - 0.7979\text{i} \\ -0.2921 + 1.0900\text{i}, & -0.2921 - 1.0900\text{i} \\ 0.2921 + 1.0900\text{i}, & 0.2921 - 1.0900\text{i} \\ 0.7979 + 0.7979\text{i}, & 0.7979 - 0.7979\text{i} \\ 1.0900 + 0.2921\text{i}, & 1.0900 - 0.2921\text{i} \end{cases}$$

2.3.6 Coeficientes do Polinômio

A função de transferência do filtro Butterworth normalizado pode ser expressa como:

$$H'(s') = \frac{H'_0}{A(s')} = \frac{H'_0}{\prod_{i=1}^{n}(s' - p_i)}, \tag{16}$$

onde p_i são os polos localizados no semiplano esquerdo do plano s', ou seja, aqueles com parte real negativa. Esses polos são selecionados dentre as $2n$ raízes complexas da equação característica $1 + \varepsilon^2(-s'^2)^n = 0$, garantindo a estabilidade do filtro.

O fator de ganho H'_0 é escolhido de forma que a magnitude da resposta em frequência seja unitária em $\Omega' = 0$, ou seja:

$$|H'(j0)| = 1. \tag{17}$$

Com isso, o ganho é determinado por:

$$H'_0 = \prod_{i=1}^{n}(-p_i). \tag{18}$$

Após selecionar as 6 raízes estáveis, utilizamos a função `poly()` do MATLAB para obter os coeficientes do polinômio $A(s')$:

```
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])
```

A saída obtida foi:

$$\texttt{ans} = [1.0000 \quad 4.3600 \quad 9.5048 \quad 13.1362 \quad 12.1033 \quad 7.0697 \quad 2.0648]$$

Portanto, a função de transferência normalizada do filtro é dada por:

$$H'(s') = \frac{2{,}0648}{s'^6 + 4{,}3600s'^5 + 9{,}5048s'^4 + 13{,}1362s'^3 + 12{,}1033s'^2 + 7{,}0697s' + 2{,}0648} \tag{19}$$

2.3.7 Desnormalização

Com os coeficientes do polinômio A(s′) obtidos a partir da função normalizada H′(s′), realizamos agora a etapa de desnormalização, que consiste em converter a função do domínio com frequência de passagem unitária para o domínio analógico real com $\Omega_p \neq 1$.

A relação entre as variáveis é dada por:

$$s' = \left(\frac{1}{a}\right)\left(\frac{s}{\Omega'_p}\right) \tag{20}$$

A desnormalização altera os coeficientes, escalando a frequência de acordo com Ω_p. Dessa forma, a função analógica desnormalizada final do filtro é:

$$H(s) = \frac{0{,}0024}{s^6 + 1{,}4166s^5 + 1{,}0034s^4 + 0{,}4506s^3 + 0{,}1349s^2 + 0{,}0256s + 0{,}0024} \tag{21}$$

Feitos esses cálculos, podemos representar a função no ambiente MATLAB com os seguintes vetores de coeficientes:

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

2.3.8 Transformação Bilinear

A transformação bilinear [1], também conhecida como substituição de Tustin, é uma técnica que converte um sistema contínuo (domínio s) em um sistema discreto (domínio z), preservando a estabilidade e mapeando todo o semiplano esquerdo de s para o interior do círculo unitário no plano z.

A substituição utilizada é dada por:

$$s \leftarrow \frac{2}{T_s} \cdot \frac{z-1}{z+1}$$

Esse mapeamento possui as seguintes propriedades:

- Garante que polos com parte real negativa em s (estáveis) sejam mapeados para dentro do círculo unitário em z;
- Comprime o eixo de frequências analógicas $\Omega \in (-\infty, \infty)$ no intervalo de frequências digitais $\omega \in [-\pi, \pi]$;
- Evita a ocorrência de aliasing, diferentemente de métodos de amostragem direta.

No ambiente MATLAB, a transformação bilinear é realizada com a função `bilinear()`, que recebe os coeficientes do filtro analógico e a frequência de amostragem. O trecho correspondente na simulação é:

```
% Transformação bilinear para obter H(z)
[num, den] = bilinear(b, a, 1/ts);
```

A saída desse comando é:

$$\texttt{num} = [0{,}0006 \quad 0{,}0036 \quad 0{,}0089 \quad 0{,}0119 \quad 0{,}0089 \quad 0{,}0036 \quad 0{,}0006]$$

$$\texttt{den} = [1{,}0000 \quad -3{,}2941 \quad 4{,}8982 \quad -4{,}0853 \quad 1{,}9929 \quad -0{,}5352 \quad 0{,}0615]$$

Esses vetores correspondem ao numerador e denominador da função de transferência do filtro digital $H(z)$, pronto para ser analisado em frequência na próxima etapa.

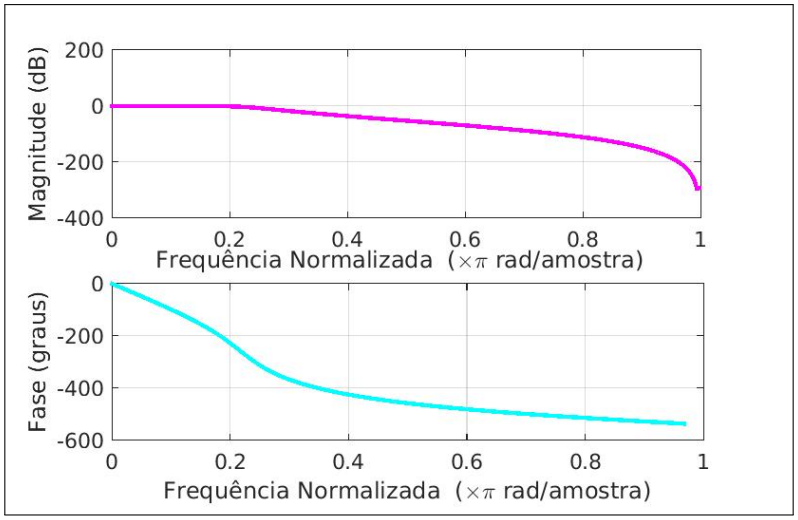
2.3.9 Resposta em Frequência

Nesta etapa, analisamos o comportamento do filtro digital no domínio da frequência. A Figura 1 apresenta, simultaneamente, a resposta em magnitude (em decibéis) e a resposta em fase (em graus), obtidas por meio da simulação no ambiente MATLAB.

Observa-se que o filtro atende às especificações do projeto, com uma atenuação de aproximadamente 0,9 na banda de passagem ($\omega \leq 0{,}2\pi$) e inferior a 0,2 na banda de rejeição ($\omega \geq 0{,}3\pi$), o que caracteriza um comportamento adequado de filtragem.

A resposta em fase apresenta uma variação pequena na banda de passagem. Apesar de o critério de projeto ser baseado principalmente na magnitude, a análise da fase é relevante em aplicações que exigem preservação da forma de onda do sinal.

Figura 1: Respostas em Magnitude e Fase do Filtro Butterworth Digital



Fonte: Própria autora

2.3.10 Script MATLAB

Ao unificar os trechos de código utilizados até então, obtemos o seguinte script completo em MATLAB:

```
clc; close all; clear all;

% Especificações do filtro
wp = 0.2*pi;                    % Frequência de passagem digital (rad)
wr = 0.3*pi;                    % Frequência de rejeição digital (rad)
ts = 2;                         % Período de amostragem (s)

% Transformação Bilinear Reversa
omega_ap = (2/ts)*tan(wp/2)  % Frequência de passagem analógica (rad/s)
```

```
omega_ar = (2/ts)*tan(wr/2)  % Frequência de rejeição analógica (rad/s)

% Normalização do filtro analógico
a = 1;                                          % Parâmetro de normalização (Butterworth: a
    = 1)
omega_p_linha = 1/a;                            % Frequência de passagem normalizada
omega_r_linha = omega_p_linha * (omega_ar/omega_ap)  % Frequência de rejeição normalizada

% Cálculo das atenuações
sigma_p = 0.1;                    % Ripple (máxima variação permitida) na passagem
sigma_r = 0.2;                    % Ripple (valor máximo) na rejeição
atenuacao_p = -20 * log10(1 - sigma_p)     % Atenuação em dB na passagem (dB)
atenuacao_r = -20 * log10(sigma_r)         % Atenuação em dB na rejeição (dB)

% Cálculo do parâmetro epsilon ()
eps = sqrt((10^(0.1*atenuacao_p))-1);

% Cálculo da ordem do filtro
numerador = log10((10^(0.1*atenuacao_r)-1) / eps^2);
denominador = 2*log10(omega_r_linha);
n = ceil(numerador/denominador);   % Ordem mínima do filtro

% Cálculo das raízes do polinômio de Butterworth
disp('>> Raízes do polinômio Butterworth:');
roots([eps^2 0 0 0 0 0 0 0 0 0 0 0 1])                % Equação característica do Butterworth (
    grau 6)

% Cálculo dos coeficientes do denominador H(s)
disp('>> Coeficientes de H(s):');
poly([-1.0900 + 0.2921i                                    % Raízes normalizadas no plano s'
      -1.0900 - 0.2921i
      -0.7979 + 0.7979i
      -0.7979 - 0.7979i
      -0.2921 + 1.0900i
      -0.2921 - 1.0900i])

% Montagem dos coeficientes H(s) desnormalizado
b = [0.0024];                                   % Numerador do filtro analógico H(s)
a = [1 1.4166 1.0034 0.4506 0.1349 0.0256 0.0024]; % Denominador do filtro analógico H(s)
fs = 1/ts;                                      % Frequência de amostragem (Hz)

% Transformação bilinear: H(s) para H(z)
[num, den] = bilinear(b, a, fs)   % H(z)

% Cálculo da resposta em frequência
freqz(num, den)
```

3 Questão 2

3.1 Enunciado

Crie, usando *MATLAB*, um sinal de entrada composto de três componentes senoidais, nas frequências 770 Hz, 852 Hz e 941 Hz, com $\Omega_s = 8\,\text{kHz}$. Projete, usando o *Simulink* ou o *MATLAB*, um filtro IIR para isolar cada componente. Documente as especificações utilizadas. Faça comentários.

3.2 Análise Inicial

A análise inicial deste problema consiste na escolha criteriosa dos parâmetros que serão utilizados na ferramenta de simulação. Foi empregada a função `fdatool`, ela apresenta uma interface gráfica

interativa do MATLAB que permite projetar e visualizar filtros digitais. Essa ferramenta oferece suporte tanto para filtros IIR quanto FIR, com ampla capacidade de personalização.

No nosso contexto, optamos pela utilização do filtro IIR do tipo Butterworth para todas as componentes senoidais. Essa escolha se justifica pelo comportamento de fase suave na banda de passagem e por sua resposta monotônica, ou seja, sem ondulações (*ripple*) tanto na faixa de passagem quanto na de rejeição.

Parâmetros Genéricos

Os parâmetros genéricos adotados, válidos para todos os três filtros (770Hz, 852Hz e 941Hz), estão apresentados na Tabela 1.

Tabela 1: Parâmetros genéricos aplicados aos filtros

Parâmetro	**Valor**
Frequência de amostragem (f_s)	8000 Hz
Atenuação da 1ª banda de rejeição (A_{stop1})	40 dB
Atenuação da 2ª banda de rejeição (A_{stop2})	60 dB
Ripple máximo na banda de passagem (A_{pass})	1 dB
Estrutura do filtro	Direct-Form II
Cálculo da ordem	Mínima automática (MATLAB)
Tipo de filtro	IIR Butterworth

Fonte: Própria autora

- **Frequência de amostragem (f_s):** foi fornecida diretamente no enunciado da questão. Representa a taxa com que o sinal é amostrado, e deve ser suficientemente alta para capturar com precisão as componentes de 770, 852 e 941 Hz, respeitando o Teorema de Nyquist.
- **Atenuação nas bandas de rejeição (A_{stop1} e A_{stop2}):** os valores de 40dB e 60dB foram definidos para garantir uma supressão efetiva das componentes fora da frequência de interesse. Isso assegura que apenas a faixa desejada seja preservada com mínima interferência das demais.
- **Ripple na banda de passagem (A_{pass}):** o valor de 1dB foi adotado para permitir uma leve ondulação no ganho dentro da faixa de passagem, garantindo fidelidade do sinal sem exigir uma ordem muito elevada do filtro.
- **Estrutura do filtro:** a estrutura *Direct-Form II* foi escolhida por ser eficiente em termos computacionais e exigir menos memória para implementação. Além disso, é apropriada para filtros IIR de ordem elevada, mantendo a estabilidade numérica.
- **Cálculo da ordem:** o MATLAB foi configurado para selecionar automaticamente a menor ordem que atenda todas as especificações de magnitude. Isso resulta em filtros mais simples, com menor custo computacional e maior estabilidade.
- **Tipo de filtro:** o filtro Butterworth foi escolhido devido à sua resposta em frequência monotônica e sua suavidade de fase, características desejáveis para filtragem de sinais senoidais.

Parâmetros Específicos por Filtro

Para isolar adequadamente cada componente senoidal, definimos bandas de passagem estreitas ao redor da frequência central de interesse, com bandas de rejeição posicionadas antes e depois

para garantir a atenuação desejada. A Tabela 2 mostra os parâmetros específicos utilizados para cada frequência.

Tabela 2: Parâmetros específicos de projeto para cada componente senoidal

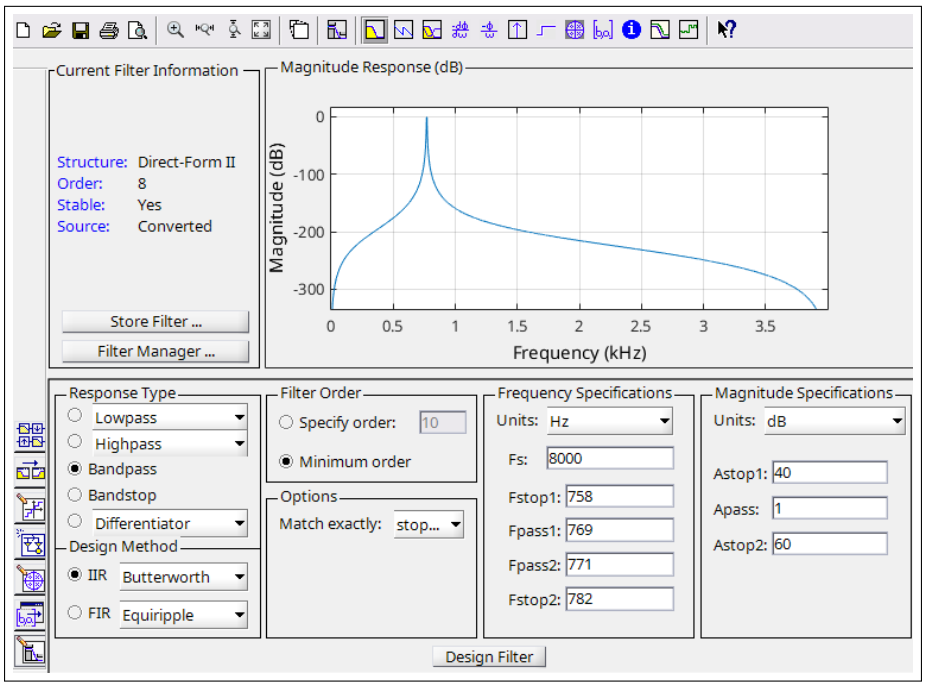
Frequência central	Fstop1 (Hz)	Fpass1 (Hz)	Fpass2 (Hz)	Fstop2 (Hz)
770 Hz	760	765	775	780
852 Hz	842	847	857	862
941 Hz	931	936	946	951

Fonte: Própria autora

- **Fstop1 e Fstop2:** representam os limites inferiores e superiores das bandas de rejeição. Nessas faixas, o filtro impõe forte atenuação, garantindo que sinais com essas frequências não interfiram na faixa de interesse.
- **Fpass1 e Fpass2:** delimitam a banda de passagem do filtro. Essa é a faixa onde a frequência central da componente senoidal está contida e deve ser preservada com mínima distorção.

A interface gráfica do `fdatool` facilita bastante a personalização das configurações desejadas, sendo necessário apenas selecionar os parâmetros apropriados e gerar o gráfico correspondente, como mostrado na Figura 2. A imagem exemplifica o caso do filtro projetado para 770 Hz.

Figura 2: Interface gráfica (`fdatool`)



Fonte: Própria autora

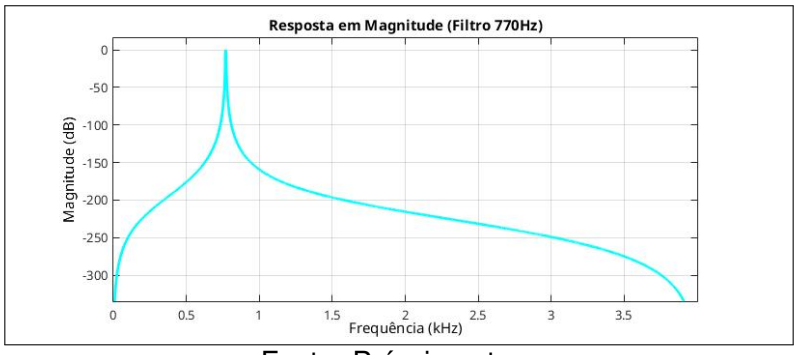
Com esses dados anotados e o conhecimento do funcionamento da ferramenta de simulação, podemos prosseguir para a análise das frequências isoladas.

3.3 Resolução

3.3.1 Magnitude

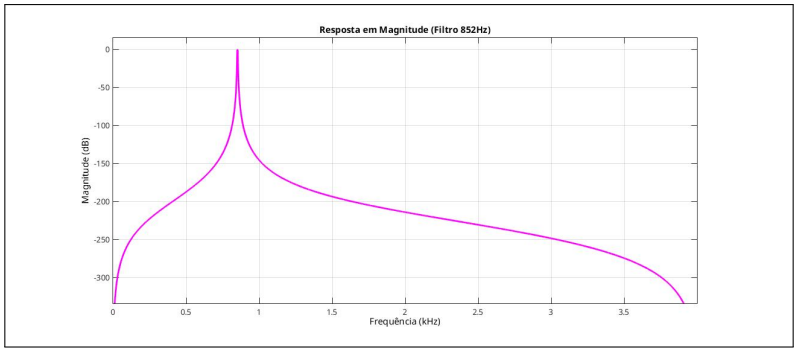
Conforme as Figuras 3, 4 e 5, verificamos a estabilidade dos filtros projetados com os parâmetros selecionados. As curvas de magnitude mostram que os filtros foram corretamente sintonizados para cada frequência-alvo (770Hz, 852Hz e 941Hz), apresentando ganho máximo próximo de 0 dB na frequência central e uma atenuação acentuada fora da banda de interesse. Isso garante que cada filtro atue de forma seletiva, isolando com eficiência a componente desejada e suprimindo as demais frequências.

Figura 3: Resposta em Magnitude (Filtro 770Hz)



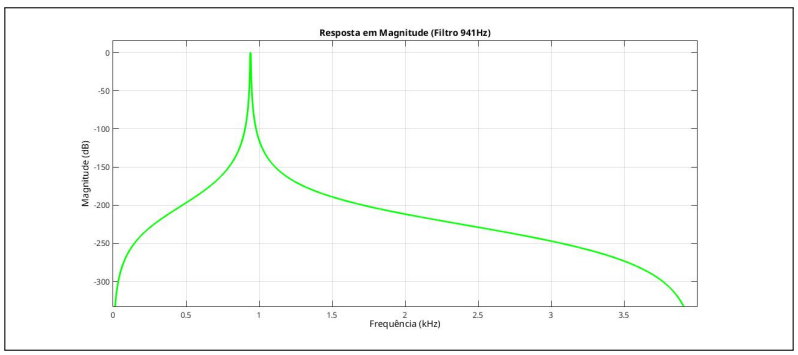
Fonte: Própria autora

Figura 4: Resposta em Magnitude (Filtro 852Hz)



Fonte: Própria autora

Figura 5: Resposta em Magnitude (Filtro 941Hz)



Fonte: Própria autora

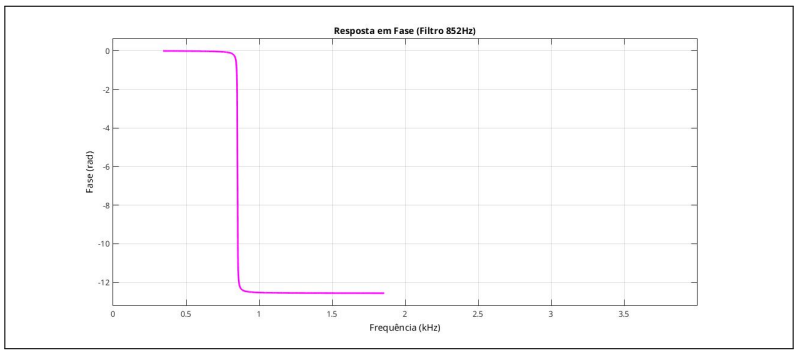
3.3.2 Fase

As Figuras 6, 7 e 8 apresentam a resposta em fase dos filtros projetados para 770Hz, 852Hz e 941Hz, respectivamente. Observa-se que, em todos os casos, a fase inicia em aproximadamente 0 graus e decresce à medida que a frequência se aproxima da frequência central de cada filtro, seguindo o comportamento esperado.

Figura 6: Resposta em Fase (Filtro 770Hz)

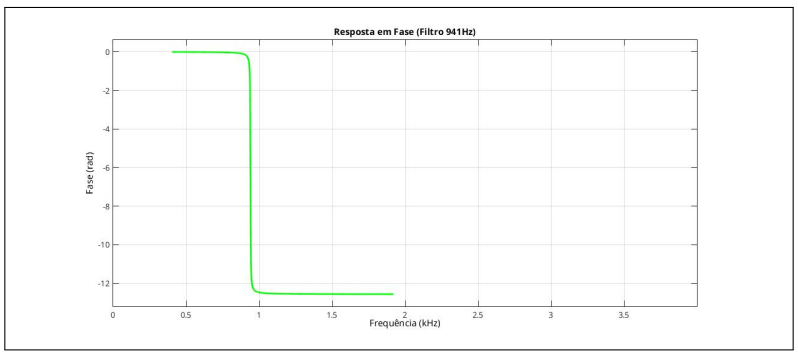
Fonte: Própria autora

Figura 7: Resposta em Fase (Filtro 852Hz)



Fonte: Própria autora

Figura 8: Resposta em Fase (Filtro 941Hz)



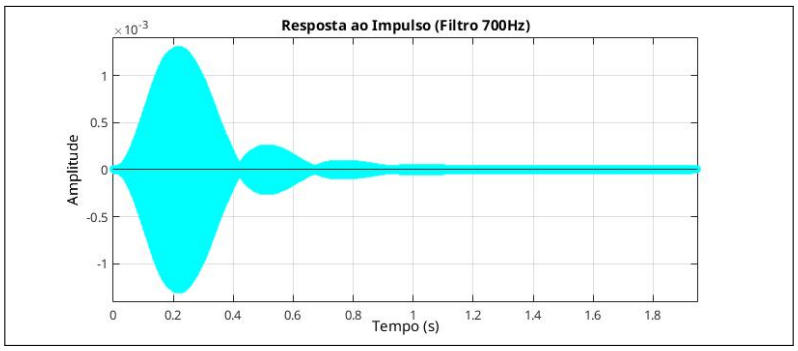
Fonte: Própria autora

3.3.3 Resposta ao Impulso

As Figuras 9, 10 e 11 apresentam a resposta ao impulso dos três filtros projetados. Analisando os gráficos, verificamos que as respostas são idênticas. Isso ocorre porque todos foram projetados

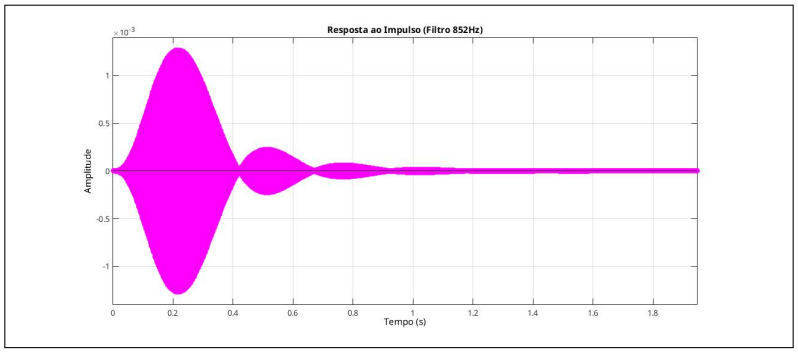
com a mesma estrutura de filtro (Butterworth), variando apenas a frequência central de atuação. A resposta ao impulso reflete a estabilidade e o comportamento característico do tipo passa-faixa.

Figura 9: Resposta ao Impulso (Filtro 770Hz)



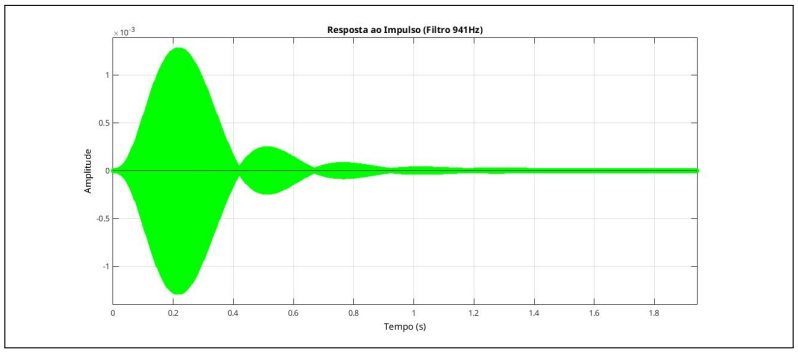
Fonte: Própria autora

Figura 10: Resposta ao Impulso (Filtro 852Hz)



Fonte: Própria autora

Figura 11: Resposta ao Impulso (Filtro 941Hz)

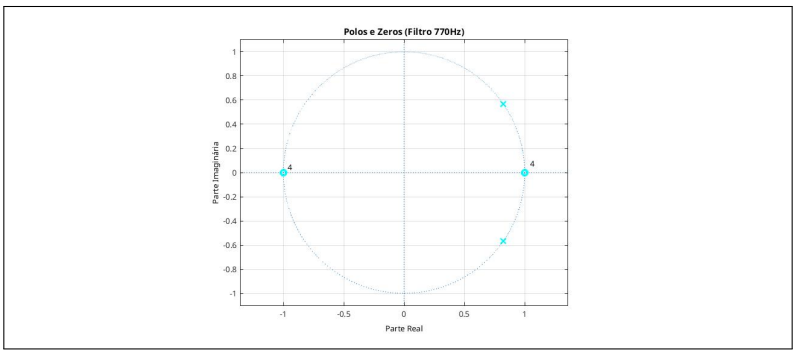


Fonte: Própria autora

3.3.4 Polos e Zeros

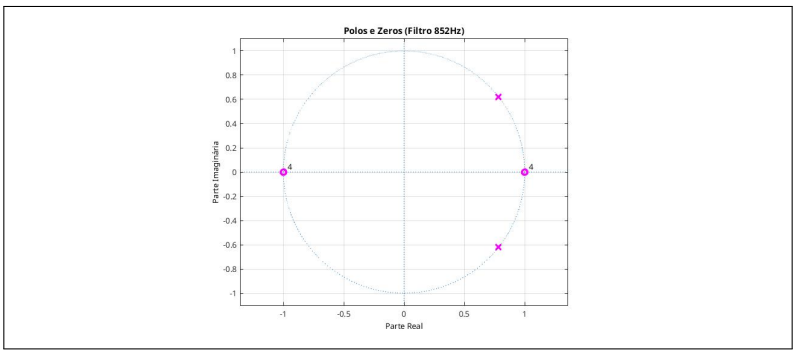
As Figuras 12, 13 e 14 mostram os diagramas de polos e zeros dos filtros digitais projetados. Observa-se que todos os filtros são estáveis, uma vez que todos os polos estão localizados dentro do círculo de raio unitário no plano-z. Essa configuração garante que o sistema responda de forma adequada às frequências de interesse, mantendo a estabilidade e a seletividade do filtro.

Figura 12: Diagrama de Polos e Zeros (Filtro 770Hz)



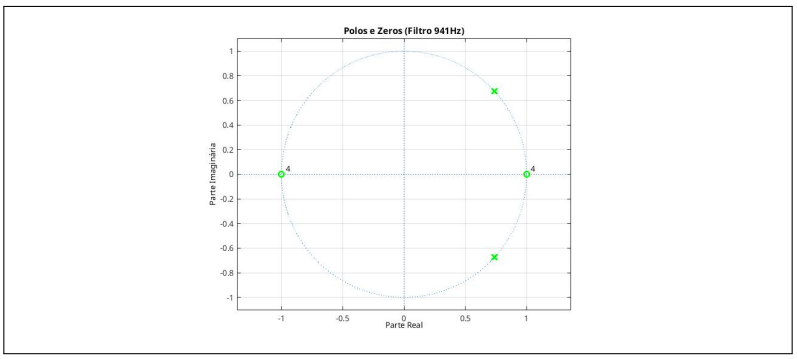
Fonte: Própria autora

Figura 13: Diagrama de Polos e Zeros (Filtro 852Hz)



Fonte: Própria autora

Figura 14: Diagrama de Polos e Zeros (Filtro 941Hz)



Fonte: Própria autora

3.3.5 Script MATLAB

A partir da personalização dos filtros digitais realizada com a ferramenta gráfica `fdatool`, é possível gerar automaticamente o código MATLAB correspondente ao projeto realizado. A seguir, são apresentados os scripts gerados para os filtros centrados em 770 Hz, 852 Hz e 941 Hz, respectivamente.

Filtro 770 Hz

```
1 function Hd = filtro770
2 %FILTRO770 Returns a discrete-time filter object.
```

```
3
4  % MATLAB Code
5  % Generated by MATLAB(R) 8.5 and the DSP System Toolbox 9.0.
6  % Generated on: 15-Jul-2025 15:06:18
7
8  % Butterworth Bandpass filter designed using FDESIGN.BANDPASS.
9
10 % All frequency values are in Hz.
11 Fs = 8000;  % Sampling Frequency
12
13 Fstop1 = 758;         % First Stopband Frequency
14 Fpass1 = 769;         % First Passband Frequency
15 Fpass2 = 771;         % Second Passband Frequency
16 Fstop2 = 782;         % Second Stopband Frequency
17 Astop1 = 40;          % First Stopband Attenuation (dB)
18 Apass  = 1;           % Passband Ripple (dB)
19 Astop2 = 60;          % Second Stopband Attenuation (dB)
20 match  = 'stopband';  % Band to match exactly
21
22 % Construct an FDESIGN object and call its BUTTER method.
23 h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
24                       Astop2, Fs);
25 Hd = design(h, 'butter', 'MatchExactly', match, 'FilterStructure', ...
26             'df2sos');
27
28 % Get the transfer function values.
29 [b, a] = tf(Hd);
30
31 % Convert to a singleton filter.
32 Hd = dfilt.df2(b, a);
33
34
35
36 % [EOF]
```

Filtro 852 Hz

```
1  function Hd = filtro852
2  %FILTRO852 Returns a discrete-time filter object.
3
4  % MATLAB Code
5  % Generated by MATLAB(R) 8.5 and the DSP System Toolbox 9.0.
6  % Generated on: 15-Jul-2025 15:14:50
7
8  % Butterworth Bandpass filter designed using FDESIGN.BANDPASS.
9
10 % All frequency values are in Hz.
11 Fs = 8000;  % Sampling Frequency
12
13 Fstop1 = 840;         % First Stopband Frequency
14 Fpass1 = 851;         % First Passband Frequency
15 Fpass2 = 853;         % Second Passband Frequency
16 Fstop2 = 864;         % Second Stopband Frequency
17 Astop1 = 40;          % First Stopband Attenuation (dB)
18 Apass  = 1;           % Passband Ripple (dB)
19 Astop2 = 60;          % Second Stopband Attenuation (dB)
20 match  = 'stopband';  % Band to match exactly
21
22 % Construct an FDESIGN object and call its BUTTER method.
23 h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
24                       Astop2, Fs);
25 Hd = design(h, 'butter', 'MatchExactly', match);
26
```

```
% Get the transfer function values.
[b, a] = tf(Hd);

% Convert to a singleton filter.
Hd = dfilt.df2(b, a);



% [EOF]
```

Filtro 941 Hz

```
function Hd = filtro941
%FILTRO941 Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 8.5 and the DSP System Toolbox 9.0.
% Generated on: 15-Jul-2025 15:36:43

% Butterworth Bandpass filter designed using FDESIGN.BANDPASS.

% All frequency values are in Hz.
Fs = 8000;  % Sampling Frequency

Fstop1 = 929;         % First Stopband Frequency
Fpass1 = 940;         % First Passband Frequency
Fpass2 = 942;         % Second Passband Frequency
Fstop2 = 953;         % Second Stopband Frequency
Astop1 = 40;          % First Stopband Attenuation (dB)
Apass  = 1;           % Passband Ripple (dB)
Astop2 = 60;          % Second Stopband Attenuation (dB)
match  = 'stopband';  % Band to match exactly

% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);

% Get the transfer function values.
[b, a] = tf(Hd);

% Convert to a singleton filter.
Hd = dfilt.df2(b, a);



% [EOF]
```

3.3.6 Aplicação ao Sinal de Entrada

A primeira etapa consiste em extrair os coeficientes de numerador e denominador de cada filtro. Para isso, utilizamos as funções geradas no `fdatool` (Seção 3.3.5). O código abaixo automatiza essa extração:

```
format long;

filtros = {@filtro770, @filtro852, @filtro941};
frequencias = [770, 852, 941];

for i = 1:length(filtros)
    Hd = filtros{i}();
```

```
    [b, a] = tf(Hd);

    fprintf('\nFiltro Butterworth %d Hz\n\n', frequencias(i));
    fprintf('Numerador (b):\n');
    for j = 1:length(b)
        fprintf('b(%d) = %.18f\n', j, b(j));
    end
    fprintf('\nDenominador (a):\n');
    for j = 1:length(a)
        fprintf('a(%d) = %.18f\n', j, a(j));
    end
end
```

A saída desse script foi registrada e formatada para melhor visualização:

Filtro Butterworth 770 Hz

b = [0.000000000007648975, 0, -0.000000000030595899, 0, 0.000000000045893848, 0, -0.000000000030595899, 0, 0.000000000007648975]

a = [1.000000000000000000, -6.573968478516325931, 20.197703598146055981, -37.435767531546105147, 45.612066895335409811, -37.354424964373507123, 20.110025517173959031, -6.531208646296375875, 0.991336860368820738]

Filtro Butterworth 852 Hz

b = [0.000000000007674623, 0, -0.000000000030698492, 0, 0.000000000046047739, 0, -0.000000000030698492, 0, 0.000000000007674623]

a = [1.000000000000000000, -6.267919673124780822, 18.723854036105780807, -34.153268845102040530, 41.404068891609767888, -34.078996547913362747, 18.642505949254868369, -6.227116450421420701, 0.991329630860964928]

Filtro Butterworth 941 Hz

b = [0.000000000007698251, 0, -0.000000000030793004, 0, 0.000000000046189506, 0, -0.000000000030793004, 0, 0.000000000007698251]

a = [1.000000000000000000, -5.906378913715164103, 17.073282256470818652, -30.558427928155534659, 36.834783842538861620, -30.491922158432025469, 16.999048319437147114, -5.867899772274874692, 0.991322986969082609]

A segunda etapa consiste em montar o sinal de entrada composto pelas três frequências e aplicar os filtros correspondentes. O script a seguir realiza essa tarefa:

```
clc; close all; clear all;

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

% Componentes do sinal de entrada
f1 = 770;
```

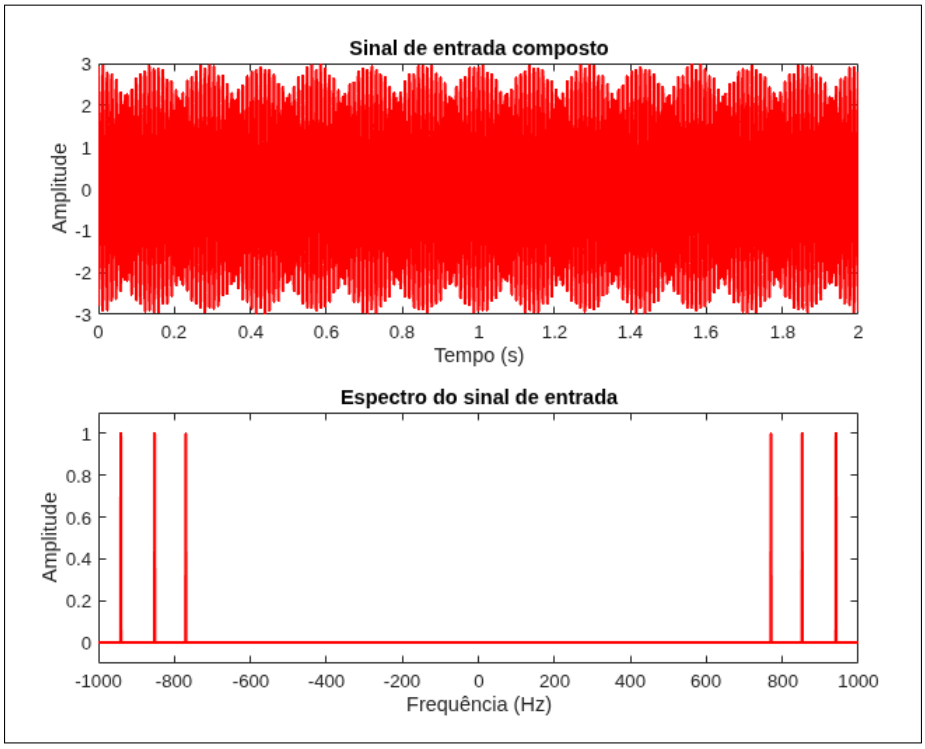
```
f2 = 852;
f3 = 941;

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

% 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;

% Plot do sinal composto
figure;
subplot(2,1,1);
plot(t, s_t, 'r', 'LineWidth', 1.5);
ylabel('Amplitude');
xlabel('Tempo (s)');
title('Sinal de entrada composto');

subplot(2,1,2);
plot(freq, S_f, 'r', 'LineWidth', 1.5);
xlabel('Frequência (Hz)');
ylabel('Amplitude');
title('Espectro do sinal de entrada');
xlim([-1000 1000]);
ylim([-0.1 1.1]);

%% 770 Hz
num_770 = [
    0.000000000007648975; 0;
   -0.000000000030595899; 0;
    0.000000000045893848; 0;
   -0.000000000030595899; 0;
    0.000000000007648975
];

den_770 = [
    1.000000000000000000;
   -6.573968478516325931;
   20.197703598146055981;
  -37.435767531546105147;
   45.612066895335409811;
  -37.354424964373507123;
   20.110025517173959031;
   -6.531208646296375875;
    0.991336860368820738
];

s_770hz = filter(num_770, den_770, s_t);
S_770hz = fftshift(abs(2*fft(s_770hz)/L));

figure;
subplot(2,1,1);
plot(freq, S_f, 'r', 'LineWidth', 1.5);
xlabel('Frequência (Hz)'); ylabel('Amplitude');
title('Espectro do sinal de entrada');
xlim([-1000 1000]); ylim([-0.1 1.1]);

subplot(2,1,2);
plot(freq, S_770hz, 'c', 'LineWidth', 1.5);
xlabel('Frequência (Hz)'); ylabel('Amplitude');
title('Componente filtrada (770 Hz)');
```

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

%% 852 Hz
num_852 = [
    0.000000000007674623; 0;
   -0.000000000030698492; 0;
    0.000000000046047739; 0;
   -0.000000000030698492; 0;
    0.000000000007674623
];

den_852 = [
    1.000000000000000000;
   -6.267919673124780822;
   18.723854036105780807;
  -34.153268845102040530;
   41.404068891609767888;
  -34.078996547913362747;
   18.642505949254868369;
   -6.227116450421420701;
    0.991329630860964928
];

s_852hz = filter(num_852, den_852, s_t);
S_852hz = fftshift(abs(2*fft(s_852hz)/L));

figure;
subplot(2,1,1);
plot(freq, S_f, 'r', 'LineWidth', 1.5);
xlabel('Frequência (Hz)'); ylabel('Amplitude');
title('Espectro do sinal de entrada');
xlim([-1000 1000]); ylim([-0.1 1.1]);

subplot(2,1,2);
plot(freq, S_852hz, 'm', 'LineWidth', 1.5);
xlabel('Frequência (Hz)'); ylabel('Amplitude');
title('Componente filtrada (852 Hz)');
xlim([-1000 1000]); ylim([-0.1 1.1]);

%% 941 Hz
num_941 = [
    0.000000000007698251; 0;
   -0.000000000030793004; 0;
    0.000000000046189506; 0;
   -0.000000000030793004; 0;
    0.000000000007698251
];

den_941 = [
    1.000000000000000000;
   -5.906378913715164103;
   17.073282256470818652;
  -30.558427928155534659;
   36.834783842538861620;
  -30.491922158432025469;
   16.999048319437147114;
   -5.867899772274874692;
    0.991322986969082609
];

s_941hz = filter(num_941, den_941, s_t);
S_941hz = fftshift(abs(2*fft(s_941hz)/L));

```

```
figure;
subplot(2,1,1);
plot(freq, S_f, 'r', 'LineWidth', 1.5);
xlabel('Frequência (Hz)'); ylabel('Amplitude');
title('Espectro do sinal de entrada');
xlim([-1000 1000]); ylim([-0.1 1.1]);

subplot(2,1,2);
plot(freq, S_941hz, 'g', 'LineWidth', 1.5);
xlabel('Frequência (Hz)'); ylabel('Amplitude');
title('Componente filtrada (941 Hz)');
xlim([-1000 1000]); ylim([-0.1 1.1]);
```

Com isso, geramos o sinal de entrada composto, mostrado na Figura 15, juntamente com seu espectro, contendo as três frequências de interesse (770 Hz, 852 Hz e 941 Hz).

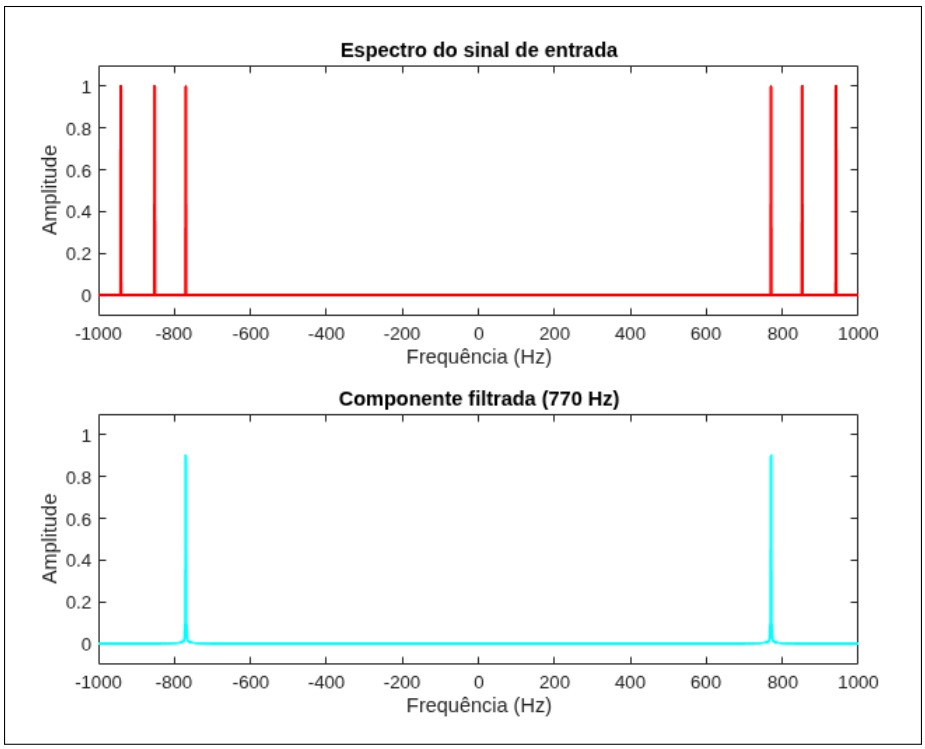
Figura 15: Sinal de Entrada



Fonte: Própria autora

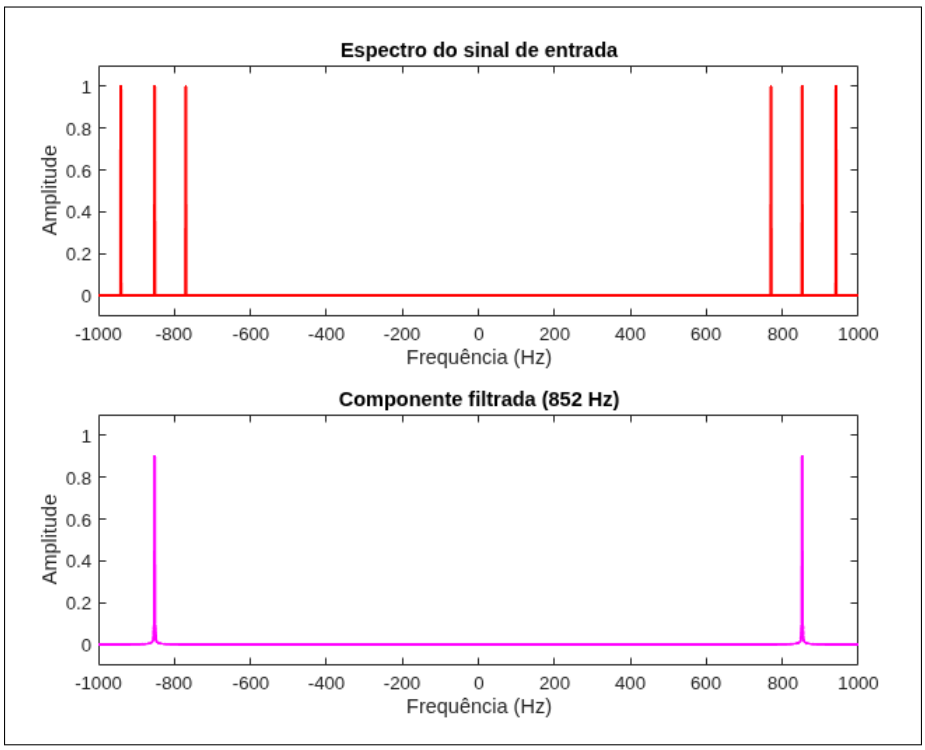
A seguir, aplicamos os filtros projetados individualmente para cada frequência. O resultado da filtragem pode ser observado nas Figuras 16, 17 e 18, onde visualizamos claramente a extração das componentes específicas de cada sinal. O processo foi bem-sucedido em isolar as frequências desejadas, validando a eficiência dos filtros Butterworth implementados.

Figura 16: Extração da Componente 770 Hz



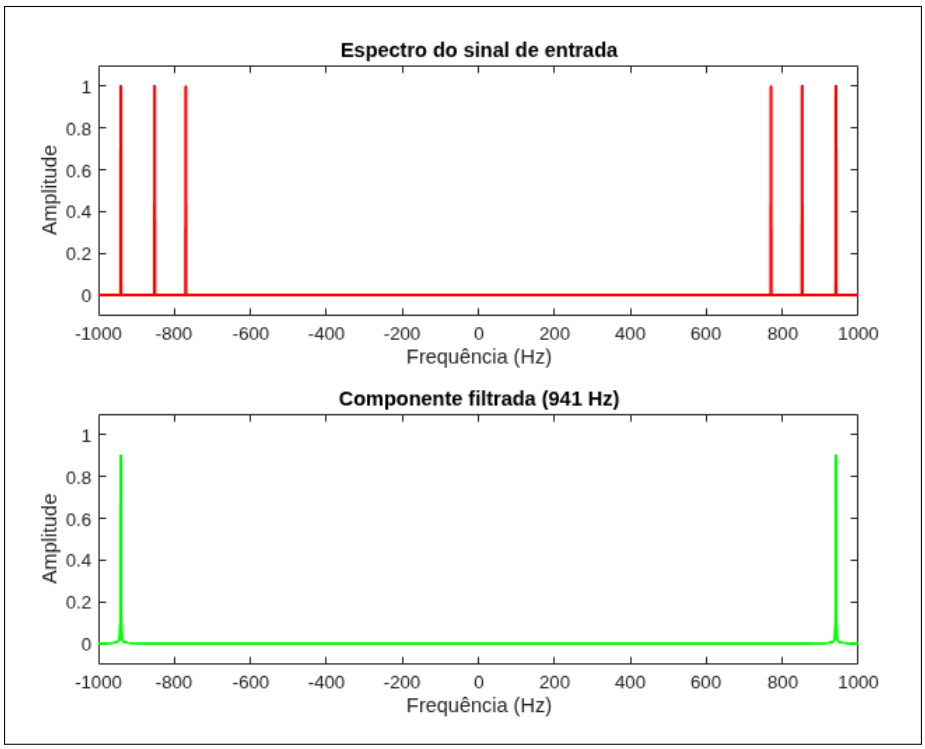
Fonte: Própria autora

Figura 17: Extração da Componente 852 Hz



Fonte: Própria autora

Figura 18: Extração da Componente 941 Hz



Fonte: Própria autora

4 Conclusão

Este trabalho abordou o projeto e a análise de filtros IIR do tipo Butterworth, demonstrando que é possível atender às exigências de banda de passagem e banda de rejeição

As simulações realizadas no MATLAB permitiram validar o desempenho dos filtros projetados, por meio da análise da resposta em frequência, verificação da estabilidade e avaliação das atenuações obtidas.

Com base nos resultados, confirmou-se que o filtro Butterworth é eficaz na separação de componentes espectrais. Sua capacidade de isolar frequências específicas com precisão reforça a relevância dos filtros IIR no processamento digital de sinais.

Referências

[1] Wikipedia. *Bilinear transform*. Acesso em: 17 jul. 2025. 2025. URL: <https://en.wikipedia.org/wiki/Bilinear_transform>.

[2] Wikipedia. *Filtro IIR*. Acesso em: 15 jul. 2025. 2024. URL: <https://pt.wikipedia.org/wiki/Filtro_IIR>.