Desenvolvimento de uma plataforma interativa de processamento de sinais

Mateus Fuga Osmarin

4 de maio de 2021

Sumário

Resumo

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.

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 é uma técnica que pode ser aplicada 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 a FDATool 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, dada a grande quantidade de bibliotecas disponíveis para todo tipo de problemas.

Dessa forma, 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.

Capítulo 2

Referenciais teóricos

2.1 Sinais

Em essência, um sinal é algo que contém alguma informação, geralmente sobre o estado de um sistema físico [4]. 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:

$$x = x(t), t \in \mathbb{R} \tag{2.1}$$

A figura 2.1 mostra um exemplo de sinal contínuo no tempo.

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

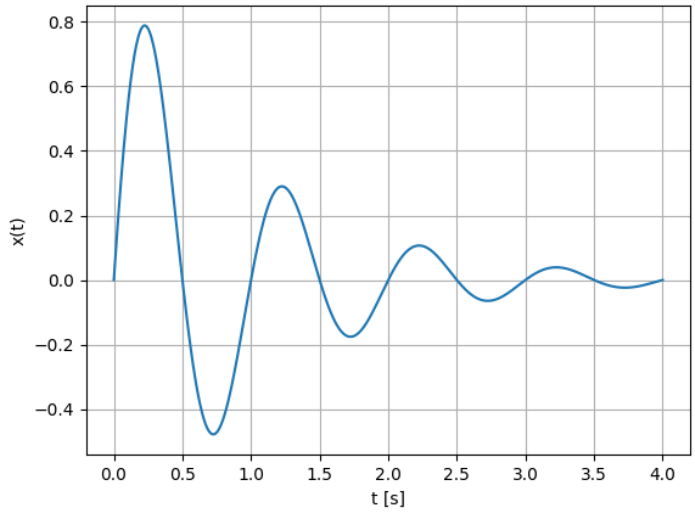
$$x = x[n], n \in \mathbb{Z} \tag{2.2}$$

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

$$x_d[n] = x_a(nT_s) \tag{2.3}$$

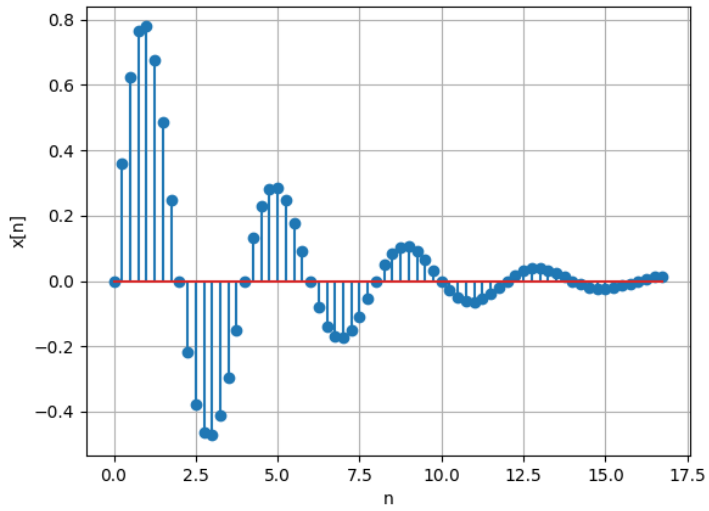
onde x_d é a versão discretizada do sinal analógico x_a e T_s representa o período de amostragem. A figura 2.2 mostra o sinal da figura 2.1 discretizado.

Figura 2.1: Sinal contínuo no tempo



Fonte: Autoria própria

Figura 2.2: Sinal discreto no tempo



Fonte: Autoria própria

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

$$\delta(t) = \begin{cases} 0 & \text{para} \quad t \neq 0 \\ \infty & \text{para} \quad t = 0 \end{cases} \tag{2.4}$$

satisfazendo a restrição

$$\int_{-\infty}^{+\infty} \delta(t)dt = 1 \tag{2.5}$$

Uma propriedade notória do impulso unitário é que

$$\int_{-\infty}^{+\infty} f(t)\delta(t - t_0)dt = f(t_0) \tag{2.6}$$

sendo conhecida como propriedade da amostragem.

No caso discreto, o impulso unitário é definido como:

$$\delta[n] = \begin{cases} 0 & \text{para} \quad n \neq 0 \\ 1 & \text{para} \quad n = 0 \end{cases} \tag{2.7}$$

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

$$\sum_{n=-\infty}^{+\infty} x[n]\delta[n - n_0] = x[n_0] \tag{2.8}$$

2.2 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:

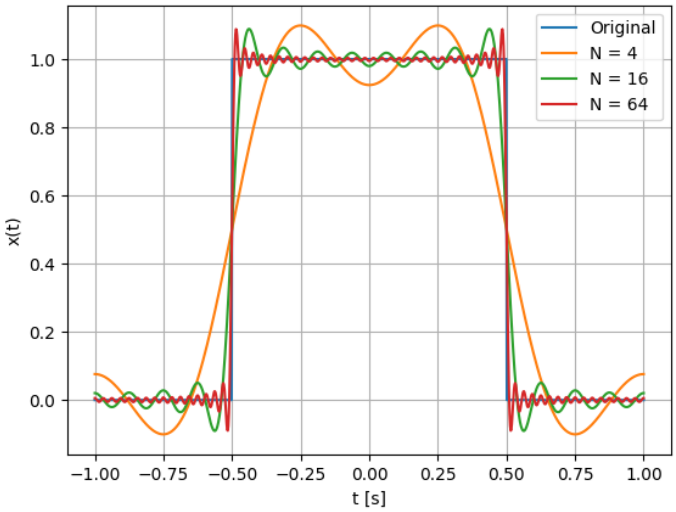
$$x_N(t) = \sum_{n=-N}^{+N} c_n \mathrm{e}^{\mathrm{i}\frac{2\pi}{T}nt} \tag{2.9}$$

onde

$$c_n = \frac{1}{T}\int_T x(t)\mathrm{e}^{-\mathrm{i}\frac{2\pi}{T}nt}dt \tag{2.10}$$

são os coeficientes da série, e N é teoricamente infinito. A figura 2.3 mostra a expansão em série de Fourier do retângulo unitário para diferentes valores de N.

Figura 2.3: 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:

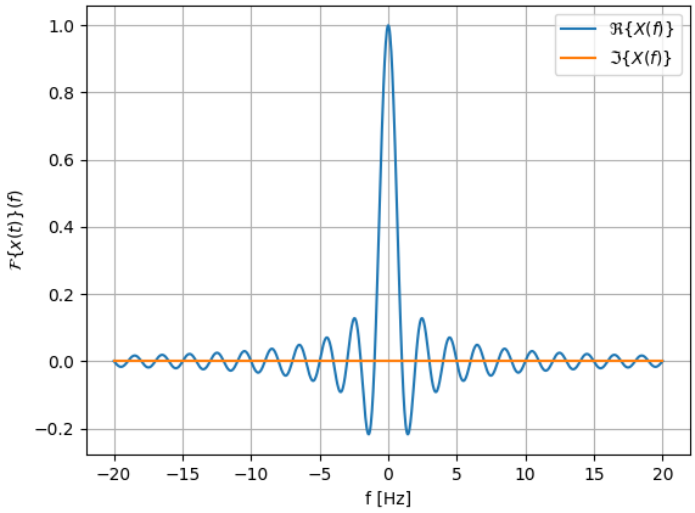
$$x(t) = \int_{-\infty}^{+\infty} \widehat{x}(f) \mathrm{e}^{\mathrm{i}2\pi ft} df \tag{2.11}$$

onde

$$\widehat{x}(f) = \mathcal{F}\{x(t)\}(f) = \int_{-\infty}^{+\infty} x(t) \mathrm{e}^{-\mathrm{i}2\pi ft} dt \tag{2.12}$$

e $\widehat{x}(f)$ é conhecida como a transformada de Fourier de $x(t)$. Segundo esta definição, f tem unidades de *hertz*. A figura 2.4 mostra a transformada de Fourier do retângulo unitário.

Figura 2.4: 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

$$\begin{aligned} x_T(t) = &\quad x(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT) \\ = &\sum_{n=-\infty}^{+\infty} x(nT)\delta(t - nT) \\ = &\sum_{n=-\infty}^{+\infty} x[n]\delta(t - nT) \end{aligned} \tag{2.13}$$

e aplicar a transformada de Fourier, tem-se

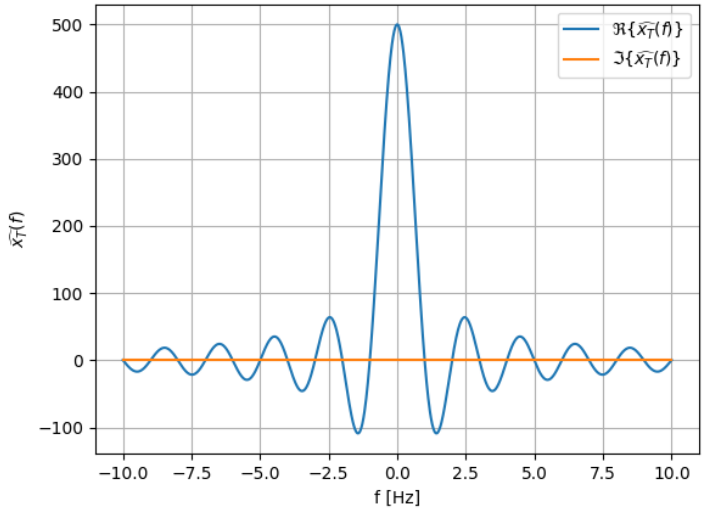
$$\begin{aligned} \widehat{x}_T(f) = &\int_{-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x[n]\delta(t - nT)\mathrm{e}^{-\mathrm{i}2\pi ft}dt \\ = &\sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{+\infty} \delta(t - nT)\mathrm{e}^{-\mathrm{i}2\pi ft}dt \\ = &\sum_{n=-\infty}^{+\infty} x[n]\mathrm{e}^{-\mathrm{i}2\pi fnT} \end{aligned} \tag{2.14}$$

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:

$$x[n] = T \int_{1/T} \widehat{x}_T(f) \mathrm{e}^{\mathrm{i}2\pi f n T} df \tag{2.15}$$

A figura 2.5 mostra a transformada de tempo discreto de Fourier do retângulo unitário.

Figura 2.5: 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:

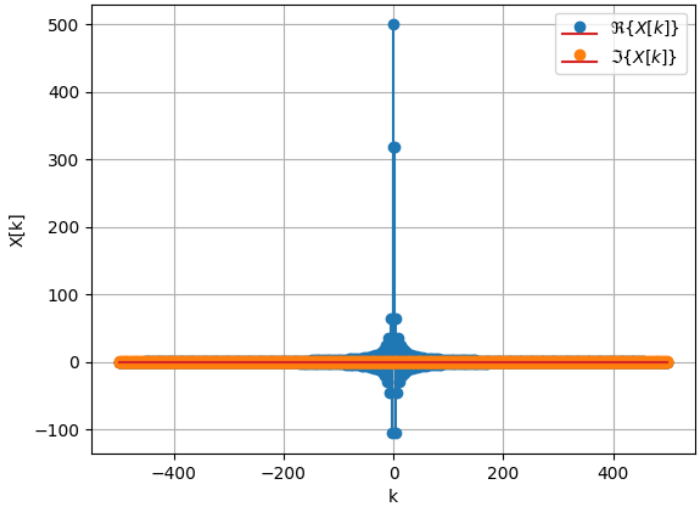
$$\begin{aligned} X[k] = \quad & \widehat{x}_T \left(\frac{k}{NT} \right) \\ & = \sum_{n=-\infty}^{+\infty} x[n] \mathrm{e}^{-\mathrm{i}2\pi \frac{k}{NT} nT} \\ & = \sum_{n=-\infty}^{+\infty} x[n] \mathrm{e}^{-\mathrm{i}2\pi \frac{k}{N} n} \end{aligned} \tag{2.16}$$

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

$$x[n] = \frac{1}{N} \sum_{k=-\infty}^{+\infty} X[k] \mathrm{e}^{\mathrm{i}2\pi \frac{k}{N} n} \tag{2.17}$$

A figura 2.6 mostra a transformada discreta de Fourier do retângulo unitário.

Figura 2.6: 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:

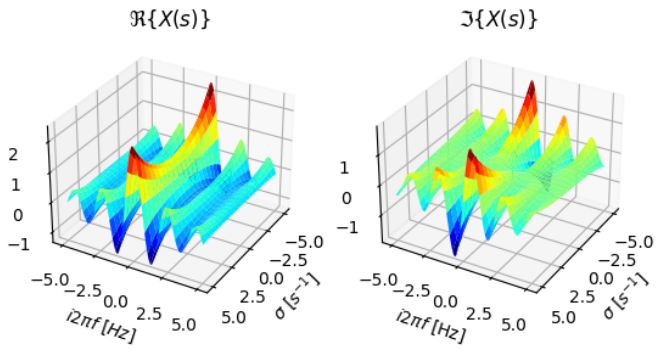
$$X(s) = \mathcal{L}\{x(t)\}(s) = \int_{-\infty}^{+\infty} x(t) \mathrm{e}^{-st} dt \tag{2.18}$$

onde

$$s = \sigma + \mathrm{i}\omega = \sigma + \mathrm{i}2\pi f \tag{2.19}$$

A figura 2.7 mostra a transformada de Laplace do retângulo unitário.

Figura 2.7: 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:

$$x(t) = \mathcal{L}^{-1}\{X(s)\}(t) = \frac{1}{\mathrm{i}2\pi} \lim_{T\to\infty} \int_{\gamma-\mathrm{i}T}^{\gamma+\mathrm{i}T} X(s)\mathrm{e}^{st} ds \tag{2.20}$$

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 (2.13) e aplicando a transformada de Laplace, tem-se

$$\begin{aligned} X_T(s) &= \int_{-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x[n]\delta(t-nT)\mathrm{e}^{-st} dt \\ &= \sum_{n=-\infty}^{+\infty} x[n] \int_{-\infty}^{+\infty} \delta(t-nT)\mathrm{e}^{-st} dt \\ &= \sum_{n=-\infty}^{+\infty} x[n]\mathrm{e}^{-snT} \end{aligned} \tag{2.21}$$

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

$$X(z) = \mathcal{Z}\{x[n]\}(z) = \sum_{n=-\infty}^{+\infty} x[n]z^{-n}, z \in \mathbb{C} \tag{2.22}$$

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

$$x[n] = \mathcal{Z}^{-1}\{X(z)\}[n] = \frac{1}{\mathrm{i}2\pi} \oint_{\mathcal{C}} X(z)z^{n-1}dz \tag{2.23}$$

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

2.3 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:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{2.24}$$

para o caso contínuo e

$$(f * g)[n] = \sum_{k=-\infty}^{+\infty} f[k]g[n-k] \tag{2.25}$$

para o caso discreto.

Uma propriedade importante da convolução é

$$x(t) * \delta(t) = \int_{-\infty}^{+\infty} x(\tau)\delta(t-\tau)d\tau = x(t) \tag{2.26}$$

para o caso contínuo e

$$x[n] * \delta[n] = \sum_{k=-\infty}^{+\infty} x[k]\delta[n-k] = x[n] \tag{2.27}$$

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:

$$\begin{aligned}
\mathcal{L}\{(f*g)(t)\}(s) &= \int_{t=-\infty}^{+\infty}\left[\int_{\tau=-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau\right]\mathrm{e}^{-st}dt \\
&= \int_{\tau=-\infty}^{+\infty}\int_{t=-\infty}^{+\infty} f(\tau)g(t-\tau)\mathrm{e}^{-st}dtd\tau \\
&= \int_{\tau=-\infty}^{+\infty} f(\tau)\int_{t=-\infty}^{+\infty} g(t-\tau)\mathrm{e}^{-st}dtd\tau \\
&= \int_{\tau=-\infty}^{+\infty} f(\tau)\int_{u=-\infty}^{+\infty} g(u)\mathrm{e}^{-s(u+\tau)}dud\tau \\
&= \int_{\tau=-\infty}^{+\infty} f(\tau)\mathrm{e}^{-s\tau}d\tau\int_{u=-\infty}^{+\infty} g(u)\mathrm{e}^{-su}du \\
&= \mathcal{L}\{f(t)\}(s)\mathcal{L}\{g(t)\}(s)
\end{aligned} \tag{2.28}$$

ou seja, no domínio da frequência a operação de convolução corresponde à multiplicação. A equação 2.28 é conhecida como teorema do convolução.

2.4 Sistemas lineares e invariantes no tempo

Um sistema é essencialmente algo que mapeia um sinal de entrada em um sinal de saída [1], ou seja:

$$y(t) = \mathcal{H}\{x(t)\} \tag{2.29}$$

para um sistema analógico e

$$y[n] = \mathcal{H}\{x[n]\} \tag{2.30}$$

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:

$$\mathcal{H}\{ax_1(t) + bx_2(t)\} = a\mathcal{H}\{x_1(t)\} + b\mathcal{H}\{x_2(t)\} \tag{2.31}$$

para o caso contínuo e

$$\mathcal{H}\{ax_1[n] + bx_2[n]\} = a\mathcal{H}\{x_1[n]\} + b\mathcal{H}\{x_2[n]\} \tag{2.32}$$

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:

$$\mathcal{H}\{x(t)\} = y(t) \Leftrightarrow \mathcal{H}\{x(t - t_0)\} = y(t - t_0) \tag{2.33}$$

para o caso contínuo e

$$\mathcal{H}\{x[n]\} = y[n] \Leftrightarrow \mathcal{H}\{x[n - n_0]\} = y[n - n_0] \tag{2.34}$$

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 2.26 e 2.27, sinais podem ser representados por uma convolução com o impulso unitário, tem-se

$$\begin{aligned} y(t) = &\quad \mathcal{H}\{x(t) * \delta(t)\} = \mathcal{H}\{\int_{-\infty}^{+\infty} x(\tau)\delta(t - \tau)d\tau\} \\ = &\quad \int_{-\infty}^{+\infty} x(\tau)\mathcal{H}\{\delta(t - \tau)\}d\tau \\ = &\quad \int_{-\infty}^{+\infty} x(\tau)h(t - \tau)d\tau \\ = &\quad x(t) * h(t) \end{aligned} \tag{2.35}$$

para o caso contínuo e

$$\begin{aligned} y[n] = &\quad \mathcal{H}\{x[n] * \delta[n]\} = \mathcal{H}\{\sum_{k=-\infty}^{+\infty} x[k]\delta[n - k]\} \\ = &\quad \sum_{k=-\infty}^{+\infty} x[k]\mathcal{H}\{\delta[n - k]\} \\ = &\quad \sum_{k=-\infty}^{+\infty} x[k]h[n - k] \\ = &\quad x[n] * h[n] \end{aligned} \tag{2.36}$$

para o caso discreto. Nas equações 2.35 e 2.36, $h(t)$ 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 2.28 fornece:

$$Y(s) = \mathcal{L}\{x(t) * h(t)\} = X(s)H(s) \tag{2.37}$$

para um sistema analógico e

$$Y(z) = \mathcal{Z}\{x[n] * h[n]\} = X(z)H(z) \tag{2.38}$$

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

2.5 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:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] \tag{2.39}$$

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:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{M} b_k z^{-k} \tag{2.40}$$

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:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k] \tag{2.41}$$

Aplicando a transformada Z, tem-se

$$Y(z) = \sum_{k=0}^{M} b_k X(z) z^{-k} - \sum_{k=1}^{N} a_k Y(z) z^{-k} \tag{2.42}$$

de onde, ao se rearranjar, obtem-se

$$Y(z)\left(1+\sum_{k=1}^{N} a_k z^{-k}\right) = X(z)\left(\sum_{k=0}^{M} b_k z^{-k}\right) \tag{2.43}$$

ou seja,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}} \tag{2.44}$$

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* [7], o algoritmo de Remez [3] e a transformação bilinear [1] aplicada aos clássicos filtros de Butterworth, Chebyshev, Cauer e Bessel.

Capítulo 3

Resultados

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, bem como será tratado sobre sua forma de uso, incluindo exemplos.

A ferramenta foi desenvolvida em Python 3, utilizando as bibliotecas GTK para interface gráfica, NumPy para computação numérica eficiente, SciPy para projeto dos filtros e Matplotlib 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 [6]. Instruções de instalação estão inclusas no README do repositório.

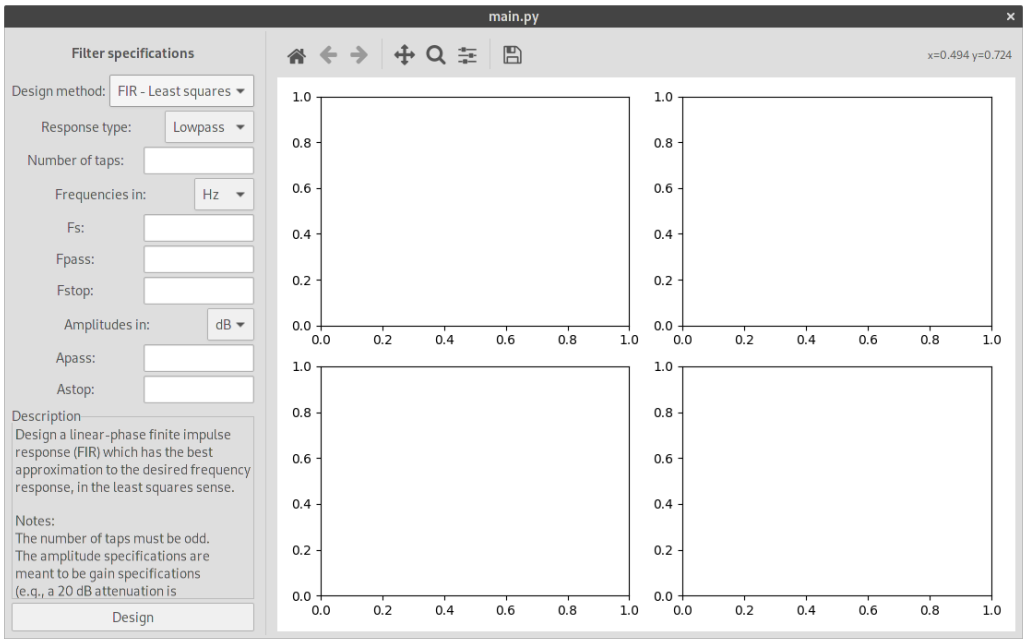
O programa foi estruturado na seguinte hierarquia:

- 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 gŕaficos 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 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 [2]

A tela inicial do programa é mostrada na figura 3.1. O sistema consiste de duas seções: as especificações do filtro a ser projetado e os gráficos que mostram os resultados do processo.

Figura 3.1: 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 inter-

face 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 [7], o algoritmo de Remez minimiza o erro máximo [3].

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 [5], que permite serializar um objeto do Python em um arquivo que pode ser posteriormente carregado e usado em qualquer aplicação Python.

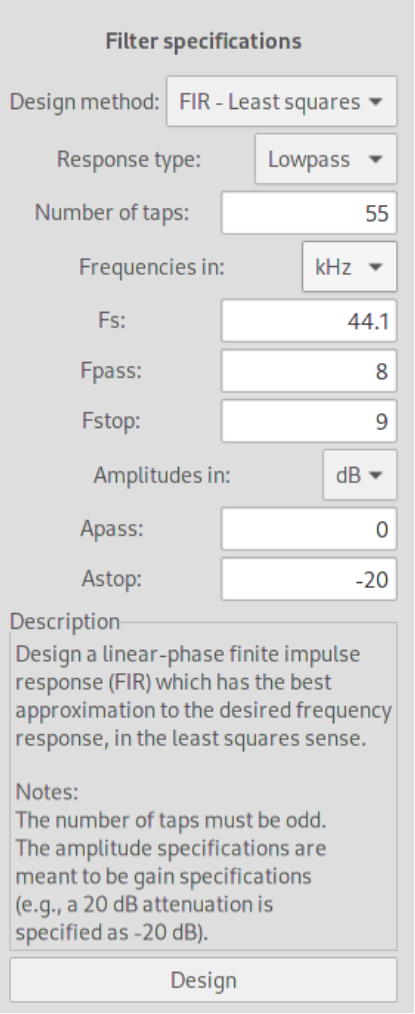
3.1 Exemplos

Nesta seção será tratado o desenvolvimento 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.

3.1.1 FIR - 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 3.2.

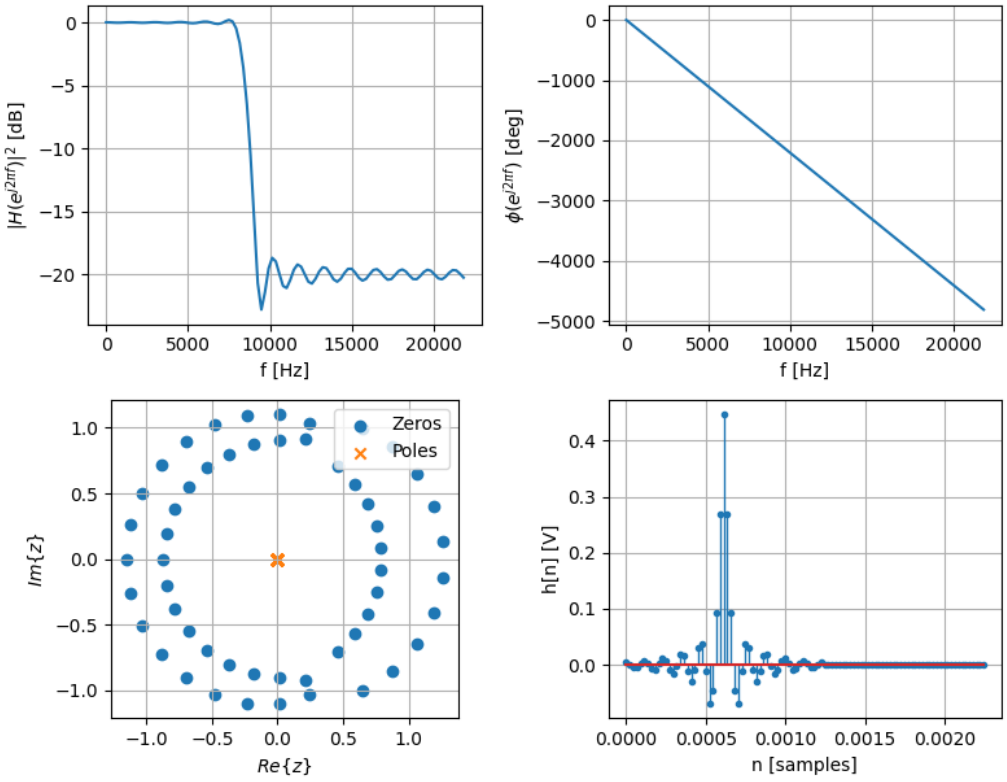
Figura 3.2: Especificação de um filtro passa-baixas usando Least Squares



Fonte: Autoria Própria

Os resultados obtidos para estas especificações podem ser visualizados na figura 3.3

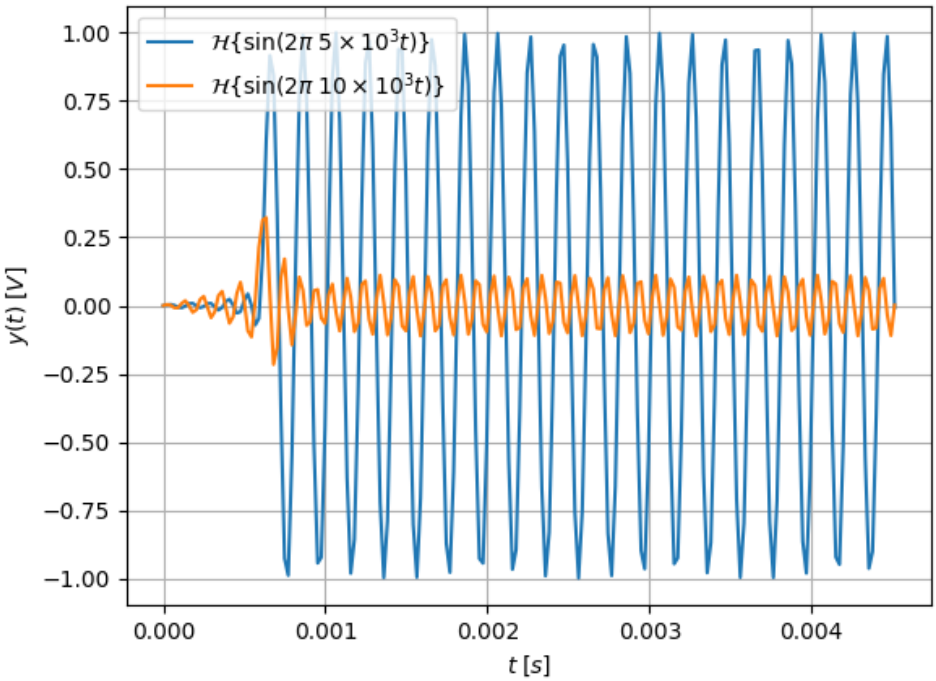
Figura 3.3: Resultados do filtro passa-baixas utilizando Least Squares



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 3.4.

Figura 3.4: Aplicação do filtro passa-baixas a dois tons

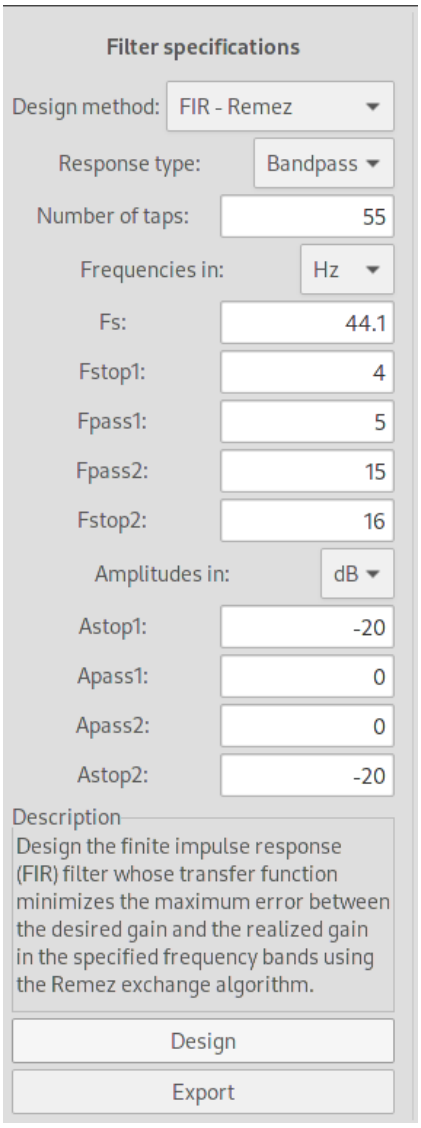


Fonte: Autoria Própria

3.1.2 FIR - 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 3.5.

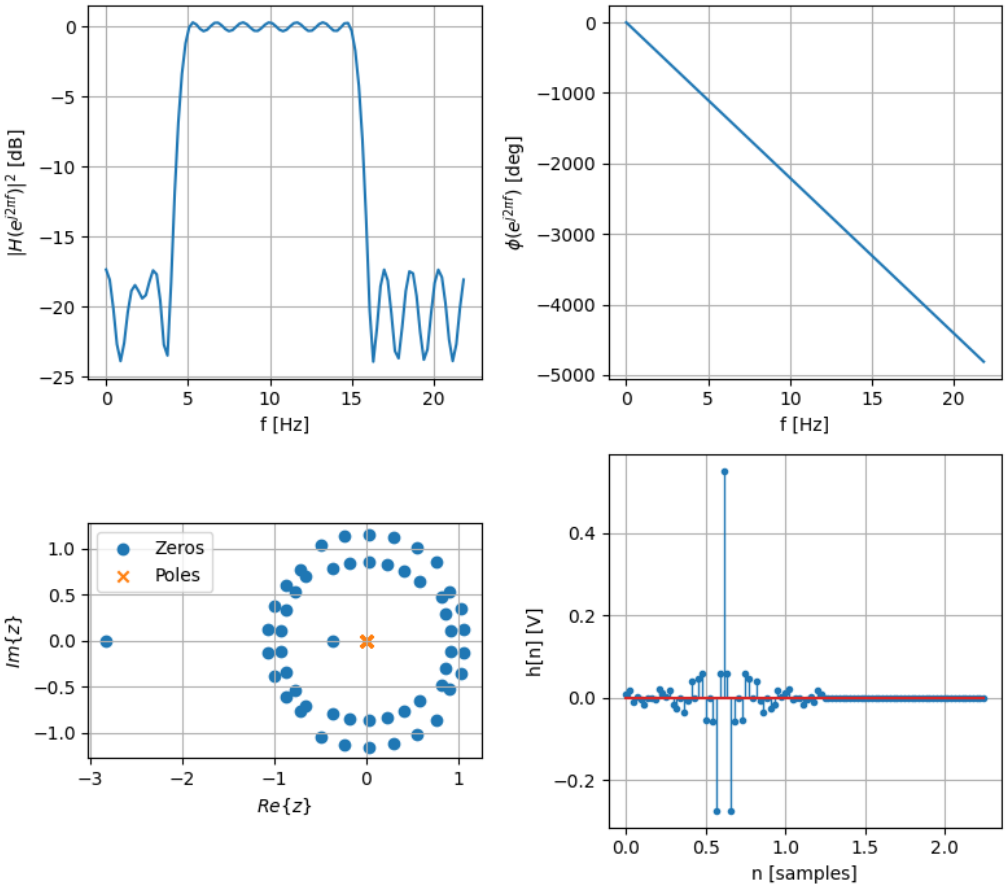
Figura 3.5: Especificação de um filtro passa-faixa usando o algoritmo de Remez



Fonte: Autoria Própria

Para este filtro, os resultados obtidos podem ser vistos na figura 3.6.

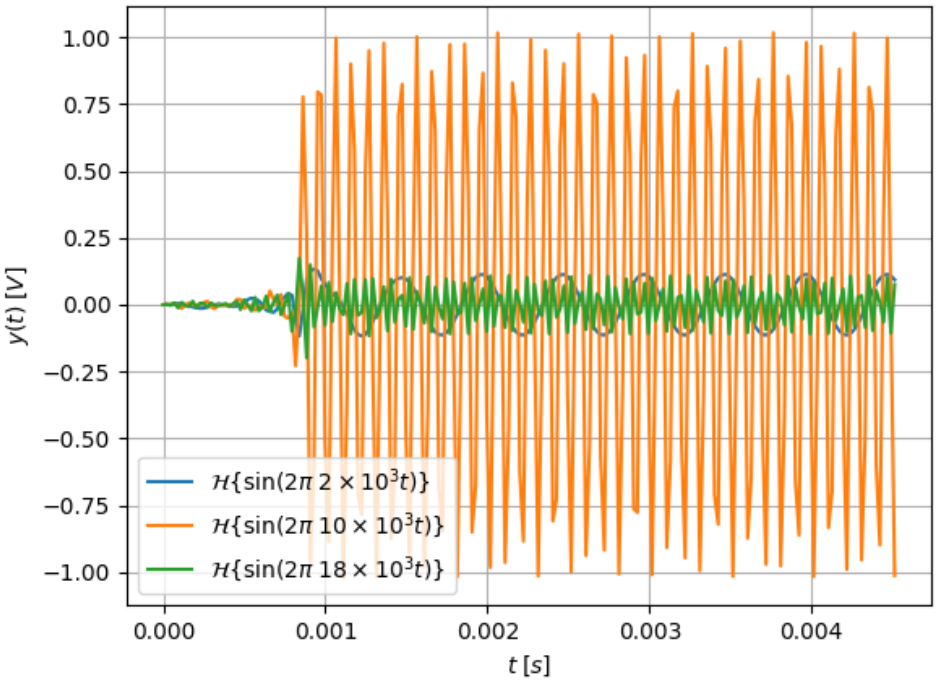
Figura 3.6: Resultados do filtro passa-faixa utilizando o algoritmo de Remez



Fonte: Autoria Própria

A aplicação do filtro obtido a tons de 2 kHz, 10 kHz e 18 kHz resulta na figura 3.7.

Figura 3.7: Aplicação do filtro passa-faixa a três tons



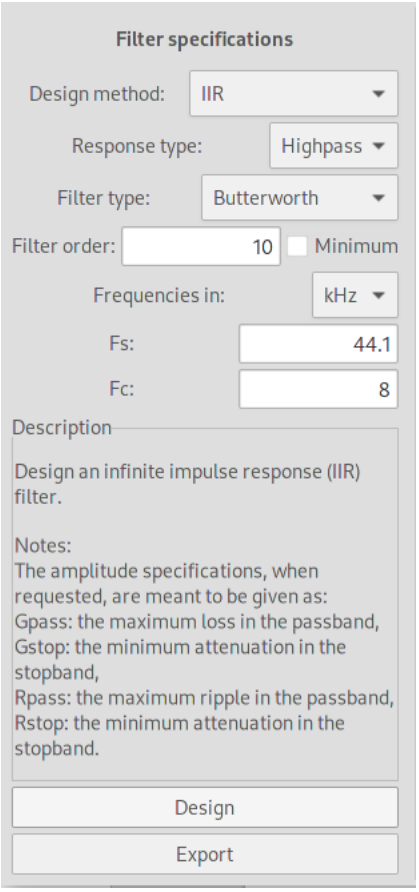
Fonte: Autoria Própria

3.1.3 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 10 com frequência de corte de 8 kHz, a descrição do filtro seria feita como na figura 3.8.

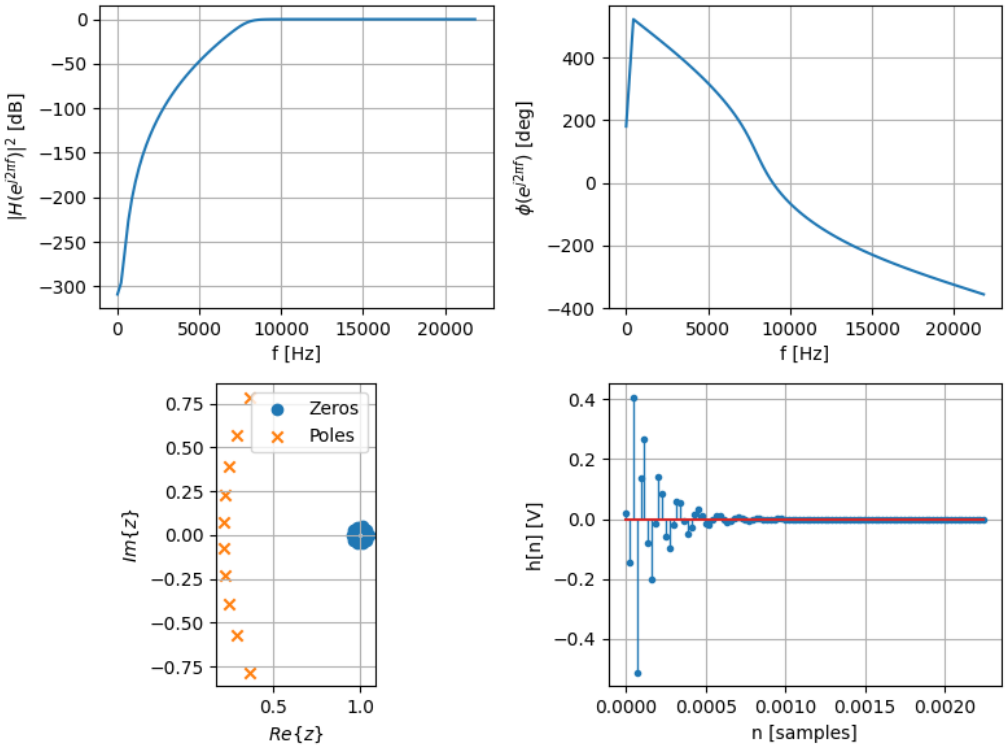
Figura 3.8: Especificação de um filtro passa-altas de Butterworth



Fonte: Autoria Própria

O filtro projetado apresenta as características exibidas na figura 3.9.

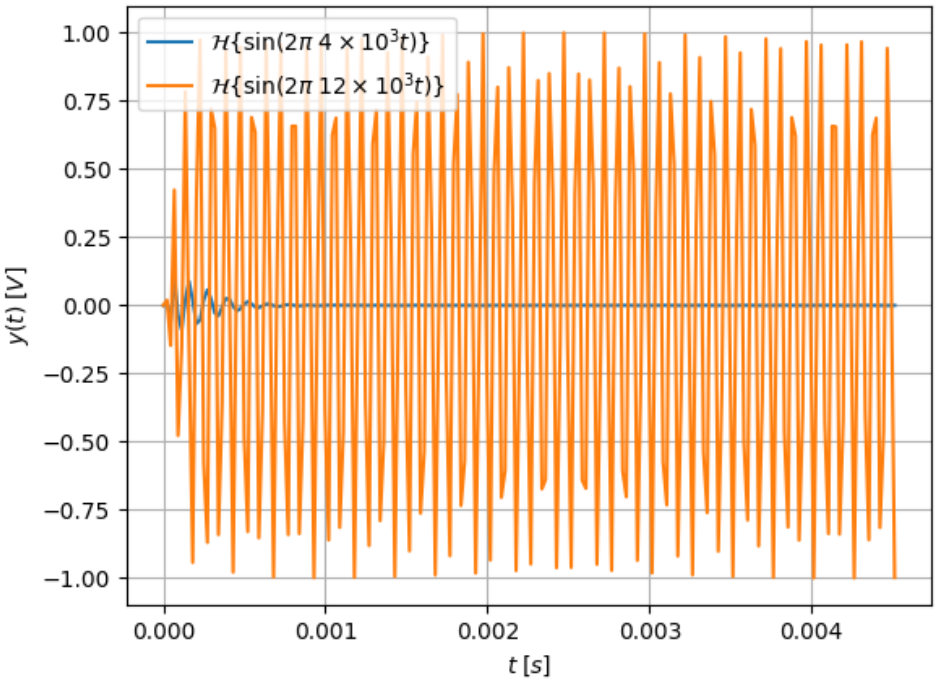
Figura 3.9: Resultados do filtro passa-altas de Butterworth



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 3.10.

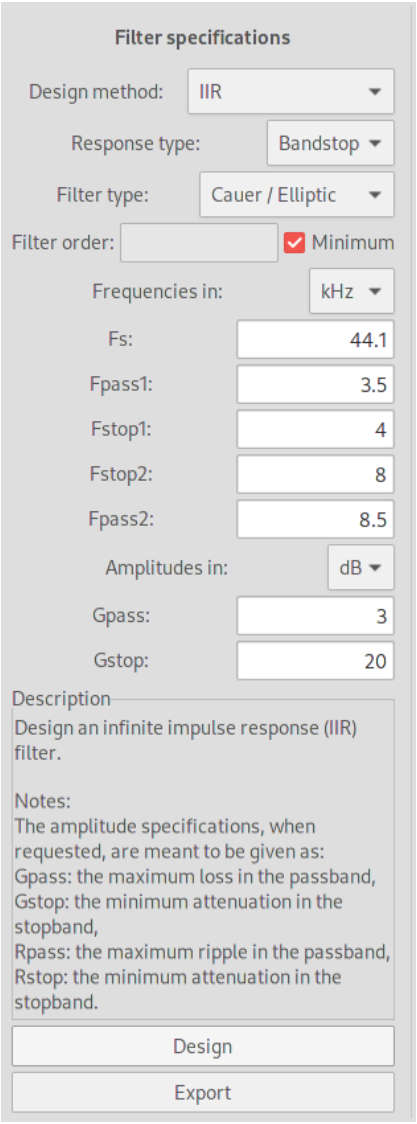
Figura 3.10: Aplicação do filtro passa-altas a dois tons



Fonte: Autoria Própria

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 3.11.

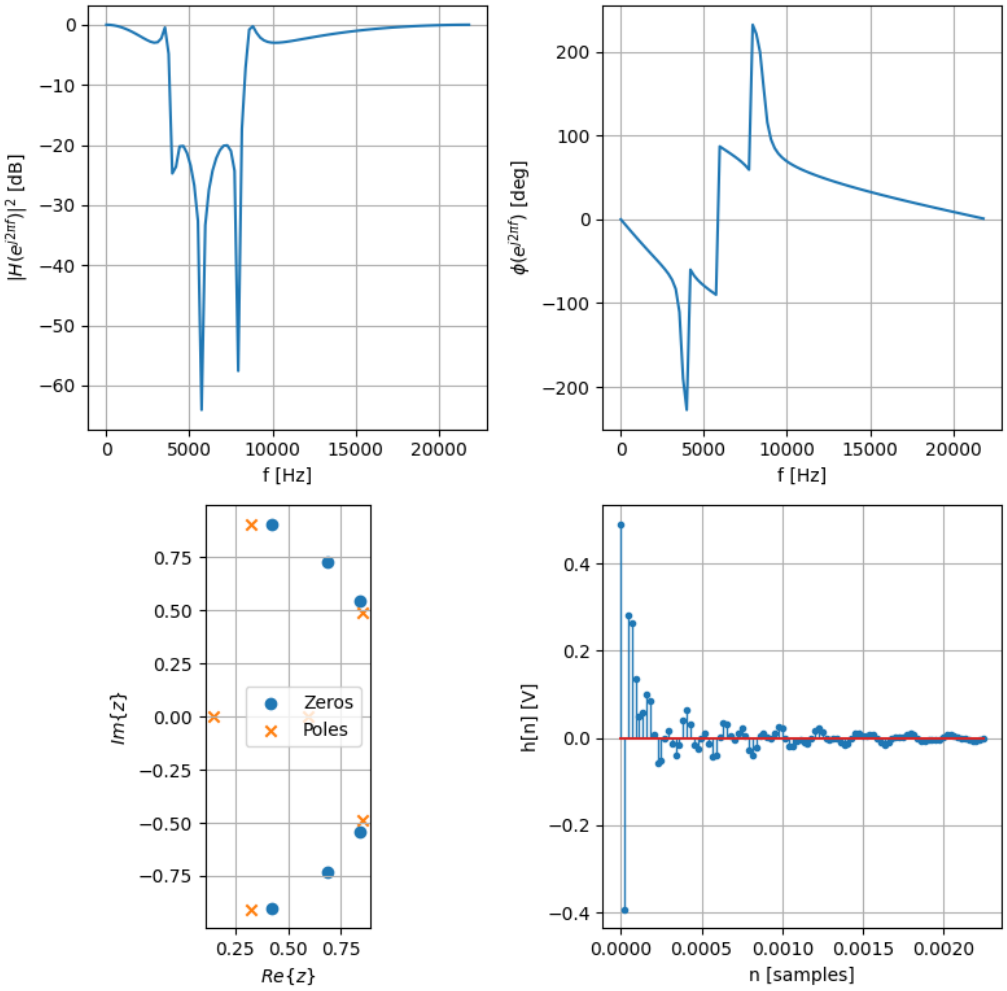
Figura 3.11: Especificação de um filtro rejeita-faixa de Cauer



Fonte: Autoria Própria

Os resultados obtidos para este filtro podem ser vistos na figura 3.12.

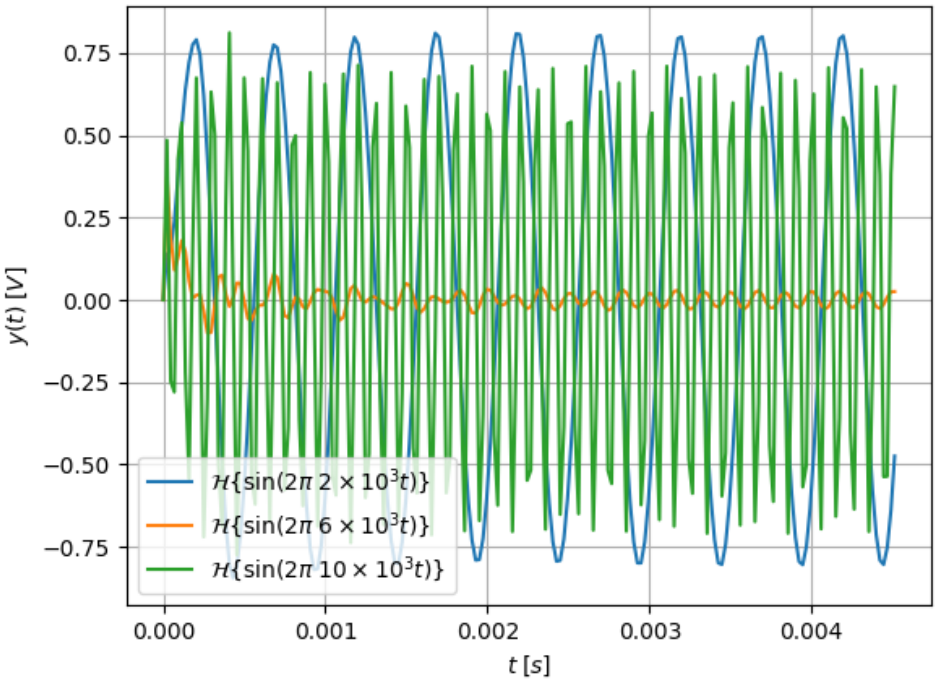
Figura 3.12: Resultados do filtro rejeita-faixa de Cauer



Fonte: Autoria Própria

Aplicando o filtro desenvolvido a tons de 2 kHz, 6 kHz e 10 kHz, obtém-se a figura 3.13.

Figura 3.13: Aplicação do filtro rejeita-faixa a três tons



Fonte: Autoria Própria

Capítulo 4

Considerações finais

O programa desenvolvido neste trabalho é capaz de realizar o projeto de filtros digitais segundo métodos clássicos da literatura, podendo ser expandido a novos métodos. Ainda assim, possíveis melhorias foram identificadas para trabalhos futuros. Uma funcionalidade interessante seria a possibilidade de, na própria aplicação, fazer a entrada de um sinal para realizar o processamento com o filtro projetado, pois atualmente só é possível exportar o filtro para um arquivo a ser utilizado posteriormente. Sobre o formato de exportação, também podem ser implementados novos formatos, como CSV, JSON e até mesmo o formato de arquivos do MATLAB. 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.

Bibliografia

[1] Paulo S. R. Diniz, Eduardo A. B. da Silva e Sergio L. Netto. *Digital Signal Processing*. Cambridge University Press.

[2] Erich Gamma et al. *Design Patterns. Elements of Reusable Object-Oriented Software*. 21 de out. de 1994.

[3] *A unified approach to the design of optimum FIR linear phase digital filters* 20.6 (nov. de 1973). DOI: 10.1109/TCT.1973.1083764.

[4] Alan V. Oppenheim, Ronald W. Schafer e John R. Buck. *Discrete-Time Signal Processing*. Prentice Hall.

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

[6] *Repositório do trabalho*. URL: https://github.com/mateusosmarin/tcc (acesso em 07/05/2021).

[7] Ivan Selesnick. "Linear-Phase FIR Filter Design by Least Squares". Em: (5 de ago. de 2002).