UFC

SINAIS E SISTEMAS

PROCESSAMENTO DIGITAL DE SINAIS

FILTRO PASSA-BAIXA

Equipe:

1. FELIPE CORDEIRO DE SOUSA 571941

1 Introdução

Com a vinda de meios de comunicações digitais através da popularização da internet por volta dos anos 80/90 e, entre outros fatores emergentes no mundo digital, potencializaram o futuro promissor do processamento de sinais devido a aplicações e plataformas de hardware que surgem constantemente, cada vez mais sofisticadas. As aplicações de processamento de sinais abrangem uma finidade de áreas, como medicina, computação, telecomunicações, arqueologia e geofísica, entre outras áreas. Apesar de nunca se perguntarmos sobre as altas qualidades de áudio, vídeo com resolução de 4K, sistemas de comunicação em tempo real, serviços de streaming e sistemas de multimídias, no geral, são baseados completamente em processamento de sinais digitais. Oppenheim and Schafer (2013); Oppenheim (2010)

A quantidade cada vez maior de aplicações e a demanda por algoritmos cada vez mais sofisticados andam lado a lado com o rápido desenvolvimento da tecnologia de dispositivos para implementar sistemas de processamento de sinais. De acordo com algumas estimativas, mesmo com as limitações iminentes da Lei de Moore, a capacidade de processamento de microprocessadores dedicados a processamento de sinais e de computadores pessoais provavelmente aumentará em várias ordens de grandeza nos próximos dez anos. Evidentemente, a importância e o papel do processamento de sinais continuarão a se expandir em um ritmo acelerado no futuro. Oppenheim and Schafer (2013); Oppenheim (2010)

O processamento de sinais lida com a representação, a transformação e a manipulação de sinais e da informação que os sinais contêm. Por exemplo, podemos querer separar dois ou mais sinais que foram combinados por alguma operação, como adição, multiplicação ou convolução, ou então amplificar algum componente do sinal ou estimar algum parâmetro de um modelo de sinal. Uma mudança continuada e relevante para tecnologias digitais foi resultado da rápida evolução de computadores e microprocessadores digitais e dos chips de baixo custo para a conversão analógico-digital (A/D) e digital-analógico (D/A). Essas evoluções na tecnologia foram reforçadas por muitos desenvolvimentos teóricos importantes, como o algoritmo da transformada de Fourier rápida (FFT, do inglês fast Fourier transform), modelagem paramétrica de sinal, técnicas multitaxas, implementação de filtros polifásicos e novas formas de representar sinais, como expansões em wavelets. Apenas como um exemplo dessa mudança, os sistemas de comunicação por rádio analógicos estão evoluindo para os "software rádios" reconfiguráveis, que são implementados quase exclusivamente com computação digital. Oppenheim and Schafer (2013)

O processamento em tempo discreto de sinais é baseado no processamento de sequências numéricas indexadas sobre as variáveis inteiras, em vez de funções de uma variável independente contínua. Em processamento digital de sinais (PDS), os sinais são representados por sequências de números com precisão finita, e o processamento é implementado usando computação digital. O termo mais geral processamento em tempo discreto de sinais inclui o processamento digital de sinais como um caso especial, mas também inclui a possibilidade de que sequências de amostras (dados amostrados) possam ser processadas com outras tecnologias de tempo discreto. Frequentemente, a distinção entre os termos processamento em tempo discreto de sinais e processamento digital de sinais tem importância menor, pois ambos tratam de sinais de tempo discreto. Oppenheim and Schafer (2013)

2 Atividade

A atividade consiste em filtrar sinais de áudio (sinais de voz), que foram corrompidos pela adição de um ruído de alta frequência. O filtro a ser desenhado deve ser do tipo passa-baixa de resposta ao impulso finita (FIR) e deverá ser parametrizado através da escolha dos seguintes parâmetros:

- ωc: Frequência de corte ou largura do filtro;
- L: Ordem ou comprimento do filtro.

Em linhas gerais, esta parametrização tem o objetivo de eliminar ou reduzir o ruído da melhor forma possível.

2.1 Tarefa

A seguir todas as questões necessárias para completar a tarefa do trabalho computacional:

1. Ler o arquivo .wav para obter o sinal x[n] em forma de vetor.
2. Plotar o gráfico de x[n] no tempo e de seu espectro de frequência.
    - Deve-se avaliar o espectro do sinal. O que pode ser dito sobre como deve ser projetado o filtro passa-baixa a partir dessa análise em frequência?
3. Implementar o filtro h[n] com os parâmetros ωc e L, tendo como base a fundamentação teórica indicada.
4. Plotar o gráfico de h[n] no tempo, a magnitude e a fase de sua resposta em frequência
    - Discorra sobre as diferenças observadas entre a descrição teórica e a implementação prática do filtro.
5. Aplicar o filtro projetado e analisar o impacto da qualidade da filtragem ao variar os parâmetros L e ωc.
    - Comente os resultados para alguns valores de sua escolha e indique qual o melhor o valor obtido.
6. Plotar o sinal filtrado no tempo e na frequência e comparar o antes e depois.
7. Mensurar, de forma subjetiva, a qualidade da filtragem ao escutar o áudio filtrado.

3 Resultados

De acordo com as questões apresentadas anteriormente, podemos mostrar os resultados, realizados em python, e o passo a passo utilizado para obter tais resultados. Para cada pergunta, mostraremos a motivação das respostas, além dos gráficos de cada sinal, aplicação do nosso filtro e as discussões sobre cada resolução.

1. Ler o arquivo .wav para obter o sinal x[n] em forma de vetor.

Solution: Inicialmente, é preciso entender o cenário do dado no qual estamos trabalhando, nesse trabalho, nossos dados foram dois áudios no formato .wav, portanto para que seja possível encontrar o sinal x[n] como vetor nesse cenário, dentre várias possibilidades disponíveis para a leitura de um áudio, a biblioteca scipy disponibiliza um módulo (scipy.io.wavfile) voltado apenas para leitura de sinais que estão no formato de áudio (.wav). Podemos analisar o código usado para obter o sinal x[n] em forma de vetor, veja:

```
import numpy as np
from scipy.io.wavfile import read, write

# Obtaining x[n] in vector form from 01.wav file.
sample_rate, data = read("./data/01.wav")
```

2. Plotar o gráfico de x[n] no tempo e de seu espectro de frequência.

Solution: Inicialmente, para que seja possível analisarmos os sinais diante de uma perspectiva visual, teremos que colocar os dados em função do tempo, pois estamos vendo n amostras e seus respectivos valores. Portanto, precisamos criar o vetor de tempo, para que seja possível visualizarmos o sinal em função, de fato, do tempo. Para isso, utilizamos esse código:

```
import numpy as np
from scipy.io.wavfile import read, write

# Obtaining x[n] in vector form from 01.wav file.
sample_rate, data = read("./data/01.wav")
N = len(data)
T = 1/sample_rate
print(f"N mero de amostras: {sample_rate}\nTamanho do sinal: {N}")

t = np.arange(0, N/sample_rate, T)
```

```
print(f"Vetor tempo: {t}")

# Converting x[n] -> x[jw]
f = fftfreq(N,T)
transf = np.abs(fft(data))
```

No código acima, precisamos identificar o tamanho do sinal, o período do sinal e, por fim, conseguimos criar o vetor tempo através de um vetor que vai de 0 até o tamanho do áudio, de T em T. Então, somente após isso, conseguimos organizar o nosso sinal para, posteriormente, convertermos ele para a frequência. Com o vetor tempo criado, analisaremos visualmente como ficou o sinal (do áudio 1) tanto no domínio do tempo quanto no da frequência.

Figure 1: Análise do áudio 1

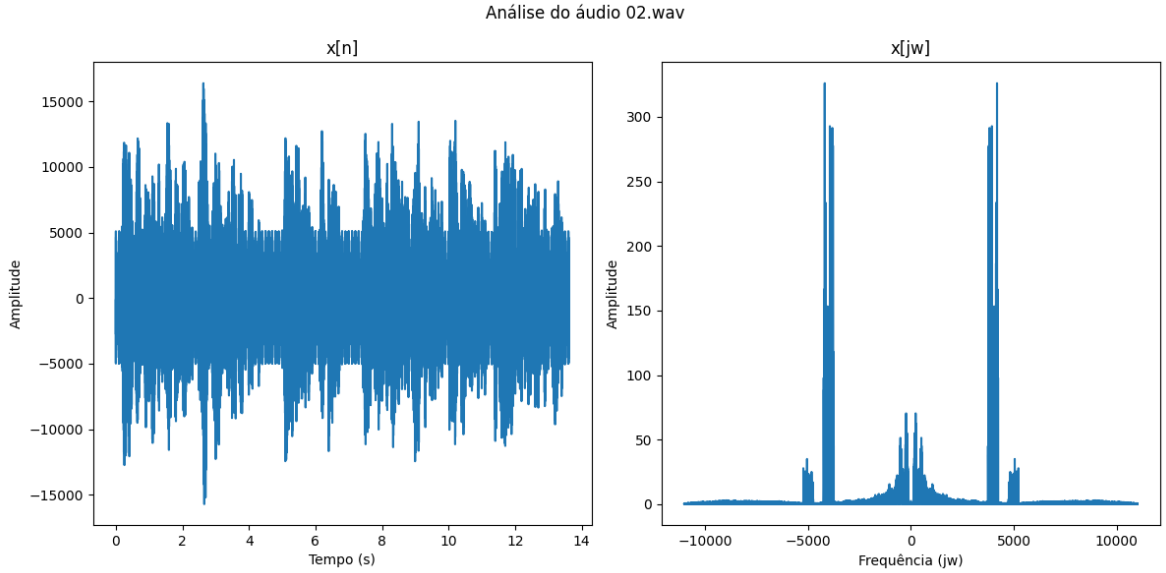
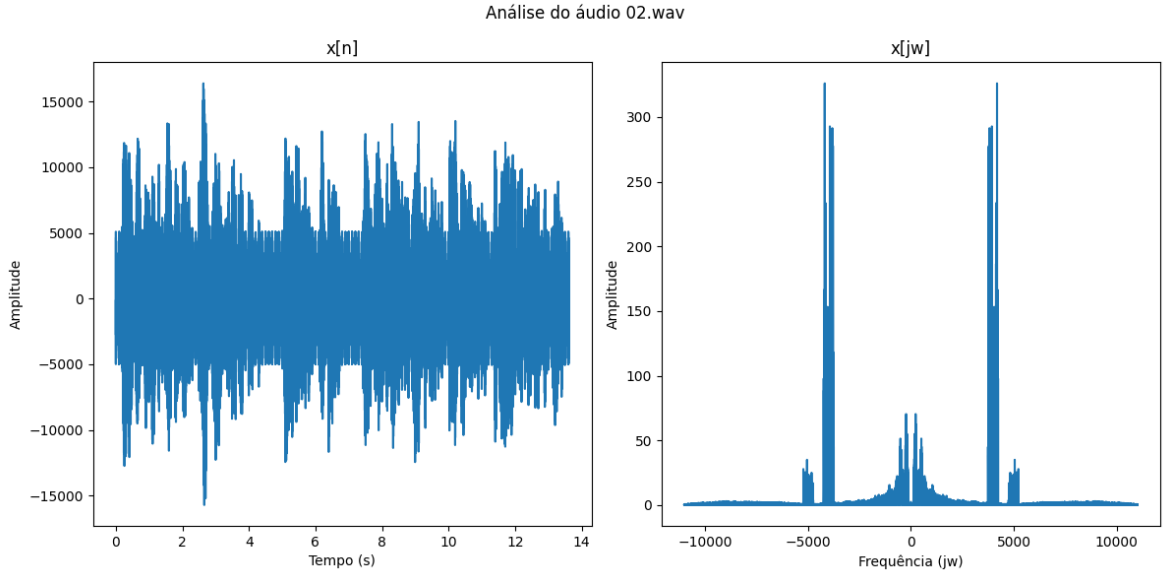


Figure 2: Análise do áudio 2



3. Implementar o filtro h[n] com os parâmetros ωc e L, tendo como base a fundamentação teórica indicada.

Solution: O filtro foi implementado a partir do código abaixo:

```
from scipy.signal import butter

# 3. Filtro h[n]
def low_pass_filter(wc, l, sample_rate):
    # Design low-pass filter
    nyq = 0.5 * sample_rate # Applying Nyquist Theorem
    normal_wc = wc/nyq
    b, a = butter(l, normal_wc, btype="lowpass")
    return b, a
```

Ainda sobre a implementação desse filtro, a biblioteca scipy fornece o filtro Butterworth para que possamos utilizá-los. O filtro Butterworth é um dos mais utilizados devido à sua resposta maximamente plana na banda passante. A implementação do filtro, matemáticamente, é essa:

$$H(\omega) = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2L}}} \tag{1}$$

Virtanen et al. (2020)

onde ω_c é a frequência de corte e L é a ordem do filtro, que define a inclinação da atenuação na banda de transição. Sobre o uso do filtro através da função, utilizamos o Teorema de Nyquist, que estabelece que a frequência de amostragem f_s deve ser pelo menos duas vezes a maior frequência presente no sinal Oppenheim (2010); Oppenheim and Schafer (2013):

$$\text{nyq} = \frac{\text{sample\_rate}}{2} \tag{2}$$

A normalização da frequência de corte é feita por:

$$\text{normal\_wc} = \frac{\omega_c}{\text{nyq}} \tag{3}$$

A forma de diferença finita do filtro, O filtro Butterworth retorna coeficientes (b, a) que descrevem a seguinte equação de diferença:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_L z^{-L}}{1 + a_1 z^{-1} + \cdots + a_L z^{-L}} \tag{4}$$

Essa equação representa um filtro digital implementado via a transformação bilinear:

$$s \to \frac{1 - z^{-1}}{1 + z^{-1}} \tag{5}$$

O parâmetro L define a inclinação da transição, na qual, é suave a transição entre as bandas é suave para valores pequenos de L. Enquanto que para valores grandes de L, a atenuação é mais abrupta, mas pode introduzir efeitos como atraso na resposta impulsiva.

4. Plotar o gráfico de h[n] no tempo, a magnitude e a fase de sua resposta em frequência.

Solution: A fase, magnitude e o filtro projetado h[n] no tempo, podendo ser visualizado logo abaixo:

A partir da imagem, de ambos os áudios, podemos observar alguns fatores como atraso de grupo dependente da frequência devido a variações lineares no filtro h[n], principalmente, visto em cenários que envolve comunicações e áudio. Também, pela fase não linear do filtro, prova que não se trata de um sinal que goza da propriedade de estabilidade. Além desses fatores citados anteriormente, podemos também concluir que temos modificações decorrentes de truncamentos realizados de maneira automática pelos computadores que acaba influenciando na fase do sinal.

5. Aplicar o filtro projetado e analisar o impacto da qualidade da filtragem ao variar os parâmetros L e ωc.

Solution: O filtro foi projetado e testado para diferentes valores de frequência de corte (ωc) e ordem do sinal (L). Intuitivamente, decidimos partir de uma frequência de 800, variando de 100 em 100, até chegarmos em 1300, o valor final de 1300 foi o valor escolhido como melhor após análise qualitativa e subjetiva do áudio filtrado, além disso, os valores de L foram variados partindo de 3 e indo até 7,

Figure 3: h[n] no domínio do tempo

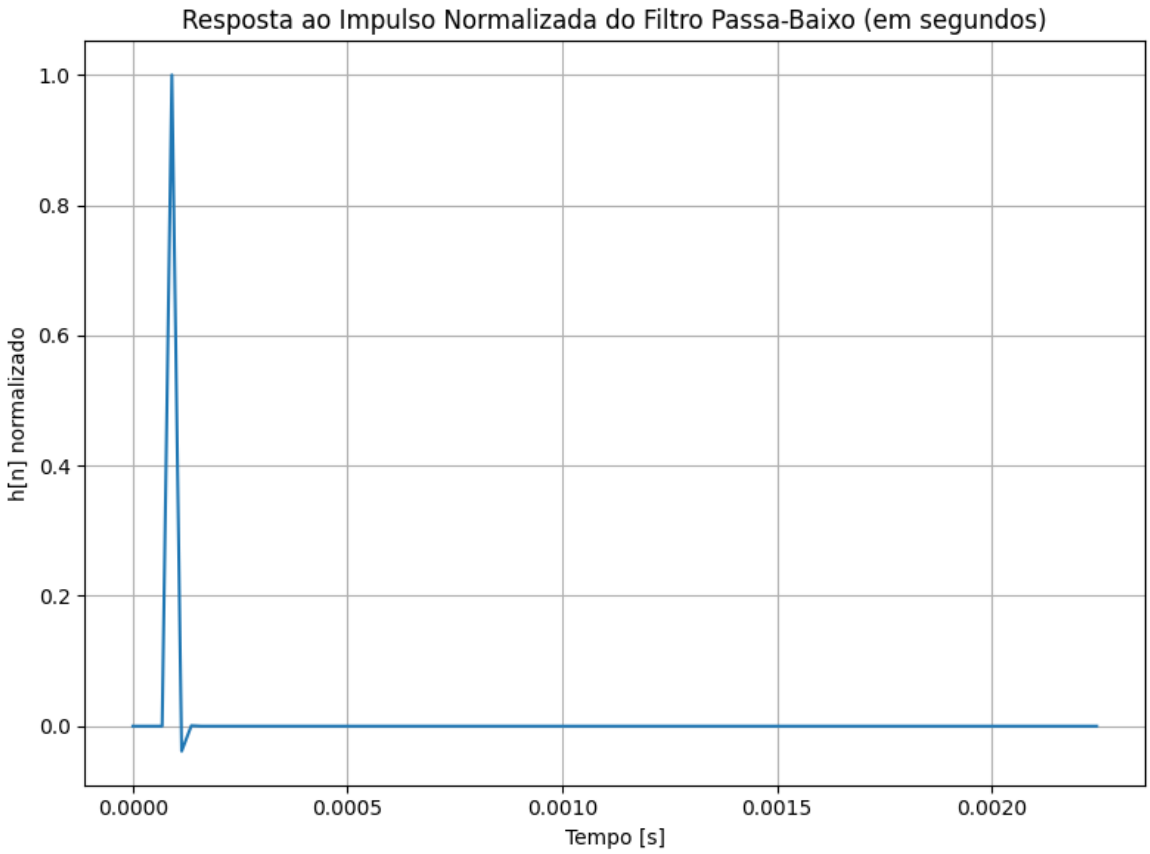


Figure 4: Áudio 1 - Visualização da magnitude e fase do sinal

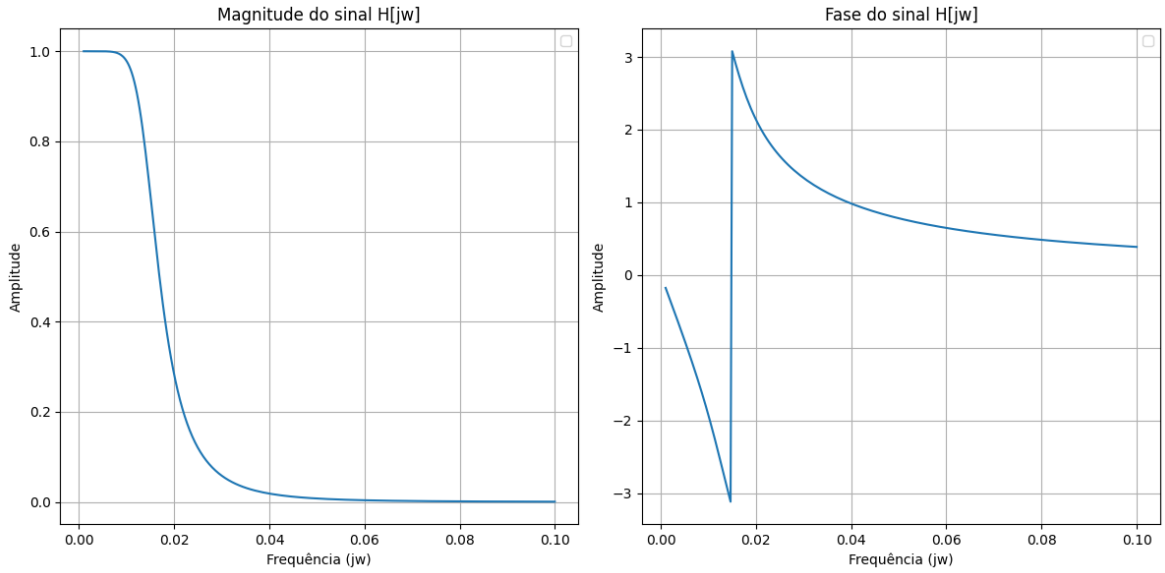
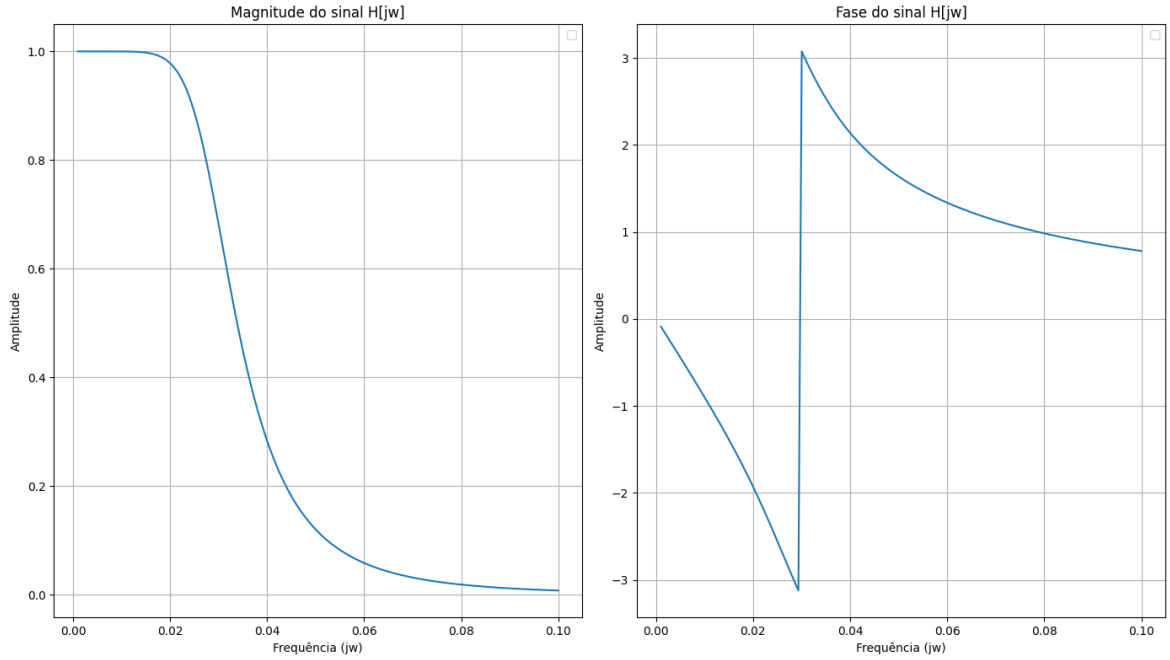


Figure 5: Áudio 2 - Visualização da magnitude e fase do sinal



variando de 1 em 1. Essa metodologia foi aplicado em ambos os áudios. A seguir, mostraremos uma série de gráficos referente aos diferentes parâmetros testados.

A seguir, usando a mesma metodologia, fazemos também para o áudio 2:

6. Plotar o sinal filtrado no tempo e na frequência e comparar o antes e depois.

Solution: Após aplicação do filtro passa-baixa projetado no sinal original, tivemos resultados significativos, ou seja, o ruído contido tanto no áudio 1 quanto no 2 foram devidamente eliminados, veja visualmente como a aplicação do filtro passa-baixa influenciou, diante de uma perspectiva visual, no sinal original:

Ao final, visualmente, não foi possível notar diferenças tão significativas com a mudança dos parâmetros, mas ao ouvir os áudios variando os parâmetros é possível notar muita diferença.

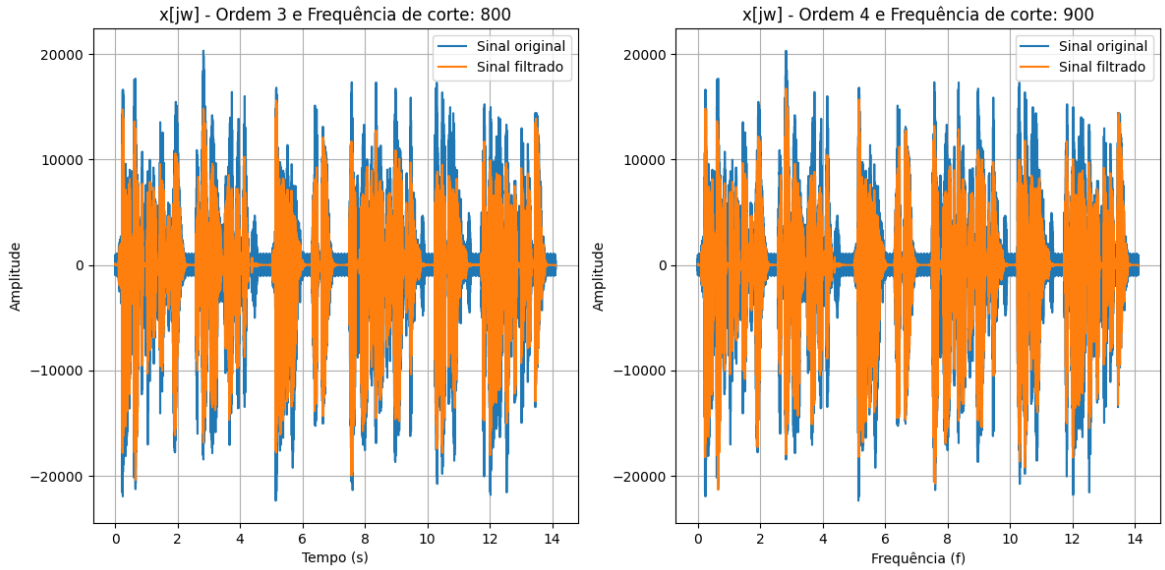
7. Mensurar de forma subjetiva a qualidade da filtragem ao escutar o áudio filtrado.

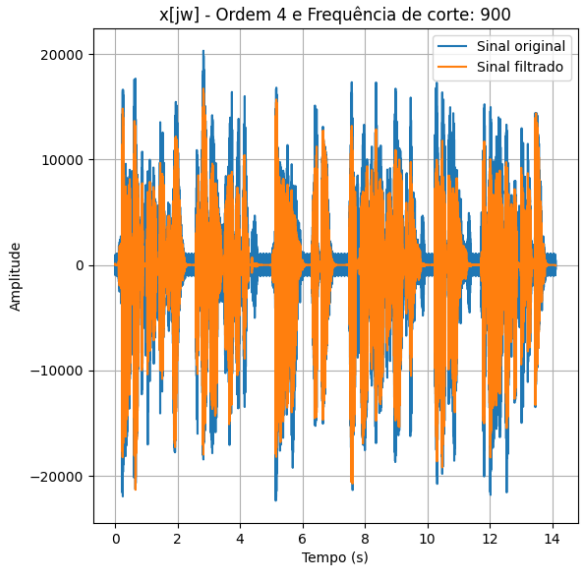
Solution: Após analisar os áudios filtrados, tanto o áudio 1 quanto o áudio 2, pudemos observar uma melhora significativa na qualidade do áudio, principalmente, analisando na remoção do ruído um pouco mais agudo que a voz padrão presente no áudio. Com a aplicação do nosso filtro passa-baixa também percebeu-se que a voz padrão no áudio ficou um pouco abafada, ou seja, o uso do filtro no sinal original acabou influenciando em janelas do sinal que também eram imprescindíveis para a clareza da voz padrão no áudio, o que é totalmente esperado uma vez que, o filtro não remove completamente apenas o ruído que queremos, mas também remove algumas informações qualitativas para o entendimento do áudio, a grande questão é entender o melhor trade-off para cada cenário.

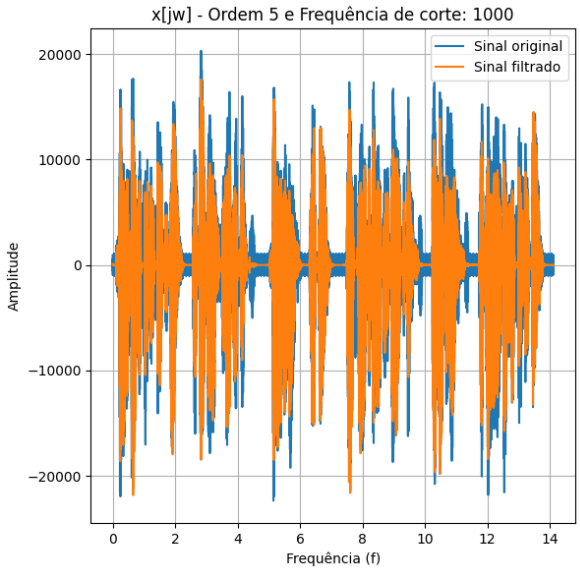
4 Conclusão

Ao fim desse trabalho computacional, foi possível compreender, de maneira prática e interativa, a partir de um problema de engenharia a potencialidade do uso de processamentos digitais de sinais, especificamente a leitura de áudios usando Python, o conceito de filtros, seus usos e tipos, representação de um sinal no domínio do tempo e a partir da aplicação da transformada de fourier encontrar o mesmo sinal no domínio da frequência e enteder a potencialidade do Python para análises de sinais. A grande importância do uso de filtros para filtrar sinais e como isso consegue influenciar outras áreas como a de sistemas de multimídias e telecomounicações. Outro ponto importante, foi o avanço na aprendizagem ao combinar o conhecimento teórico previamente estabelecido a partir das aulas em prática nesses trabalhos.

Figure 6: Áudio 1







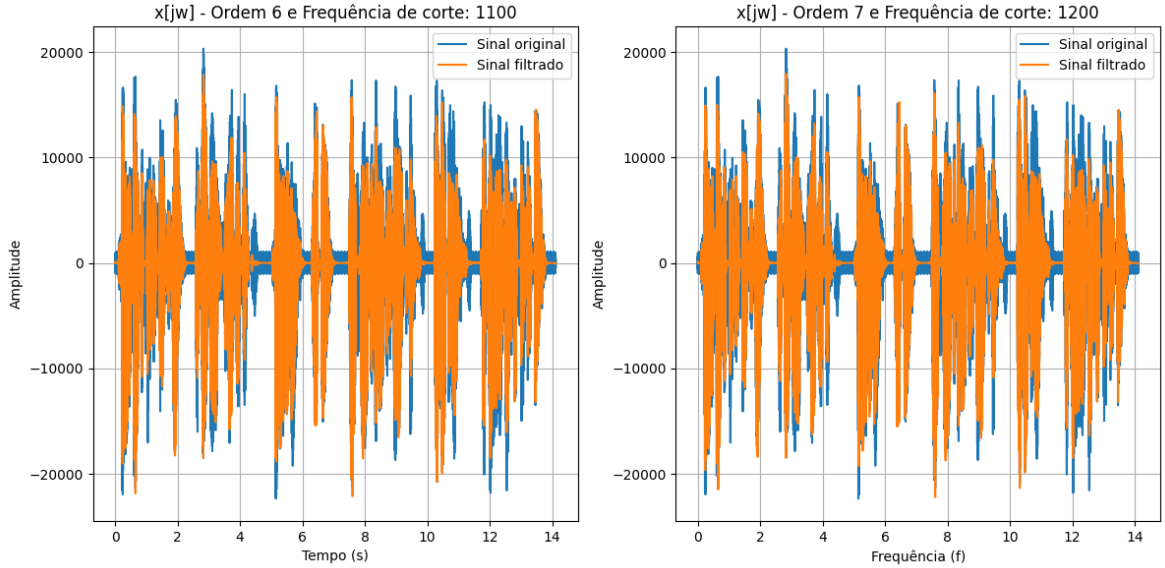
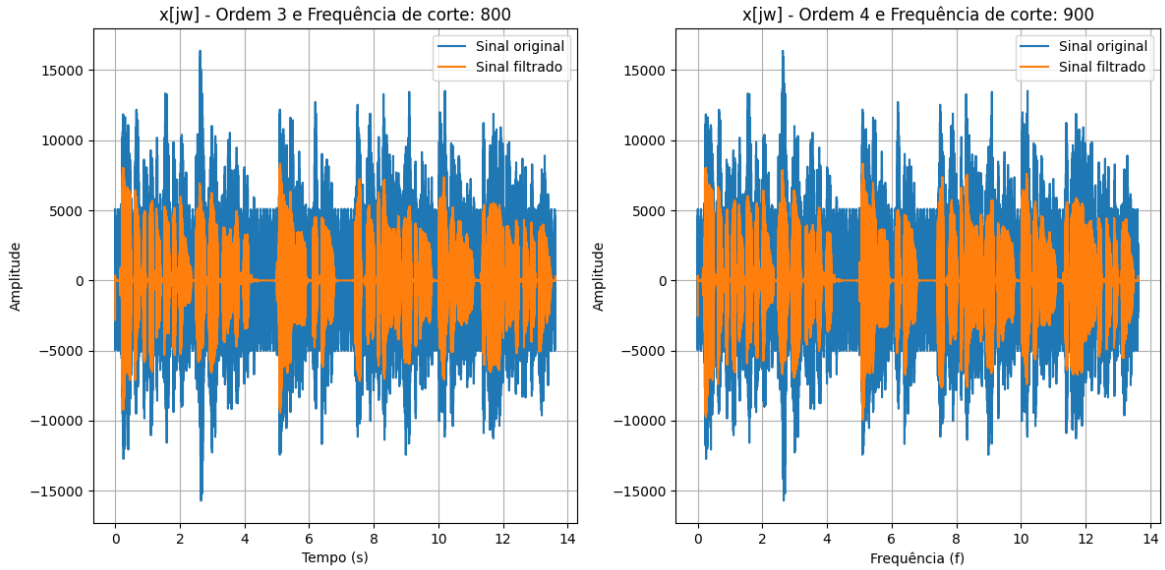
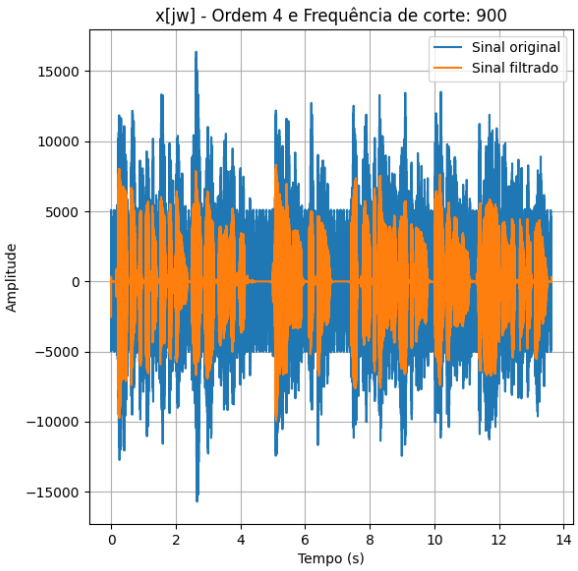
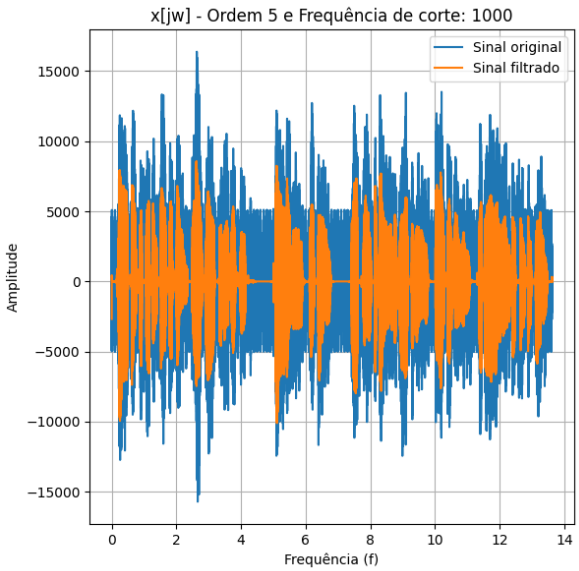
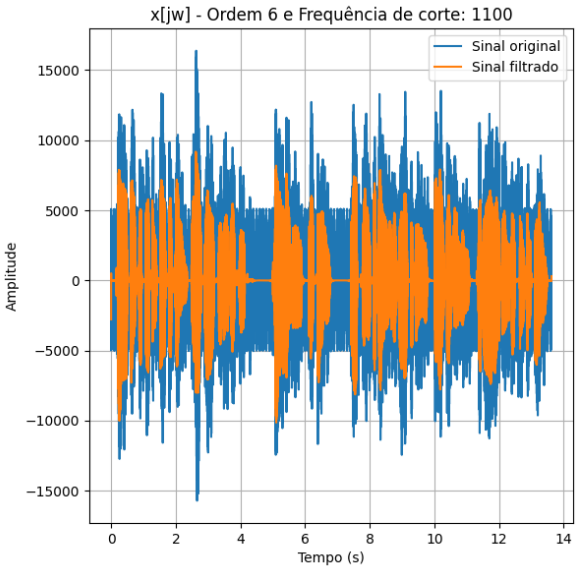


Figure 7: Áudio 2




x[jw] - Ordem 4 e Frequência de corte: 900
Sinal original
Sinal filtrado
Amplitude
Tempo (s)


x[jw] - Ordem 5 e Frequência de corte: 1000
Sinal original
Sinal filtrado
Amplitude
Frequência (f)


x[jw] - Ordem 6 e Frequência de corte: 1100
Sinal original
Sinal filtrado
Amplitude
Tempo (s)

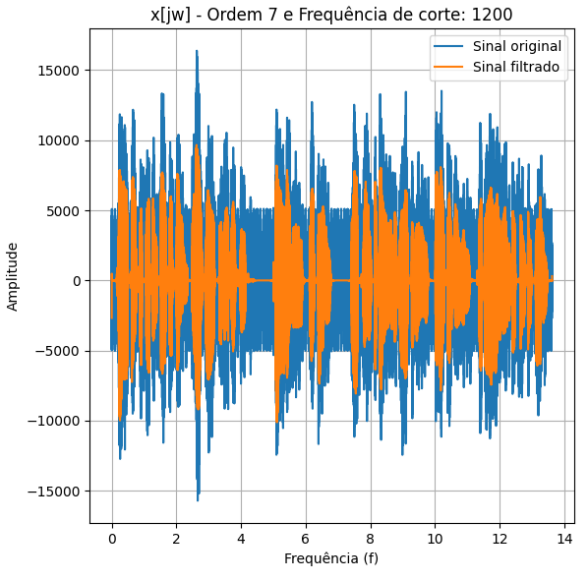

x[jw] - Ordem 7 e Frequência de corte: 1200
Sinal original
Sinal filtrado
Amplitude
Frequência (f)

Figure 8: Filtro Passa Baixa - Audio 1

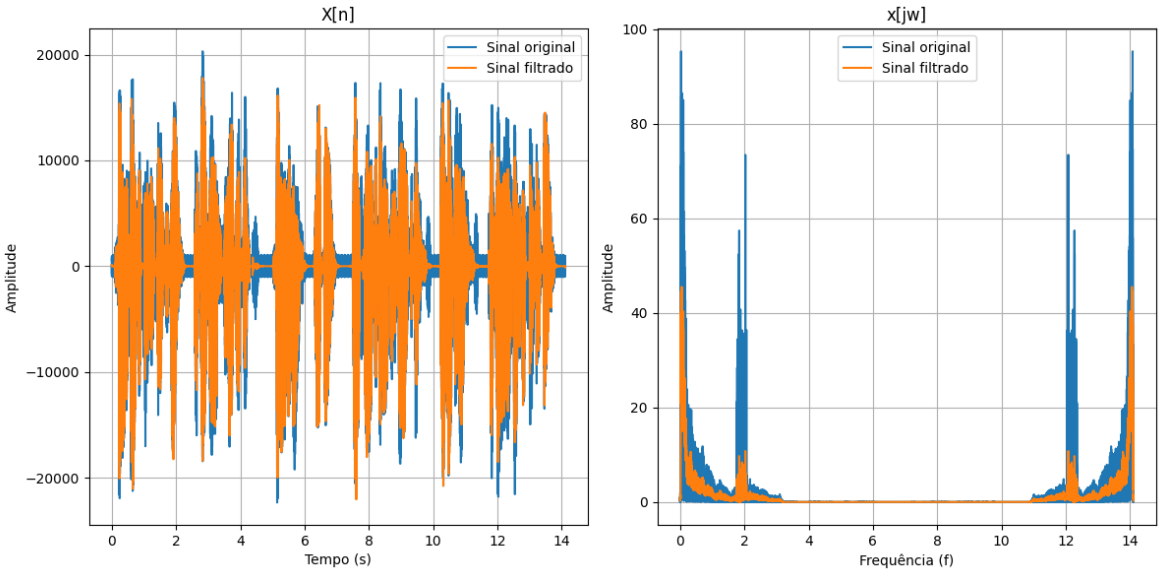
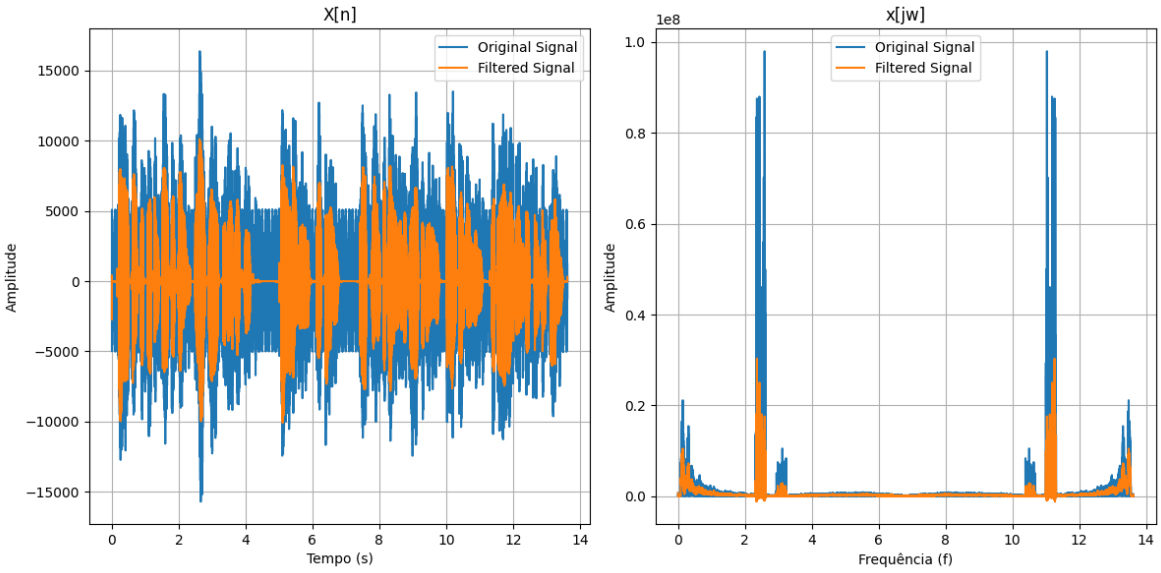


Figure 9: Filtro Passa Baixa - Audio 2



References

Oppenheim, A. V. (2010). *Sinais e sistemas*. Prentice-Hall.

Oppenheim, A. V. and Schafer, R. W. (2013). Processamento em tempo discreto de sinais. *Tradução de Daniel Vieira*, 3.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.