Universidade Tecnológica Federal do Paraná
Câmpus Apucarana
Departamento de Engenharia de Computação
Processamento Digital de Sinais

ATIVIDADE PRÁTICA 03:
FILTROS

EDUARDO YUJI YOSHIDA YAMADA, RA: 2320606

SÉFORA DAVANSO DE ASSIS, RA: 2367777

FÁBIO PEREIRA

APUCARANA - PR
2024

Filtros Sonoros

Para este exercício utilizamos como sinal de entrada o mesmo sinal do Trabalho 2, ou seja, a soma do áudio com o assovio, e projetamos um filtro rejeita-faixa para remover o sinal de assovio com a Linguagem Python.

Especificação do Filtro:

A seguir temos a resposta em frequência do sinal de entrada e definimos a banda de rejeição do filtro em: 1000 até 2000 Hz, o que podemos ver no gráfico e no código a abaixo:

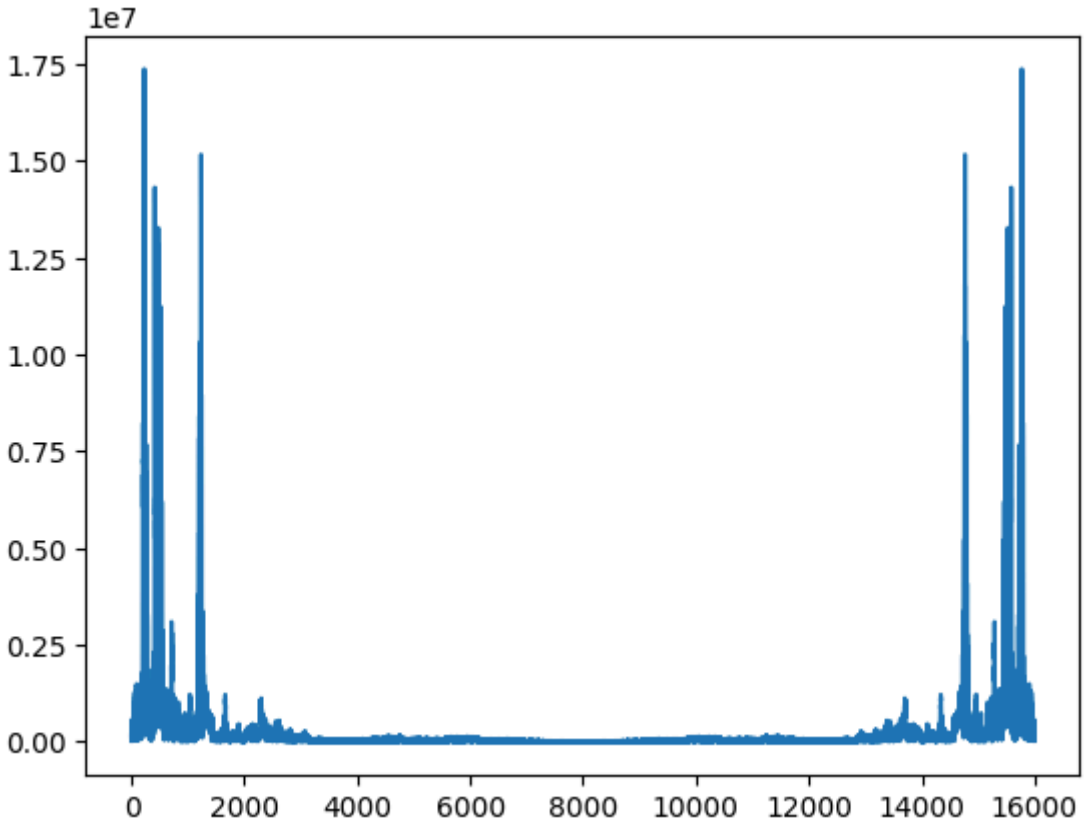

Figura 1: Gráfico de resposta em frequência do sinal de entrada.
Fonte: Autoria própria.

```
# -*- coding: utf-8 -*-
"""At3.ipynb

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1egbNI83JidHXJjNztapcs8V9muzlUyUl
"""

import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile
from IPython.display import Audio, display
from scipy import signal
```

```
from scipy.signal import freqz

rate, data = wavfile.read("y1.wav")
rate, len(data)

X = np.fft.fft(data)
freqs = np.linspace(0, rate, len(data))
plt.plot(freqs, np.abs(X))
```

Fonte: Autoria própria.

Projeto do Filtro:

Usando Python foi possível projetar um Filtro IIR Butterworth de ordem 4, dado que somos do Grupo C. Os coeficientes encontrados foram: [0.59630237, -4.04416589, 12.67064433, -23.75859186, 29.07674731, -23.75859186, 12.67064433, -4.04416589, 0.59630237] e [1 , -5.91468303, 16.16694843, -26.51038408, 28.45809905, -20.46048452, 9.63001584, -2.71996386, 0.35557738]

Encontrados por meio do código abaixo e depois plotamos a resposta em frequência do Filtro e os polos e os zeros.

```
def plota_b_a(b, a, fs):
    w_res, h_res = signal.freqz(b, a, fs=fs)
    plt.plot(w_res, np.abs(h_res), label='resp freq')
    plt.legend()
    plt.title('Resp do filtro')

b, a = signal.butter(N=4, Wn = [1000, 2000], btype='bandstop', output='ba',
fs=16000)
plota_b_a(b, a, 16000)

print(b)

print(a)

filtered_data = signal.lfilter(b, a, data)

#filtered_data = filtered_data / np.max(np.abs(filtered_data))

plt.plot(freqs, np.abs(np.fft.fft(filtered_data)))

# Reproduzir áudio original
print("Áudio original:")
display(Audio(data, rate=rate))

# Reproduzir áudio filtrado
print("Áudio filtrado:")
display(Audio(filtered_data, rate=rate))

!pip install zplane
import zplane as zp

# Polos e zeros:
tf = signal.TransferFunction(b, a, dt=1/16000)
zp.pz(tf)
```

Fonte: Autoria própria.

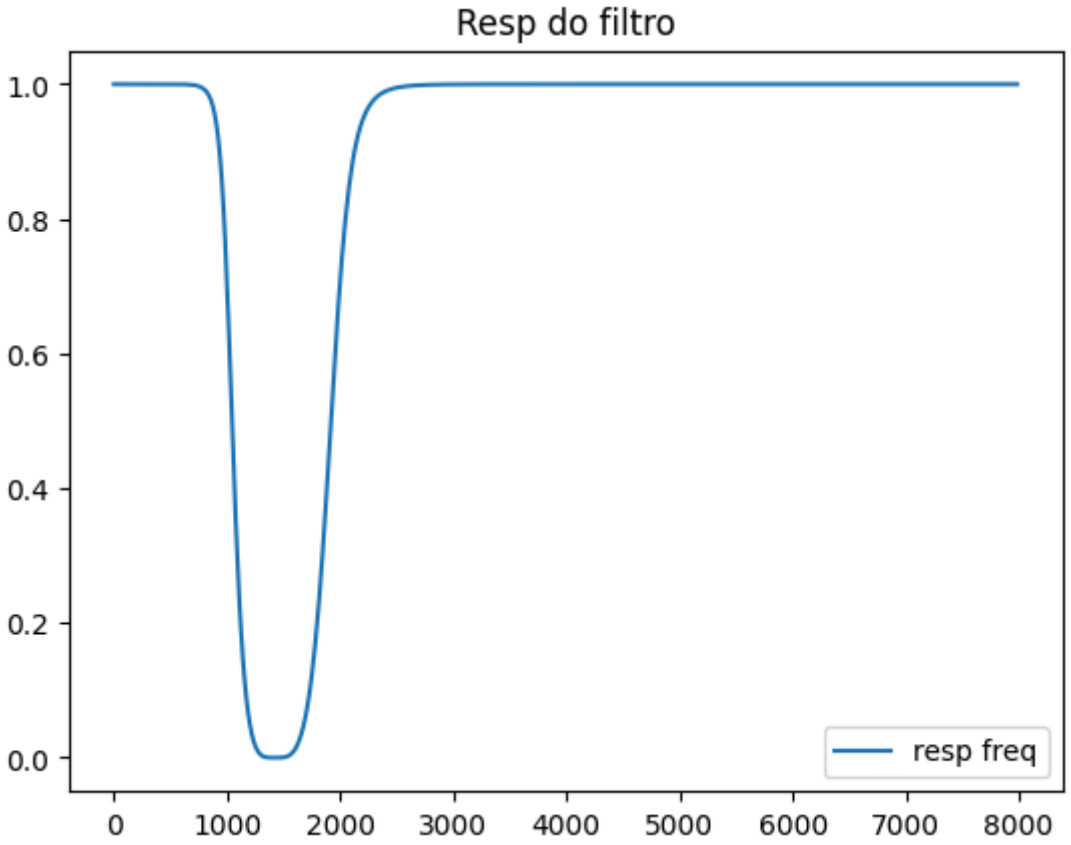

Figura 2: Resposta em frequência do Filtro.
Fonte: Autoria própria.

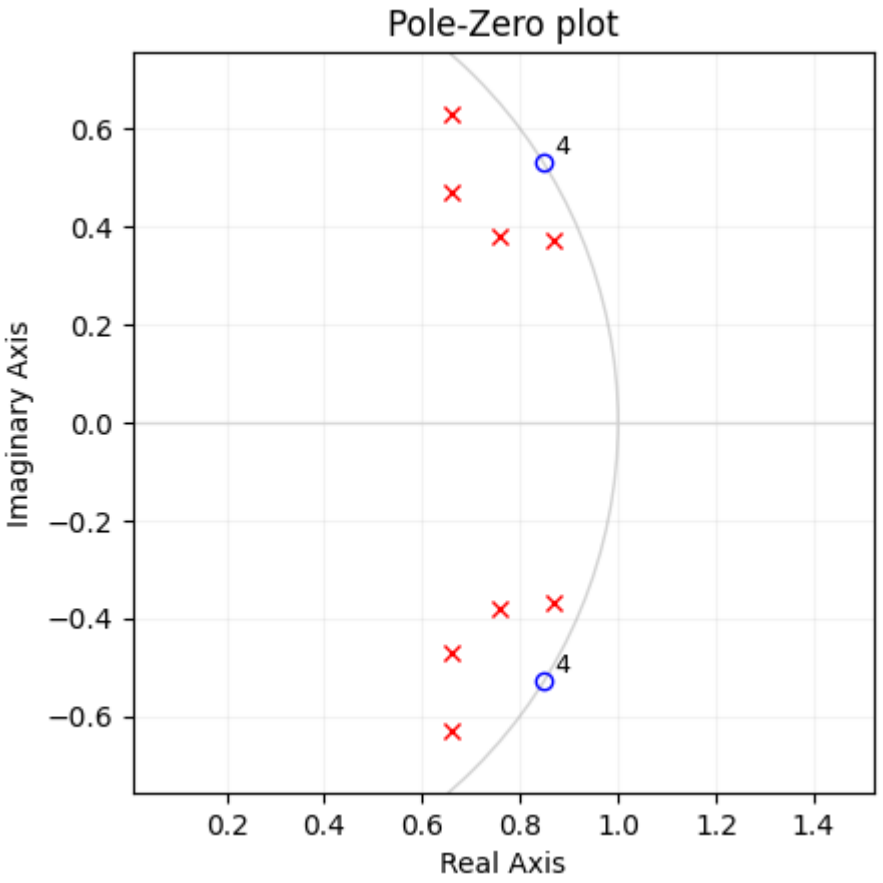

Figura 3: Polos e zeros.
Fonte: Autoria própria.

Conclusão:

Depois de plotar o espectro do sinal de saída, percebemos que ele se assemelha muito ao inicial da Figura 1, além de que ao escutar o sinal de saída pode-se concluir que o filtro foi bem executado, dado que apenas a voz pode ser ouvida, sem o assovio.

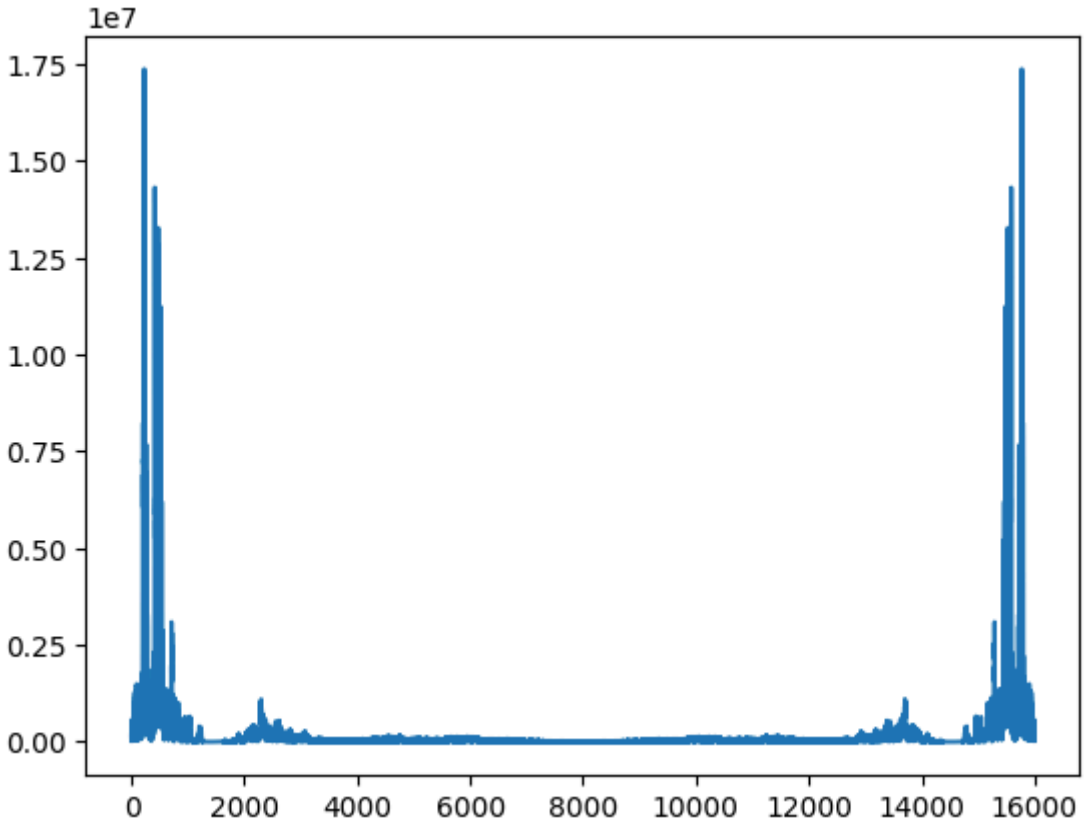

Figura 4: Espectro sinal de saída.
Fonte: Autoria própria.