ECE 438 - Laboratory 9a
Speech Processing (Week 1)

Last Updated on March 29, 2022

Date:4/6
Section:

Name	Signature	Time spent outside lab
Student Name #1 [Ruixiang Wang]		
Student Name #2 [---%]		

	Below expectations	Lacks in some respect	Meets all expectations
Completeness of the report			
Organization of the report			
Quality of figures: *Correctly labeled with title, x-axis, y-axis, and name(s)*			
Understanding differences between voiced/unvoiced segments(30 pts): *Python plots and code(zero cross), questions*			
Understanding and implementation of short-time DTFT (30 pts): *Python plots and code(DFTwin), questions*			
Understanding and implementation of spectrogram (40 pts): *Python plots and code(specgm), questions, formant estimates*			

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf
import IPython.display as ipd
from helper import hamming
```

In [2]:
```
# specify the size of the plot
plt.rcParams['figure.figsize'] = (16, 6)

# make sure the plot is displayed in this notebook
%matplotlib inline

# for auto-reloading extenrnal modules
%load_ext autoreload
%autoreload 2
```

Exercise 2.2: Classification of Voiced/Unvoiced Speech

1. Load the audio `Start.wav` using `start, fs = sf.read("Start.wav")`.

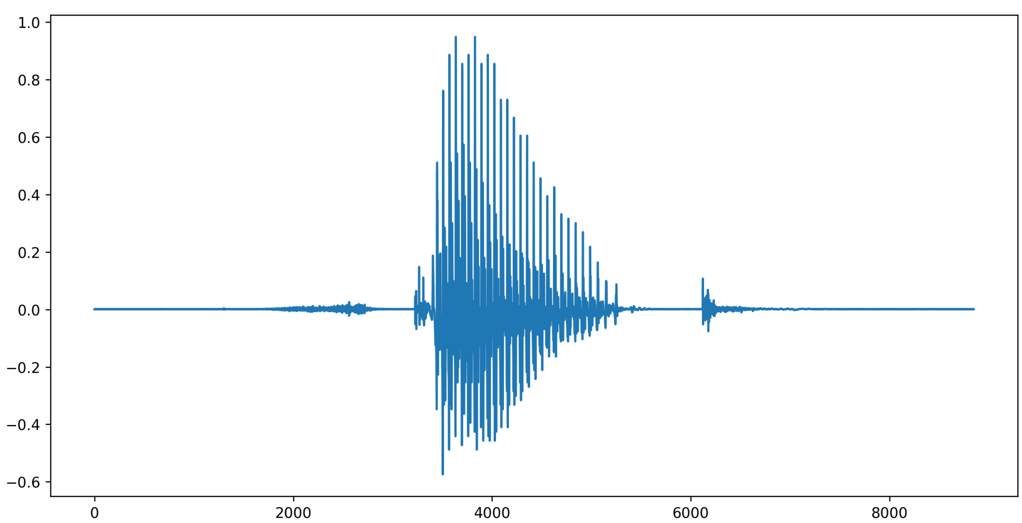
In [3]:
```
start, fs = sf.read("Start.wav")
```

2. Plot (not stem) the speech signal. Then identify two segments of the signal: one segment that is voiced and a second segment that is unvoiced. To identify them, you can choose one from the following options:

- a. Circle the regions of the plot of the speech signal corresponding to these two segments.
- b. Plot the regions corresponding to these two segments separately in new cells.
- c. Print the starting and ending indices of these two regions.

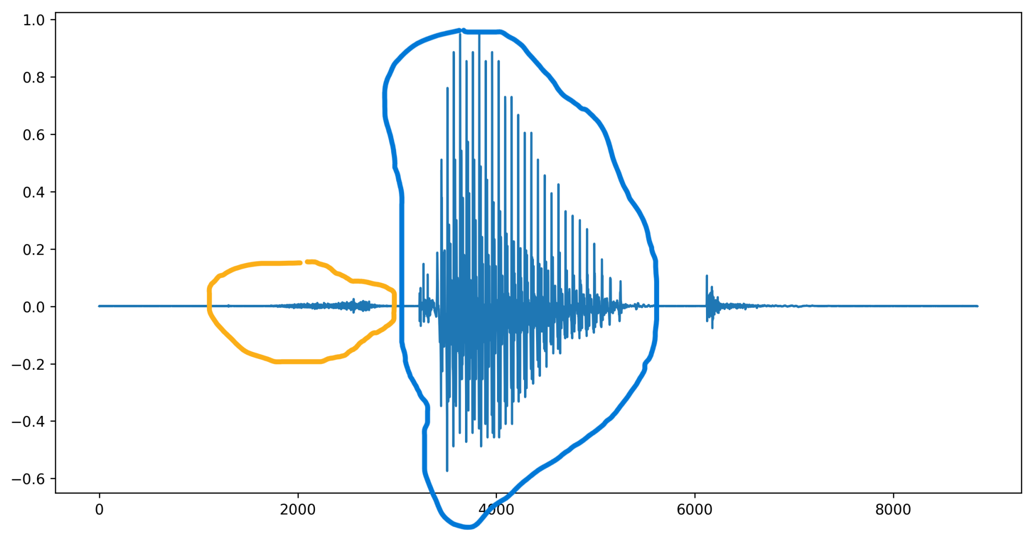
In [4]:
```
# temporarily make the plot interactive
%matplotlib notebook
# specify the size of the plot
plt.rcParams['figure.figsize'] = (12, 6)
```

In [6]:
```
plt.plot(start)
```



Out[6]:
```
[<matplotlib.lines.Line2D at 0x15e2b34bdf0>]
```

In [ ]:
```
# make the plot not interactive
%matplotlib inline
# specify the size of the plot
plt.rcParams['figure.figsize'] = (16, 6)
```



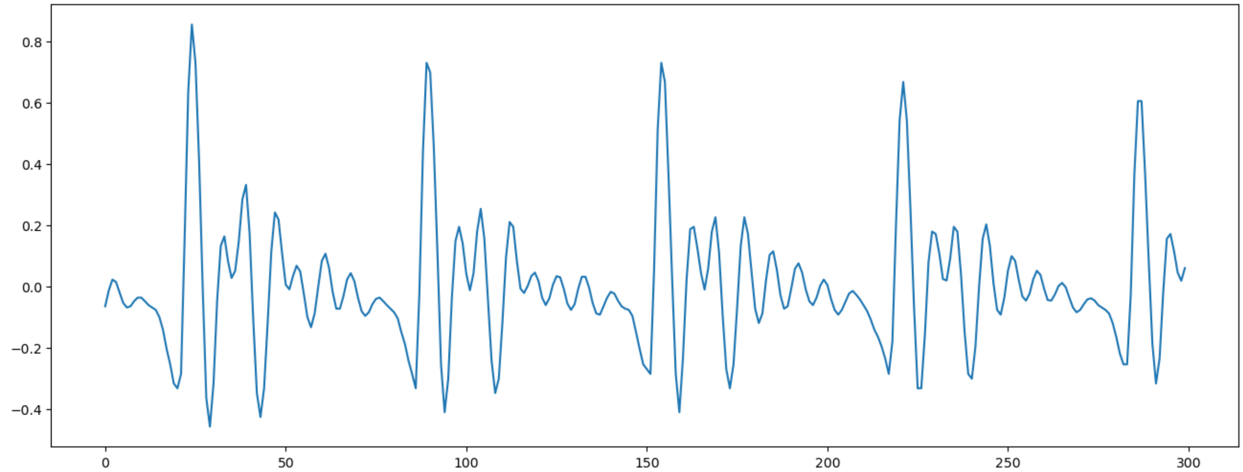
2. Extract 300 samples from the voiced segment of the speech into a NumPy vector called `VoicedSig` . Also, extract 300 samples from the unvoiced segment of the speech into a NumPy vector called `UnvoicedSig` .

In [12]:
```
UnvoicedSig = start[2000:2300]
VoicedSig = start[4000:4300]
```

3. Plot the two signals, `VoicedSig` and `UnvoicedSig` .

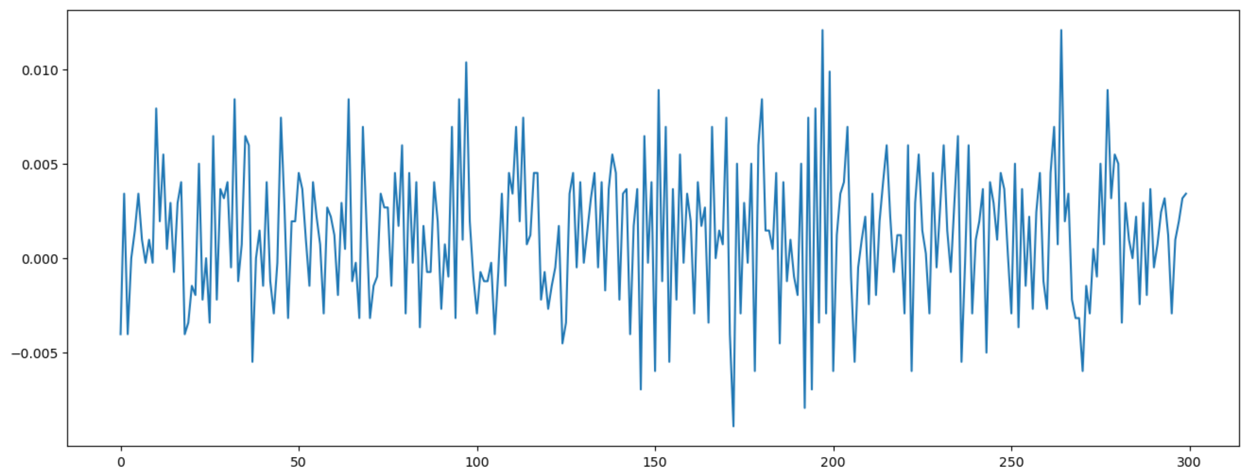
In [13]:
```
plt.plot(VoicedSig)
```

Out[13]: [<matplotlib.lines.Line2D at 0x15e2e653ac0>]



In [14]:
```
plt.plot(UnvoicedSig)
```

Out[14]: [<matplotlib.lines.Line2D at 0x15e2e6c1640>]



4. Explain how you selected your voiced and unvoiced regions.

By observation

5. Estimate the pitch period for the voiced segment. Keep in mind that these speech signals are sampled at 8 KHz, which means that the time between samples is 0.125 milliseconds (ms). Typical values for the pitch period are 8 ms for male speakers, and 4 ms for female speakers. Based on this, would you predict that the speaker is male, or female?

By multiplying sample size for one pitch period with time, I persume the speaker is male.

6. Complete the function below that calculates equation (1) to compute the average energy.

In [21]:
```
def get_average_energy(x):
    """
    Parameters
    ---
    x: the input signal

    Returns
    ---
    P: the average energy of the signal
    """
    P = np.sum(x[n]**2 for n in range(1,len(x)))/len(x)
    return P
```

7. Use this function to compute the average energy of the voiced and unvoiced segments that you plotted above. Print the values.

In [24]:
```
Voiced_e = get_average_energy(VoicedSig)
Unvoiced_e = get_average_energy(UnvoicedSig)
print(Voiced_e)
print(Unvoiced_e)
```

```
0.042868237495422366
1.5203505754470825e-05

C:\Users\rxw14\AppData\Local\Temp\ipykernel_17928\2085720812.py:11: DeprecationWarning: Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.
  P = np.sum(x[n]**2 for n in range(1,len(x)))/len(x)
```

8. For which segment is the average energy greater?

Voiced segment has greater average energy

---

9. Complete the function below to compute the number of zero-crossings that occur within a vector.

In [31]:
```
def get_zero_cross(x):
    """
    Parameters
    ---
    x: the input signal

    Returns
    ---
    cnt: the number of zero-crossings
    """
    cnt = 0
    for i in range(1,len(x)):
        if (x[i] > 0)and(x[i-1]<0)or(x[i] < 0)and(x[i-1]>0):
            cnt += 1

    return cnt
```

10. Compute and print the numbers of zero-crossings of both `VoicedSig` and `UnvoicedSig` .

In [32]:
```
Voiced_cross = get_zero_cross(VoicedSig)
Unvoiced_cross = get_zero_cross(UnvoicedSig)
print(Voiced_cross)
print(Unvoiced_cross)
```

```
56
161
```

11. Which segment has more zero-crossings?

Unvoiced signal has more zero-crossing

Exercise 3.2

1. Complete the function below to compute the DFT of a windowed length L segment of the vector x.

Note:

- You should use a Hamming window of length L to window x
- Your window should start at the index m of the signal x.
- Your DFTs should be of length N
- You may use `hamming(L)` function to obtain the Hamming window.
- You may use `np.fft.fft()` function to compute the DFTs.

In [49]:
```
def DFTwin(x, L, m, N):
    """
    Parameters
    ---
    x: the input signal
    L: the length of the Hamming window
    m: the starting index of the signal x where the Hamming window is applied
    N: the length of DFT

    Returns
    ---
    X: the DFT of a windowed length L segment of x
    """
    a = hamming(L)*x[m:m+L]
    X = np.fft.fft(a, N)
    return np.abs(X)
```

2. Load the file `go.au` .

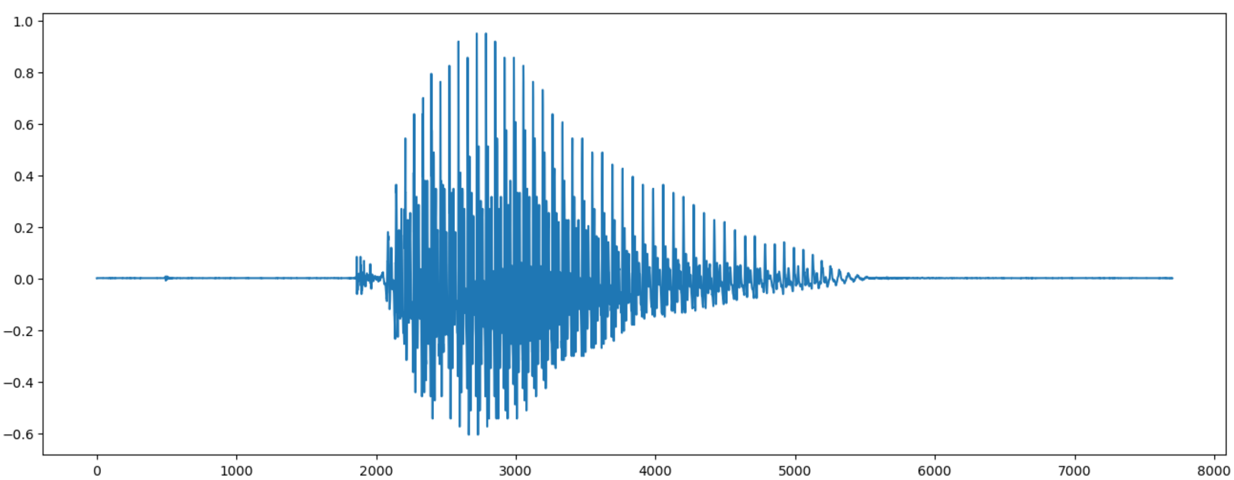
In [34]:
```
go, fs = sf.read("go.au")
```

3. Plot the signal and locate a voiced region, which should cover six pitch periods.

In [35]:
```
# temporarily make the plot interactive
%matplotlib notebook
%matplotlib notebook
import matplotlib.pyplot as plt
# specify the size of the plot
plt.rcParams['figure.figsize'] = (12, 6)
```

In [50]:
```
plt.plot(go)
```

Out[50]:
```
[<matplotlib.lines.Line2D at 0x15e2dc2d970>]
```



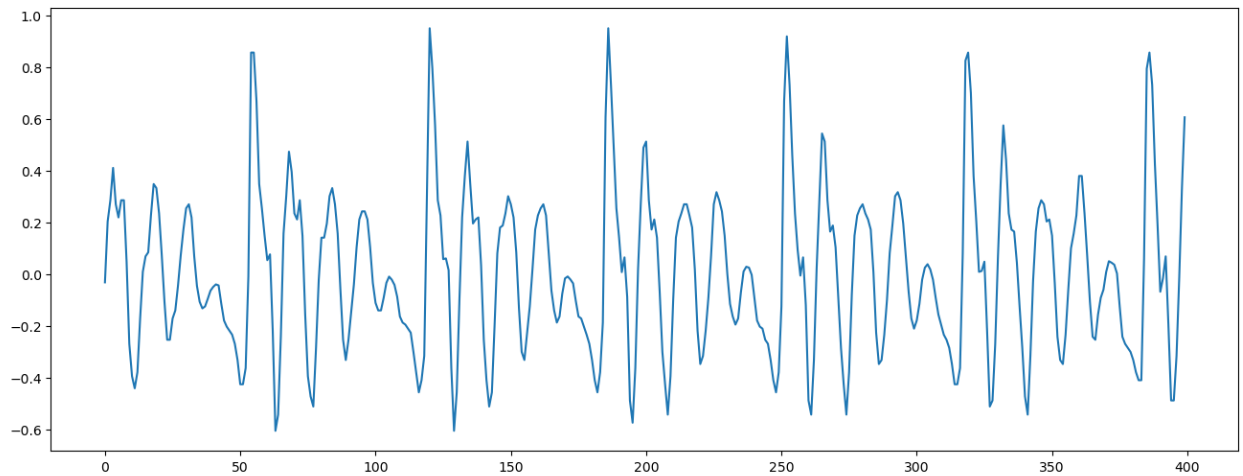
A voiced region that cover six pitch period is [2600:3000]

In [37]:
```
# make the plot not interactive
%matplotlib inline
import matplotlib.pyplot as plt
# specify the size of the plot
plt.rcParams['figure.figsize'] = (16, 6)
```

4. Plot the voiced region.

In [44]:
```
plt.plot(go[2600:3000])
```

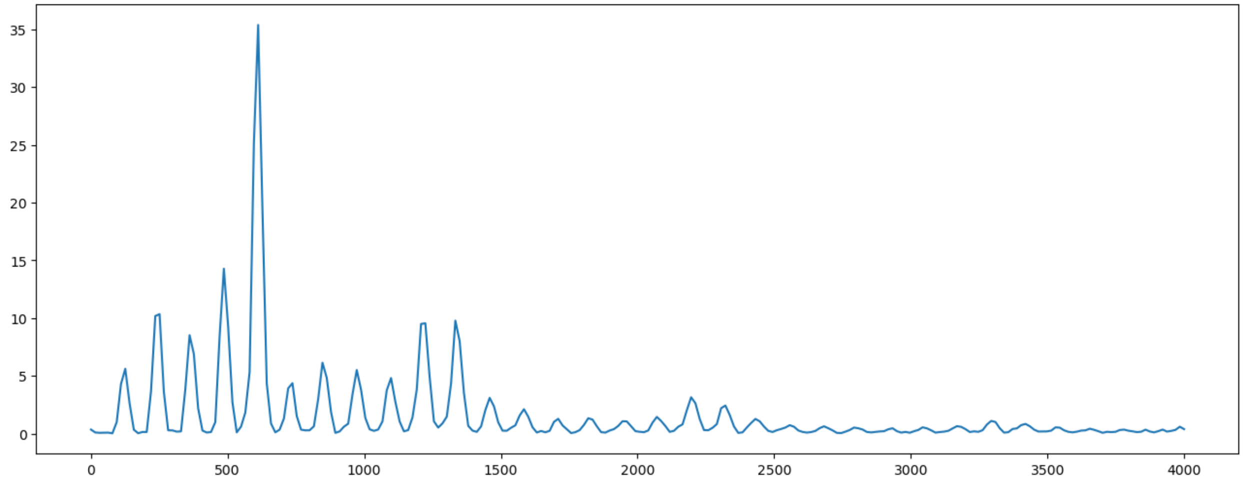
Out[44]: [<matplotlib.lines.Line2D at 0x15e2bd5d190>]



5. Use `DFTwin()` to compute a 512-point DFT of the speech segment, with a window that covers 6 pitch periods within the voiced region, and then plot it for $0 \leq \omega \leq \pi$.

In [86]:
```
DFT_go = DFTwin(go, 400, 2600, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_go[:512//2])
```

Out[86]: [<matplotlib.lines.Line2D at 0x15e3a0cb2b0>]



6. Describe the general shape of the spectrum, and estimate the formant frequencies for the region of voiced speech.

Looks like some impulses. The formant frequencies roughly is 600Hz

Exercise 3.4

1. Complete the function below that can create a spectrogram using your `DFTwin()` function from the previous section. You will do this by creating a matrix of windowed DFTs, oriented as described above.

Important hints:

- You can start by initializing `A` as an empty list, then keep appending the DFT results from `DFTwin()` to it. At the end, convert it to a NumPy array by `A = np.array(A)`.
- After you do the step above, you should find that the frequency components are on the x-axis, but we want them to be on the y-axis. You might use `np.transpose()`.
- Your `DFTwin()` function returns the DT spectrum for frequencies between 0 and 2π. Therefore, you will only need to use the first or second half of these DFTs.
- The statement `B[:, n]` references the entire nth (zero-based index) column of the matrix `B`.

In [54]:
```
def Specgm(x, L, overlap, N):
    """
    Parameters
    ---
    x: the input signal
    L: the window length
    overlap: the number of points common to successive windows
    N: the number of points computed in each DFT

    Returns
    ---
    A: the matrix of spectrogram
    """

    A = []
    for m in range(0,len(x)-L+1,L-overlap):
        A.append(DFTwin(x,L,m,N))

    A = np.array(A)
    A = A[:,:N//2]
    A = np.transpose(A)
    return A
```

2. Load the file `signal.npy` using `np.load()` . Plot the signal.

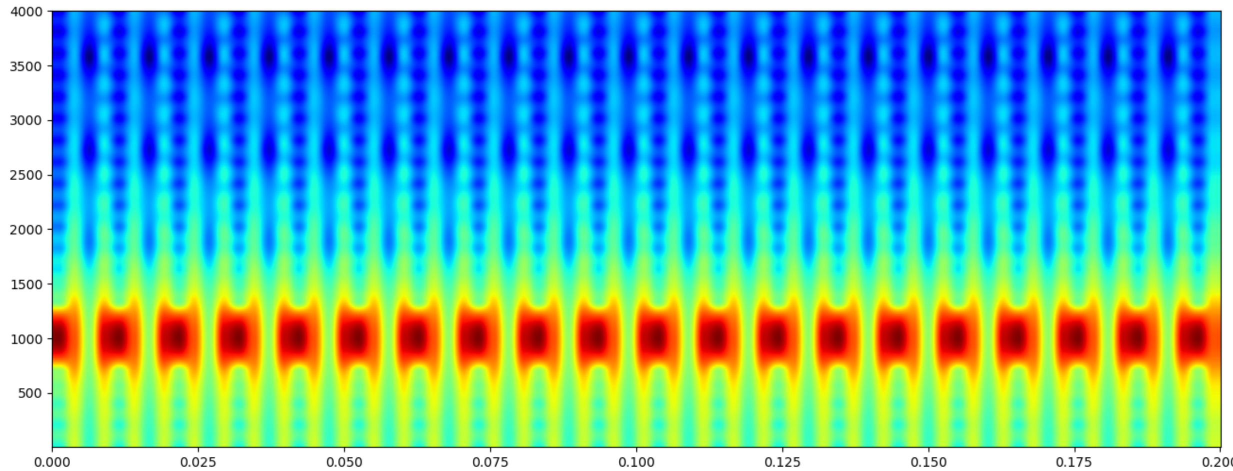
In [61]:
```
signal = np.load('signal.npy')
```

3. Plot the magnitude (in dB) of the wideband spectrogram, using a window length of 50 samples and an overlap of 30 samples.

- In labeling the axes of the image, assume a sampling frequency of 8 KHz. Then the frequency will range from 0 to 4000 Hz.
- Set the parameter `extent=[0, len(signal) / 8000, 1, 4001]` in `plt.imshow()` to correctly label the axes.
- Set the parameter `aspect='auto'` in `plt.imshow()` .
- You can get a pseudo-color mapping by setting the parameter `cmap='jet'` in `plt.imshow()` .
- Set the parameter `origin='lower'` or `origin='upper'` in `plt.imshow()` to place the origin of your plot in the lower/upper left corner, depending on your matrix returned by `Specgm()` .
- For more information, see the online help for the [`plt.imshow()` (https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.axes.Axes.imshow.html)](https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.axes.Axes.imshow.html) command.

In [90]:
```
A = Specgm(signal, 50, 30, 512)
A = 20*np.log10(A)
plt.imshow(A,extent=[0,len(signal)/8000,1,4001], aspect='auto',cmap='jet',origin='lower')
```

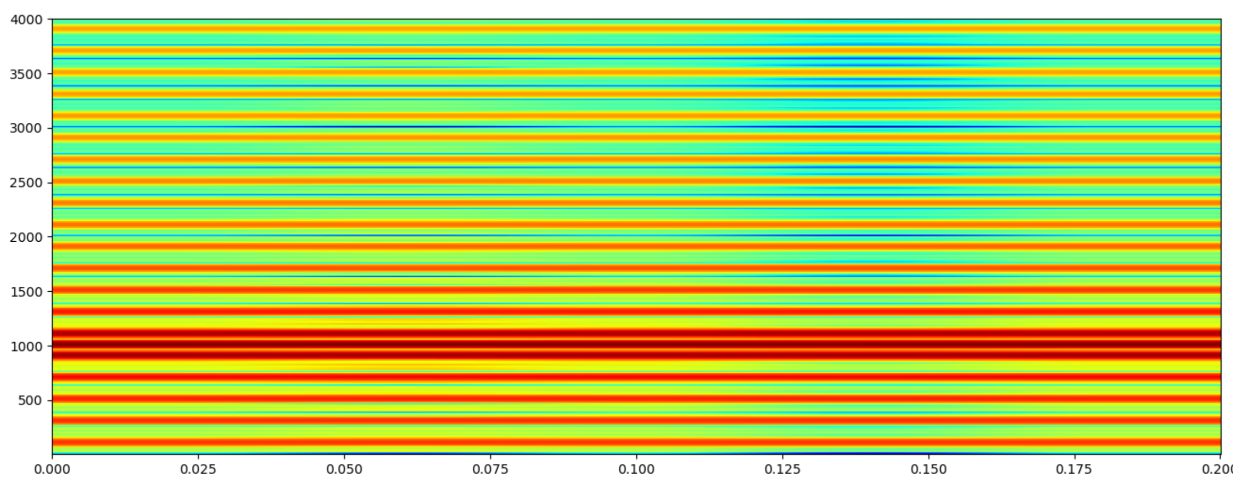
Out[90]: <matplotlib.image.AxesImage at 0x15e3afe9d90>



4. Plot the magnitude (in dB) of the narrowband spectrogram, using a window length of 320 samples and an overlap of 60 samples.

In [118]:
```
A1 = Specgm(signal, 320, 60, 512)
A1 = 20*np.log10(A1)
plt.imshow(A1,extent=[0,len(signal)/8000,1,4001], aspect='auto',cmap='jet',origin='lower'
```

Out[118]: <matplotlib.image.AxesImage at 0x15e42c6a130>



5. Do you see vertical striations in the wideband spectrogram? Similarly, do you see horizontal striations in the narrowband spectrogram? In each case, what causes these lines, and what does the spacing between them represent?

```
Yes. In wideband spectrogram, because the window is small to cover one pitch period,
it represent frequency better than time. For narrowband spectrogram the window cover
multiple pitch period, so the time representation is better. The line is performance
of signal along the line. Spacing is missing information.
```

Exercise 3.6

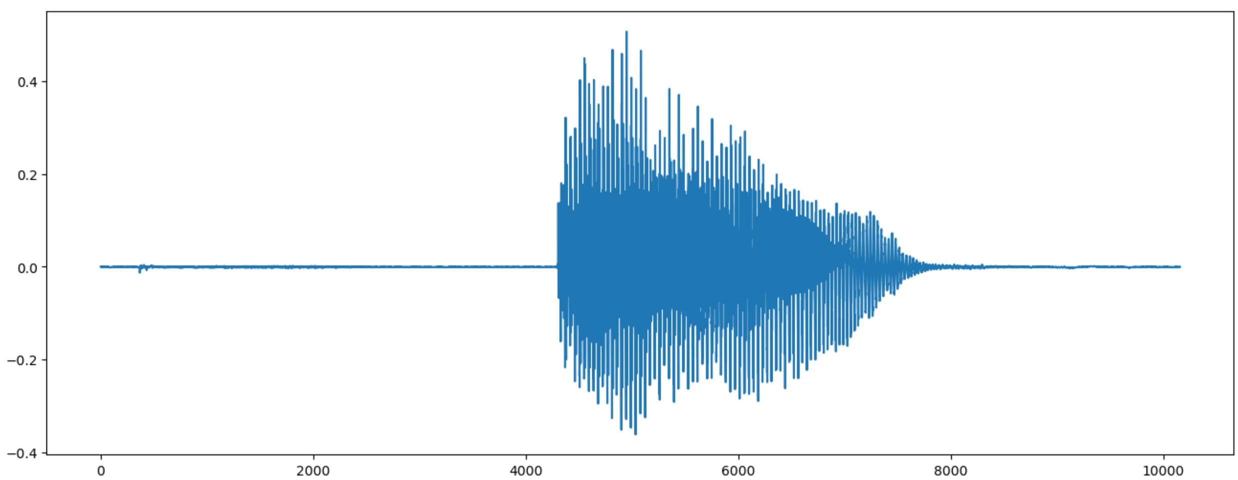
1. Run the following code to load the vowel utterances *a*, *e*, *i*, *o*, and *u* from a female speaker.

In [70]:
```
a = np.load("a.npy")
e = np.load("e.npy")
i = np.load("i.npy")
o = np.load("o.npy")
u = np.load("u.npy")
```

2. Plot the magnitude (in dB) of the narrowband spectrogram of each of the utterances.

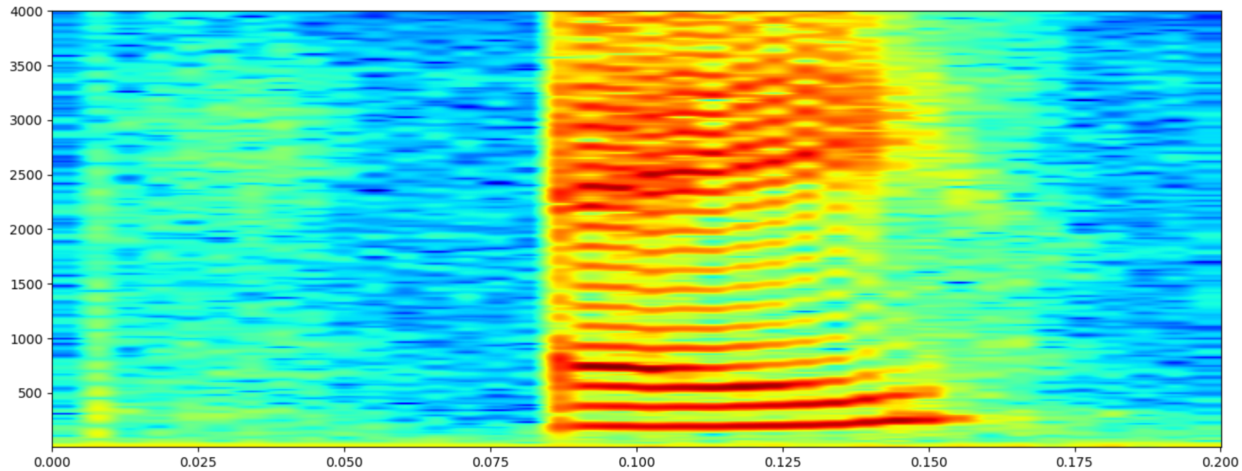
In [107]:
```
plt.plot(a)
```

Out[107]:
```
[<matplotlib.lines.Line2D at 0x15e3d4a1d90>]
```



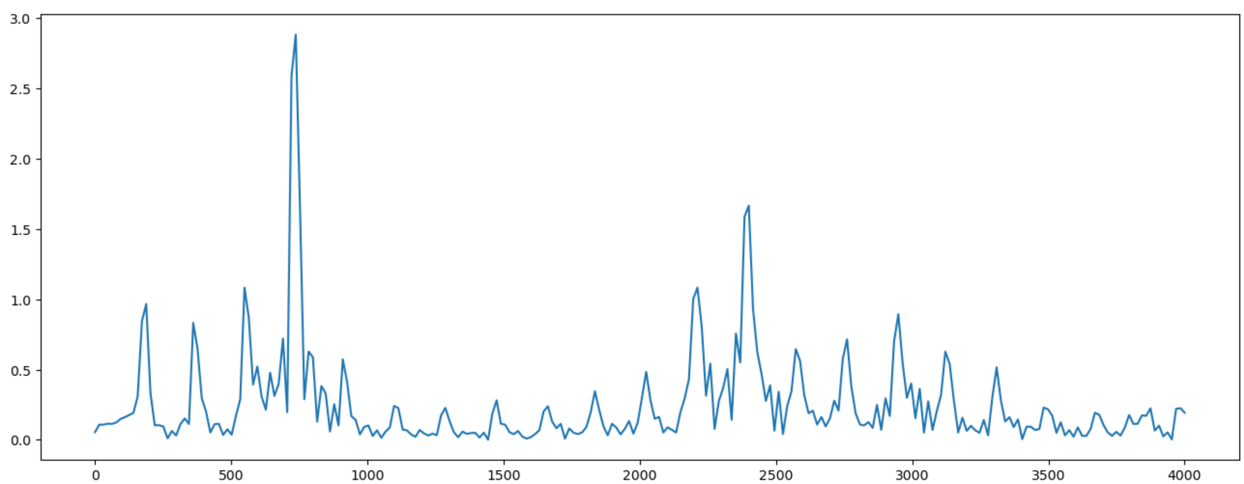
In [97]:
```
Aa = Specgm(a, 320, 60, 512)
Aa = 20*np.log10(Aa)
plt.imshow(Aa,extent=[0,len(signal)/8000,1,4001], aspect='auto',cmap='jet',origin='lower'
```

Out[97]: <matplotlib.image.AxesImage at 0x15e3c546940>



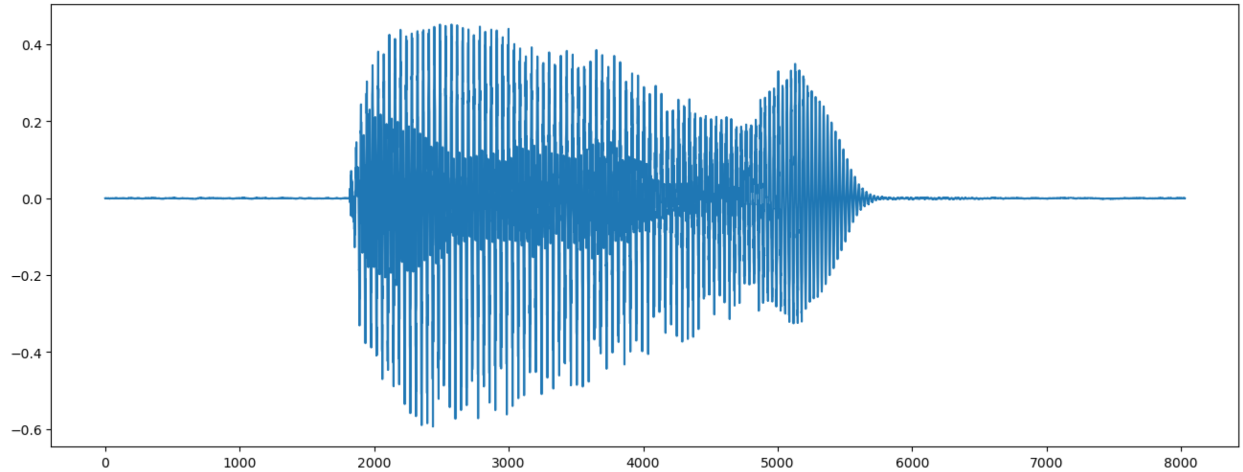
In [108]:
```
DFT_a = DFTwin(a, 3800, 4200, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_a[:512//2])
```

Out[108]: [<matplotlib.lines.Line2D at 0x15e3e71aeb0>]



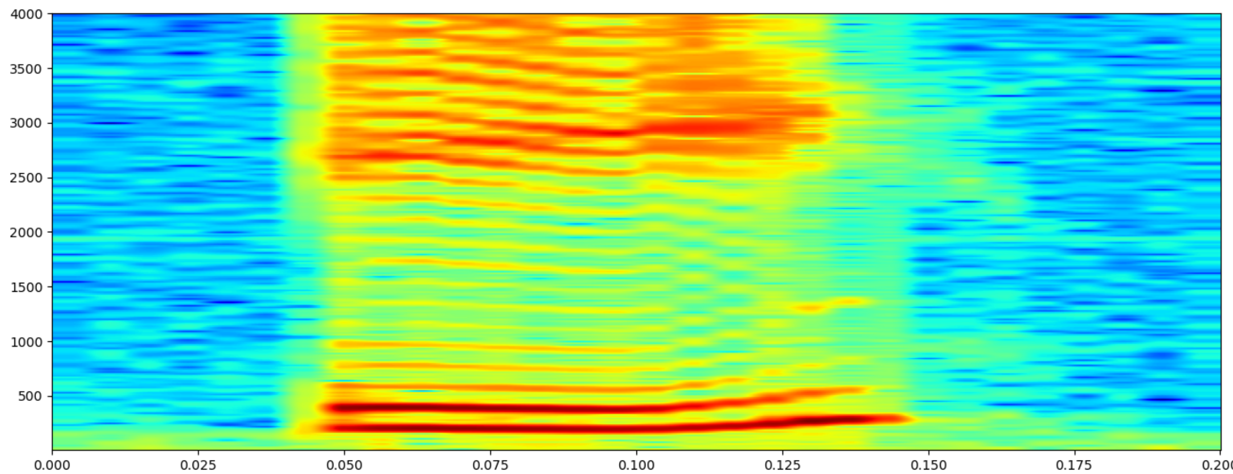
In [109]:
```
plt.plot(e)
```

Out[109]: [<matplotlib.lines.Line2D at 0x15e3d4a1220>]



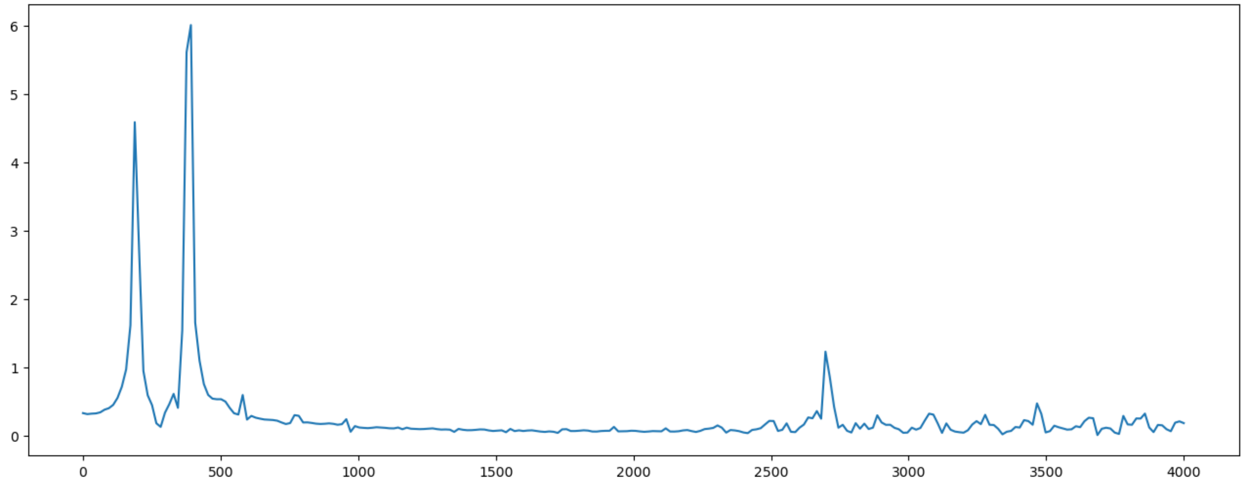
In [99]:
```
Ae = Specgm(e, 320, 60, 512)
Ae = 20*np.log10(Ae)
plt.imshow(Ae,extent=[0,len(signal)/8000,1,4001], aspect='auto',cmap='jet',origin='lower'
```

Out[99]: <matplotlib.image.AxesImage at 0x15e3c8d20d0>



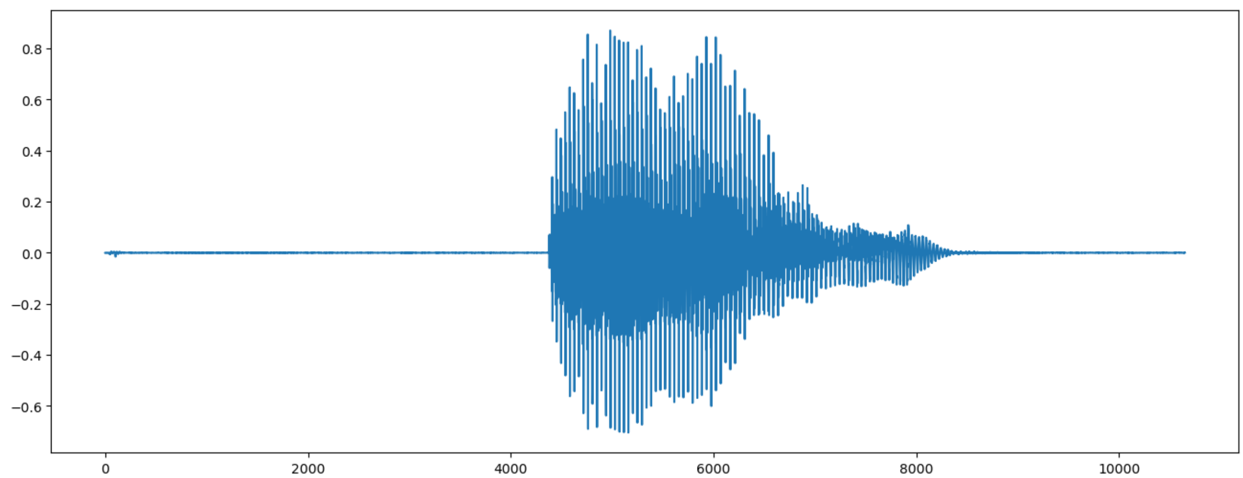
In [110]:
```
DFT_e = DFTwin(e, 4000, 1800, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_e[:512//2])
```

Out[110]: [<matplotlib.lines.Line2D at 0x15e3fa7a790>]



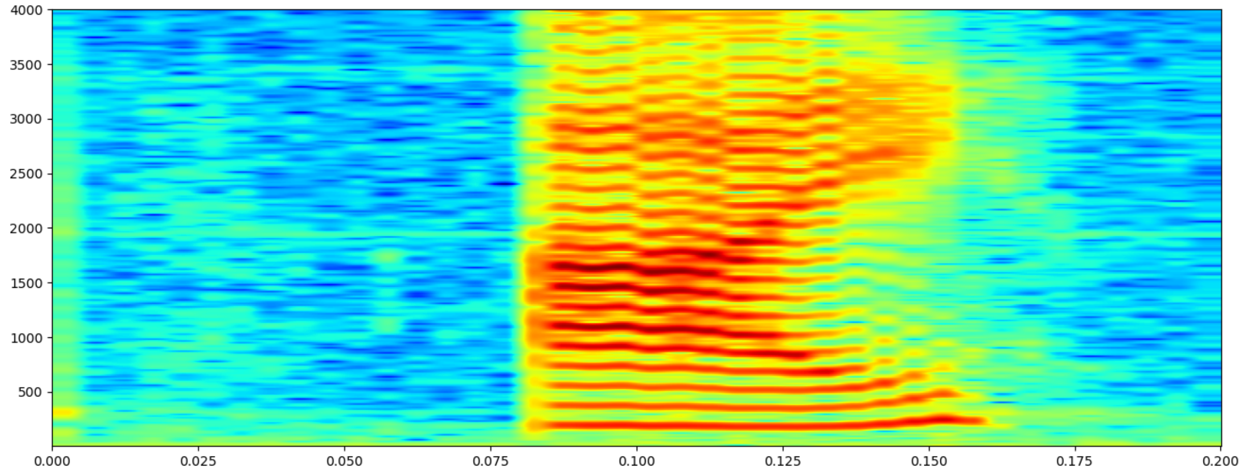
In [111]:
```
plt.plot(i)
```

Out[111]: [<matplotlib.lines.Line2D at 0x15e3fae5820>]



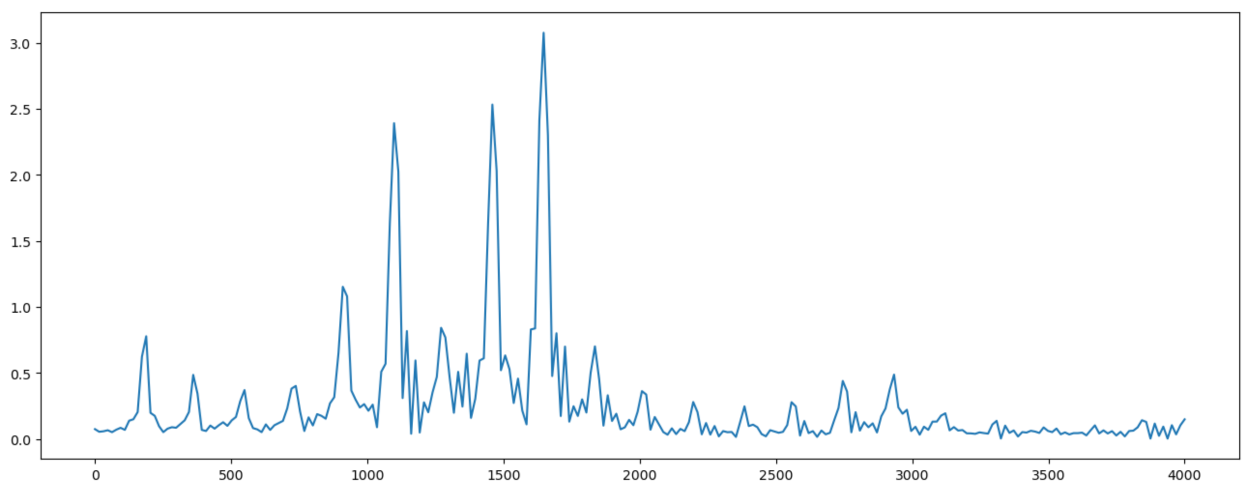
In [101]:
```
Ai = Specgm(i, 320, 60, 512)
Ai = 20*np.log10(Ai)
plt.imshow(Ai,extent=[0,len(signal)/8000,1,4001], aspect='auto',cmap='jet',origin='lower'
```

Out[101]: <matplotlib.image.AxesImage at 0x15e3d1ca6a0>



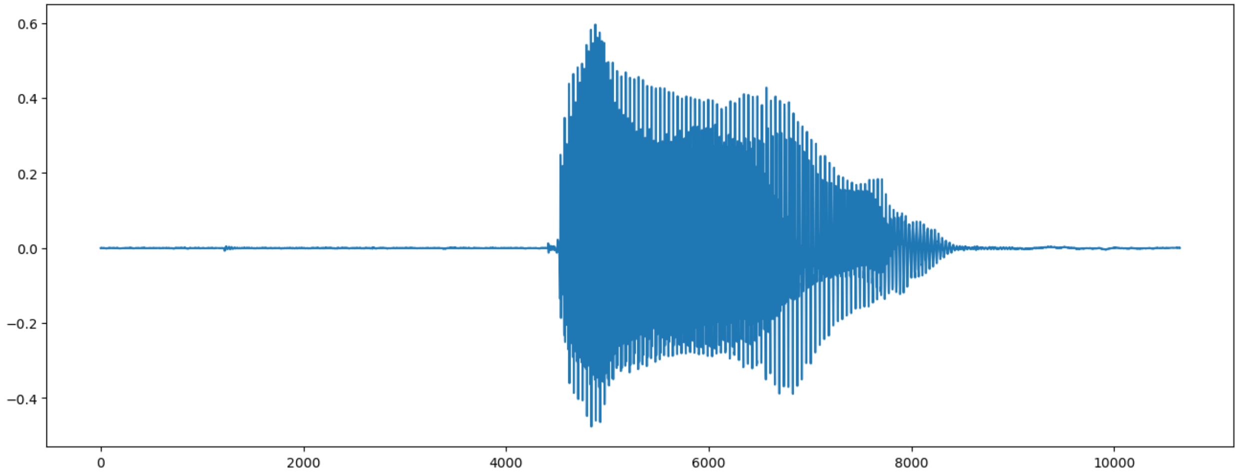
In [112]:
```
DFT_i = DFTwin(i, 4000, 4200, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_i[:512//2])
```

Out[112]: [<matplotlib.lines.Line2D at 0x15e3fdadaf0>]



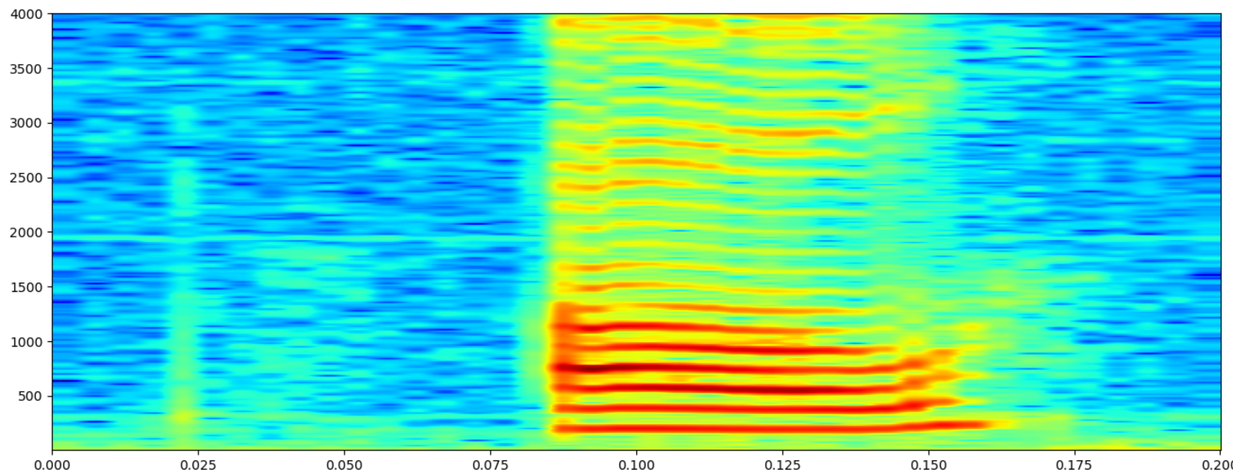
In [114]:
```
plt.plot(o)
```

Out[114]: [<matplotlib.lines.Line2D at 0x15e3e7555b0>]



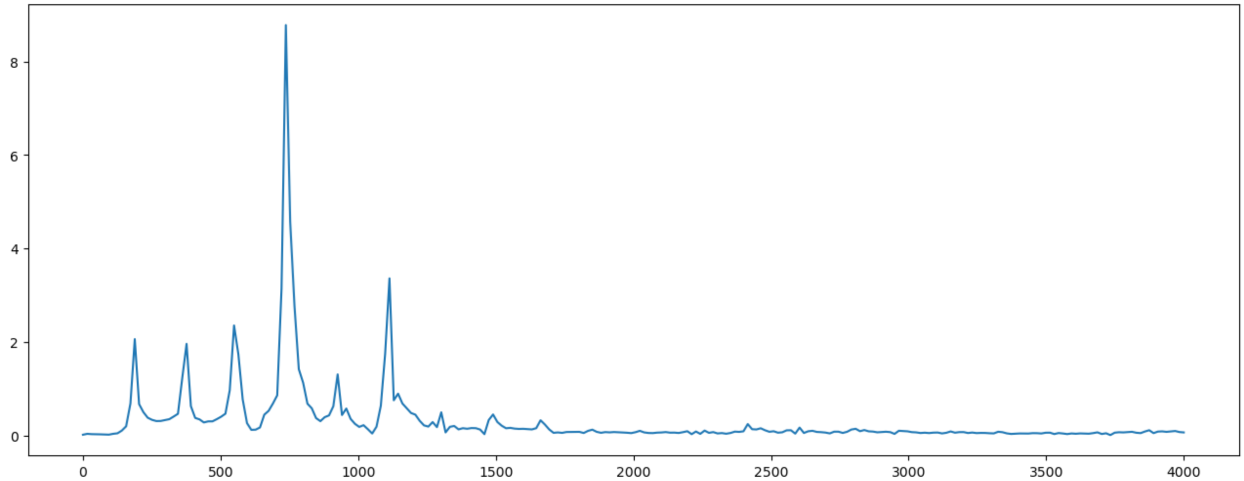
In [103]:
```
Ao = Specgm(o, 320, 60, 512)
Ao = 20*np.log10(Ao)
plt.imshow(Ao,extent=[0,len(signal)/8000,1,4001], aspect='auto',cmap='jet',origin='lower'
```

Out[103]: <matplotlib.image.AxesImage at 0x15e3d7c9e50>



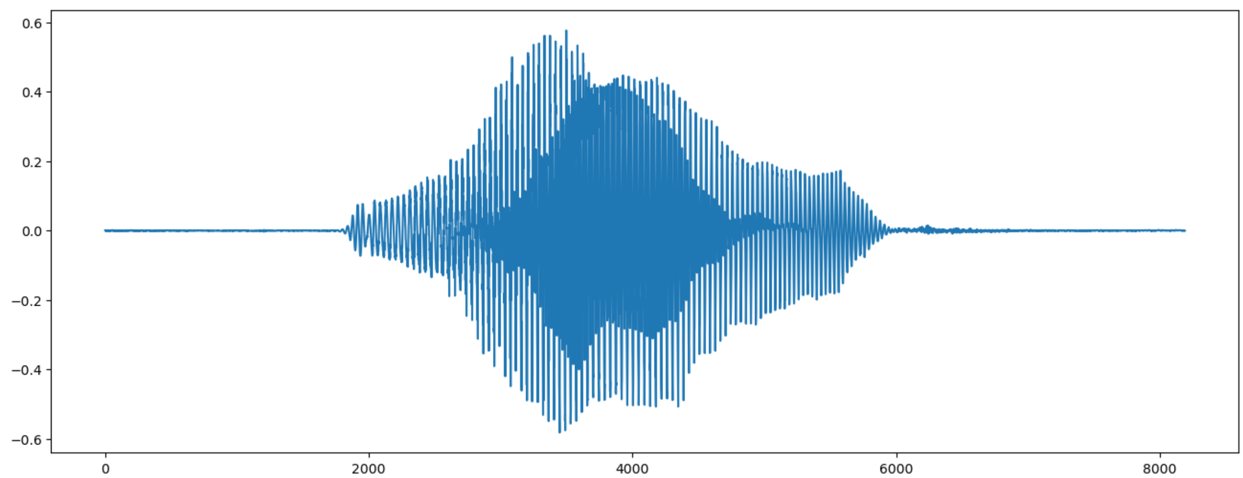
In [115]:
```
DFT_o = DFTwin(o, 4000, 4500, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_o[:512//2])
```

Out[115]: [<matplotlib.lines.Line2D at 0x15e411e0610>]



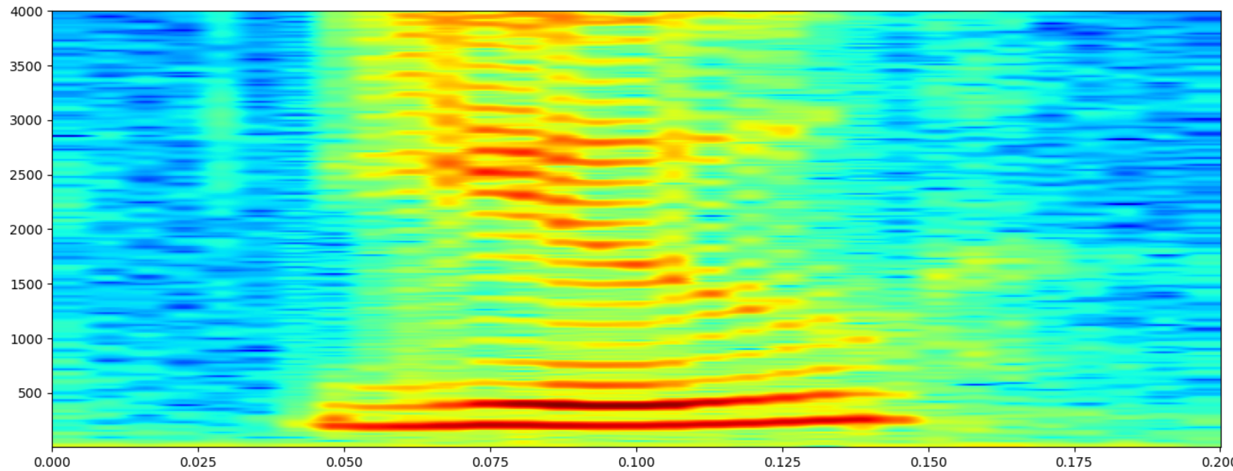
In [116]:
```
plt.plot(u)
```

Out[116]: [<matplotlib.lines.Line2D at 0x15e41241a30>]



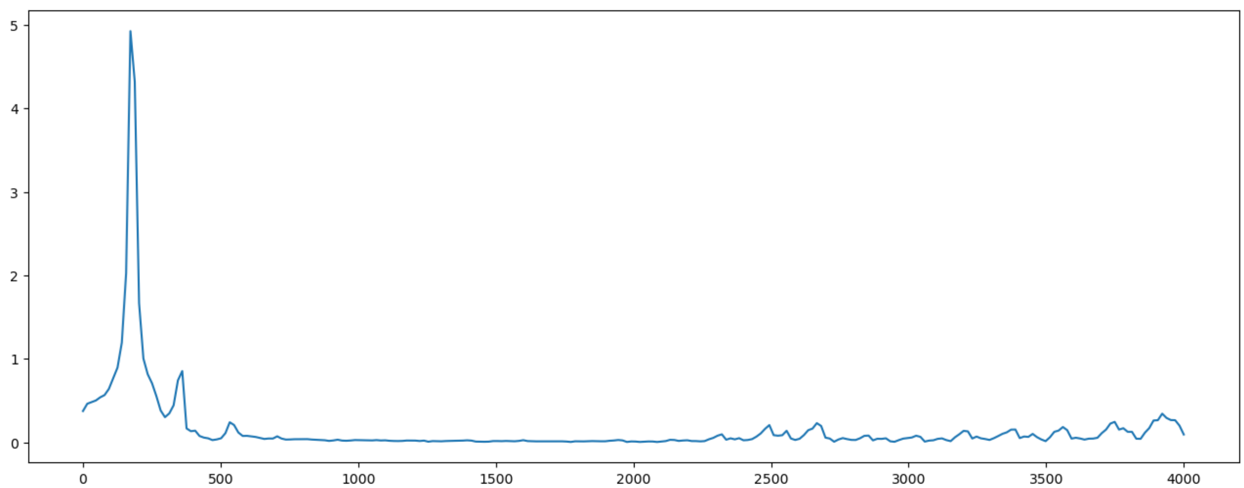
In [105]:
```
Au = Specgm(u, 320, 60, 512)
Au = 20*np.log10(Au)
plt.imshow(Au,extent=[0,len(signal)/8000,1,4001], aspect='auto',cmap='jet',origin='lower'
```

Out[105]: <matplotlib.image.AxesImage at 0x15e3ddd7d60>



In [120]:
```
DFT_u = DFTwin(u, 2000, 1900, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_u[:512//2])
```

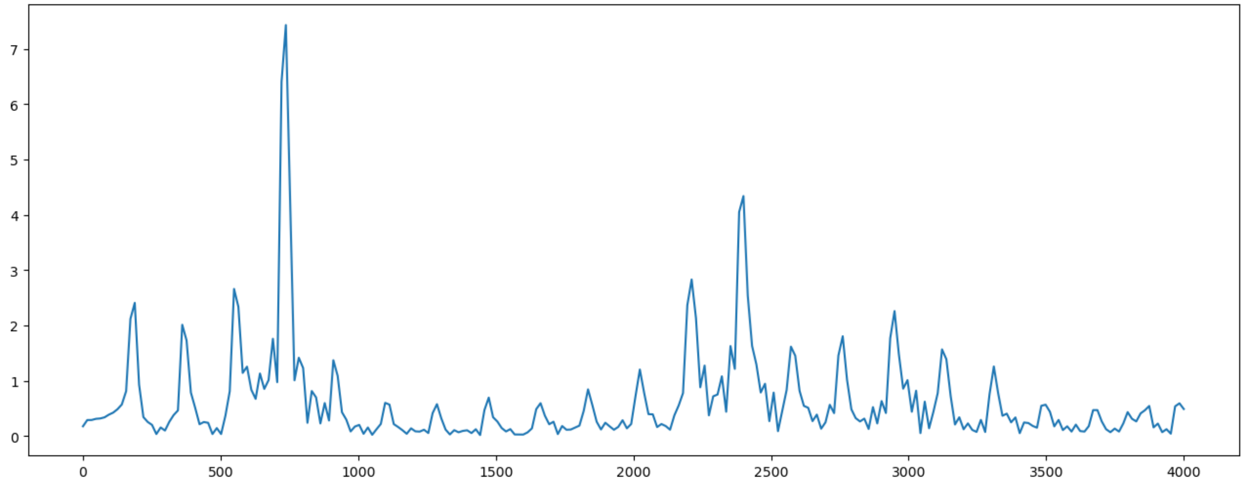
Out[120]: [<matplotlib.lines.Line2D at 0x15e42fddca0>]



3. For the vowels a and u, estimate the first two formant frequencies using the functions you created in the previous sections. Make your estimates at a time frame toward the beginning of the utterance, and another set of estimates toward the end of the utterance. You may want to use both the `Specgm()` and `DFTwin()` functions to determine the formants. Plot these four points in the vowel triangle provided in the file `vowel_triangle.pdf`. For each vowel, draw a line connecting the two points, and draw an arrow indicating the direction the formants are changing as the vowel is being uttered.

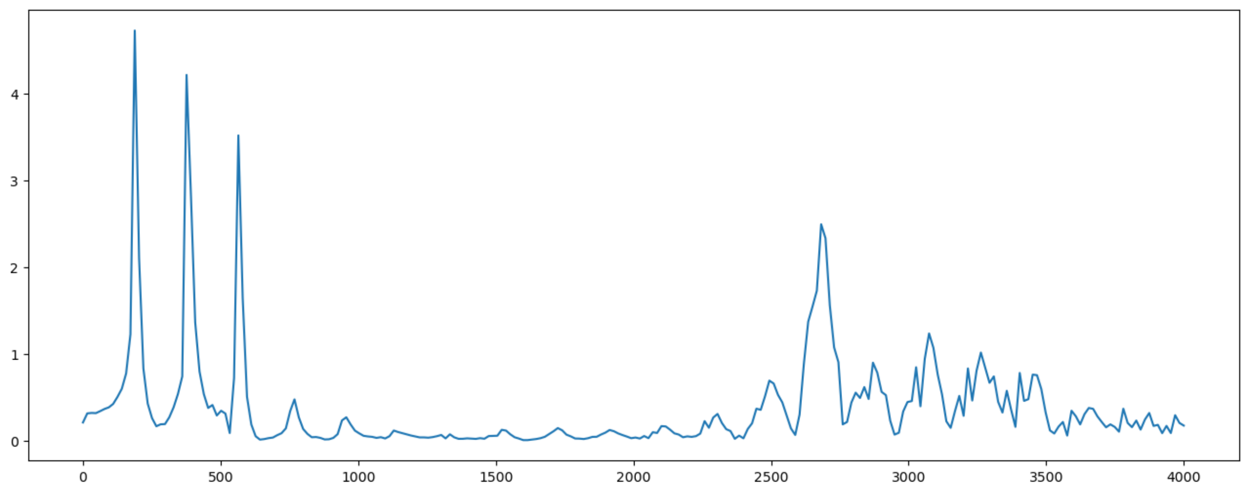
In [127]:
```
DFT_a1 = DFTwin(a, 1800, 4200, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_a1[:512//2])
```

Out[127]: [<matplotlib.lines.Line2D at 0x15e448affa0>]



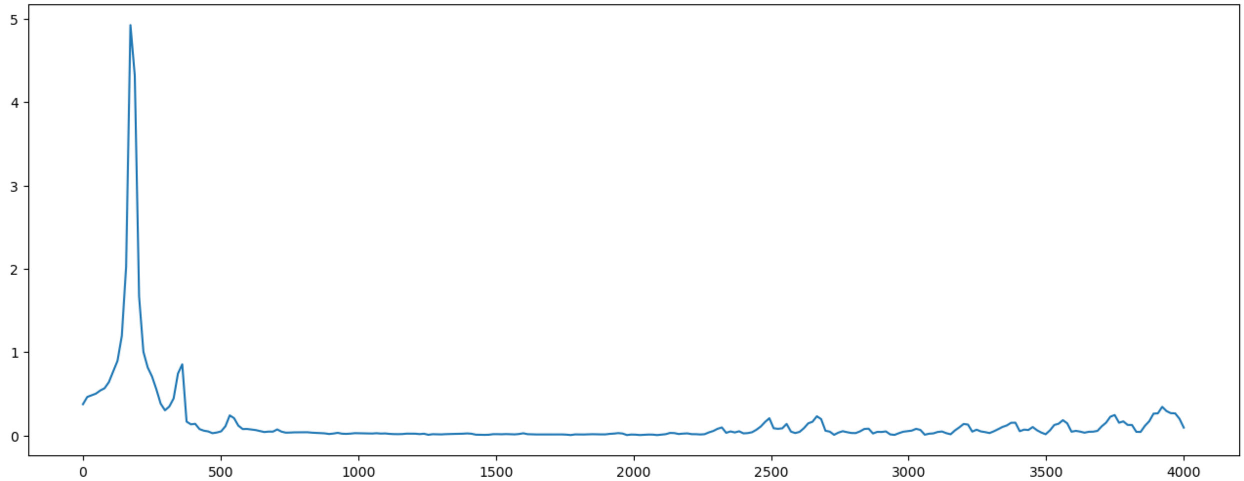
In [128]:
```
DFT_a2 = DFTwin(a, 1800, 6000, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_a2[:512//2])
```

Out[128]: [<matplotlib.lines.Line2D at 0x15e44ba1cd0>]



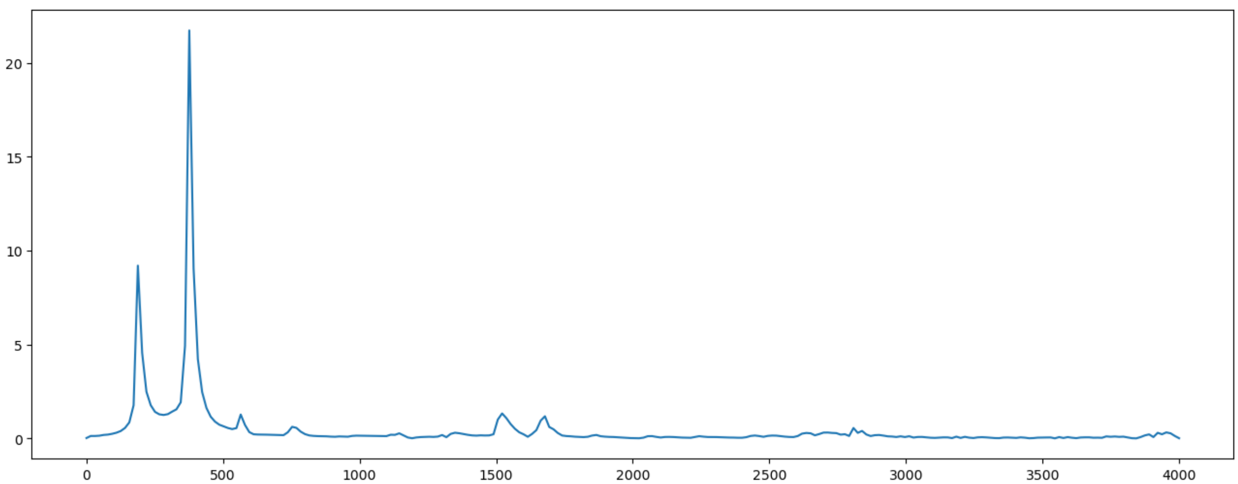
In [125]:
```
DFT_u1 = DFTwin(u, 2000, 1900, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_u1[:512//2])
```

Out[125]: [<matplotlib.lines.Line2D at 0x15e4426a370>]



In [126]:
```
DFT_u2 = DFTwin(u, 2000, 3900, 512)
w = np.linspace(0, np.pi, 512//2)
f = w/(2*np.pi)*8000
plt.plot(f, DFT_u2[:512//2])
```

Out[126]: [<matplotlib.lines.Line2D at 0x15e442d0460>]



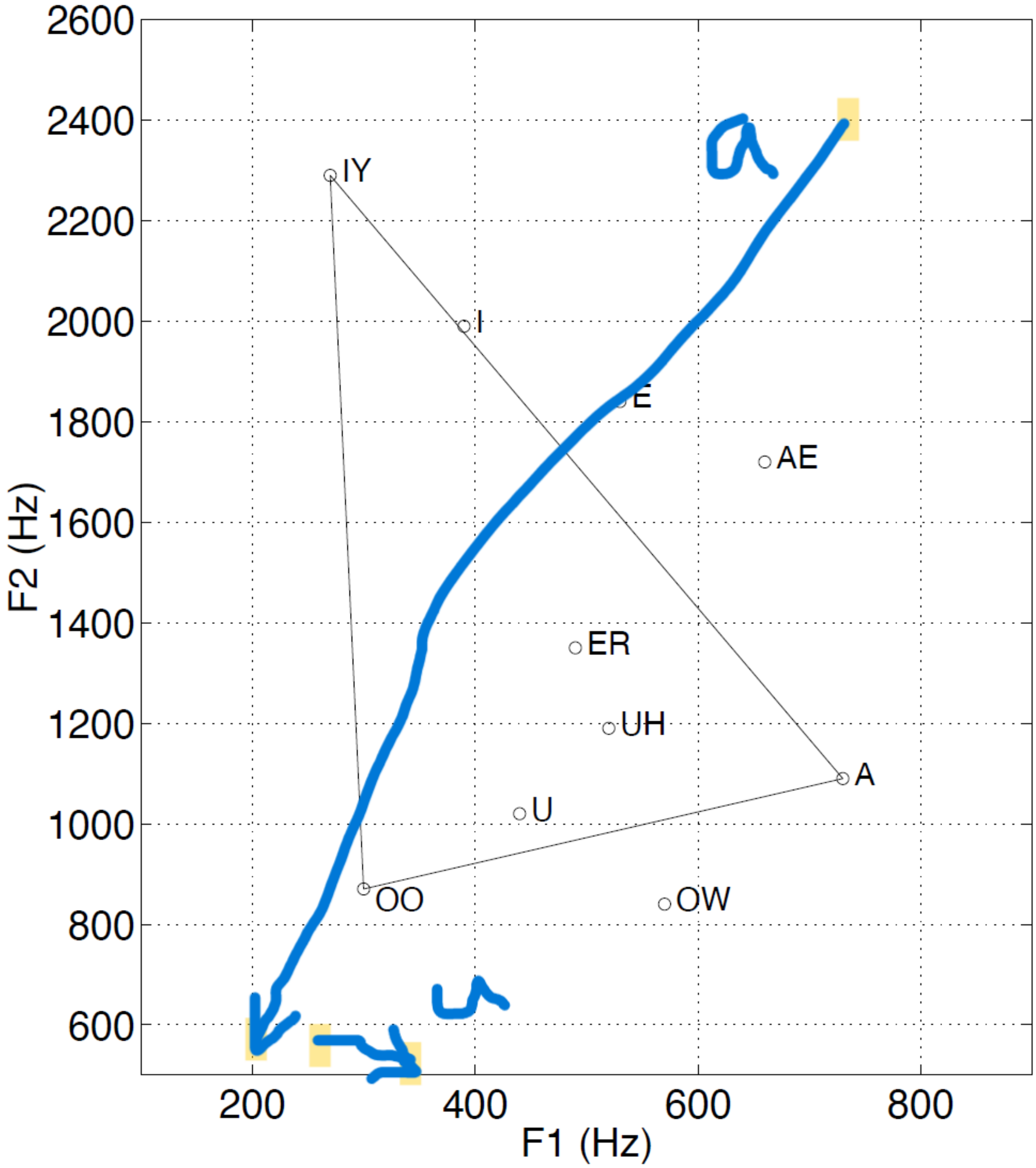


Figure 6: The Vowel Triangle

remember to append `vowel_triangle.pdf` when you submit the report on Gradescope.