UNIVERSIDADE ESTADUAL DE MARINGÁ
DEPARTAMENTO DE ENGENHARIA QUÍMICA
ENGENHARIA ELÉTRICA

MATEUS FUGA OSMARIN

DESENVOLVIMENTO DE UMA PLATAFORMA INTERATIVA DE PROCESSAMENTO DE SINAIS

TRABALHO DE CONCLUSÃO DE CURSO

MARINGÁ
2021

MATEUS FUGA OSMARIN

DESENVOLVIMENTO DE UMA PLATAFORMA INTERATIVA DE PROCESSAMENTO DE SINAIS

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Química da Universidade Estadual de Maringá, como requisito parcial para a obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Rafael Krummenauer

Maringá
2021

MATEUS FUGA OSMARIN

DESENVOLVIMENTO DE UMA PLATAFORMA INTERATIVA DE PROCESSAMENTO DE SINAIS

Relatório final apresentado à Universidade Estadual de Maringá como parte das exigências para a obtenção do título de Engenheiro Eletricista.

Maringá, 07 de maio de 2021.

BANCA EXAMINADORA

Prof. Rafael Krummenauer
Orientador

Prof. Abel Fidalgo Alves
Membro titular

Emilio Soitsi Zukeram Júnior
Membro titular

AGRADECIMENTOS

Resumo

OSMARIN, Mateus F. **Desenvolvimento de uma Plataforma Interativa de Processamento de Sinais**. 2021. 39 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia Elétrica. Universidade Estadual de Maringá. Maringá, 2021.

Este trabalho trata do desenvolvimento de uma plataforma interativa de processamento de sinais. A partir de uma interface gráfica, é possível realizar o projeto de filtros digitais tanto do tipo IIR quanto FIR, utilizando técnicas clássicas da literatura. O sistema busca endereçar a dificuldade de visualização no aprendizado do tópico de processamento de sinais, fazendo-se uma ferramenta didática alternativa.

Palavras-chave: Processamento de sinais. Projeto de filtros digitais. Visualização. Python.

Abstract

OSMARIN, Mateus F. **Development of a Signal Processing Interactive Platform**. 2021. 39 p. Undergraduate thesis - Electrical Engeneering. Universidade Estadual de Maringá. Maringá, 2021.

This paper addresses the development of an interactive signal processing platform. By means of a graphical user interface, it is possible to design both IIR and FIR filters, using classic techniques. The system aims to deal with the visualization issues in the learning of signal processing topic, being an alternative didatic tool.

Keywords: Signal processing. Digital filter design. Visualization. Python.

Lista de Figuras

Sumário

Capítulo 1

Introdução

A disciplina de processamento de sinais é uma área ampla que permite entender e tratar matematicamente sinais, transformando-os segundo as necessidades envolvidas na sua aplicação. Majoritariamente, encontra-se no dia-a-dia sinais de natureza contínua no tempo, como a temperatura de um corpo, o som de um instrumento musical ou a velocidade de um automóvel. Ainda, existem sinais de natureza inerentemente discreta no tempo, como é o caso do preço de uma ação, as temperaturas máxima e mínima diárias de uma cidade ou o número diário de novos casos de COVID-19. Entretanto, é importante ressaltar que mesmo sinais de natureza contínua podem ser tratados de forma discreta por meio de procedimentos de amostragem adequados. Dessa forma, o processamento digital de sinais pode ser aplicado a uma infinidade de casos.

Diante da generalidade envolvida, tem-se também um nível de abstração elevado no formalismo matemático que fundamenta a disciplina, o que resulta em dificuldades no aprendizado dessa importante área. Nesse sentido, a utilização de ferramentas como o Filter Designer do MATLAB traz benefícios tanto de produtividade para profissionais da área como facilitam o aprendizado para estudantes do tópico, por abordar de forma visual e intuitiva o projeto de filtros digitais. Utilizando esse tipo de ferramenta, pode-se variar parâmetros de projeto e visualizar o comportamento dos sistemas projetados de forma eficiente e facilitada, com gráficos que exibem as principais características do processo. Contudo, o MATLAB se trata de um software pago e, não obstante, tem-se observado um aumento crescente no interesse pela linguagem de programação Python por estudantes e profissionais da área da ciência e tecnologia, dada a grande quantidade de bibliotecas disponíveis para todo tipo de problemas.

Capítulo 2

Objetivos

2.1 Objetivos gerais

O presente trabalho tem como objetivo o desenvolvimento de um software livre para projeto de filtros digitais utilizando a linguagem de programação Python, construindo uma interface gráfica amigável em GTK para aumento de produtividade e facilitar o aprendizado na área de processamento de sinais, contando com uma revisão de conceitos importantes a servir como referência para utilização do sistema.

2.2 Objetivos específicos

Os objetivos específicos deste trabalho consistem no desenvolvimento de uma aplicação com interface gráfica que

- possibilite projetar filtros FIR e IIR utilizando técnicas clássicas da literatura;
- permita visualizar as principais características dos filtros projetados, como resposta em frequência (amplitude e fase), localização de polos e zeros no plano complexo e resposta ao impulso;
- tenha a opção de exportar o filtro projetado para que seja possível utilizá-lo externamente.

Capítulo 3

Referenciais teóricos

3.1 Sinais

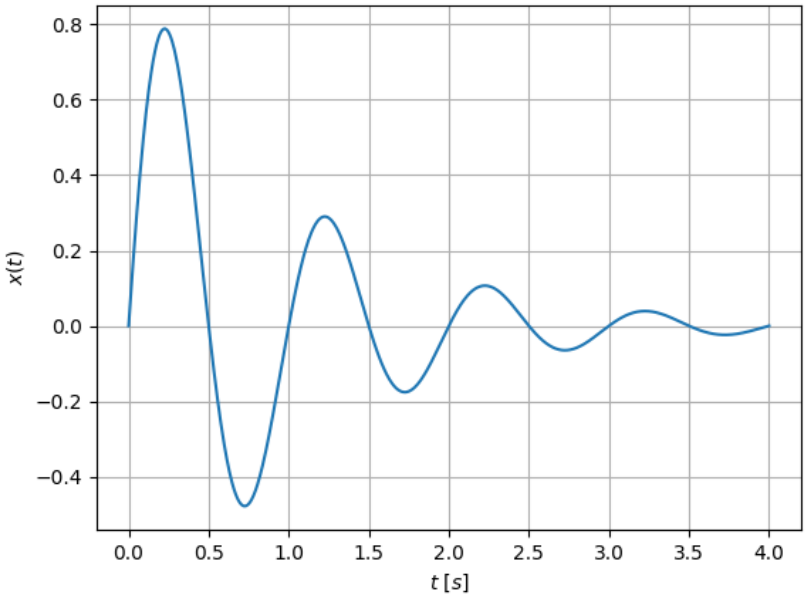
Em essência, um sinal é algo que contém alguma informação, geralmente sobre o estado de um sistema físico [1]. Podemos definir matematicamente um sinal como sendo uma função, de uma ou mais variáveis independentes. A natureza dessas variáveis é diversa, sendo o tempo e o espaço as mais usuais. O som é um exemplo de sinal temporal, enquanto a temperatura em uma sala ao longo do dia se refere a um sinal espacial e temporal. No decorrer deste trabalho, serão considerados sinais de uma variável, sendo tomada como temporal por convenção, mas a teoria é igualmente válida para outros domínios.

Um sinal contínuo no tempo é definido como sendo uma função do tempo definida para todos os instantes:

$$x = x(t), t \in \mathbb{R} \tag{3.1}$$

A figura 3.1 mostra um exemplo de sinal contínuo no tempo.

Figura 3.1: Sinal contínuo no tempo



Fonte: Autoria própria

Um sinal discreto no tempo, por outro lado, é definido como uma sequência numérica:

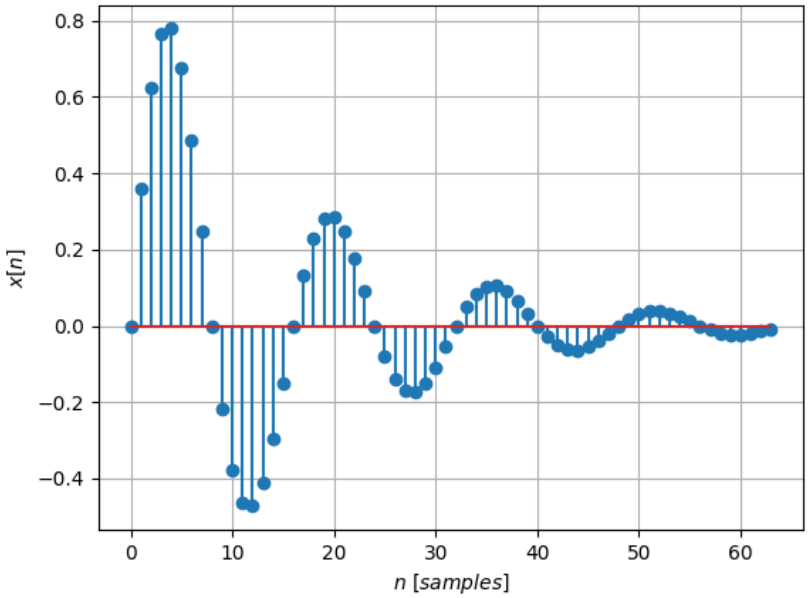
$$x = x[n], n \in \mathbb{Z} \tag{3.2}$$

Para tratar discretamente sinais contínuos no tempo, pode-se amostrar o sinal contínuo em intervalos de tempo igualmente espaçados:

$$x_d[n] = x_a(nT_s) \tag{3.3}$$

onde x_d é a versão discretizada do sinal analógico x_a e T_s representa o período de amostragem. A figura 3.2 mostra o sinal da figura 3.1 discretizado.

Figura 3.2: Sinal discreto no tempo



Fonte: Autoria própria

Para discretizar um sinal adequadamente, sem perda de informação e sem gerar distorção (*aliasing*), o teorema da amostragem de Nyquist-Shannon [2] afirma que a amostragem deve ser feita a uma taxa f_s com pelo menos o dobro da maior frequência no espectro do sinal f_m, ou seja

$$f_s \geq 2f_m \tag{3.4}$$

De forma equivalente, no domínio do tempo, tem-se que o período de amostragem T_s deve ser

$$T_s \leq \frac{1}{2f_m} \tag{3.5}$$

Dentre os sinais existentes, o impulso unitário ou delta de Dirac, é um dos mais importantes. Este sinal é definido, no caso contínuo, como:

$$\delta(t) = \begin{cases} 0 & \text{para} \quad t \neq 0 \\ \infty & \text{para} \quad t = 0 \end{cases} \tag{3.6}$$

satisfazendo a restrição

$$\int_{-\infty}^{+\infty} \delta(t)dt = 1 \tag{3.7}$$

Uma propriedade notória do impulso unitário é que

$$\int_{-\infty}^{+\infty} f(t)\delta(t - t_0)dt = f(t_0) \tag{3.8}$$

sendo conhecida como propriedade da amostragem.

No caso discreto, o impulso unitário é definido como:

$$\delta[n] = \begin{cases} 0 & \text{para} \quad n \neq 0 \\ 1 & \text{para} \quad n = 0 \end{cases} \tag{3.9}$$

A propriedade da amostragem para o caso discreto é expressa como:

$$\sum_{n=-\infty}^{+\infty} x[n]\delta[n - n_0] = x[n_0] \tag{3.10}$$

3.2 Transformadas de Fourier, Laplace e Z

Até aqui, os sinais foram descritos no domínio do tempo. É possível, no entanto, representar sinais no domínio da frequência através das transformações conhecidas como transformada de Fourier, transformada de Laplace e transformada Z. O princípio de partida é a série de Fourier, que possibilita a representação de uma função T-periódica como uma série de senos e cossenos ou, equivalentemente, exponenciais complexas.

A série de Fourier é definida como:

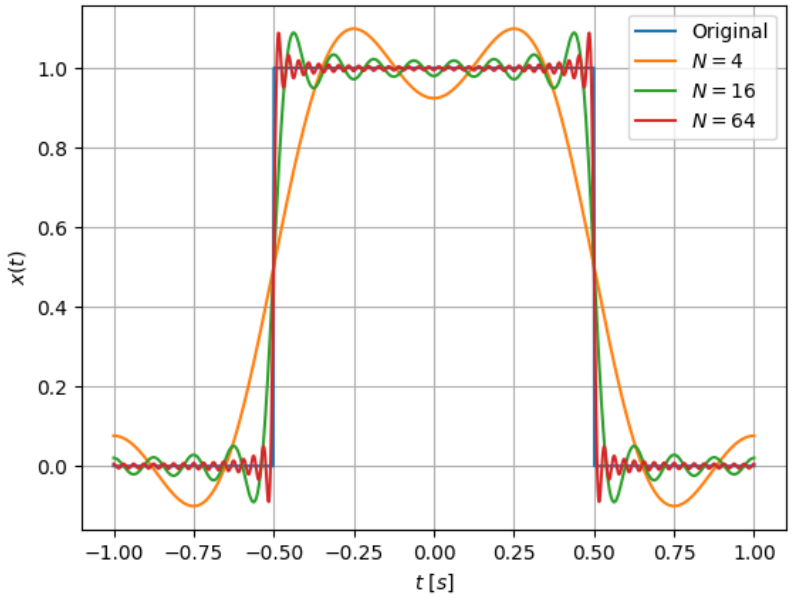
$$x_N(t) = \sum_{n=-N}^{+N} c_n \mathrm{e}^{\mathrm{i}\frac{2\pi}{T}nt} \tag{3.11}$$

onde

$$c_n = \frac{1}{T}\int_T x(t)\mathrm{e}^{-\mathrm{i}\frac{2\pi}{T}nt}dt \tag{3.12}$$

são os coeficientes da série, e N é teoricamente infinito. A figura 3.3 mostra a expansão em série de Fourier do retângulo unitário para diferentes valores de N.

Figura 3.3: Expansão em série de Fourier do retângulo unitário



Fonte: Autoria própria

Para funções não-periódicas, generaliza-se a série de Fourier considerando o período como sendo infinito. No caso limite, tem-se:

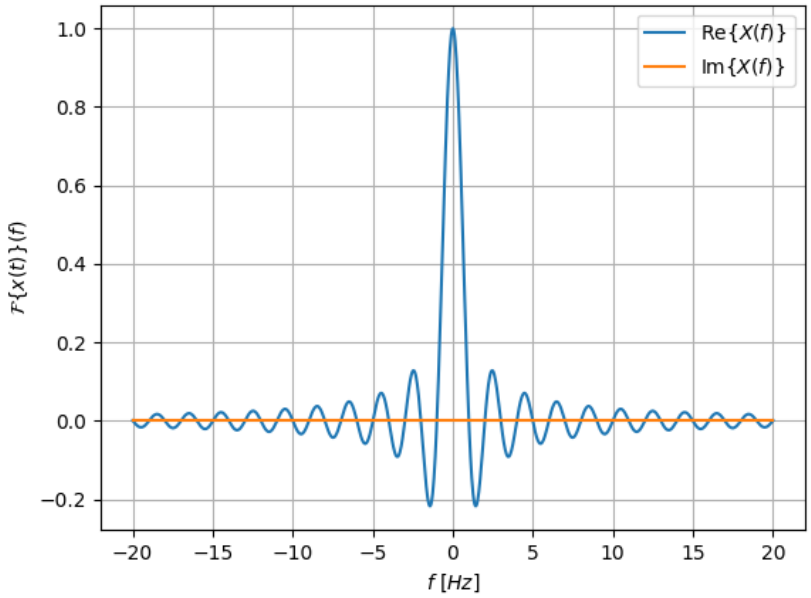
$$x(t) = \int_{-\infty}^{+\infty} \widehat{x}(f) \mathrm{e}^{\mathrm{i}2\pi ft} df \tag{3.13}$$

onde

$$\widehat{x}(f) = \mathcal{F}\{x(t)\}(f) = \int_{-\infty}^{+\infty} x(t) \mathrm{e}^{-\mathrm{i}2\pi ft} dt \tag{3.14}$$

e $\widehat{x}(f)$ é conhecida como a transformada de Fourier de $x(t)$. Segundo esta definição, f tem unidades de $hertz$. A figura 3.4 mostra a transformada de Fourier do retângulo unitário.

Figura 3.4: Transformada de Fourier do retângulo unitário



Fonte: Autoria própria

Ao amostrar um sinal com um período de amostragem T por meio de um pente de Dirac

$$\begin{aligned} x_T(t) = &\quad x(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT) \\ = &\sum_{n=-\infty}^{+\infty} x(nT)\delta(t - nT) \\ = &\sum_{n=-\infty}^{+\infty} x[n]\delta(t - nT) \end{aligned} \tag{3.15}$$

e aplicar a transformada de Fourier, tem-se

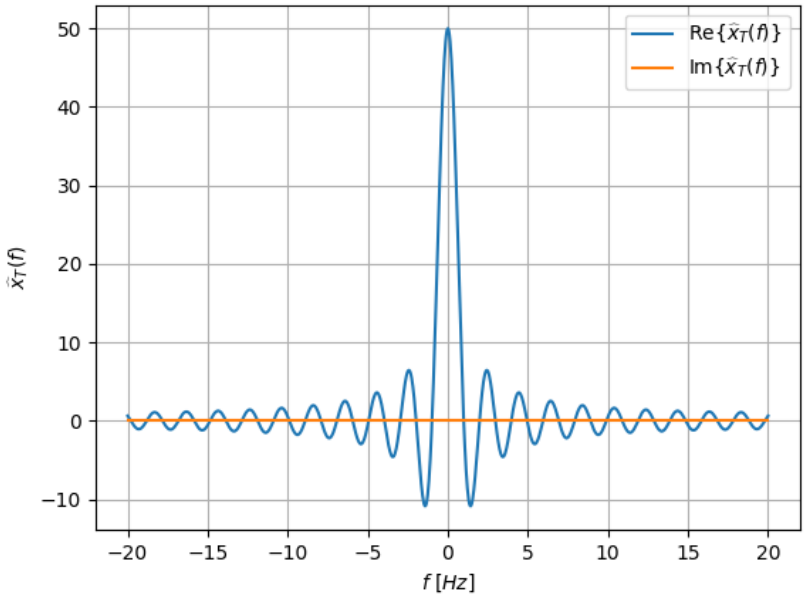
$$\begin{aligned} \widehat{x}_T(f) = &\int_{-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x[n]\delta(t - nT)\mathrm{e}^{-\mathrm{i}2\pi ft}dt \\ = &\sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{+\infty} \delta(t - nT)\mathrm{e}^{-\mathrm{i}2\pi ft}dt \\ = &\sum_{n=-\infty}^{+\infty} x[n]\mathrm{e}^{-\mathrm{i}2\pi fnT} \end{aligned} \tag{3.16}$$

ao que se obtém a chamada transformada de tempo discreto de Fourier. Para obter o sinal original, a transformada de tempo discreto de Fourier inversa é:

$$x[n] = T \int_{1/T} \widehat{x}_T(f)\mathrm{e}^{\mathrm{i}2\pi fnT}df \tag{3.17}$$

A figura 3.5 mostra a transformada de tempo discreto de Fourier do retângulo unitário.

Figura 3.5: Transformada de tempo discreto de Fourier do retângulo unitário



Fonte: Autoria própria

Nesse caso, a discretização acontece no domínio do tempo, mas a frequência ainda é contínua. Discretizando a transformada de tempo discreto de Fourier para N amostras de um ciclo, obtém-se:

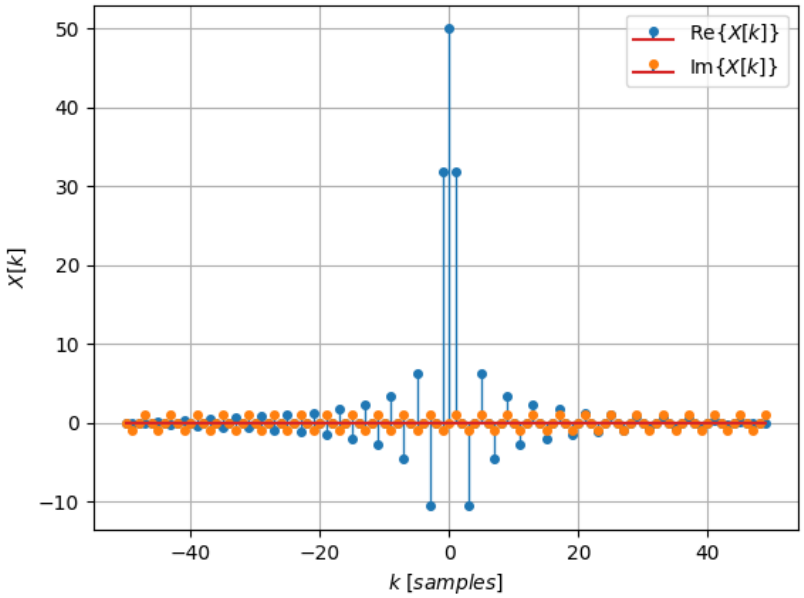
$$\begin{aligned} X[k] = \quad & \widehat{x}_T\left(\frac{k}{NT}\right) \\ & = \sum_{n=-\infty}^{+\infty} x[n] \mathrm{e}^{-\mathrm{i}2\pi \frac{k}{NT} nT} \\ & = \sum_{n=-\infty}^{+\infty} x[n] \mathrm{e}^{-\mathrm{i}2\pi \frac{k}{N} n} \end{aligned} \tag{3.18}$$

o que define a transformada discreta de Fourier. Para obter o sinal original, a transformada discreta de Fourier inversa é:

$$x[n] = \frac{1}{N} \sum_{k=-\infty}^{+\infty} X[k] \mathrm{e}^{\mathrm{i}2\pi \frac{k}{N} n} \tag{3.19}$$

A figura 3.6 mostra a transformada discreta de Fourier do retângulo unitário.

Figura 3.6: Transformada discreta de Fourier do retângulo unitário



Fonte: Autoria própria

A transformada de Fourier pode ainda ser generalizada ao se considerar uma frequência complexa. O resultado é a transformada de Laplace:

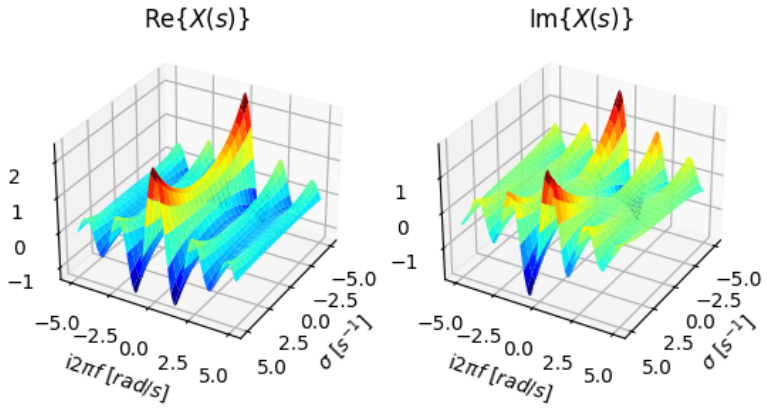
$$X(s) = \mathcal{L}\{x(t)\}(s) = \int_{-\infty}^{+\infty} x(t)\mathrm{e}^{-st}dt \tag{3.20}$$

onde

$$s = \sigma + \mathrm{i}\omega = \sigma + \mathrm{i}2\pi f \tag{3.21}$$

A figura 3.7 mostra a transformada de Laplace do retângulo unitário.

Figura 3.7: Transformada de Laplace do retângulo unitário



Fonte: Autoria própria

Nota-se também que a transformada de Laplace pode ser vista como sendo a transformada de Fourier de $x(t)\mathrm{e}^{-\sigma t}$. Para recuperar o sinal original, a transformada de Laplace inversa é:

$$x(t) = \mathcal{L}^{-1}\{X(s)\}(t) = \frac{1}{\mathrm{i}2\pi} \lim_{T\to\infty} \int_{\gamma-\mathrm{i}T}^{\gamma+\mathrm{i}T} X(s)\mathrm{e}^{st} ds \tag{3.22}$$

onde γ é um número real tal que o caminho de integração esteja na região de convergência de $F(s)$.

Novamente, amostrando um sinal por meio de um pente de Dirac (3.15) e aplicando a transformada de Laplace, tem-se

$$\begin{aligned} X_T(s) = & \int_{-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x[n]\delta(t-nT)\mathrm{e}^{-st} dt \\ = & \sum_{n=-\infty}^{+\infty} x[n] \int_{-\infty}^{+\infty} \delta(t-nT)\mathrm{e}^{-st} dt \\ = & \sum_{n=-\infty}^{+\infty} x[n]\mathrm{e}^{-snT} \end{aligned} \tag{3.23}$$

da qual, ao tomar $z = \mathrm{e}^{sT}$, obtém-se a transformada Z de um sinal discreto:

$$X(z) = \mathcal{Z}\{x[n]\}(z) = \sum_{n=-\infty}^{+\infty} x[n]z^{-n}, z \in \mathbb{C} \tag{3.24}$$

Para recuperar o sinal original, a transformada Z inversa é:

$$x[n] = \mathcal{Z}^{-1}\{X(z)\}[n] = \frac{1}{\mathrm{i}2\pi} \oint_{\mathcal{C}} X(z) z^{n-1} dz \tag{3.25}$$

onde $\mathcal{C}$ é um caminho fechado percorrido no sentido anti-horário, contendo a origem e inteiramente na região de convergência de $X(z)$.

3.3 Convolução

Dados dois sinais, é possível realizar um conjunto de operações sobre os mesmos. Dentre estas, tem-se as bem conhecidas operações de soma, subtração, multiplicação, divisão, derivação e integração. Em processamento de sinais, uma operação conhecida como convolução é muito utilizada, sendo definida por:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau) g(t - \tau) d\tau \tag{3.26}$$

para o caso contínuo e

$$(f * g)[n] = \sum_{k=-\infty}^{+\infty} f[k] g[n - k] \tag{3.27}$$

para o caso discreto.

Uma propriedade importante da convolução é

$$x(t) * \delta(t) = \int_{-\infty}^{+\infty} x(\tau) \delta(t - \tau) d\tau = x(t) \tag{3.28}$$

para o caso contínuo e

$$x[n] * \delta[n] = \sum_{k=-\infty}^{+\infty} x[k] \delta[n - k] = x[n] \tag{3.29}$$

para o caso discreto. Esta propriedade exprime o fato de que o impulso unitário corresponde a uma identidade para a convolução.

Ainda, outra propriedade importante da convolução está em sua relação com as transformadas de Fourier, Laplace e Z. Tomando a transformada de Laplace como exemplo,

tem-se:

$$\begin{aligned}
\mathcal{L}\{(f*g)(t)\}(s) &= \int_{t=-\infty}^{+\infty}\left[\int_{\tau=-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau\right]\mathrm{e}^{-st}dt \\
&= \int_{\tau=-\infty}^{+\infty}\int_{t=-\infty}^{+\infty} f(\tau)g(t-\tau)\mathrm{e}^{-st}dtd\tau \\
&= \int_{\tau=-\infty}^{+\infty} f(\tau)\int_{t=-\infty}^{+\infty} g(t-\tau)\mathrm{e}^{-st}dtd\tau \\
&= \int_{\tau=-\infty}^{+\infty} f(\tau)\int_{u=-\infty}^{+\infty} g(u)\mathrm{e}^{-s(u+\tau)}dud\tau \\
&= \int_{\tau=-\infty}^{+\infty} f(\tau)\mathrm{e}^{-s\tau}d\tau\int_{u=-\infty}^{+\infty} g(u)\mathrm{e}^{-su}du \\
&= \mathcal{L}\{f(t)\}(s)\mathcal{L}\{g(t)\}(s)
\end{aligned} \tag{3.30}$$

ou seja, no domínio da frequência a operação de convolução corresponde à multiplicação. A equação 3.30 é conhecida como teorema do convolução.

3.4 Sistemas lineares e invariantes no tempo

Um sistema é essencialmente algo que mapeia um sinal de entrada em um sinal de saída [3], ou seja:

$$y(t) = \mathcal{H}\{x(t)\} \tag{3.31}$$

para um sistema analógico e

$$y[n] = \mathcal{H}\{x[n]\} \tag{3.32}$$

para um sistema discreto, onde $\mathcal{H}\{\cdot\}$ denota o sistema.

Dentre todos os sistemas possíveis, tem-se classes de sistemas que são interessantes pelas propriedades que apresentam. Um sistema linear é um sistema que tem a propriedade:

$$\mathcal{H}\{ax_1(t) + bx_2(t)\} = a\mathcal{H}\{x_1(t)\} + b\mathcal{H}\{x_2(t)\} \tag{3.33}$$

para o caso contínuo e

$$\mathcal{H}\{ax_1[n] + bx_2[n]\} = a\mathcal{H}\{x_1[n]\} + b\mathcal{H}\{x_2[n]\} \tag{3.34}$$

para o caso discreto. Sistemas com essa propriedade tem a característica de, por exemplo, ao se dobrar a amplitude da entrada, dobrar-se também a amplitude da saída.

Um sistema invariante no tempo é um sistema que tem a propriedade:

$$\mathcal{H}\{x(t)\} = y(t) \Leftrightarrow \mathcal{H}\{x(t-t_0)\} = y(t-t_0) \tag{3.35}$$

para o caso contínuo e

$$\mathcal{H}\{x[n]\} = y[n] \Leftrightarrow \mathcal{H}\{x[n-n_0]\} = y[n-n_0] \tag{3.36}$$

para o caso discreto. Sistemas com essa propriedade tem a característica de ao se atrasar a entrada, atrasar-se também a saída pela mesma quantidade.

Sistemas que são lineares e invariantes no tempo podem ser tratados matemáticamente de forma generalizada como se segue. Sendo $\mathcal{H}\{\cdot\}$ um sistema linear e invariante no tempo, e como, de acordo com as equações 3.28 e 3.29, sinais podem ser representados por uma convolução com o impulso unitário, tem-se

$$\begin{aligned} y(t) = &\quad \mathcal{H}\{x(t) * \delta(t)\} = \mathcal{H}\{\int_{-\infty}^{+\infty} x(\tau)\delta(t-\tau)d\tau\} \\ = &\quad \int_{-\infty}^{+\infty} x(\tau)\mathcal{H}\{\delta(t-\tau)\}d\tau \\ = &\quad \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \\ = &\quad x(t) * h(t) \end{aligned} \tag{3.37}$$

para o caso contínuo e

$$\begin{aligned} y[n] = &\quad \mathcal{H}\{x[n] * \delta[n]\} = \mathcal{H}\{\sum_{k=-\infty}^{+\infty} x[k]\delta[n-k]\} \\ = &\quad \sum_{k=-\infty}^{+\infty} x[k]\mathcal{H}\{\delta[n-k]\} \\ = &\quad \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \\ = &\quad x[n] * h[n] \end{aligned} \tag{3.38}$$

para o caso discreto. Nas equações 3.37 e 3.38, $h(t)$ e $h[n]$ representam a resposta do sistema a um impulso unitário. Assim, um sistema linear e invariante no tempo pode ser descrito totalmente por sua resposta ao impulso.

No domínio da frequência, o teorema da convolução 3.30 fornece:

$$Y(s) = \mathcal{L}\{x(t) * h(t)\} = X(s)H(s) \tag{3.39}$$

para um sistema analógico e

$$Y(z) = \mathcal{Z}\{x[n] * h[n]\} = X(z)H(z) \tag{3.40}$$

para um sistema discreto. Chama-se $H(s)$ e $H(z)$ a função de transferência do sistema $\mathcal{H}\{\cdot\}$.

3.5 Filtros digitais

Em geral, um filtro digital é um sistema discreto cujo objetivo é destacar ou reduzir determinadas características de um sinal. Neste trabalho serão abordados filtros digitais lineares e invariantes no tempo.

Filtros não-recursivos são caracterizados por uma equação de diferenças do tipo:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] \tag{3.41}$$

onde os coeficientes b_k estão diretamente relacionados à resposta ao impulso do sistema, ou seja, $b_k = h[k]$. Aplicando a transformada Z, tem-se que a função de transferência do sistema é dada por:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{M} b_k z^{-k} \tag{3.42}$$

Devido ao fato de a resposta ao impulso deste sistema ser de duração finita, filtros não-recursivos são também conhecidos como filtros de resposta ao impulso finito (FIR, do inglês *finite impulse response*).

Filtros recursivos, por outro lado, são caracterizados por uma equação de diferenças do tipo:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k] \tag{3.43}$$

Aplicando a transformada Z, tem-se

$$Y(z) = \sum_{k=0}^{M} b_k X(z) z^{-k} - \sum_{k=1}^{N} a_k Y(z) z^{-k} \tag{3.44}$$

de onde, ao se rearranjar, obtem-se

$$Y(z)\left(1 + \sum_{k=1}^{N} a_k z^{-k}\right) = X(z)\left(\sum_{k=0}^{M} b_k z^{-k}\right) \tag{3.45}$$

ou seja,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{3.46}$$

é a função de transferência do sistema. Filtros recursivos, pelo fato de que a saída do sistema depende de saídas passadas, apresentam uma resposta ao impulso de duração infinita, dessa forma são também conhecidos como filtros de resposta ao impulso infinito (IIR, do inglês *infinite impulse response*).

Neste trabalho serão abordados projetos de filtros digitais por métodos de otimização como *Least Squares* [4], o algoritmo de Remez [5] e a transformação bilinear [3] aplicada aos clássicos filtros de Butterworth, Chebyshev, Cauer e Bessel.

Capítulo 4

Metodologia

O principal objetivo deste trabalho foi o desenvolvimento de uma aplicação com interface gráfica para projeto de filtros digitais. Neste capítulo, serão apresentados detalhes sobre a elaboração do software.

A ferramenta foi desenvolvida em Python 3 [6], utilizando as bibliotecas GTK [7, 8], para interface gráfica, NumPy [9] para computação numérica eficiente, SciPy [10] para projeto dos filtros e Matplotlib [11] para visualização de dados. O código foi escrito totalmente pelo autor, de forma modular, facilitando modificações e a inclusão de novas funcionalidades, estando disponível publicamente no Github [12]. Instruções de instalação estão inclusas no README do repositório.

O programa foi estruturado na seguinte hierarquia:

- `main.py`, é o módulo principal responsável por montar toda a aplicação;
- `fdatool/utils.py`, apresenta utilitários para conversão entre diferentes unidades de medida de frequência (Hz, kHz, MHz e GHz) e amplitude (dB, V e W), bem como uma classe representado eventos que podem ocorrer durante a execução do programa, permitindo o registro de callbacks;
- `fdatool/widgets/inputs.py`, constrói os componentes de entrada de dados da aplicação, adicionando controle de estado aos mesmos;
- `fdatool/widgets/specs.py`, trata dos componentes de especificações de frequência e amplitude;
- `fdatool/widgets/common.py`, contém componentes gráficos recorrentes como botões e componente para seleção de ordem dos filtros;
- `fdatool/widgets/figures.py`, onde são construídos os gráficos da aplicação;
- `fdatool/filters/firls.py`, este módulo implementa os filtros FIR utilizando o método Least Squares;

- `fdatool/filters/remez.py`, cuja função é o projeto de filtros FIR pelo algoritmo de Remez;
- `fdatool/filters/iir.py`, contém os filtros IIR clássicos que são discretizados pela transformação bilinear;
- `fdatool/filters/factory.py`, permite a obtenção de um dos métodos de projeto por meio de uma fábrica [13].

O software foi escrito levando em consideração padrões de projeto [13] como *decorator*, *observer*, *state* e *factory method*, sendo adaptados pelo autor de modo a harmonizar com o ecossistema Python.

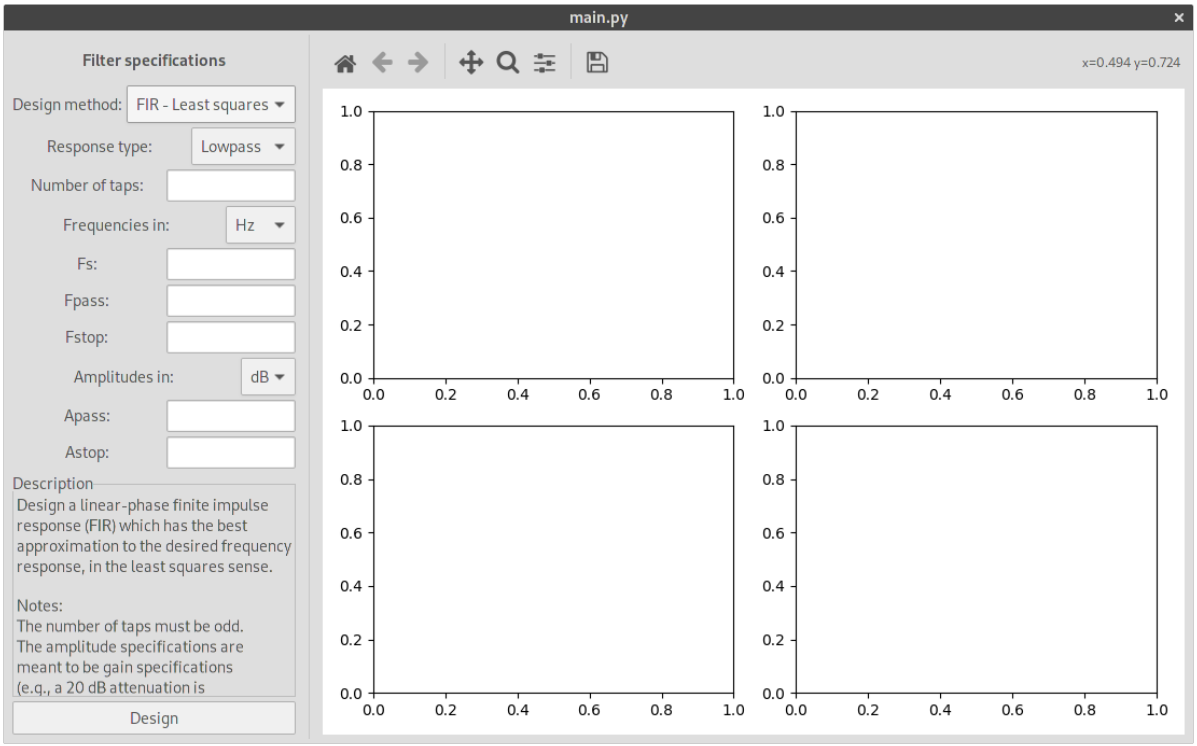
Capítulo 5

Resultados

Neste capítulo será apresentada a aplicação desenvolvida, bem como sua forma de uso. Ainda, será realizado o projeto de alguns filtros utilizando a ferramenta construída. Os filtros terão frequência de amostragem de 44.1 kHz, frequência típica de áudio digital com qualidade de CD.

A tela inicial do programa é mostrada na figura 5.1. O sistema consiste de duas seções: as especificações do filtro a ser projetado e os gráficos que mostram os resultados do processo.

Figura 5.1: Tela inicial da ferramenta desenvolvida



Fonte: Autoria própria

A utilização do sistema começa pela escolha de um método de projeto, sendo que cada método possui características próprias e até mesmo a interface do sistema muda de acordo para acomodar cada metodologia. Ainda, cada método conta com uma breve descrição de seu funcionamento e observações a respeito da forma de utilização. Os método de projeto de filtros FIR utilizando o Least Squares e o algoritmo de Remez apresentam uma descrição bastante semelhante, porém diferem quanto ao seu princípio de funcionamento: enquanto o Least Squares realiza o projeto a partir da minimização do erro quadrático médio [4], o algoritmo de Remez minimiza o erro máximo [5].

Em seguida, faz-se necessária a escolha de um tipo de resposta para o filtro: passa-baixas, passa-altas, passa-faixa ou rejeita-faixa. Dependendo do tipo selecionado, serão exibidas entradas correspondentes para a correta descrição do filtro desejado. Na sequência, deve-se então fazer a especificação em frequência e em amplitude, de acordo com as unidades escolhidas.

A partir de todas as especificações, ao clicar no botão *Design*, o filtro será projetado e os resultados exibidos nos quatro gráficos que apresentam a resposta em frequência (amplitude e fase), polos e zeros da função de transferência e a resposta ao impulso do sistema.

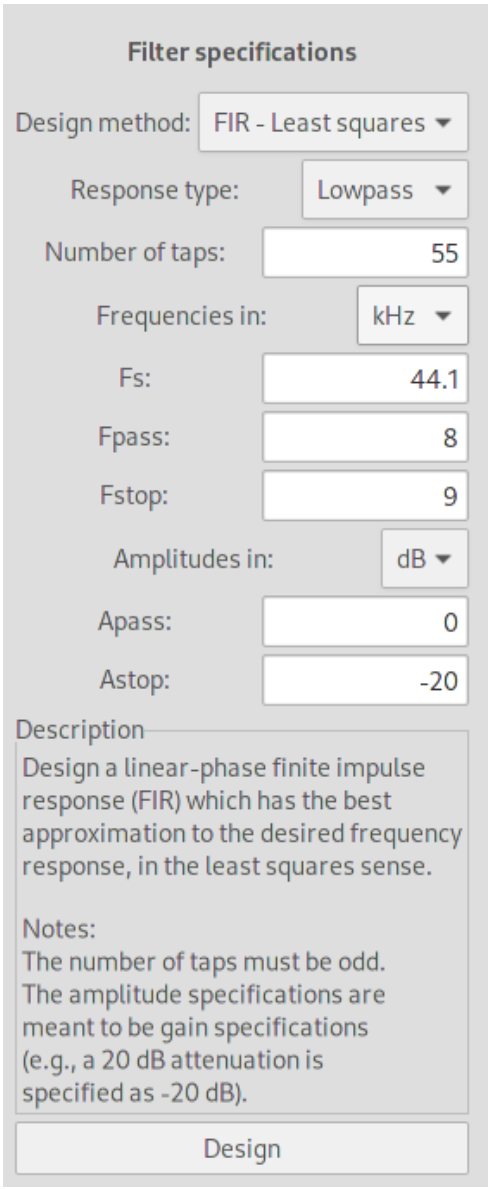
Com um filtro projetado também é habilitada uma opção para exportação do filtro utilizando Pickle [14], que permite serializar um objeto do Python em um arquivo que pode ser posteriormente carregado e usado em qualquer aplicação Python. O objeto exportado trata-se de uma instância da classe `scipy.signal.dlti` [10], o que permite a filtragem de sinais e a obtenção das diversas características do filtro, tais como sua função de transferência, resposta em frequência e resposta ao impulso.

Nas próximas seções serão detalhados cada um dos métodos de projeto de filtros digitais oferecidos pela plataforma, mostrando a forma pela qual as especificações devem ser feitas, bem como os resultados obtidos. Cada filtro projetado será também exportado no formato `.pickle` e utilizado para efetivamente filtrar alguns sinais sintetizados programaticamente.

5.1 Filtros FIR pelo método Least Squares

Para desenvolver um filtro passa-baixas com 55 taps (coeficientes) cuja banda passante vai até 8 kHz, tendo ganho unitário (0 dB), e a banda rejeitada iniciando em 9 kHz, com atenuação de 20 dB (um ganho de -20 dB), a especificação do filtro seria como mostra a figura 5.2.

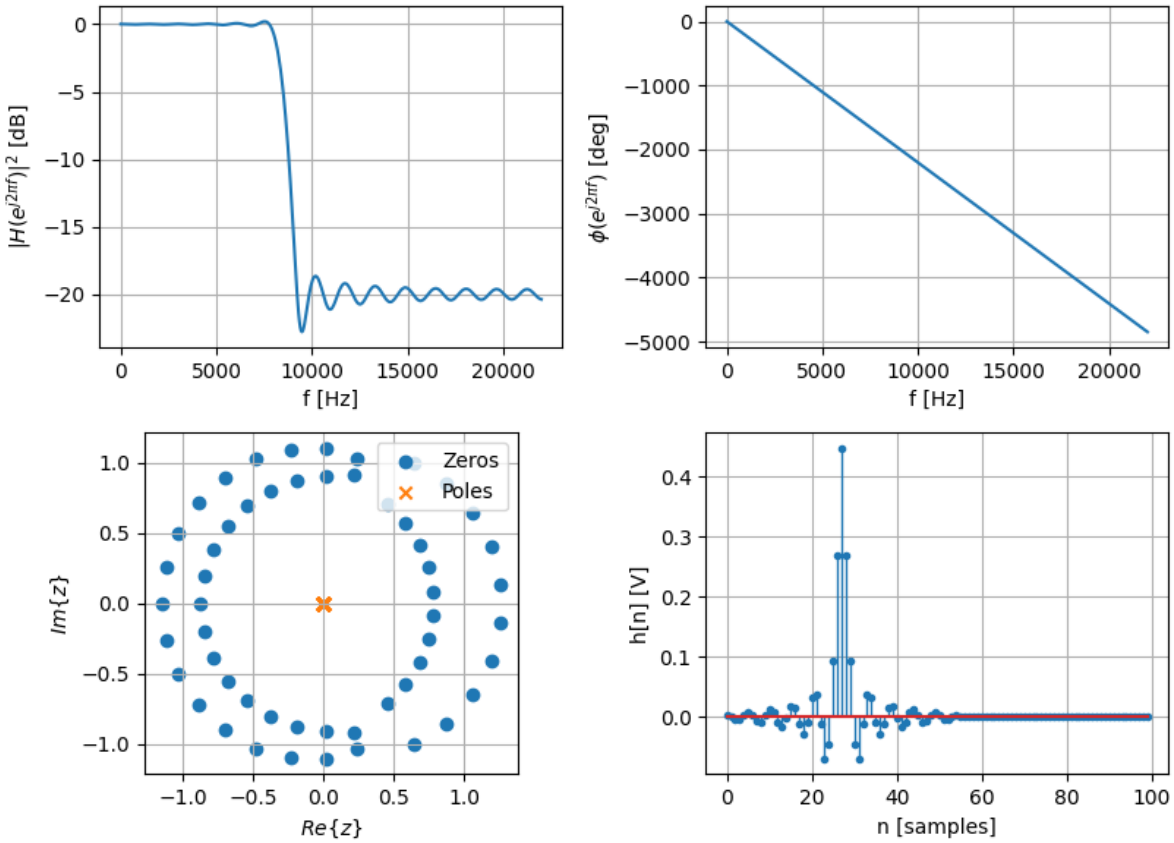
Figura 5.2: Especificação de um filtro passa-baixas usando Least Squares



Fonte: Autoria Própria

Os resultados obtidos para estas especificações podem ser visualizados na figura 5.3

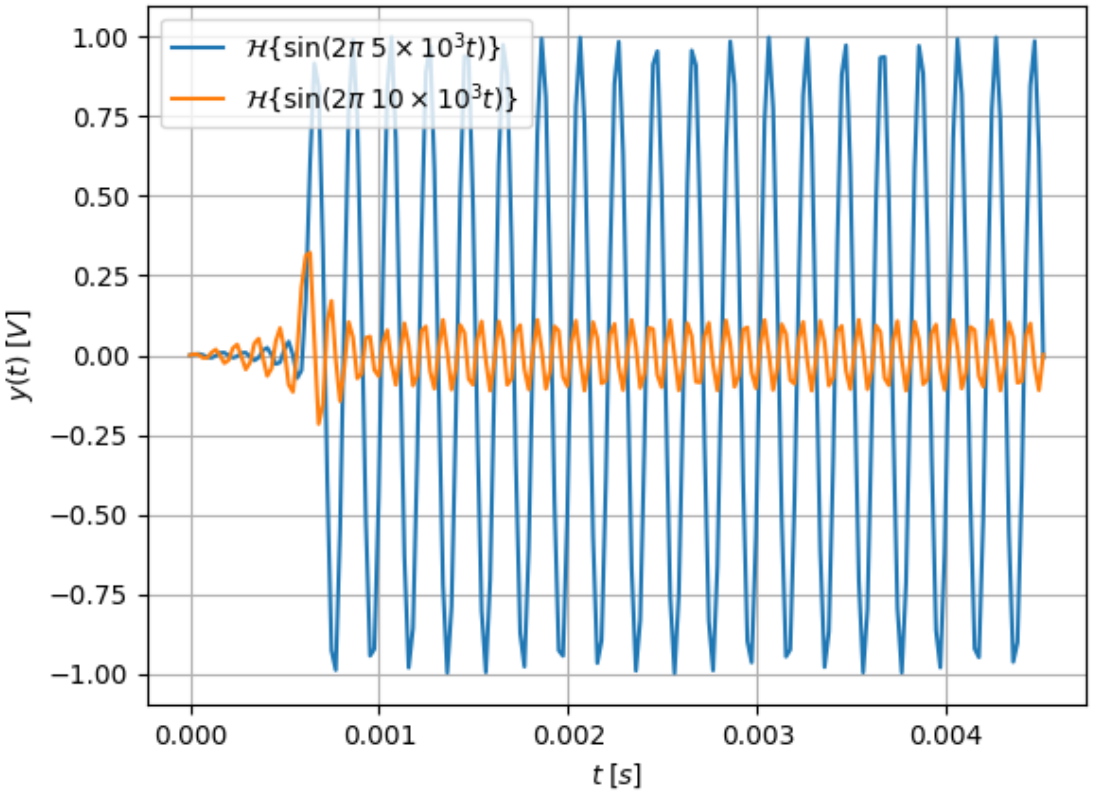
Figura 5.3: Resultados do filtro passa-baixas utilizando Least Squares



Fonte: Autoria Própria

Exportando o filtro e aplicando-o a um tom de 5 kHz e um de 10kHz, pode-se observar o efeito da filtragem na figura 5.4.

Figura 5.4: Aplicação do filtro passa-baixas a dois tons



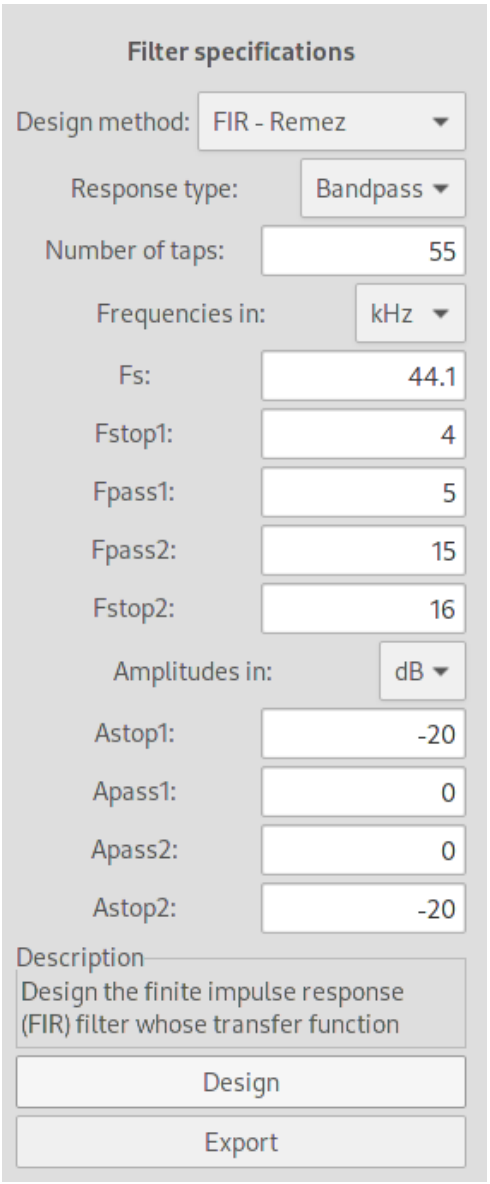
Fonte: Autoria Própria

Pode-se perceber que as características desejadas no filtro foram implementadas com sucesso observando sua resposta em frequência na figura 5.3. Quanto aos dois tons sintetizados e filtrados, nota-se que o tom de 5 kHz passa praticamente sem sofrer atenuações, enquanto o tom de 10 kHz é reduzido drasticamente. Também é possível notar um atraso na saída do filtro, o que é decorrente do tamanho do mesmo (55 coeficientes), sendo também esperado quando se leva em conta a fase da resposta em frequência, que apresenta uma inclinação elevada.

5.2 Filtros FIR pelo algoritmo de Remez

Supondo que se deseje projetar um filtro passa-faixa com 55 taps, cuja banda passante vai de 5 kHz a 15 kHz, com ganho unitário (0 dB), e bandas rejeitadas com 1 kHz de transição e atenuações de 20 dB (ganho de -20 dB), então a especificação do filtro deve ser feita de acordo com a figura 5.5.

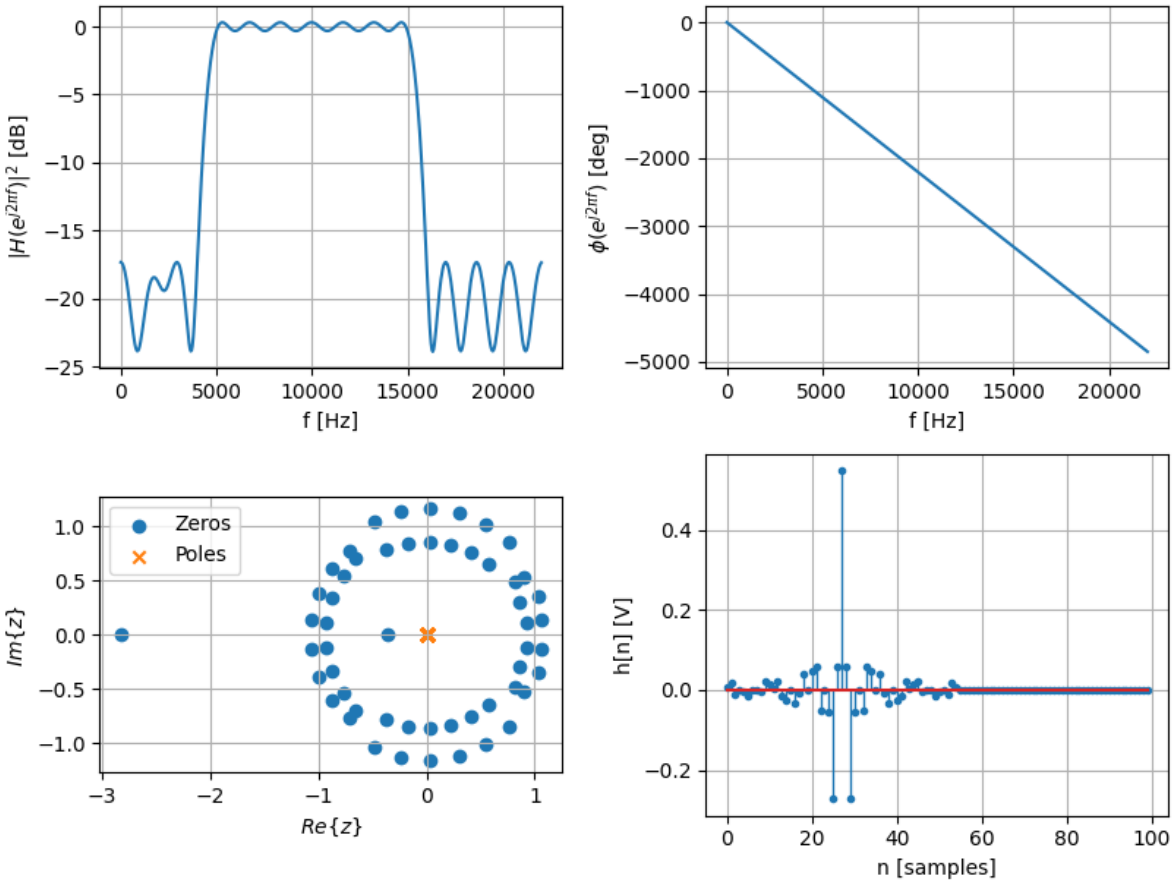
Figura 5.5: Especificação de um filtro passa-faixa usando o algoritmo de Remez



Fonte: Autoria Própria

Para este filtro, os resultados obtidos podem ser vistos na figura 5.6.

Figura 5.6: Resultados do filtro passa-faixa utilizando o algoritmo de Remez

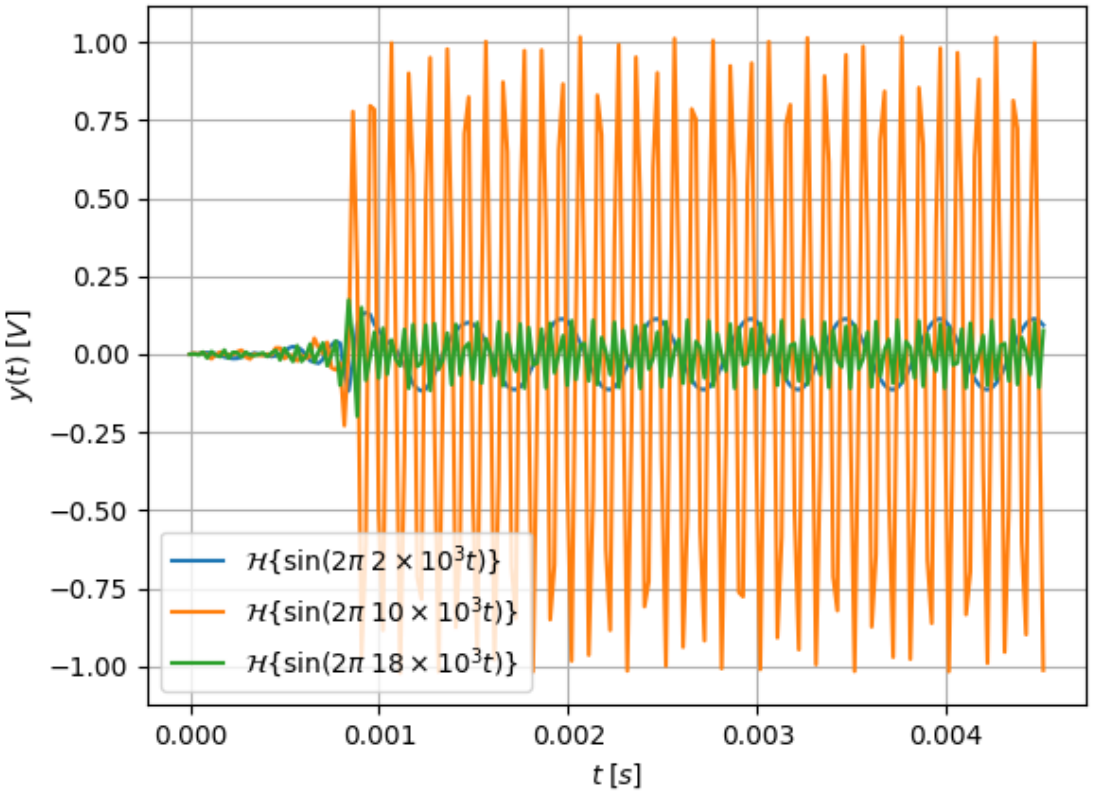


Fonte: Autoria Própria

Aqui, fica evidenciado que o filtro projetado está de acordo com as especificações propostas, sendo possível observar uma característica específica do algoritmo de Remez: a amplitude da resposta em frequência apresenta *ripples* de igual amplitude, motivo pelo qual estes filtros também são conhecidos como filtros *equiripple*.

A aplicação do filtro obtido a tons de 2 kHz, 10 kHz e 18 kHz resulta na figura 5.7. Para os três tons processados pelo filtro é observada a passagem do tom de 10 kHz, que está dentro da banda passante, enquanto as frequências de 2 kHz e 18 kHz são atenuadas.

Figura 5.7: Aplicação do filtro passa-faixa a três tons



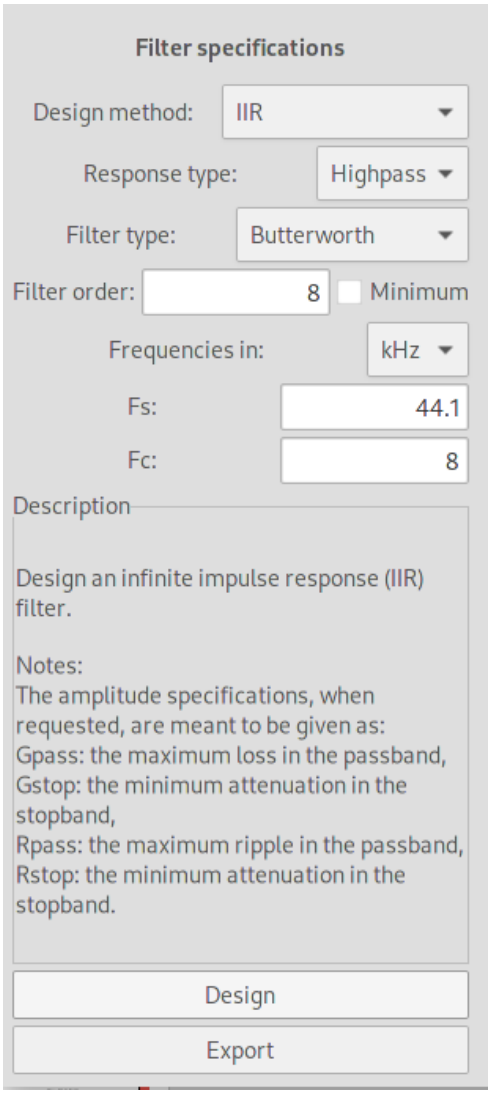
Fonte: Autoria Própria

5.3 Filtros IIR

Para os filtros IIR, existe a possibilidade de projetar o filtro de acordo com especificações bem definidas em relação a resposta desejada bem como de definir a ordem do filtro diretamente. Ao especificar um filtro de ordem mínima, será construído um filtro com a mínima ordem necessária para atingir as especificações.

No caso de um filtro passa-altas Butterworth de ordem 8 com frequência de corte de 8 kHz, a descrição do filtro seria feita como na figura 5.8.

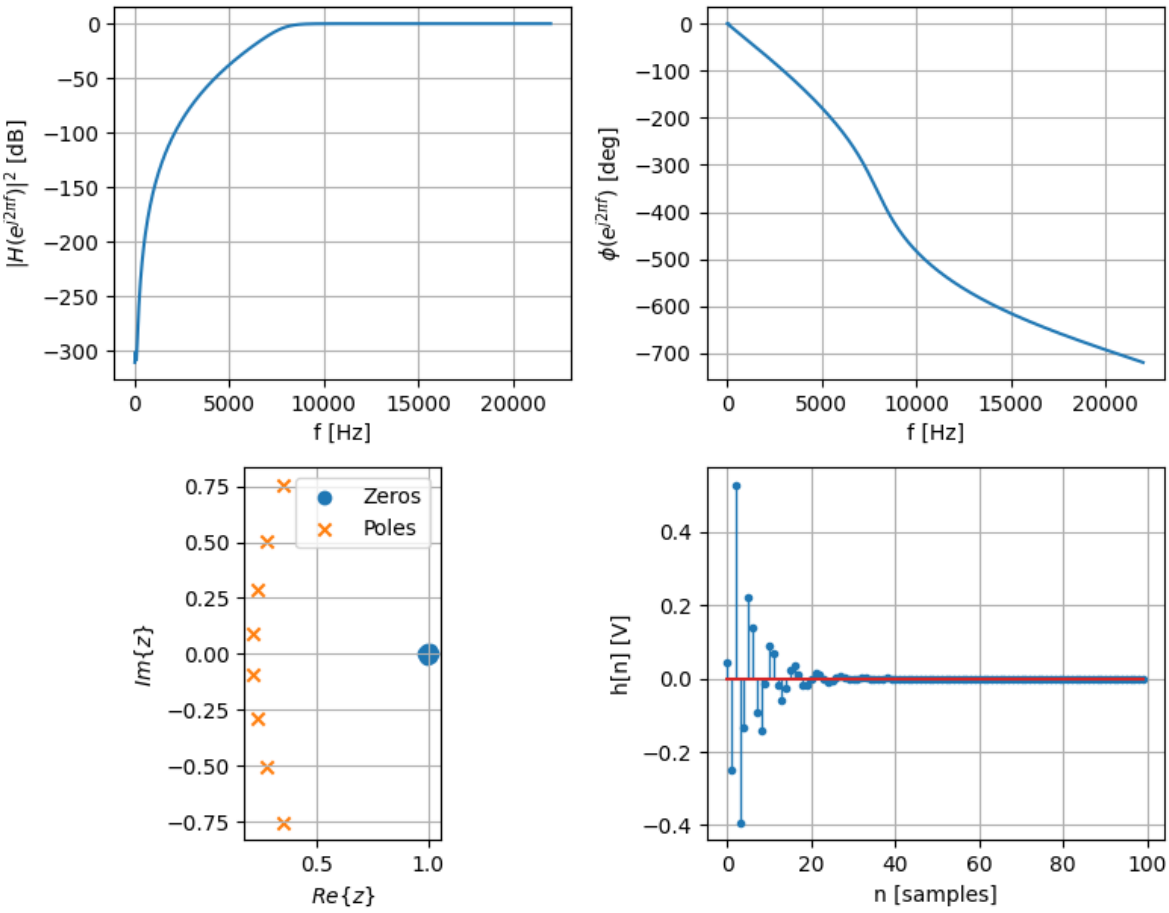
Figura 5.8: Especificação de um filtro passa-altas de Butterworth



Fonte: Autoria Própria

O filtro projetado apresenta as características exibidas na figura 5.9. Como é esperado de um filtro Butterworth, tem-se que a amplitude da resposta em frequência do filtro obtido é extremamente plana na banda de passagem.

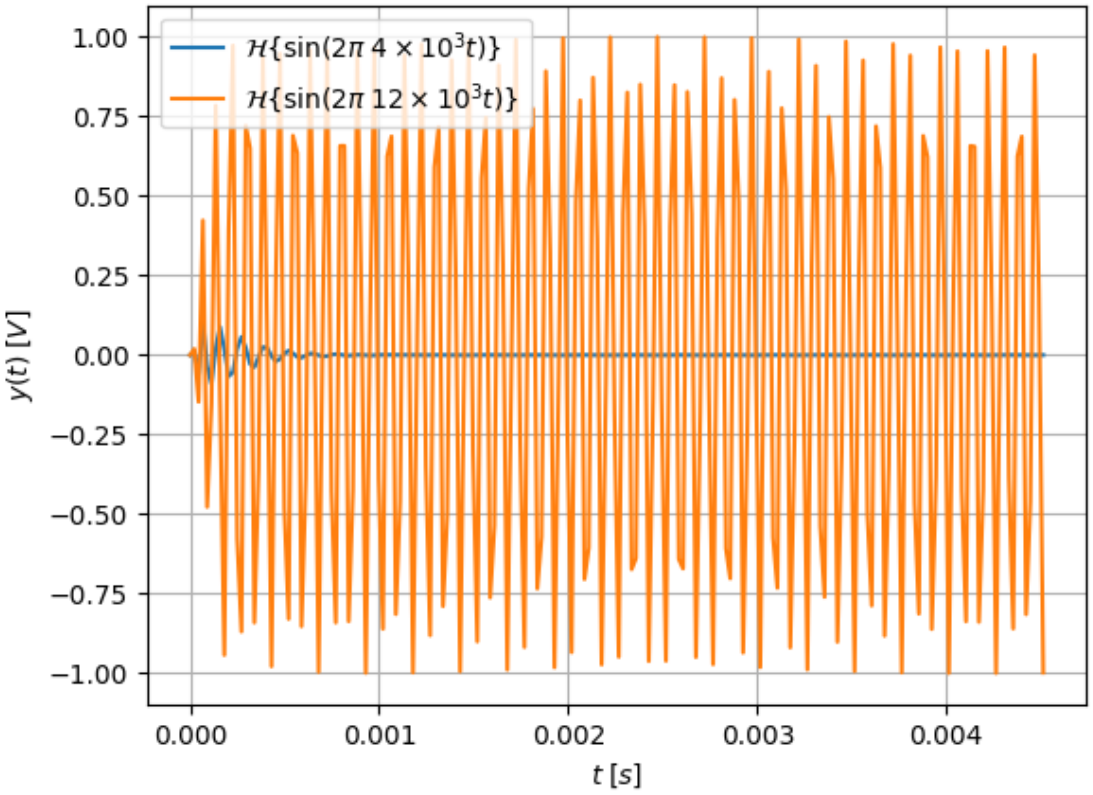
Figura 5.9: Resultados do filtro passa-altas de Butterworth



Fonte: Autoria Própria

Utilizando o filtro para processar um tom de 4 kHz e um de 12 kHz, obtem-se os resultados mostrados na figura 5.10.

Figura 5.10: Aplicação do filtro passa-altas a dois tons

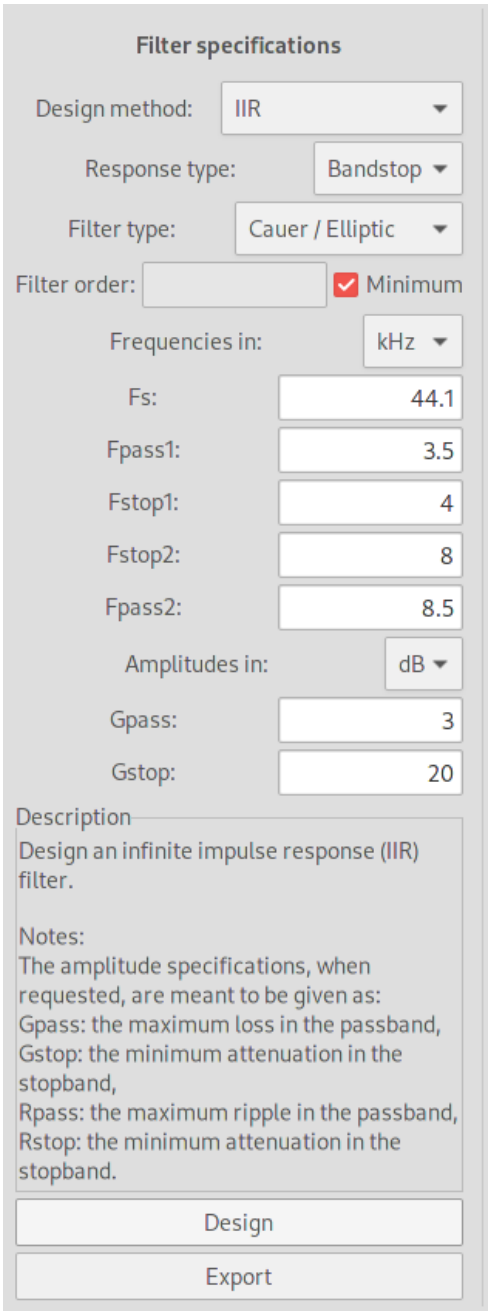


Fonte: Autoria Própria

Neste caso, a grande atenuação proporcionada pelo filtro faz com que o tom de 4 kHz seja praticamente eliminado na saída do mesmo, enquanto o tom de 12 kHz pode passar sem problemas. Vale destacar aqui o pequeno atraso introduzido pelo filtro, em relação aos filtros apresentados até então, devido a fase de sua resposta em frequência ter uma inclinação consideravelmente menor. Isto se dá em grande parte pela diferença no número de coeficientes ou ordem dos filtros considerados.

Já para um filtro rejeita-faixa de ordem mínima cuja banda rejeitada vai de 4 kHz a 8 kHz, com 500 Hz de faixa de transição, uma atenuação de 20 dB na banda rejeitada e atenuação máxima de 3 dB na banda passante, as especificações seriam como na figura 5.11.

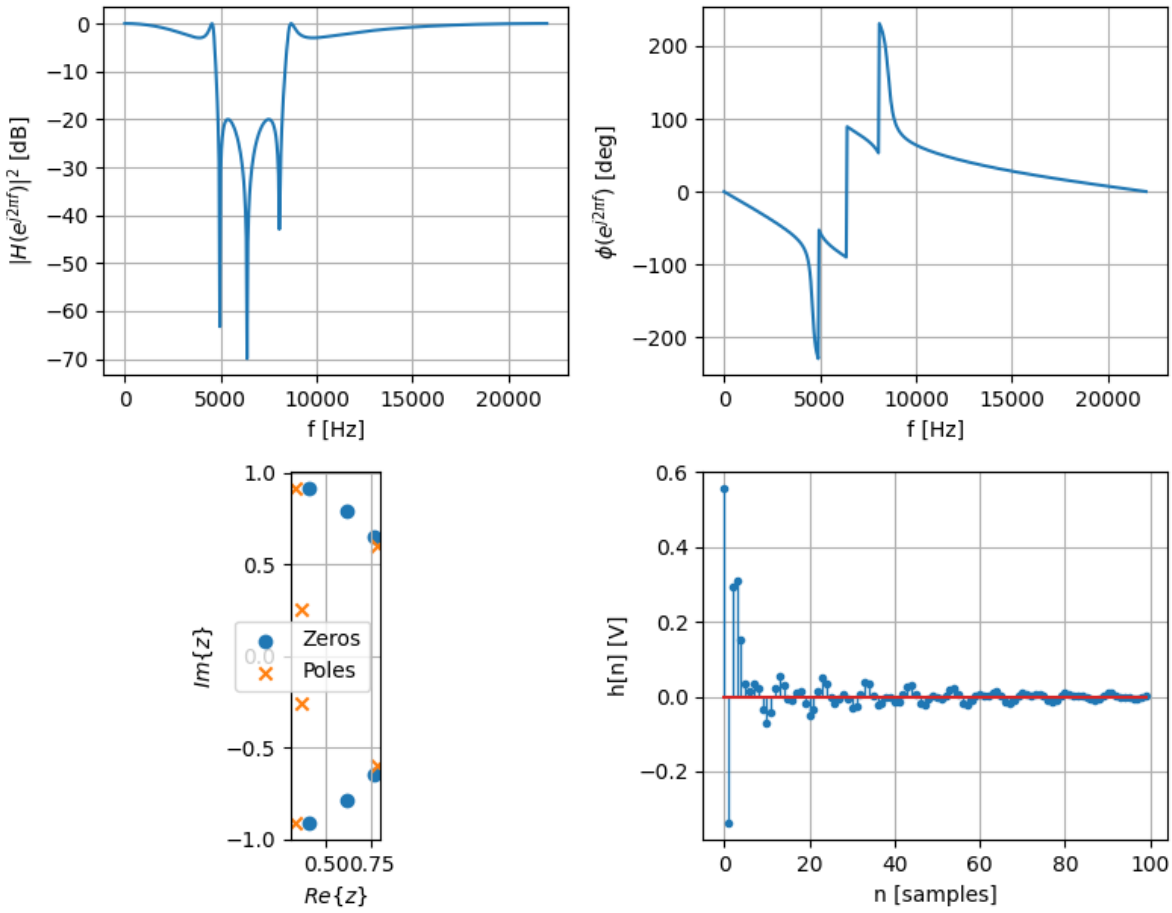
Figura 5.11: Especificação de um filtro rejeita-faixa de Cauer



Fonte: Autoria Própria

Os resultados obtidos para este filtro podem ser vistos na figura 5.12.

Figura 5.12: Resultados do filtro rejeita-faixa de Cauer

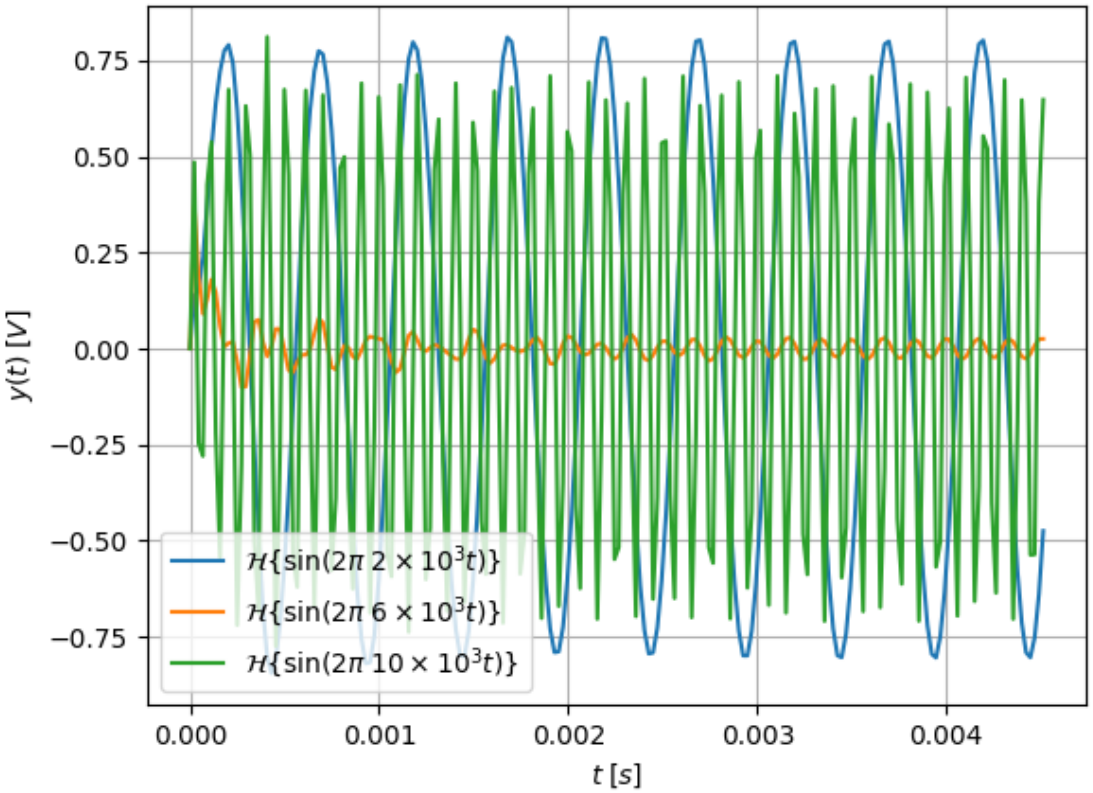


Fonte: Autoria Própria

É possível observar que as características esperadas pelo filtro de Cauer foram obtidas, sendo respeitadas as atenuações mínima de de 20 dB na banda rejeitada e máxima de 3 dB na banda passante. Nota-se ainda que a atenuação pode atingir valores bastante elevados, alcançando até 70 dB.

Aplicando o filtro desenvolvido a tons de 2 kHz, 6 kHz e 10 kHz, obtém-se a figura 5.13.

Figura 5.13: Aplicação do filtro rejeita-faixa a três tons



Fonte: Autoria Própria

Os tons processados pelo filtro de Cauer mostram bem a atenuação de 3 dB nas frequências de 2 kHz e 10 kHz, resultando em amplitudes de aproximadamente 0.7, enquanto que o tom de 6 kHz foi fortemente atenuado.

Capítulo 6

Conclusão

Devido à dificuldade de visualização no aprendizado de processamento de sinais, imposta pelo elevado grau de abstração envolvido no formalismo matemático da teoria abrangida, foi desenvolvido neste trabalho uma ferramenta capaz de realizar o projeto de filtros digitais segundo métodos clássicos da literatura. A plataforma construída contempla filtros FIR utilizando o método Least Squares e o algoritmo de Remez, sendo também tratados os clássicos filtros IIR de Butterworth, Chebyshev I e II, Cauer e Bessel. A aplicação é também capaz de fornecer diversos gráficos com intuito de permitir a visualização das principais características dos filtros projetados, como sua resposta em frequência, a localização de seus polos e zeros no plano complexo e sua resposta ao impulso. Por fim, tem-se ainda a funcionalidade de exportar o filtro construído para um arquivo em formato `.pickle` para a utilização em outras aplicações Python.

Com a possibilidade de realizar o projeto de filtros digitais por meio de interface gráfica com o usuário, a plataforma desenvolvida cumpre com o objetivo de tornar o processo mais prático para o projetista de filtros, bem como capacita o aluno com uma ferramenta alternativa durante o aprendizado do tópico de processamento de sinais.

6.1 Trabalhos futuros

Possíveis melhorias foram identificadas para trabalhos futuros, tais quais:

- funcionalidade para permitir a filtragem de sinais diretamente pela plataforma;
- adição de outros formatos de exportação (`.json`, `.csv`, `.mat`, etc);
- implementação de especificações mais complexas para filtros FIR, como filtros de Hilbert e até mesmo filtros multibandas;
- possibilidade de edição dos polos e zeros do filtro graficamente;

- processamento em tempo real de sinais captados via interface serial com o filtro projetado.

Ainda, devido a natureza também didática da ferramenta desenvolvida, o *feedback* de alunos a respeito de sua utilização pode no futuro apontar para melhorias de *layout* que tornem a plataforma mais intuitiva.

Capítulo 7

Bibliografia

[1] Alan V. Oppenheim, Ronald W. Schafer e John R. Buck. *Discrete-Time Signal Processing*. 2ª ed. Prentice Hall, 31 de dez. de 1998. ISBN: 0-13-754920-2.

[2] B. P. Lathi e Zhi Ding. *Modern Digital and Analog Communication Systems*. 4ª ed. Oxford University Press, 2 de jul. de 2009. ISBN: 978-0-19-538493-2.

[3] Paulo S. R. Diniz, Eduardo A. B. da Silva e Sergio L. Netto. *Digital Signal Processing*. 2ª ed. Cambridge University Press, set. de 2010. ISBN: 978-0521887755.

[4] Ivan Selesnick. *Linear-Phase FIR Filter Design by Least Squares*. OpenStax CNX, 9 de ago. de 2005. URL: `https://cnx.org/contents/eb1ecb35-03a9-4610-ba87-41cd771c95f2@7` (acesso em 07/05/2021).

[5] J. McClellan e T. Parks. "A unified approach to the design of optimum FIR linear-phase digital filters". Em: *IEEE Transactions on Circuit Theory* 20.6 (1973), pp. 697–701. DOI: `10.1109/TCT.1973.1083764`.

[6] *Python*. URL: `https://www.python.org` (acesso em 07/05/2021).

[7] *The GTK Project - A free and open-source cross-platform widget toolkit*. URL: `https://www.gtk.org` (acesso em 07/05/2021).

[8] *The Python GTK+ 3 Tutorial*. URL: `https://python-gtk-3-tutorial.readthedocs.io/en/latest/` (acesso em 07/05/2021).

[9] *The fundamental package for scientific computing with Python*. URL: `https://numpy.org` (acesso em 07/05/2021).

[10] *Scientific computing tools for Python*. URL: `https://www.scipy.org/about.html` (acesso em 07/05/2021).

[11] *Matplotlib: Visualization with Python*. URL: `https://matplotlib.org` (acesso em 07/05/2021).

[12] URL: `https://github.com/mateusosmarin/tcc` (acesso em 07/05/2021).

[13] Erich Gamma et al. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 21 de out. de 1994. ISBN: 978-0201633610.

[14] *Python Object Serialization*. URL: `https://docs.python.org/3/library/pickle.html` (acesso em 07/05/2021).

Apêndice - Demonstração de uso do filtro exportado

```python
import pickle

import matplotlib.pyplot as plt
import numpy as np
from scipy import signal

# Nome do arquivo do filtro exportado
filename = "filter.pickle"

# O arquivo deve ser aberto para leitura em formato binario (rb)
with open(filename, "rb") as file:
    # O metodo pickle.load permite carregar o objeto serializado
    tf = pickle.load(file)

# Obtem o periodo e frequencia de amostragem definidos no filtro
Ts = tf.dt
fs = 1 / Ts

# Constroi um vetor de tempo amostrado em Ts
t = np.arange(0, 200 * Ts, Ts)

# Sintetiza dois tons de frequencias 10 kHz e 1 kHz
x1 = np.sin(2 * np.pi * 10e3 * t)
x2 = np.sin(2 * np.pi * 1e3 * t)

# Aplica-se o filtro utilizando o metodo tf.output
t, y1 = tf.output(x1, t)
t, y2 = tf.output(x2, t)

# Obtem a resposta em frequencia do filtro
f, h = signal.freqz(tf.num, tf.den, fs=fs)

# Calcula a amplitude da resposta em dB
mag = 20 * np.log(np.abs(h))
```

```
35 # Calcula a fase da resposta em graus
36 phase = np.degrees(np.unwrap(np.angle(h)))
37
38 fig, axes = plt.subplots(2, 1)
39 fig.canvas.set_window_title("Resposta em frequencia")
40
41 # Cria figura para amplitude da resposta em frequencia
42 axes[0].plot(f, mag)
43 axes[0].set_xlabel(r"$f\ [Hz]$")
44 axes[0].set_ylabel(r"$|H(\mathrm{i} 2 \pi f)|^2\ [dB]$")
45 axes[0].grid(True)
46
47 # Cria figura para fase da resposta em frequencia
48 axes[1].plot(f, phase)
49 axes[1].set_xlabel(r"$f\ [Hz]$")
50 axes[1].set_ylabel(r"$\angle{H(\mathrm{i} 2 \pi f)\ [deg]}$")
51 axes[1].grid(True)
52
53 fig.tight_layout()
54
55 # Exibe sinais sintetizados
56 plt.figure("Sinais sintetizados")
57 plt.plot(t, x1, label=r"$\sin(2 \pi\ 10 \times 10^3 t)$")
58 plt.plot(t, x2, label=r"$\sin(2 \pi\ 1 \times 10^3 t)$")
59 plt.xlabel(r"$t\ [s]$")
60 plt.ylabel(r"$x(t)$")
61 plt.grid(True)
62 plt.legend()
63
64 # Exibe sinais filtrados
65 plt.figure("Sinais filtrados")
66 plt.plot(t, y1,
67          label=r"$\mathcal{H}\{\sin(2 \pi\ 10 \times 10^3 t)\}$")
68 plt.plot(t, y2,
69          label=r"$\mathcal{H}\{\sin(2 \pi\ 1 \times 10^3 t)\}$")
70 plt.xlabel(r"$t\ [s]$")
71 plt.ylabel(r"$y(t)$")
72 plt.grid(True)
73 plt.legend()
74
75 plt.show()
```