hw3

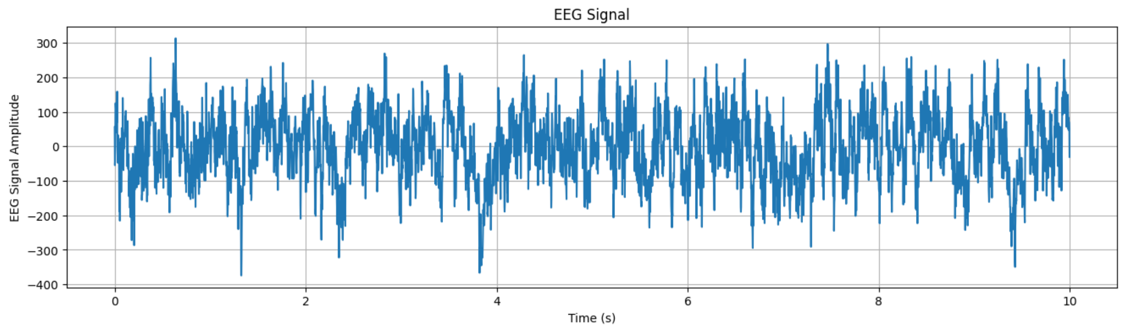
February 5, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from scipy.signal import welch

data = pd.read_csv('HW3_Cogs118C_eeg.csv')
```

[2]:
```
data
```

[2]:
```
       Time         EEG
0     0.000   55.681915
1     0.002  -54.693130
2     0.004  -41.902405
3     0.006   33.800045
4     0.008  125.065987
…       …          …
4994  9.988   46.774258
4995  9.990   71.314636
4996  9.992   62.213642
4997  9.994   49.834541
4998  9.996  -30.724459

[4999 rows x 2 columns]
```

[3]:
```
plt.figure(figsize = (16, 4))
plt.plot(data['Time'], data['EEG'])
plt.xlabel('Time (s)')
plt.ylabel('EEG Signal Amplitude')
plt.title('EEG Signal')
plt.grid(True)
plt.show()
```



Question 8: What is the Sampling Rate (F_s) of the recording?

Inverse difference between two consecutivete time stamps: $0.002 - 0.0 = 0.002 = \Delta t$

0.1 Answer:

$F_s = \frac{1}{\Delta t} = \frac{1}{0.002} = 500 = \text{signal sampled 500 Hz times per second}$

[4]:
```
time_stamps = data['Time']
#sampling interval from sample to sample
sampling_interval = time_stamps[1] - time_stamps[0]
Fs = 1 / sampling_interval
print("Sampling Rate of the Recording: ", Fs)
```

```
Sampling Rate of the Recording:  500.0
```

Question 9: What is the duration of the recording, T?

Duration = difference between final and initial time stamps

0.2 Answer:

$T = t_{\text{final}} - t_{\text{initial}} = 9.996 - 0.0 = \ 9.996 = 10s$

[5]:
```
T = time_stamps.iloc[-1] - time_stamps.iloc[0]
print("\nDuratiion of the Recording: ", T)
```

```

Duratiion of the Recording:  9.996
```

Question 10: Calculate the discrete Fourier transform of the signal, using numpy.fft.fft. This will generate a new vector containing 4999 complex Fourier coefficients, $X[k]$. What frequency (in Hz) corresponds to the lowest non-zero component, $X[k = 1]$?

0.3 Answer:

Calculating Frequency for bin k(f) $\rightarrow f = \frac{kFs}{N} \rightarrow$

$$\text{frequency corresponding to lowest non-zero component } (X[k = 1]) = \frac{1 \times 500}{4999} \approx 0.10002000400080016\,\text{Hz}$$

Notes: - This does not calculate amplitude or content of $X[k]$ rather it calculates the physical frequency in Hz that corresponds to a given DFT bin k, which tells us which realworld frequency $X[k]$ represents $(X[k = 1]$ *or* $X[1]$ corresponds to a frequency of $0.10002000400080016\ Hz$, excluding the DC component at $k = 0$ - - $X[k = 1]$: sum of all N individuals contributions for n = 0 through N - 1 and represents the complex amplitude (magnitude and phase) of the signal frequency $f = \frac{kFs}{N}$ - - N:total number of samples = 4999 - - F_s : sampling rate = 500 - - Direct Current (DC)/Zero Frequency: k = 0 component represents average or mean of the entire signal, corresponding to the frequency of 0 Hz. This component is help us in understanding the overall bias or offset present in the signal but does not contribute to the signal's oscillatory behavior or its frequency content above 0 Hz

This calculation allows us to understand the fundamental frequency component present in the signal

Discrete Fourier Transform (DFT) Formula: $X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N}kn}$ - used to calculate complex amplitude of frequency component k of signal - - N:total number of samples - - $x[n]$: given signal in the time domain - - $X[k]$ represents the signal in the frequency domain - - $e^{-\frac{i2\pi}{N}kn}$: complex exponential function that oscillates according to the frequency bin k - - k is the index in the frequency domain ranging from 0 to $N - 1$

Calculating Frequency content of signal for $X[1]$: $x[0] = 55.681915$, $x[1] = -54.693130$, $x[2] = -41.902405$, $x[N - 1] = ...$

$X[1] = \left(55.681915 \cdot e^{-\frac{i2\pi}{4999} \cdot 1 \cdot 0}\right) + \left(-54.693130 \cdot e^{-\frac{i2\pi}{4999} \cdot 1 \cdot 1}\right) + \left(-41.902405 \cdot e^{-\frac{i2\pi}{4999} \cdot 1 \cdot 2}\right) + ... =$ the full $X[1]$ value, which would represent the magnitude and phase of the signal's frequency component at 0.10002000400080016 Hz

In short, f = frequency and $X[k]$ = complex amplitude, and (DFT): $X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N}kn}$ = frequency content

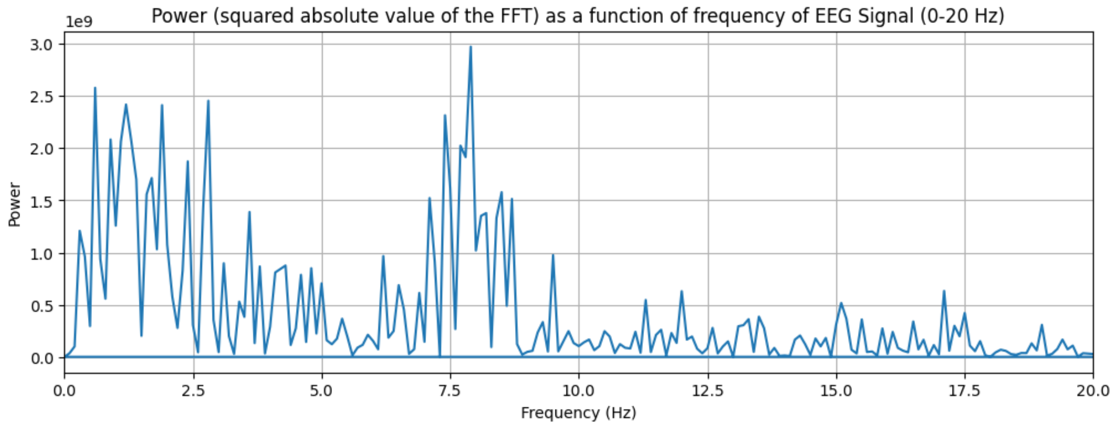
[6]:
```
#Discrete Fourier transform (DFT)
eeg_signal_data = data['EEG']
vector_Xk = np.fft.fft(eeg_signal_data)

#lowest non-zero component X[k = 1]
number_of_signals = len(eeg_signal_data)
all_frequencies = np.fft.fftfreq(number_of_signals, d = sampling_interval)
lowest_non_zero_freq = all_frequencies[1]
print("\nFrequency (Hz) of Lowest non-zero component of discrete Fourier␣
 ↪transform of the signal: ", lowest_non_zero_freq)
```

```
Frequency (Hz) of Lowest non-zero component of discrete Fourier transform of the
signal:  0.10002000400080016
```

Make a plot showing the power (squared absolute value of the FFT) as a function of frequency. Zoom in on the x-axis (using plt.xlim()) to show the range 0-20 Hz. Label the axes. ## Answer:

[7]:
```
#power as function of frequency
powerfunction_as_frequency = np.abs(vector_Xk) ** 2
'''
Spectrum is symmetric and signal is real, thus we extract positive
```

```
frequencies only since the signal is real
'''

plt.figure(figsize = (12, 4))
plt.plot(all_frequencies, powerfunction_as_frequency)
plt.xlim(0, 20)
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.title('Power (squared absolute value of the FFT) as a function of frequency␣
↪of EEG Signal (0-20 Hz)')
plt.grid(True)
plt.show()
```



Notes:

Power Spectral Density (PSD) is the squared absolute value of the DFT coefficients. - PSD as function of frequency: $P(f) = |X[k]|^2$ - - $P(f)$ represents power at frequency f.

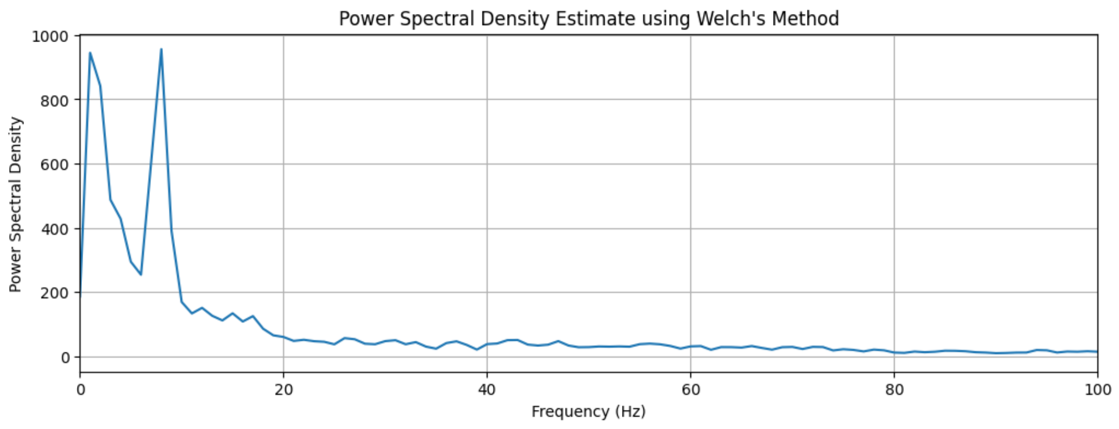
Plot shows insight into the power distribution across this range of frequencies within the EEG signal, which one can use to identify dominant frequency components for neuroscientific analysis

Now use Welch's method to estimate the power spectrum. In python you can use scipy.signal.welch. The parameter nperseg sets how many samples will be included in each data chunk. Choose nperseg=500 and plot the resulting power spectral estimate. Make sure to label the axes. Notice that there is a strong theta rhythm.

11. What is the frequency resolution for Welch's power spectral density estimate?

Welch's method formula: $\Delta f = \frac{F_s}{\text{nperseg}} = \frac{500}{500} = 1.0Hz$ - F_s = 500 Hz - nperseg = 500

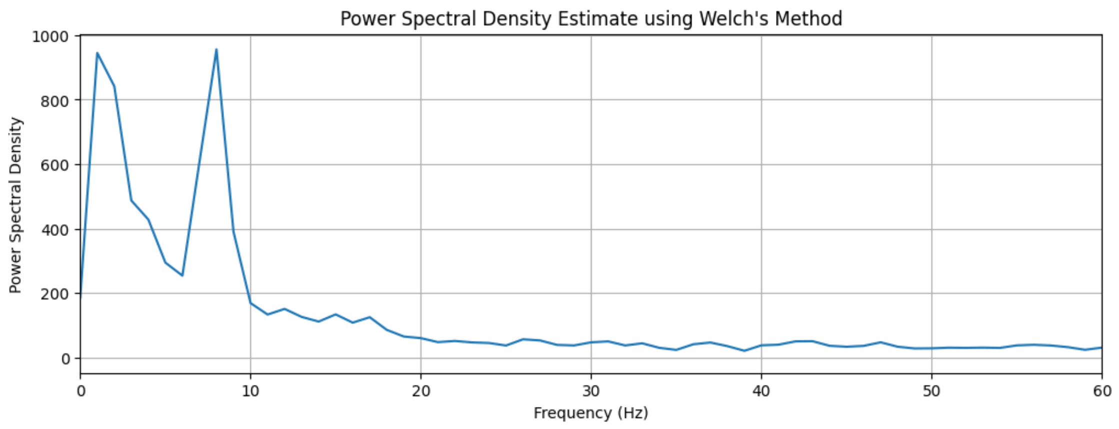
[8]:
```
welch_frequency_bins, welch_power_spectral_density = welch(eeg_signal_data, fs␣
↪= Fs, nperseg = 500)

plt.figure(figsize = (12, 4))
plt.plot(welch_frequency_bins, welch_power_spectral_density)
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power Spectral Density')
plt.title('Power Spectral Density Estimate using Welch\'s Method')
plt.xlim(0, 100)
plt.grid(True)
plt.show()
```



Notes:

Adjusting to see theta rhythm

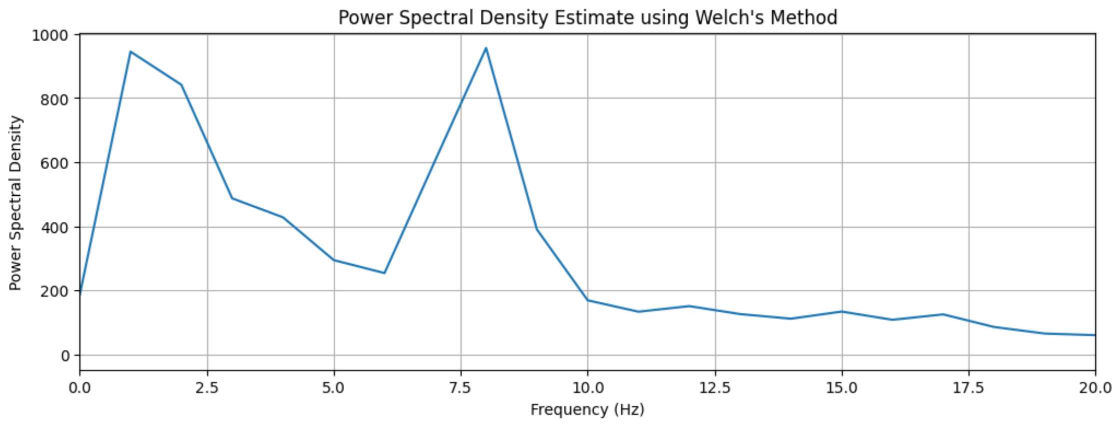
[9]:
```
welch_frequency_bins, welch_power_spectral_density = welch(eeg_signal_data, fs␣
  ↪= Fs, nperseg = 500)

plt.figure(figsize = (12, 4))
plt.plot(welch_frequency_bins, welch_power_spectral_density)
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power Spectral Density')
plt.title('Power Spectral Density Estimate using Welch\'s Method')
plt.xlim(0, 60)
plt.grid(True)
plt.show()
```



Notes:

Adjusting to 0-20Hz to capture theta rhythm

```
[10]: welch_frequency_bins, welch_power_spectral_density = welch(eeg_signal_data, fs␣
      ↪= Fs, nperseg = 500)

      plt.figure(figsize = (12, 4))
      plt.plot(welch_frequency_bins, welch_power_spectral_density)
      plt.xlabel('Frequency (Hz)')
      plt.ylabel('Power Spectral Density')
      plt.title('Power Spectral Density Estimate using Welch\'s Method')
      plt.xlim(0, 20)
      plt.grid(True)
      plt.show()
```



13. What value of npersg would give a frequency resolution of $\Delta f = 0.25$ Hz?

nperseg $= \frac{F_s}{\Delta f} = \frac{500}{0.25} = 2000$