ESTI019 - Codificação de Sinais Multimídia

Laboratório 7 - Princípios de Codificação da Voz

Prof. Mário Minami

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

OBJETIVOS

1. Determinação dos parâmetros LPC
2. Separação Sonora/Surda
3. Espectro e Envoltória LP
4. Estimação da f_0 e do Pitch

```
!pip install audiolazy

from google.colab import files
import audiolazy as lz #funções para tratar áudio
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
import librosa #para áudio
import librosa.display
import IPython.display
import math
```

```
Requirement already satisfied: audiolazy in /usr/local/lib/python3.7/dist-pack
```

1. Separação de uma Estrofe

Ouvir o áudio

```
audio1 = '/content/drive/MyDrive/Lab 7/Entre_Leva_Catia_Falada.wav'
print(audio1)
v1 , sr1 = librosa.load(audio1)
print(type(v1), type(sr1))
print(v1.shape, sr1)
```

```
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v1, rate=sr1)
```

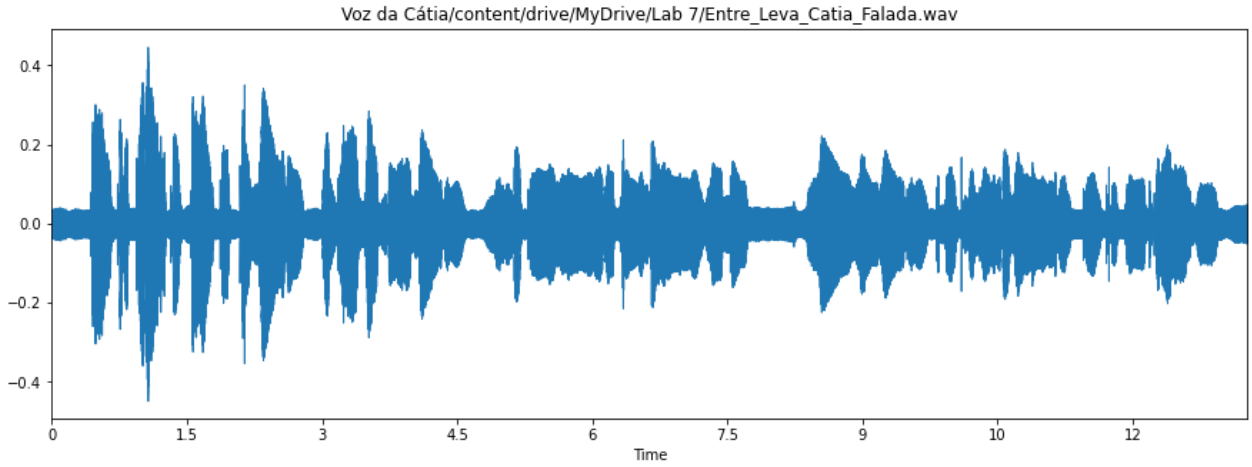
```
/content/drive/MyDrive/Lab 7/Entre_Leva_Catia_Falada.wav
<class 'numpy.ndarray'> <class 'int'>
(292805,) 22050
```

0:00 / 0:13

Visualizando o Áudio

```
plt.figure()
fig, ax = plt.subplots(figsize=(15, 5))
librosa.display.waveplot(v1, sr=sr1)
plt.title('Voz da Cátia' + audio1)
```

```
Text(0.5, 1.0, 'Voz da Cátia/content/drive/MyDrive/Lab 7/Entre_Leva_Catia_F
<Figure size 432x288 with 0 Axes>
```

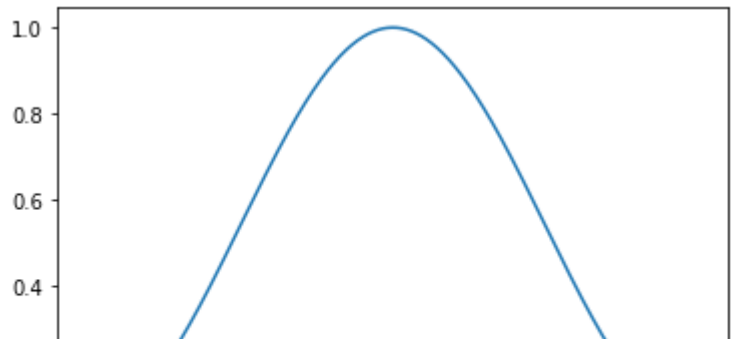


Janela de Hamming

```
fa = sr1
Ts = 0.04
Nj = int(Ts*fa)
Nseg = int(len(v1)/Nj)
Nover = int(Nj*0.5)
```

```
hm = signal.get_window('hamming', Nj)
plt.plot(hm)
```

```
[<matplotlib.lines.Line2D at 0x7f32d328eb90>]
```



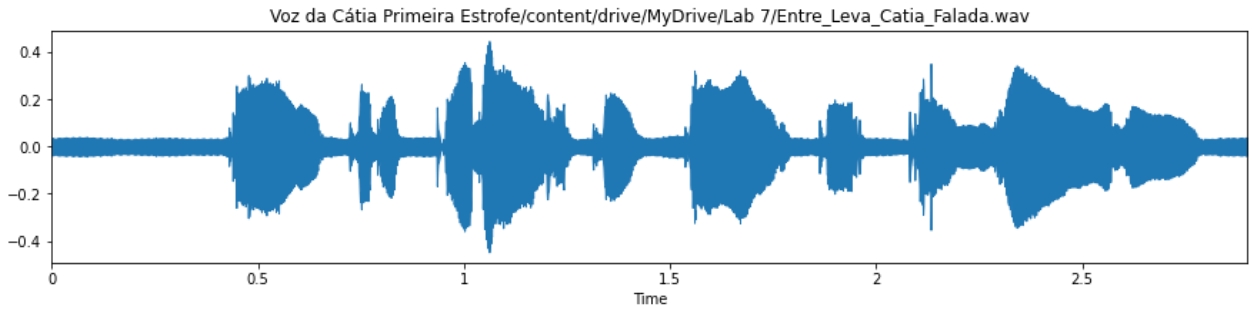
Primera estrofe

```
# Só a primeira estrofe
v11 = v1[1:64000]
plt.figure()
fig11, ax11 = plt.subplots(figsize=(15, 3))
librosa.display.waveplot(v11, sr=sr1)
plt.title('Voz da Cátia Primeira Estrofe' + audio1)
print(type(v11), type(sr1))
print(v11.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v11, rate=sr1)
```

```
<class 'numpy.ndarray'> <class 'int'>
(63999,) 22050
```

0:00 / 0:02

```
<Figure size 432x288 with 0 Axes>
```



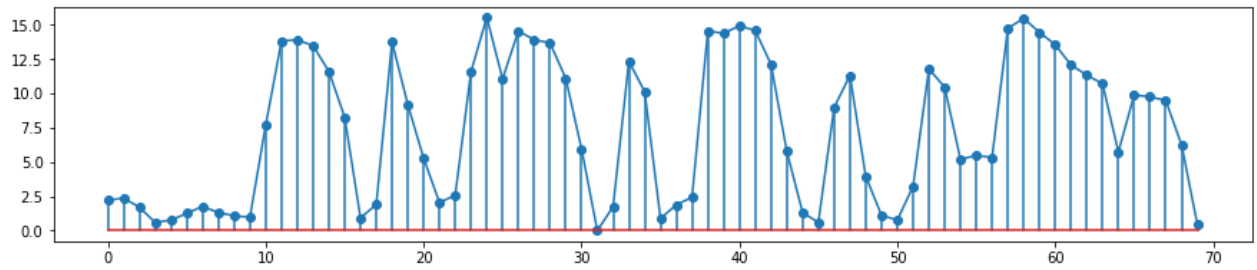
2. Cálculo dos Parâmetros LPC, separação U/UV e Espectro + Envoltória LPC

Gráfico de como está a voz sem tratamento

```
Nover = int(Nj*0.5)
```

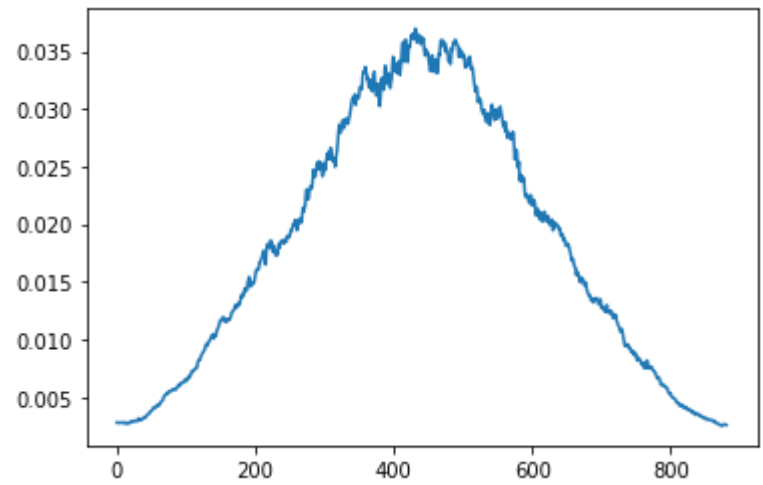
```
Nseg1 = int(len(v11)/Nj)
p = 10 #LPC = 10
E = [] #Energia de Tempo Curto
ind_voz = [0]*Nseg1 #Indicar de Presença de Voz
t = np.arange(Nj)
for l in range(1, Nseg1-1):
    xjan = v1[(l-1)*Nj+Nover:l*Nj+Nover]*hm
    x2 = list(np.array(xjan**2))
    aux = sum(x2)/Nj
    E.append(aux)
E1 = 10*np.log10(E)
E1min = np.min(E1)
E1 = E1 - E1min   # Coloca o ruído de fundo em 0 dB
fig1, ax1 = plt.subplots(figsize=(15, 3))
plt.figure(1)
plt.plot(E1)
plt.stem(E1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:18: UserWarnin
<StemContainer object of 3 artists>
```




```
#Segmento 13 com 50% de sobreposição, i = (13+1)/2, i = 7.
xj = v1[7*Nj:8*Nj]*hm
plt.plot(xj)
```

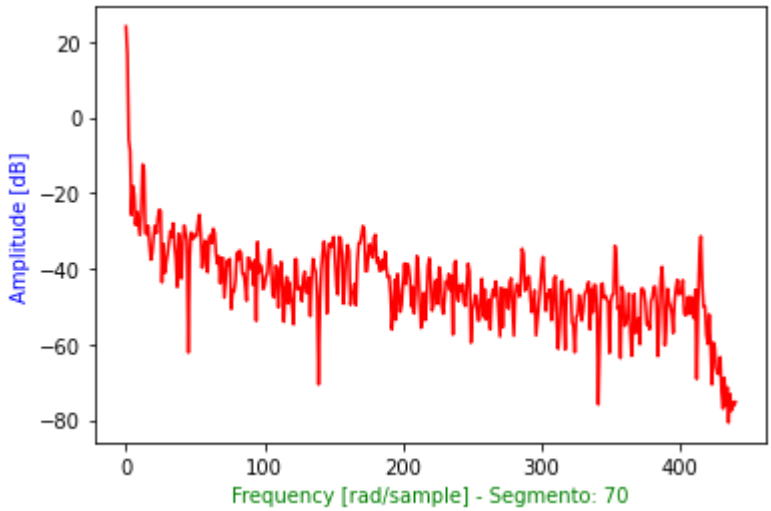
```
[<matplotlib.lines.Line2D at 0x7f32d1a17350>]
```



Espectro real

```
sp = np.fft.fft(xj)
plt.ylabel('Amplitude [dB]', color='b')
plt.xlabel('Frequency [rad/sample] - Segmento: ' + str(l), color='g')

plt.plot(20*np.log10(abs(sp[0:int(Nj/2)])), 'r')
```

```
[<matplotlib.lines.Line2D at 0x7f32d1b183d0>]
```



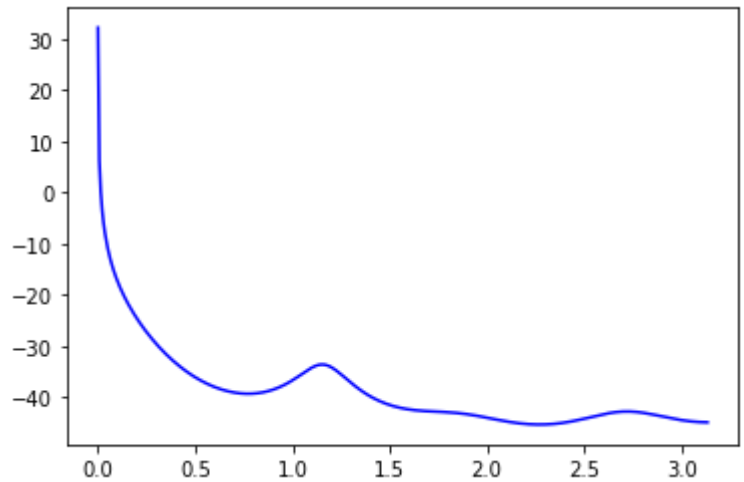

```
a_filter = lz.lpc.kautocor(xjan, p)
print(a_filter)
```

```
1 - 0.940122 * z^-1 + 0.00454464 * z^-2 + 0.0574783 * z^-3 - 0.231505 * z^-4 -
```

Modelo

```
gain_lpc = 10*np.log10(abs(a_filter.error))
print(gain_lpc)
w, h = signal.freqz(1,a_filter.numerator,worN=int(Nj/2))
LP = 20 * np.log10(abs(h)) + gain_lpc
plt.plot(w, LP, 'b')
```

```
-38.74736654372238
[<matplotlib.lines.Line2D at 0x7f32d1af6990>]
```

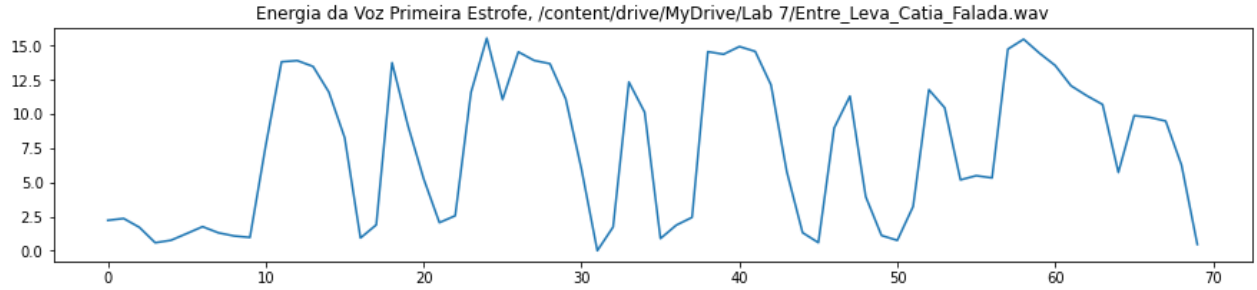


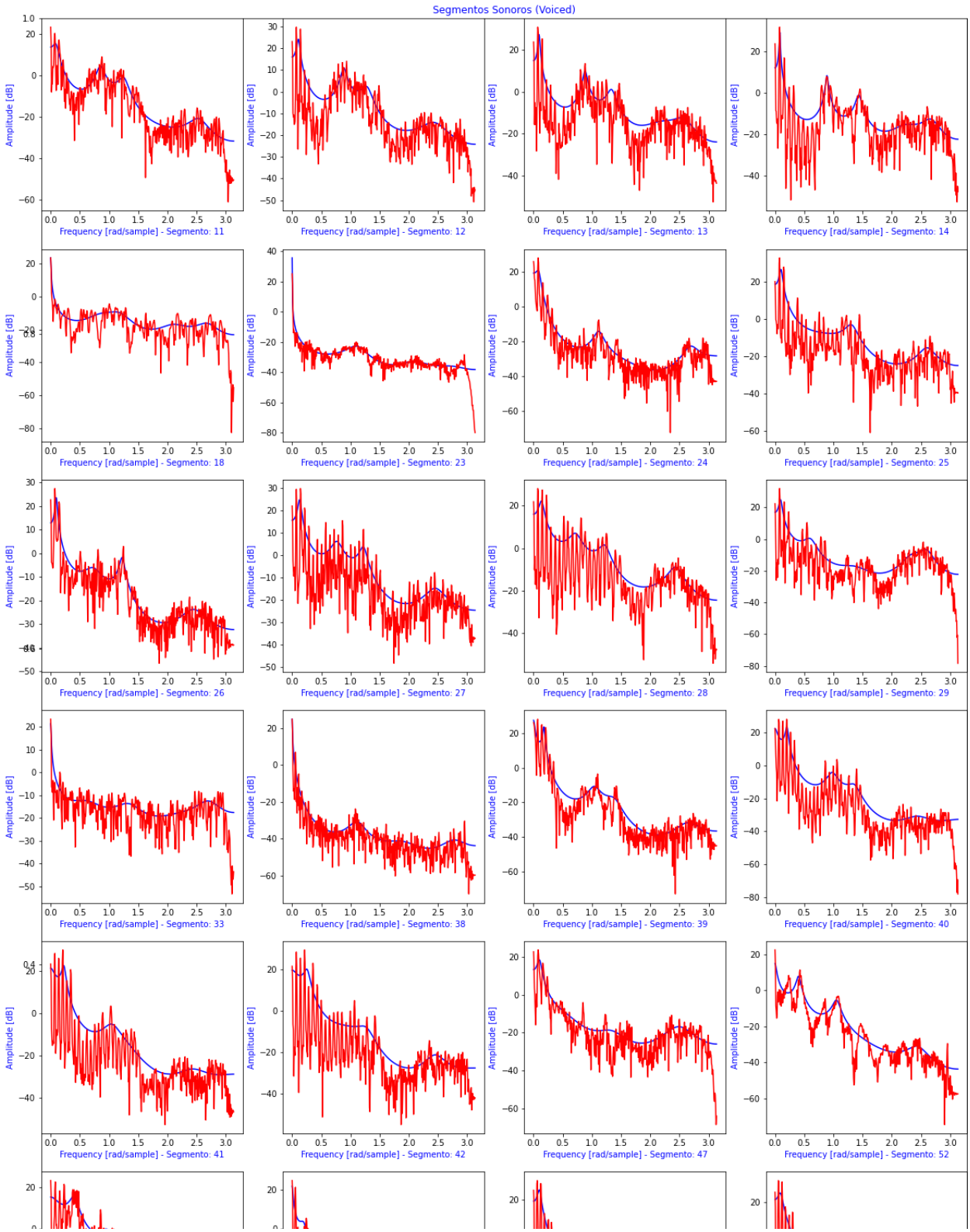
Para cada segmento

```
E1max = np.max(E1)

# ************************************************************
# OBSERVAÇÃO:
# ************************************************************
# ATENÇÃO -> aqui nos arquivos que vocês gravam pode haver diferença
# nos limiares de VAD e U/UV
# É preciso ajustar ambos olhando para os níveis da
# Energia de Tempo Curto E1
# ************************************************************
# limiar de atividade VAD
E1VAD_lim = E1max - 8
print('Limiar VAD ' + str(E1VAD_lim))
# Limiar U/UV ajustado para 30% do máximo
E1voiced_lim = E1max - 5.3
print('Limiar U/UV ' + str(E1voiced_lim))
# indicador de VAD
ind_voz = np.where(E1 > E1VAD_lim, 1, 0)
ind_voiced = np.where(E1 > E1voiced_lim, 1, 0)
tot_voz = np.sum(ind_voz)
num_voiced = np.sum(ind_voiced)
num_unvoiced = tot_voz - num_voiced
linhas_voiced = math.ceil(num_voiced/4)
linhas_unvoiced = math.ceil(num_unvoiced/4)

print('Sonoros = ' + str(num_voiced) + ' e Surdos = ' + str(num_unvoiced) )
fig1, ax1 = plt.subplots(figsize=(15, 3))
plt.figure(1)
plt.plot(E1)
plt.title('Energia da Voz Primeira Estrofe, ' + audio1)

# partição das figuras voiced
i = 0
fig2, ax2 = plt.subplots(figsize=(20, num_voiced + linhas_voiced))
plt.title('Segmentos Sonoros (Voiced)', color = 'b')

# partição das figuras unvoiced
j = 0
fig3, ax3 = plt.subplots(figsize=(20, num_unvoiced + linhas_unvoiced))
plt.title('Segmentos Surdos (Unvoiced)', color = 'g')

for l in range(1, Nseg1-2):
    # teste de VAD
    if ind_voz[l] == 1:
        xjan = v1[(l-1)*Nj+Nover:l*Nj+Nover]*hm
        a_filter = lz.lpc.kautocor(xjan, p)
        gain_lpc = np.log10(abs(a_filter.error))
        w, h = signal.freqz(1,a_filter.numerator,worN=int(Nj/2))
        LP = 20 * np.log10(abs(h)) + 10*gain_lpc
        # Teste U/UV

        if E1[l] > E1voiced_lim:
```

```
        i += 1
        ax2 = fig2.add_subplot(linhas_voiced,4,i)
        plt.figure(2)
        plt.plot(w, LP, 'b')
        plt.ylabel('Amplitude [dB]', color='b')
        plt.xlabel('Frequency [rad/sample] - Segmento: ' + str(l), color='b')
        sp = np.fft.fft(xjan)
        plt.plot(w, 20*np.log10(abs(sp[0:int(Nj/2)])), 'r')
    else:

        j += 1
        ax3 = fig3.add_subplot(linhas_unvoiced,4,j)
        plt.figure(3)
        plt.plot(w, LP, 'g')
        plt.ylabel('Amplitude [dB]', color='b')
        plt.xlabel('Frequency [rad/sample] - Segmento: ' + str(l), color='g')
        sp = np.fft.fft(xjan)
        plt.plot(w, 20*np.log10(abs(sp[0:int(Nj/2)])), 'r')
```

Limiar VAD 7.528272731468608
Limiar U/UV 10.228272731468607
Sonoros = 28 e Surdos = 8

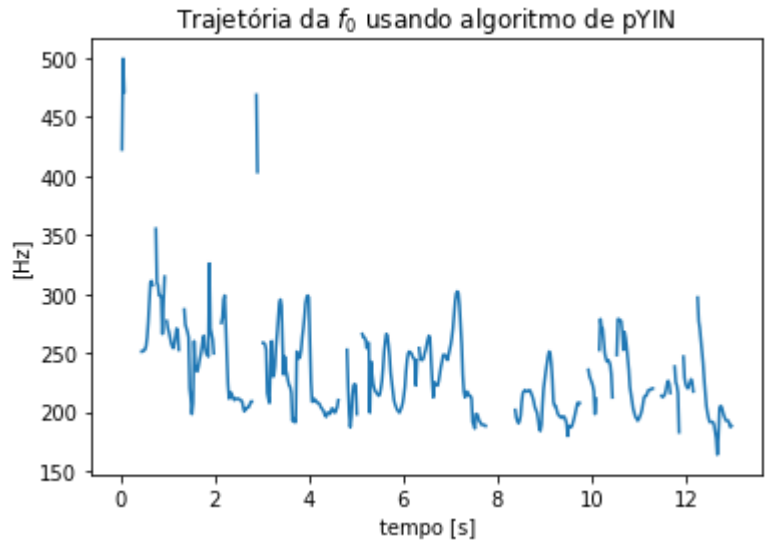




3. Estimação da f_0 e do Pitch

```
f0, voiced_flag, voiced_probs = librosa.pyin(v1, fmin=librosa.note_to_hz('C2'), fma
times = librosa.times_like(f0)
plt.plot(times,f0)
plt.ylabel('[Hz]')
plt.xlabel('tempo [s]')
plt.title('Trajetória da $f_0$ usando algoritmo de pYIN')
```

```
Text(0.5, 1.0, 'Trajetória da $f_0$ usando algoritmo de pYIN')
```



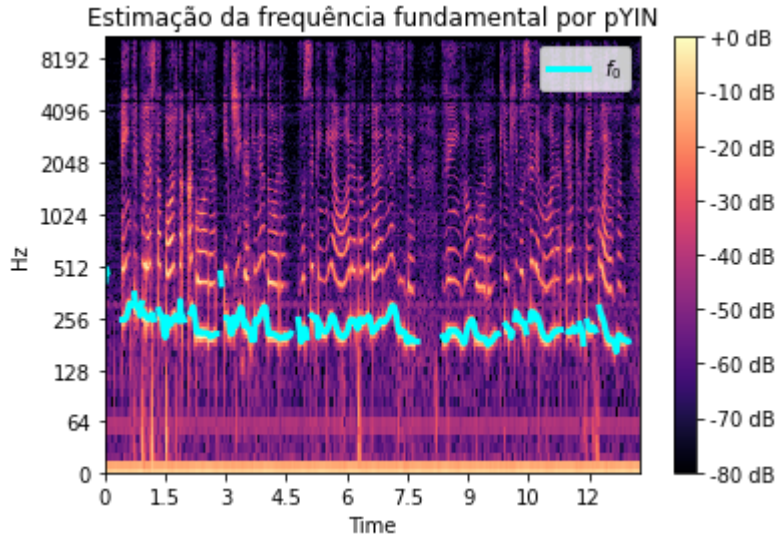

```
pitch = 1/f0
plt.plot(times,1000*pitch)
plt.ylabel('Pitch [ms]')
plt.xlabel('tempo [s]')
plt.title('Trajetória do Pitch')
```

```
Text(0.5, 1.0, 'Trajetória do Pitch')
```

Espectrograma enfatizando a f_0

```
D = librosa.amplitude_to_db(np.abs(librosa.stft(v1)), ref=np.max)
fig, ax = plt.subplots()
img = librosa.display.specshow(D, x_axis='time', y_axis='log', ax=ax)
ax.set(title='Estimação da frequência fundamental por pYIN')
fig.colorbar(img, ax=ax, format="%+2.f dB")
ax.plot(times, f0, label='$f_0$', color='cyan', linewidth=3)
ax.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x7f32d35755d0>
```



4. Repita o mesmo procedimento para a sua voz gravada

Se houver erro de separação U/UV provavelmente é devido aos níveis de gravação utilizados: ajustar os valores como na OBSERVAÇÃO deste Notebook

Jefferson

Carregando o aúdio

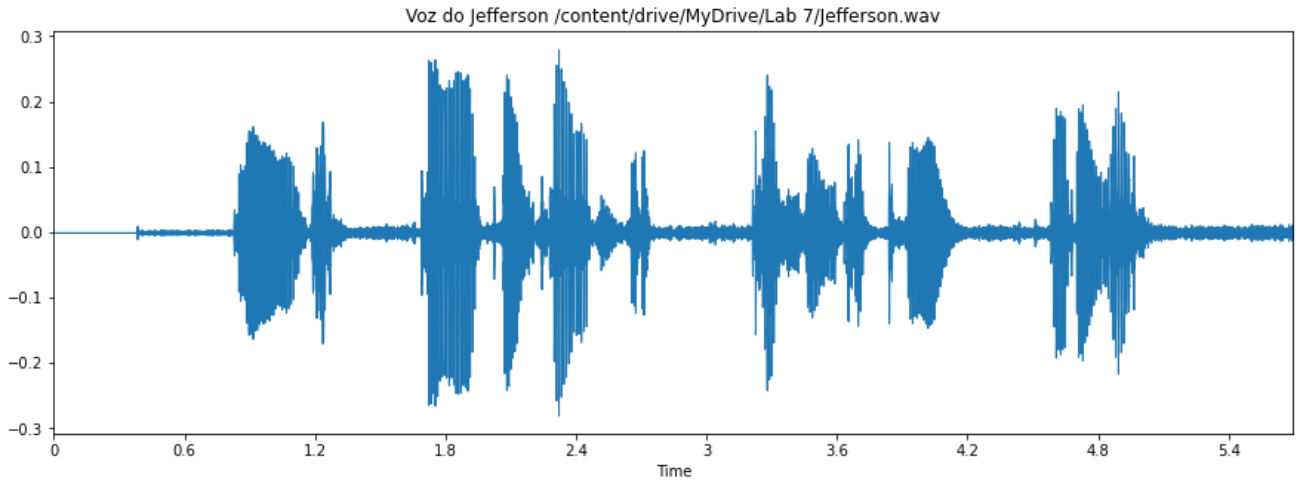
```
audio1 = '/content/drive/MyDrive/Lab 7/Jefferson.wav'
print(audio1)
v1 , sr1 = librosa.load(audio1)
print(type(v1), type(sr1))
print(v1.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v1, rate=sr1)
```

```
/content/drive/MyDrive/Lab 7/Jefferson.wav
/usr/local/lib/python3.7/dist-packages/librosa/core/audio.py:165: UserWarning:
  warnings.warn("PySoundFile failed. Trying audioread instead.")
<class 'numpy.ndarray'> <class 'int'>
(125685,) 22050
```

0:00 / 0:05

```
plt.figure()
fig, ax = plt.subplots(figsize=(15, 5))
librosa.display.waveplot(v1, sr=sr1)
plt.title('Voz do Jefferson ' + audio1)

plt.savefig('Forma de Onda Sinal Completo - Jefferson.png', format='png', dpi=300,
```

```
<Figure size 432x288 with 0 Axes>
```

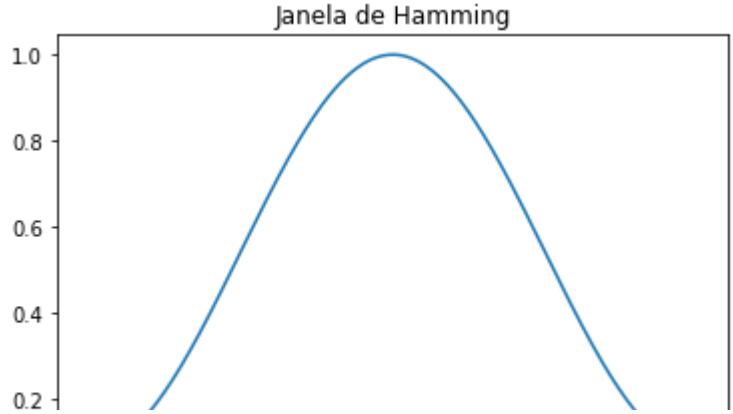



```
fa = sr1
Ts = 0.04
Nj = int(Ts*fa)
Nseg = int(len(v1)/Nj)
Nover = int(Nj*0.5)
```

Janela de Hamming

```
# Obtendo a Janela de Hamming
hm = signal.get_window('hamming', Nj)
plt.plot(hm)
plt.title("Janela de Hamming")
```

```
Text(0.5, 1.0, 'Janela de Hamming')
```



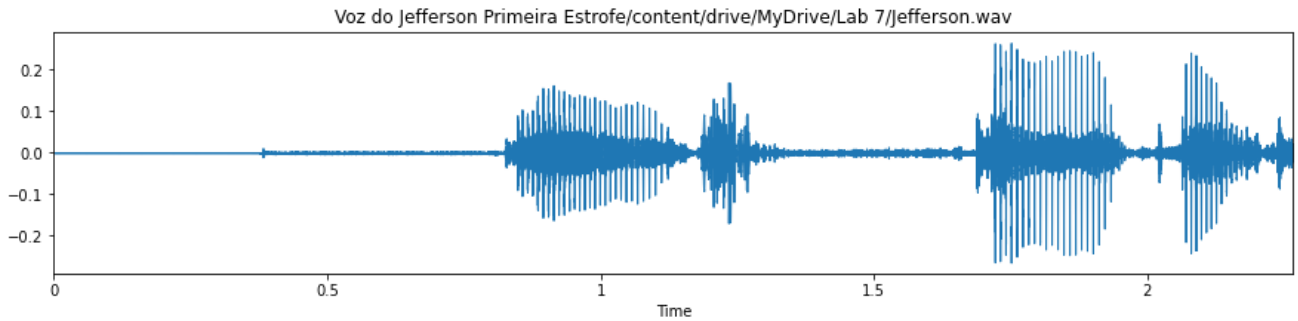
Separação do primeiro nome

```
# Separação do primeiro nome
v11 = v1[1:50000]
plt.figure()
fig11, ax11 = plt.subplots(figsize=(15, 3))
librosa.display.waveplot(v11, sr=sr1)
plt.title('Voz do Jefferson Primeira Estrofe' + audio1)

plt.savefig('Forma_de_Onda_Primeira_Estrofe - Jefferson.png', format='png', dpi=300

print(type(v11), type(sr1))
print(v11.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v11, rate=sr1)
```

```
<class 'numpy.ndarray'> <class 'int'>
(49999,) 22050
```

0:00 / 0:02

```
<Figure size 432x288 with 0 Axes>
```



2. Cálculo dos Parâmetros LPC, separação U/UV e Espectro + Envoltória LPC

[ ] ↳ *1 célula oculta*

3.Estimação da f0 e do Pitch

[ ] ↳ *2 células ocultas*

Espectograma enfatizando a f0

[ ] ↳ *1 célula oculta*

Lucas

Carregando o aúdio

```
audio1 = '/content/drive/MyDrive/Lab 7/Lucas.wav'
print(audio1)
v1 , sr1 = librosa.load(audio1)
print(type(v1), type(sr1))
print(v1.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v1, rate=sr1)
```

```
/content/drive/MyDrive/Lab 7/Lucas.wav
/usr/local/lib/python3.7/dist-packages/librosa/core/audio.py:165: UserWarning:
  warnings.warn("PySoundFile failed. Trying audioread instead.")
<class 'numpy.ndarray'> <class 'int'>
(173313,) 22050
```

0:00 / 0:07

```
plt.figure()
fig, ax = plt.subplots(figsize=(15, 5))
librosa.display.waveplot(v1, sr=sr1)
plt.title('Voz do Lucas ' + audio1)

plt.savefig('Forma de Onda Sinal Completo - Lucas.png', format='png', dpi=300, bbox
```

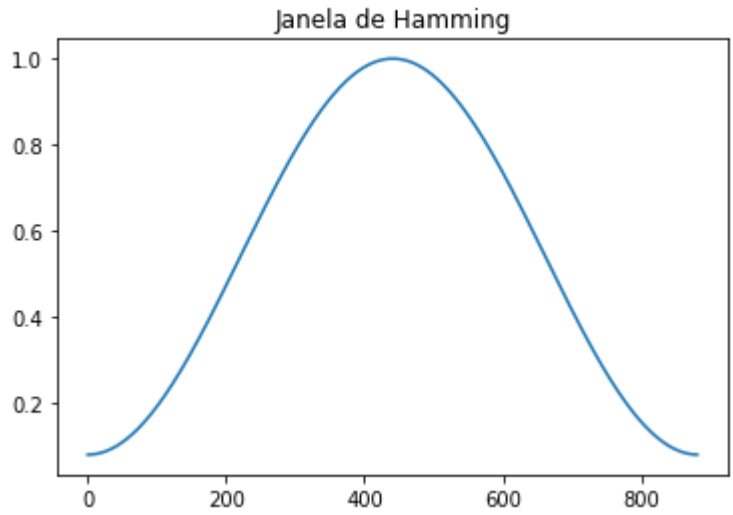
```
<Figure size 432x288 with 0 Axes>
```


```
fa = sr1
Ts = 0.04
Nj = int(Ts*fa)
Nseg = int(len(v1)/Nj)
Nover = int(Nj*0.5)
```

Janela de Hamming

```
# Obtendo a Janela de Hamming
hm = signal.get_window('hamming', Nj)
plt.plot(hm)
plt.title("Janela de Hamming")
```

```
Text(0.5, 1.0, 'Janela de Hamming')
```



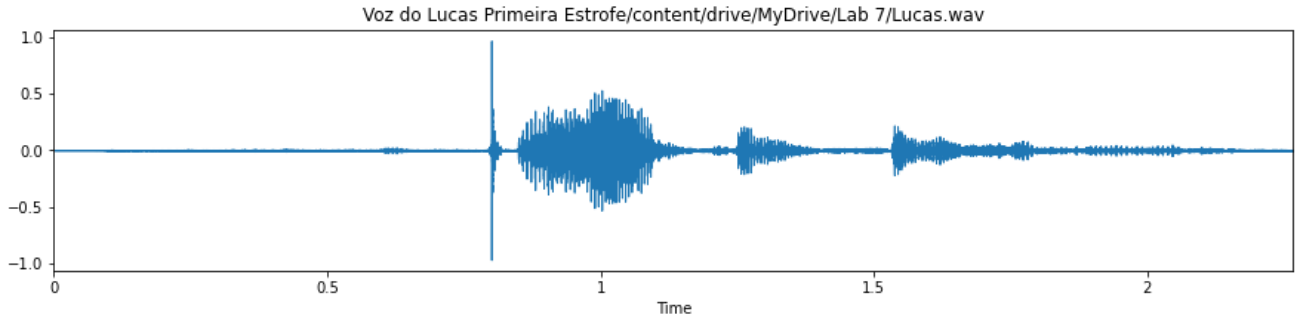
Separação do primeiro nome

```
# Separação do primeiro nome
v11 = v1[1:50000]
plt.figure()
fig11, ax11 = plt.subplots(figsize=(15, 3))
librosa.display.waveplot(v11, sr=sr1)
plt.title('Voz do Lucas Primeira Estrofe' + audio1)

plt.savefig('Forma_de_Onda_Primeira_Estrofe - Lucas.png', format='png', dpi=300, bb

print(type(v11), type(sr1))
print(v11.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v11, rate=sr1)
```

```
<class 'numpy.ndarray'> <class 'int'>
(49999,) 22050
```

0:00 / 0:02

```
<Figure size 432x288 with 0 Axes>
```



2. Cálculo dos Parâmetros LPC, separação U/UV e Espectro + Envoltória LPC

[ ] ↳ *1 célula oculta*

3.Estimação da f0 e do Pitch

[ ] ↳ *2 células ocultas*

Espectograma enfatizando a f0

[ ] ↳ *1 célula oculta*

Victor

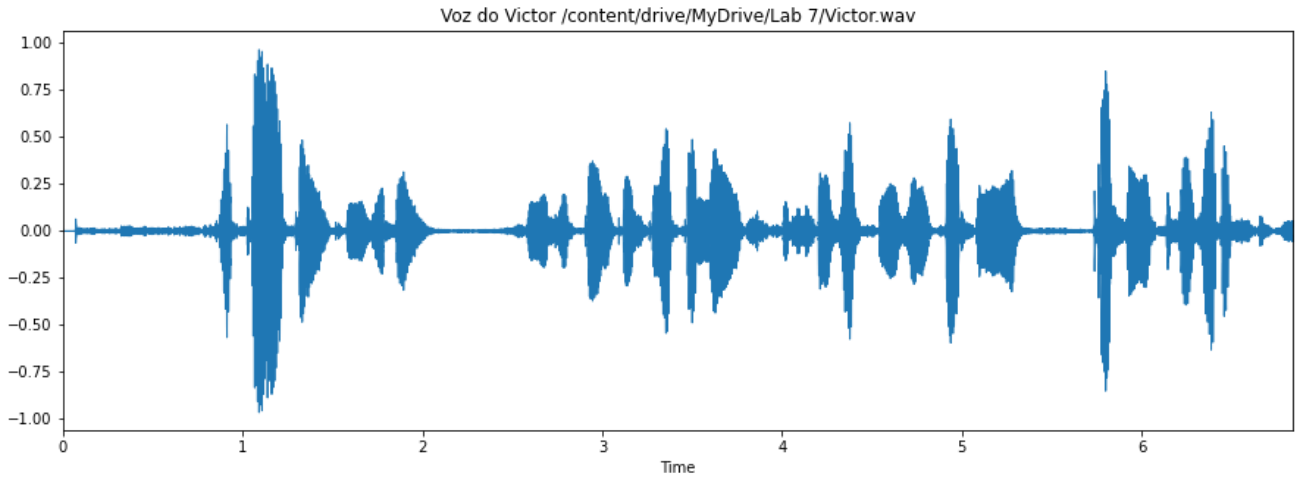
Carregando o áudio

```
audio1 = '/content/drive/MyDrive/Lab 7/Victor.wav'
print(audio1)
v1 , sr1 = librosa.load(audio1)
print(type(v1), type(sr1))
print(v1.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v1, rate=sr1)
```

```
/content/drive/MyDrive/Lab 7/Victor.wav
/usr/local/lib/python3.7/dist-packages/librosa/core/audio.py:165: UserWarning:
  warnings.warn("PySoundFile failed. Trying audioread instead.")
<class 'numpy.ndarray'> <class 'int'>
```

```
plt.figure()
fig, ax = plt.subplots(figsize=(15, 5))
librosa.display.waveplot(v1, sr=sr1)
plt.title('Voz do Victor ' + audio1)

plt.savefig('Forma de Onda Sinal Completo - Victor.png', format='png', dpi=300, bbo
```

```
<Figure size 432x288 with 0 Axes>
```

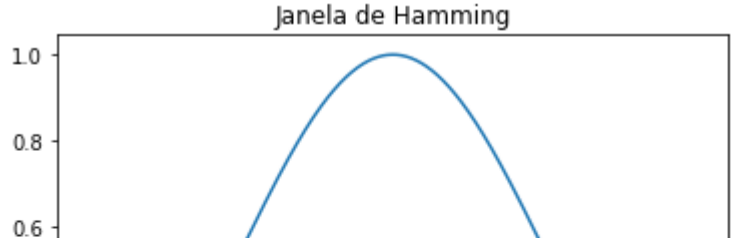



```
fa = sr1
Ts = 0.04
Nj = int(Ts*fa)
Nseg = int(len(v1)/Nj)
Nover = int(Nj*0.5)
```

Janela de Hamming

```
# Obtendo a Janela de Hamming
hm = signal.get_window('hamming', Nj)
plt.plot(hm)
plt.title("Janela de Hamming")
```

```
Text(0.5, 1.0, 'Janela de Hamming')
```



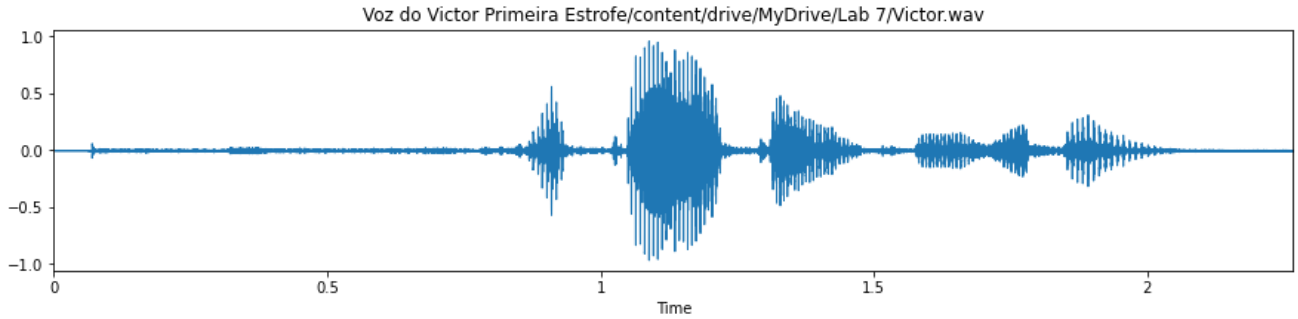
Separação do primeiro nome

```
# Separação do primeiro nome
v11 = v1[1:50000]
plt.figure()
fig11, ax11 = plt.subplots(figsize=(15, 3))
librosa.display.waveplot(v11, sr=sr1)
plt.title('Voz do Victor Primeira Estrofe' + audio1)

plt.savefig('Forma_de_Onda_Primeira_Estrofe - Victor.png', format='png', dpi=300, b

print(type(v11), type(sr1))
print(v11.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v11, rate=sr1)
```

```
<class 'numpy.ndarray'> <class 'int'>
(49999,) 22050
```

0:00 / 0:02

```
<Figure size 432x288 with 0 Axes>
```



2. Cálculo dos Parâmetros LPC, separação U/UV e Espectro + Envoltória LPC

[ ] ↳ *1 célula oculta*

3.Estimação da f0 e do Pitch

[ ] ↳ *2 células ocultas*

Espectograma enfatizando a f0

[ ] ↳ *1 célula oculta*

Muriel

Carregando o aúdio

```
audio1 = '/content/drive/MyDrive/Lab 7/Muriel.wav'
print(audio1)
v1 , sr1 = librosa.load(audio1)
print(type(v1), type(sr1))
print(v1.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v1, rate=sr1)
```

```
plt.figure()
fig, ax = plt.subplots(figsize=(15, 5))
librosa.display.waveplot(v1, sr=sr1)
plt.title('Voz do Lucas ' + audio1)

plt.savefig('Forma de Onda Sinal Completo - Muriel.png', format='png', dpi=300, bbo
```

```
fa = sr1
Ts = 0.04
Nj = int(Ts*fa)
Nseg = int(len(v1)/Nj)
Nover = int(Nj*0.5)
```

Janela de Hamming

```
# Obtendo a Janela de Hamming
hm = signal.get_window('hamming', Nj)
plt.plot(hm)
plt.title("Janela de Hamming")
```

Separação do primeiro nome

```
# Separação do primeiro nome
v11 = v1[1:50000]
plt.figure()
fig11, ax11 = plt.subplots(figsize=(15, 3))
librosa.display.waveplot(v11, sr=sr1)
plt.title('Voz do Lucas Primeira Estrofe' + audio1)

plt.savefig('Forma_de_Onda_Primeira_Estrofe - Muriel.png', format='png', dpi=300, b
```

```
print(type(v11), type(sr1))
print(v11.shape, sr1)
# Player será aberto! AGUARDE até abrir!
IPython.display.Audio(data=v11, rate=sr1)
```

2. Cálculo dos Parâmetros LPC, separação U/UV e Espectro + Envoltória LPC

```
Nover = int(Nj*0.5)
Nseg1 = int(len(v11)/Nj)
p = 10
E = []
ind_voz = [0]*Nseg1
t = np.arange(Nj)
for l in range(1, Nseg1-1):
    xjan = v1[(l-1)*Nj+Nover:l*Nj+Nover]*hm
    x2 = list(np.array(xjan**2))
    aux = sum(x2)/Nj
    E.append(aux)
E1 = 10*np.log10(E)
E1min = np.min(E1)
E1 = E1 - E1min   # Coloca o ruído de fundo em 0 dB
E1max = np.max(E1)
print('E1mín', E1min)
print('E1max', E1max)
# ****************************************************************
# OBSERVAÇÃO:
# ****************************************************************
# ATENÇÃO -> aqui nos arquivos que vocês gravam pode haver diferença
# nos limiares de VAD e U/UV
# É preciso ajustar ambos olhando para os níveis da
# Energia de Tempo Curto E1
# ****************************************************************
# limiar de atividade VAD
E1VAD_lim = E1max - 16
print('Limiar VAD ' + str(E1VAD_lim))
# Limiar U/UV ajustado para 30% do máximo
E1voiced_lim = E1max - 10.6
print('Limiar U/UV ' + str(E1voiced_lim))
# indicador de VAD
ind_voz = np.where(E1 > E1VAD_lim, 1, 0)
ind_voiced = np.where(E1 > E1voiced_lim, 1, 0)
tot_voz = np.sum(ind_voz)
num_voiced = np.sum(ind_voiced)
num_unvoiced = tot_voz - num_voiced
linhas_voiced = math.ceil(num_voiced/4)
linhas_unvoiced = math.ceil(num_unvoiced/4)

print('Sonoros = ' + str(num_voiced) + ' e Surdos = ' + str(num_unvoiced) )
fig1, ax1 = plt.subplots(figsize=(15, 3))
plt.figure(1)
plt.plot(E1)
plt.title('Energia da Voz Primeira Estrofe, ' + audio1)
```

```
plt.savefig('Energia da Voz Primeira Estrofe - Muriel.png', format='png', dpi=300,

# partição das figuras voiced
i = 0
fig2, ax2 = plt.subplots(figsize=(20, num_voiced + linhas_voiced))
plt.title('Segmentos Sonoros (Voiced)', color = 'b')

# partição das figuras unvoiced
j = 0
fig3, ax3 = plt.subplots(figsize=(20, num_unvoiced + linhas_unvoiced))
plt.title('Segmentos Surdos (Unvoiced)', color = 'g')

for l in range(1, Nseg1-2):
    # teste de VAD
    if ind_voz[l] == 1:
        xjan = v1[(l-1)*Nj+Nover:l*Nj+Nover]*hm
        a_filter = lz.lpc.kautocor(xjan, p)
        gain_lpc = np.log10(abs(a_filter.error))
        w, h = signal.freqz(1,a_filter.numerator,worN=int(Nj/2))
        LP = 20 * np.log10(abs(h)) + 10*gain_lpc
        # Teste U/UV

        if E1[l] > E1voiced_lim:

            i += 1
            ax2 = fig2.add_subplot(linhas_voiced,4,i)
            plt.figure(2)
            plt.plot(w, LP, 'b')
            plt.ylabel('Amplitude [dB]', color='b')
            plt.xlabel('Frequency [rad/sample] - Segmento: ' + str(l), color='b')
            sp = np.fft.fft(xjan)
            plt.plot(w, 20*np.log10(abs(sp[0:int(Nj/2)])), 'r')

        else:

            j += 1
            ax3 = fig3.add_subplot(linhas_unvoiced,4,j)
            plt.figure(3)
            plt.plot(w, LP, 'g')
            plt.ylabel('Amplitude [dB]', color='b')
            plt.xlabel('Frequency [rad/sample] - Segmento: ' + str(l), color='g')
            sp = np.fft.fft(xjan)
            plt.plot(w, 20*np.log10(abs(sp[0:int(Nj/2)])), 'r')
```

3.Estimação da f0 e do Pitch

```
f0, voiced_flag, voiced_probs = librosa.pyin(v1, fmin=librosa.note_to_hz('C2'), fma
times = librosa.times_like(f0)
plt.plot(times,f0)
plt.ylabel('[Hz]')
plt.xlabel('tempo [s]')
plt.title('Trajetória da $f_0$ usando algoritmo de pYIN')
```