Nome: Victor Yaegashi Setti

RA: 206362

Importando as bibliotecas que serão usadas e definindo pi = π

In [115…
```
import math
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
import scipy.io as spio
import IPython.display as ipd

pi = np.pi
```

Ignorando os warnings causados pelas funções `kaiser()` e `wavfile.read()`

In [116…
```
import warnings
warnings.filterwarnings("ignore", category=SyntaxWarning)
warnings.filterwarnings("ignore", category=spio.wavfile.WavFileWarning)
```

Copiando a função `kaiser()`

In [117…
```
def kaiser(wp,wr):

    wc = (wp + wr)/2
    d = 0.01
    Ap = 20*math.log10((1+d)/(1-d))
    Ar = -20*math.log10(d)

    if Ar < 21:
        b = 0
        D = .9222

    elif Ar < 50:
        b = 0.5842*(Ar-21)**0.4+0.07886*(Ar-21)
        D = (Ar - 7.95)/14.36
    else:
        b = .1102*(Ar-8.7)
        D = (Ar - 7.95)/14.36

    k = math.ceil(math.pi*D/(wr-wp)-.5)
    M = 2*k+1

    n = np.arange(-k,k+1,1)

    w = np.i0(b*np.sqrt(1-(4/M**2)*n**2))
    w = np.divide(w,np.i0(b))

    h = wc/math.pi*np.sinc(wc*n/math.pi)*w

    return h
```

Importando a função `espectro()` (PS: Essa função será chamada com ';' no final para suprimir o output indesejado)

In [118…
```
""" Rotina que exibe o espectro de magnitude (X(ejw)) de um sinal discret
def espectro(y):

    #modulo da transf. de Fourier
    Y = np.abs(np.fft.fft(y))
    #frequencias avaliadas
    w = np.linspace(0,2*math.pi,Y.size)

    #exibe o grafico do espectro
    plt.figure()
    plt.plot(w,Y)
    plt.xlabel('$\Omega$ [rad]', fontsize=10)
    plt.ylabel('|$Y(e^{j\Omega})$|', fontsize=10)
    plt.grid(True)
    plt.xlim((0,2*math.pi))
    plt.show()

    return Y,w
```

1) Amostragem e Aliasing

Carregando o arquivo de áudio e tornando-o Mono

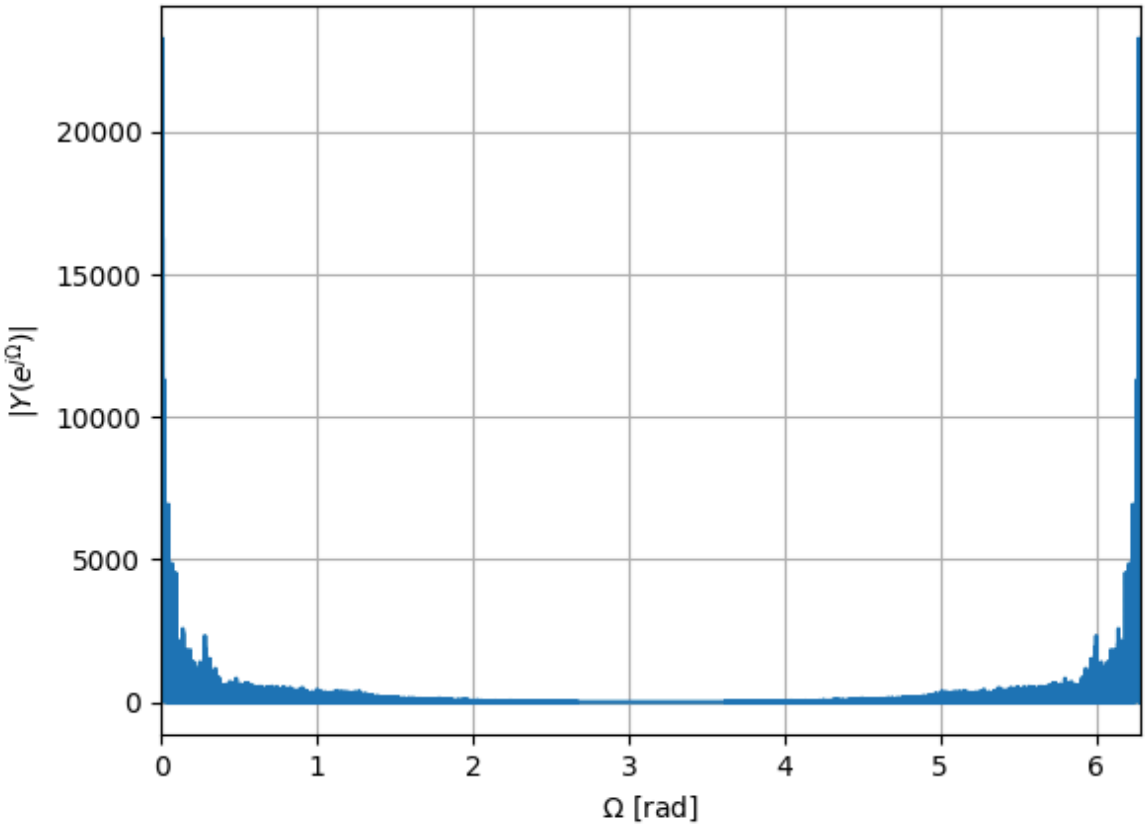
In [119…
```
Fs, y = spio.wavfile.read("creed_overcome.wav")

y = (y[:,0] + y[:,1]) / 2
```

a) Análise do conteúdo espectral

Utilizando a função espectro para plotar o áudio em um gráfico

In [120…
```
espectro(y);
```



Discussão:

Nota-se que há uma maior concentração de energia nos valores de Ω próximos de múltiplos de 2π, indicando que a maior parte da energia está nas frequências graves.

De mesmo modo, nota-se uma menor concentração de energia nos valores de Ω próximo de π, indicando que os componentes de alta frequência, ou seja, frequências mais agudas, têm menores intensidades.

Sobre a relação da frequência digital Ω com a frequência analógica f, sabe-se que:

$$f = \frac{\Omega}{2\pi} \cdot f_s$$

Com f_s sendo a taxa de amostragem que, para este sinal, é 44.1 kHz.

Assim, comparando Ω e f, temos que as bandas estão em 0 Hz e 44.1 kHz, já o meio, onde há a menor quantidade de energia, está em 22.05 kHz.

b) Decimação do sinal com M = 6

Implementando uma função que faz a decimação

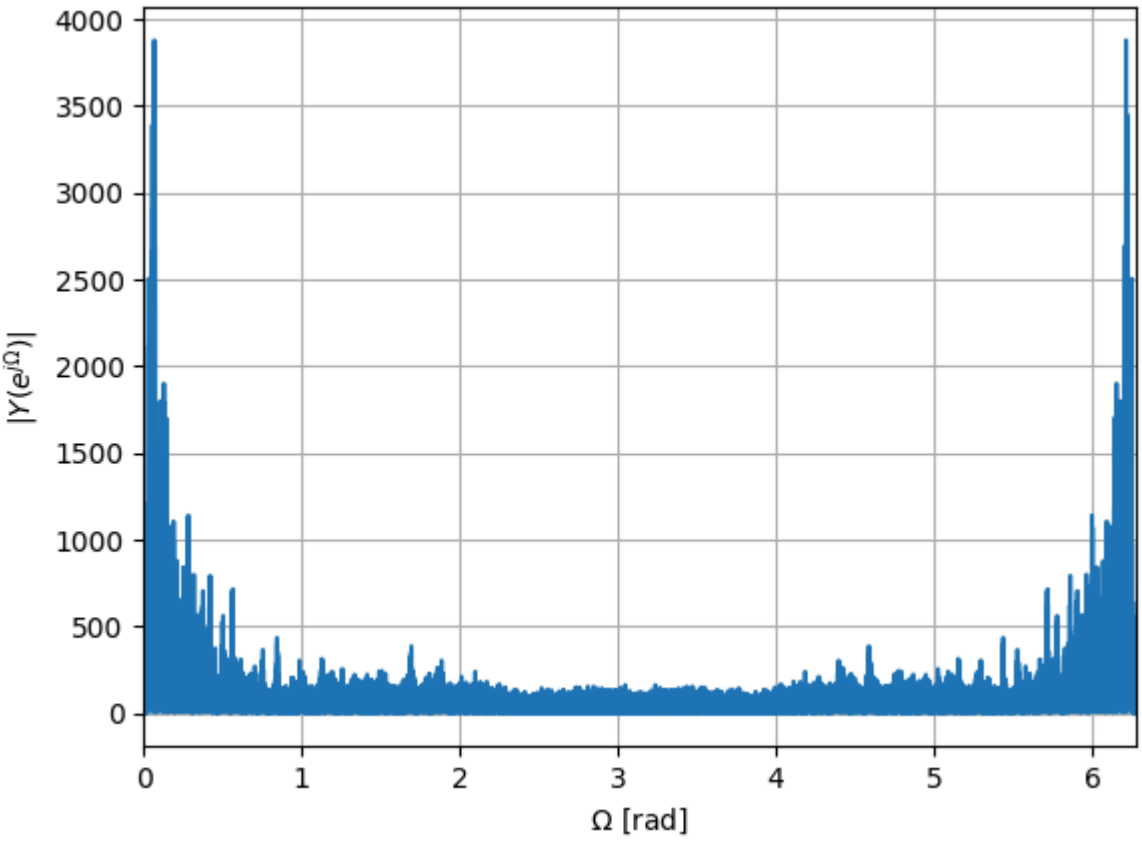
In [121…
```
def decimacao(y, m):
    return y[::m]
```

Fazendo a decimação com M = 6

In [122…
```
y_dec = decimacao(y, 6)
```

Plotando o espectro do sinal decimado `y_dec`

In [123…
```
espectro(y_dec);
```



Discussão:

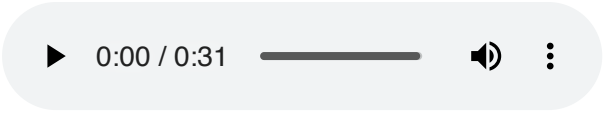
Nota-se que a energia apresenta-se de maneira mais bem distribuída ao longo do gráfico. Os pontos próximos de π não estão mais tão abaixo que os pontos próximos de 0 e 2π. Isso é resultado do aliasing e denota a perda de alguns aspectos importantes da música original, como alguns agudos.

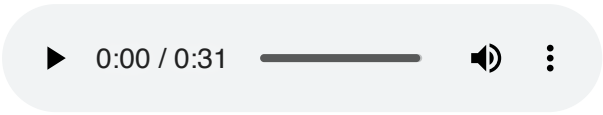
Nota-se também que a amplitude de todos os pontos está bem menor pois, ao decimar com fator M = 6, há uma divisão dessa amplitude pelo mesmo fator, uma vez que a energia total do sinal também é reduzida nessa proporção. O que é evidente uma vez que os valores mais altos do espectro do sinal original estavam em torno de 24000 e agora estão em torno de 4000. Vale ressaltar que esse diminuição não representa uma perda de energia real, apenas uma escala normalizada adequada ao novo domínio decimado.

Fazendo o display dos áudios `y` e `y_dec`, respectivamente

In [124…
```
ipd.display(ipd.Audio(y, rate=Fs))

Fs_dec = Fs / 6
ipd.display(ipd.Audio(y_dec, rate=Fs_dec))
```





Discussão:

É visível como o áudio decimado perde qualidade em diversos aspectos. Conforme previsto, há uma tendência de "homogenização" do áudio, uma vez que ele perde aspectos mais agudos. Um exemplo notável são os pratos da bateria que, no original, são marcantes e constantes, mas no decimado se mostram ausentes. Após a decimação, o som se torna mais abafado, distante, áspero e metálico.

Importante ressaltar também a presença de uma espécie de chiado ou ruído quando o vocalista canta palavras com "s". Isso é bem notável próximo da marca de tempo `0:21`, durante o trecho "now it'**s** my turn to **s**peak".

c) Minimização do Aliasing

Criando FPB para Ω_p = 0,45 [rad] e Ω_r = 2 [rad]

In [125…
```
h1 = kaiser(0.45, 2)
```

Criando FPB para Ω_p = 0,45 [rad] e Ω_r = 0,50 [rad]

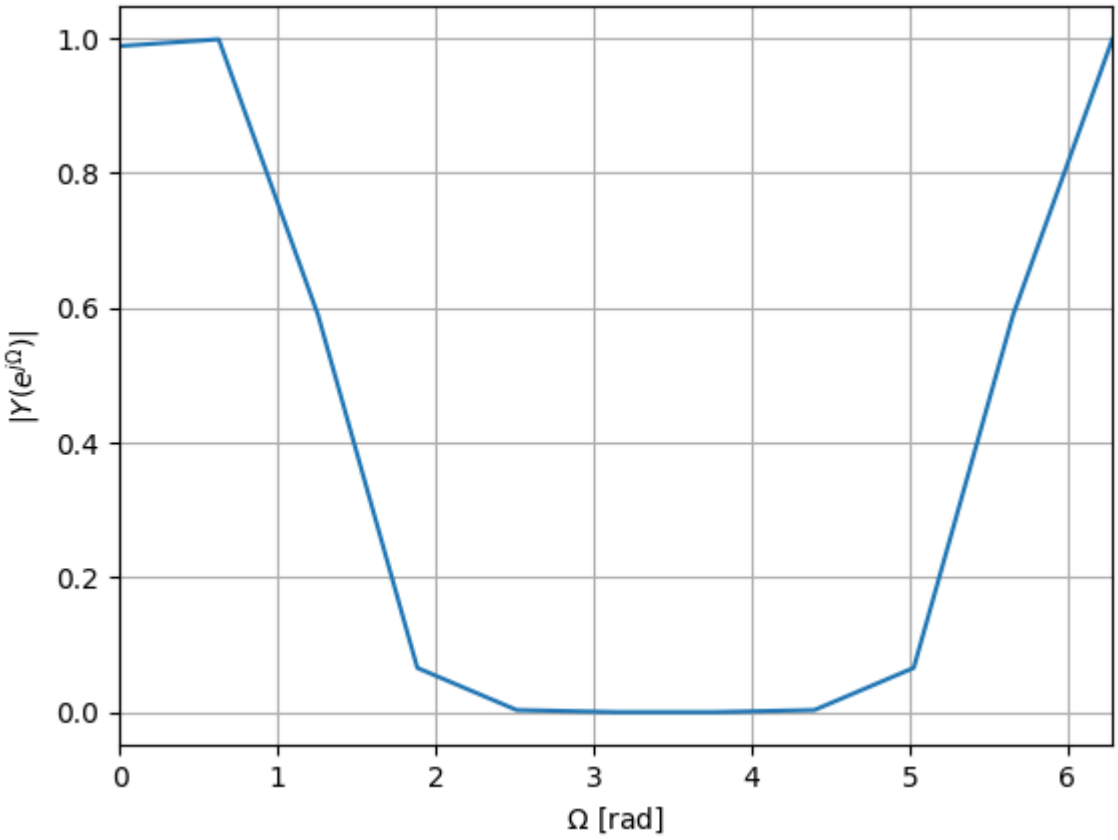
In [126…
```
h2 = kaiser(0.45, 0.50)
```

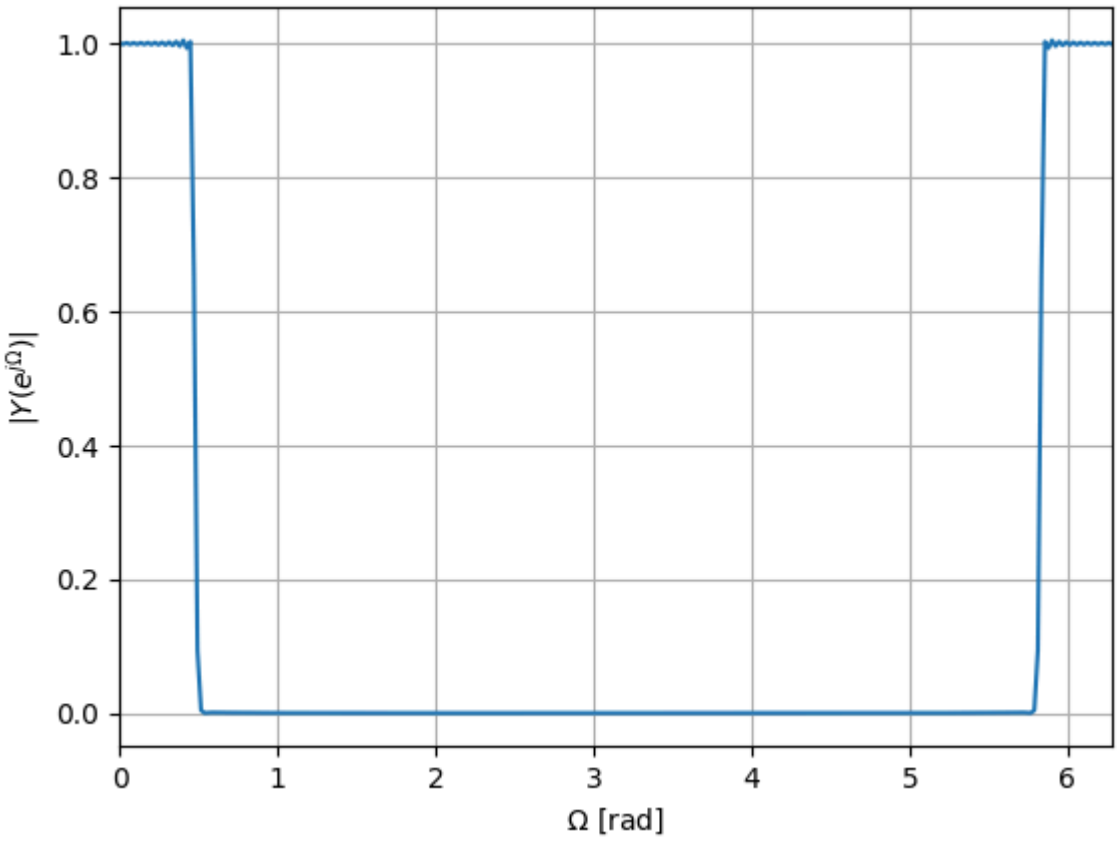
Criando FPB para Ω_p = 1,50 [rad] e Ω_r = 2 [rad]

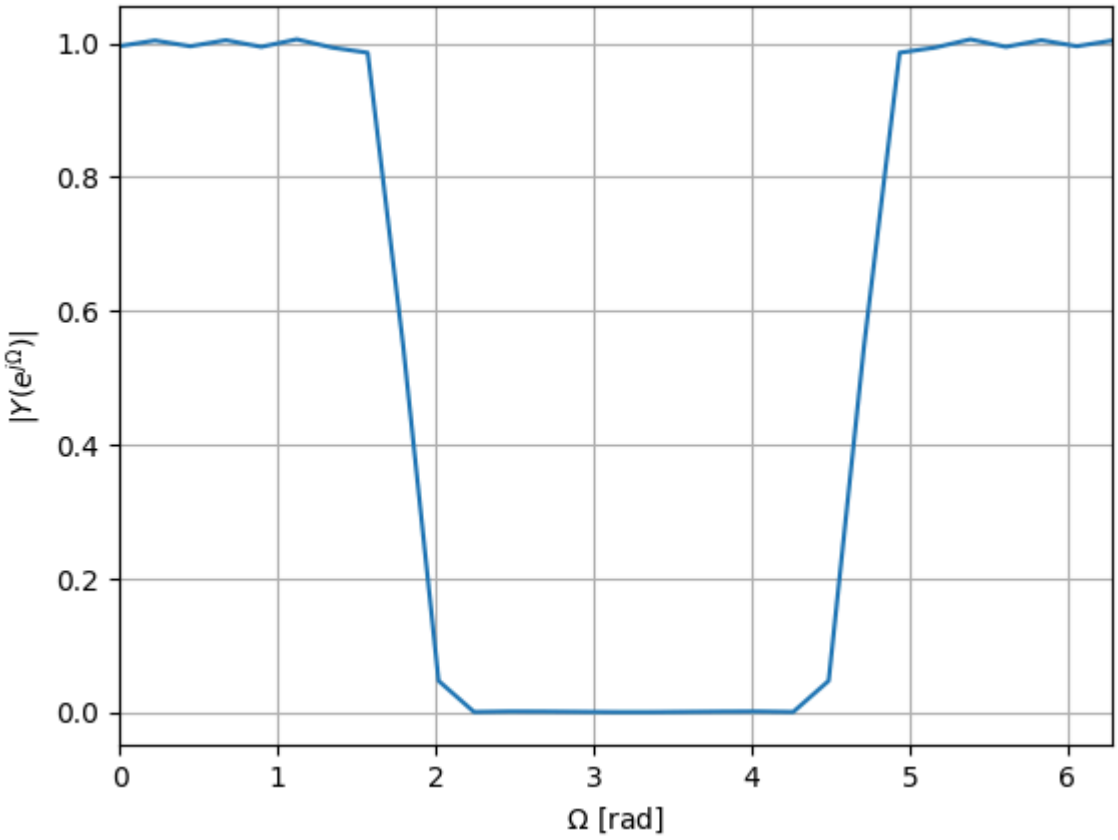
In [127…
```
h3 = kaiser(1.50, 2)
```

Plotando os filtros em um gráfico

In [128…
```
espectro(h1)
espectro(h2)
espectro(h3);
```


1.0
0.8
0.6
0.4
0.2
0.0
|Y(e^jΩ)|
0
1
2
3
4
5
6
Ω [rad]


1.0
0.8
0.6
0.4
0.2
0.0
|Y(e^jΩ)|
0
1
2
3
4
5
6
Ω [rad]



Discussão:

Nota-se, claramente, que o FPB gerado pelos valores Ω_p = 0,45 [rad] e Ω_r = 0,50 [rad] é o mais próximo de um FPB ideal, uma vez que este apresenta pouca oscilação antes e depois da banda de rejeição, bem como apresenta uma queda suficientemente brusca de 0 para 1.

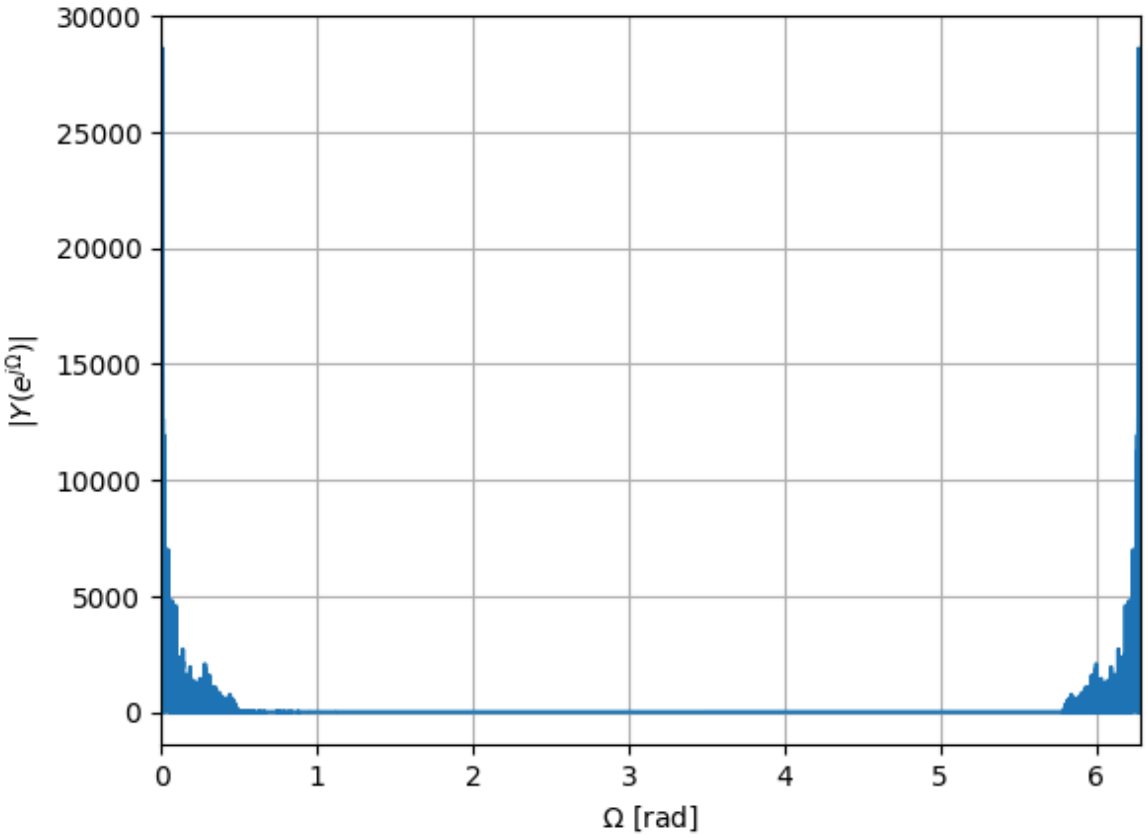
d) Filtrando o sinal

Convoluindo o sinal `y` com o FPB `h2`, gerado com os valores Ω_p = 0,45 [rad] e Ω_r = 0,50 [rad]

In [129…
```
saida = sp.signal.fftconvolve(y, h2, mode="full")
```

Plotando o espectro da saída após a convolução com o FPB

In [130…
```
espectro(saida);
```



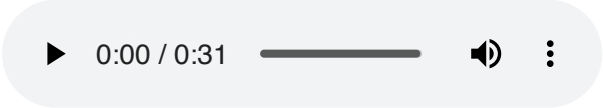
Discussão:

Nota-se que a concentração da energia nos pontos próximos de 0 e 2π se tornou ainda maior, uma vez que, após passar pelo FPB, conforme o esperado, as frequências altas próximas de π foram barradas, atenunando ainda mais as frequências baixas nos cantos do gráfico.

Toda a região do meio do gráfico apresenta-se basicamente nula. Isso causa a mesma perda de agudos que a decimação feita no item c) causou, mas prepara o terreno para que o aliasing não ocorra, uma vez que já espurga as frequências altas agora.

Fazendo o display da `saida`

In [131…
```
ipd.display(ipd.Audio(saida, rate=Fs))
```



Discussão:

Nota-se que, conforme esperado, os sons agudos, como os pratos da bateria, não são possíveis de se escutar. A qualidade do áudio caiu, tornando-o mais distante e vazio, e alguns sons agudos se perderam, mas a voz do vocalista e os instrumentos mais graves não apresentaram tanta distorção.

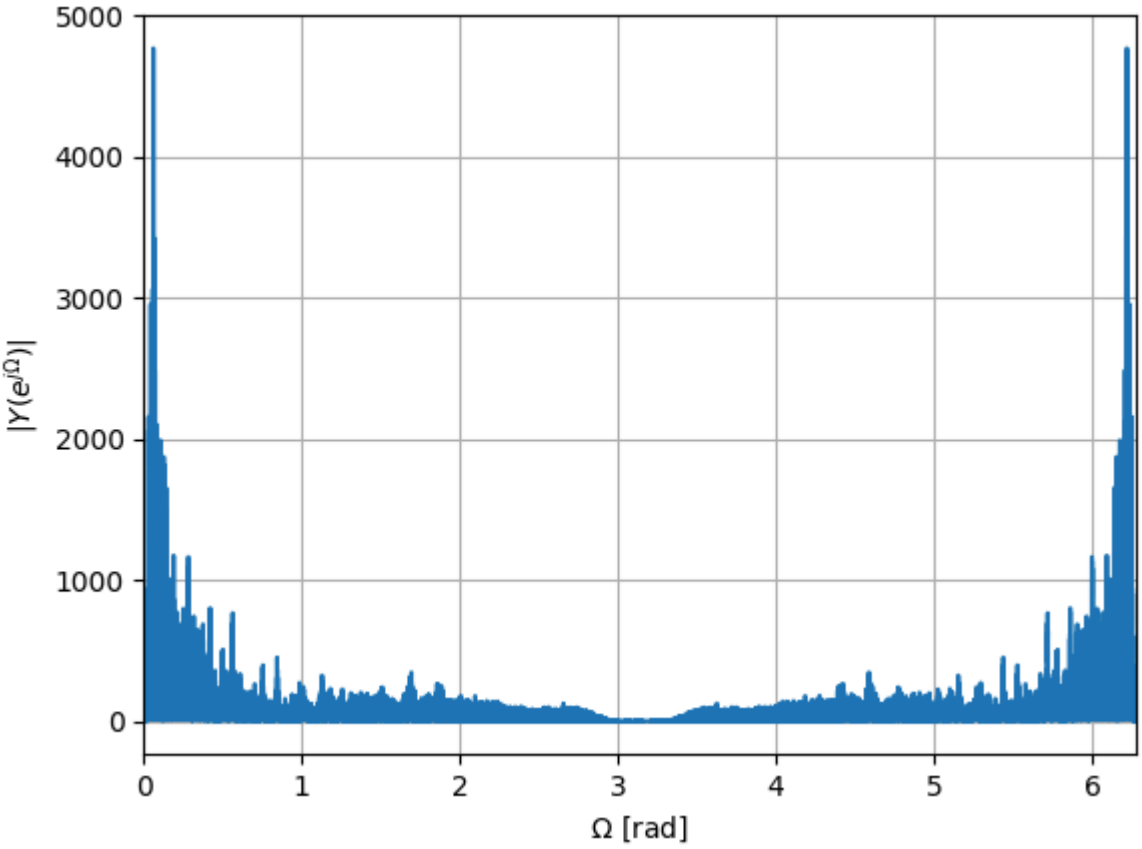
e) Decimando a saída após passar pelo FPB

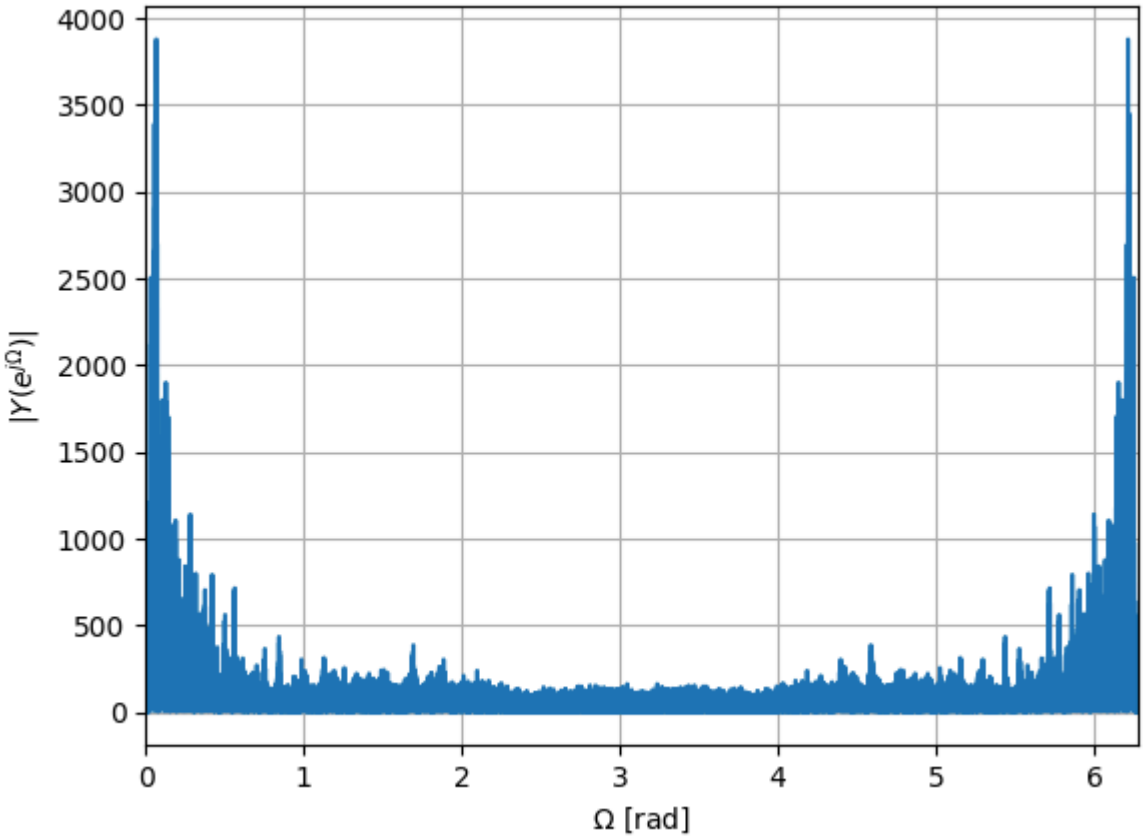
Decimando o sinal `saida` para M = 6

In [132…
```
saida_dec = decimacao(saida, 6)
```

Plotando o espectro da `saida_dec` e do `y_dec`

In [133…
```
espectro(saida_dec)
espectro(y_dec);
```





Discussão:

Nota-se que o espectro do sinal `saida_dec`, nos valores perto de π, se aproximam bem mais de 0, indicando uma supressão maior das altas frequências, mesmo após a decimação, ao contrário do espectro do sinal `y_dec`, que por sua vez apresenta uma estrutura bem mais homogênea perto de π.

Essa diferença nos valores perto de π gera diferenças, também, nos valores próximos de 0 e 2π. Na `saida_dec`, os picos se aproximam de 5000, enquanto na `y_dec` os picos não chegam nem a 4000.

Essas diferenças, em tese, resultarão em uma redução do aliasing, bem como uma maior preservação da estrutura do sinal original.

Fazendo o display da `y_dec` e da `saida_dec`

In [134…
```
ipd.display(ipd.Audio(y_dec, rate=Fs_dec))

ipd.display(ipd.Audio(saida_dec, rate=Fs_dec))
```

0:00 / 0:31

0:00 / 0:31

Discussão:

Escutando os dois áudios lado a lado, é evidente que, conforme esperado, os agudos foram perdidos em ambos (pratos da bateria), mas em `saida_dec`, ao contrário de `y_dec`, não há a distorção causada pelo aliasing nos momentos em que o vocalista canta palavras com "s". Isso é visível, novamente, na marcação de tempo `0:21`, pois no trecho "now it'**s** my time to **s**peak" o chiado ou ruído não acontece mais.

Mesmo que `saida_dec` também perca os agudos e, consequentemente, faça com que o som fique mais distante e com menos instrumentos, a ausência do aliasing torna o áudio mais bem preservado e agradável de se escutar que o `y_dec`.

2) DFT e Identificação de Frequências

f) Análise de um EEG

Abrindo o arquivo `.csv` e definindo a taxa de amostragem `Fs`

In [135…
```
eeg = np.loadtxt("EEG.txt", dtype=float)

Fs = 250
```

Calculando a transformada de Fourier de `eeg`

In [136…
```
n = len(eeg)
eeg_fft = np.fft.fft(eeg)

eeg_fft_mag = np.abs(eeg_fft) / n
```

Calculando o vetor de frequências

In [137…
```
freqs = np.fft.fftfreq(n, 1/Fs)
```

Considerando apenas o espectro positivo de `eeg_fft_mag` e `freqs`

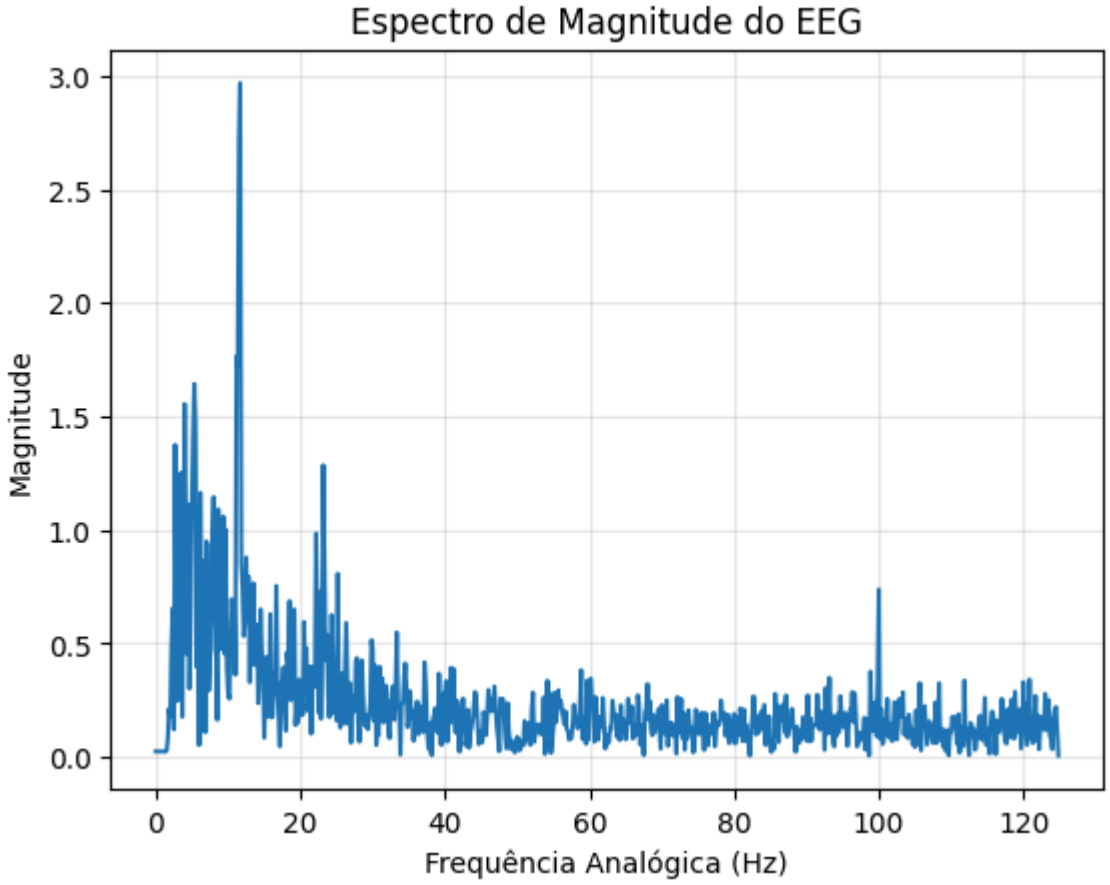
In [138…
```
half = n // 2

eeg_fft_mag_half = 2 * eeg_fft_mag[:half]

freqs_half = freqs[:half]
```

Plotando o EEG num gráfico

In [139…
```
plt.plot(freqs_half, eeg_fft_mag_half)
plt.xlabel("Frequência Analógica (Hz)")
plt.ylabel("Magnitude")
plt.title("Espectro de Magnitude do EEG")
plt.grid(True, alpha=0.3)
plt.show()
```



Indentificando a frequência de maior pico

In [140…

```
k_max = np.argmax(eeg_fft_mag_half[1:]) + 1
Fe = freqs_half[k_max]

print(f"Frequência de maior pico = {Fe:.2f} Hz")
```

```
Frequência de maior pico = 11.67 Hz
```