**CSE 3048 PROJECT: Demodulating A Composite Signal**

**Sinan Göçmen – 150 120 519**

File: 81.txt

Modulation frequency 1: 5KHz      Message 1: Trouble, Your Mind (not sure)

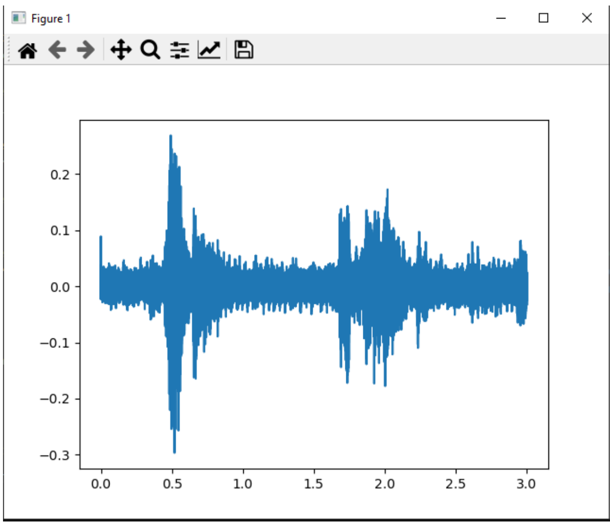
Modulation frequency 2: 10KHz     Message 2: Guilty, Excellent

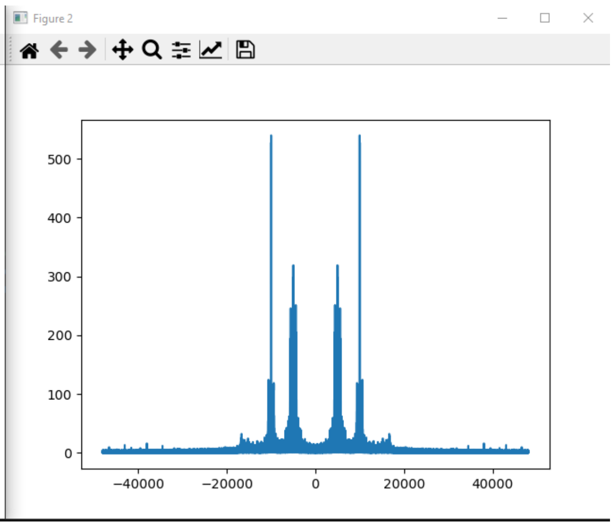
## Explanation

Stage1: Taking composite signal and observing the FT of it.

First, I imported the file and took the Fourier transform of it. Then, I plot the results in both time and frequency domains.

```
fs = 96000 # sample rate
seconds = 3 # recording duration
modulation_frequency = 5000

my_recording = np.loadtxt('81.txt')
time = np.linspace(0,seconds,my_recording.shape[0]) # start, stop, sample count

plt.figure(1) # composite signal in time domain
plt.plot(time, my_recording)

frequencies = np.linspace(-fs/2,fs/2,fs*seconds)
my_recording_in_frequency = fftshift(fft(my_recording))

plt.figure(2) # composite signal in frequency domain
plt.plot(frequencies, np.abs(my_recording_in_frequency))
```

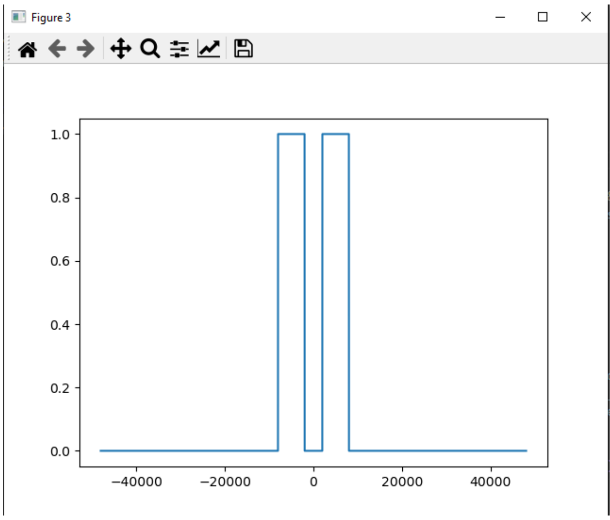


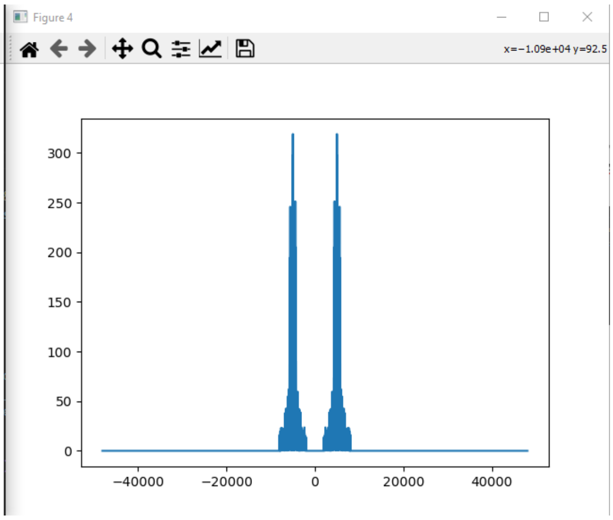



Stage 2: Bandpass Filtering

When we observe the signal, we can see that their modulation frequencies are 5KHz and 10KHz. We should create a bandpass filter accordingly. We take +- 3 KHz of them. The creation of the bandpass filter and an example for the first signal (5KHz) are as follows.

```
bandpass_filter = np.where(((frequencies >= (-modulation_frequency-3000)) & (frequencies <= (-modulation_frequency+3000))) |
                           ((frequencies >= (modulation_frequency-3000)) & (frequencies <= modulation_frequency+3000)),1,0)
bandpass_filtered_record = np.multiply(my_recording_in_frequency, bandpass_filter)

plt.figure(3) # bandpass filter
plt.plot(frequencies, bandpass_filter)

plt.figure(4) # bandpass filtered recording
plt.plot(frequencies, np.abs(bandpass_filtered_record))
```



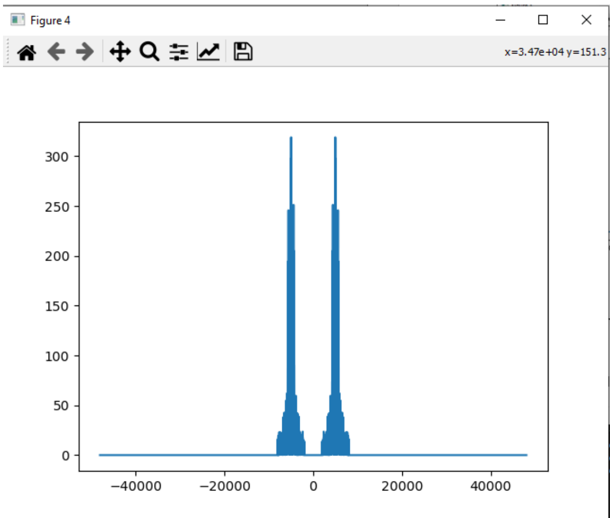


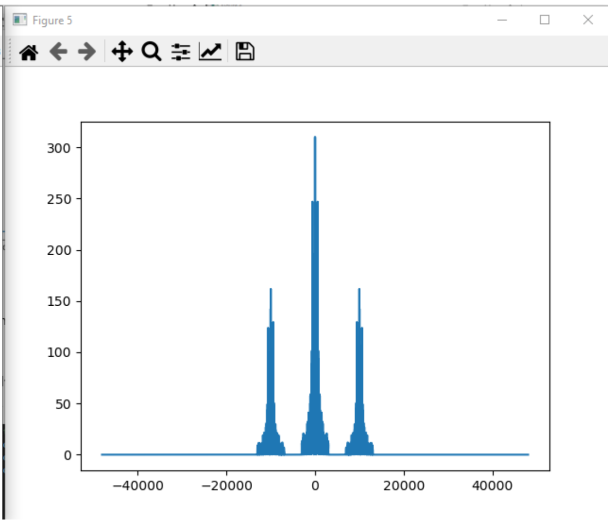

Stage 3: Multiplying with cos(2pift) in time domain.

I preferred multiplying my signal with cos(2pift) in time domain instead of convolving it in frequency domain. So, I took the inverse fourier transform of the signal and did the multiplication.

After that, I took its fourier transform again to progress. The code for these operations and plot results are as follows:

```
#stage2 : multiply with cos(2*pi*f*t) in time domain
bandpass_filtered_record_in_time = ifft(ifftshift(bandpass_filtered_record))
bandpass_filtered_record_stage2 = np.multiply(bandpass_filtered_record_in_time,np.cos(2*np.pi*modulation_frequency*time))
bandpass_filtered_record_stage2_in_frequency = fftshift(fft(bandpass_filtered_record_stage2))

plt.figure(5)
plt.plot(frequencies, np.abs(bandpass_filtered_record_stage2_in_frequency))
```

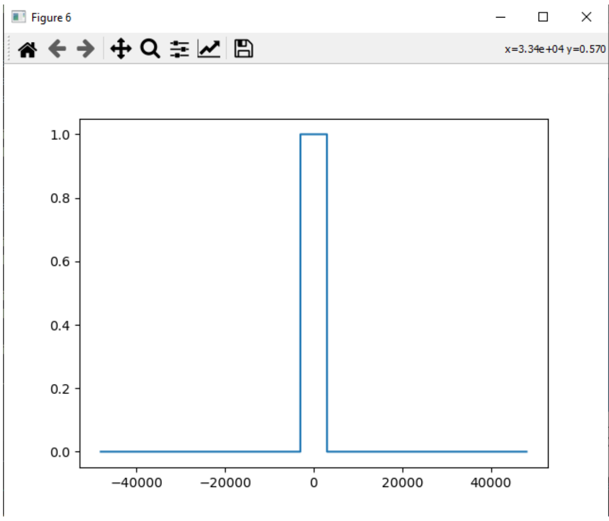


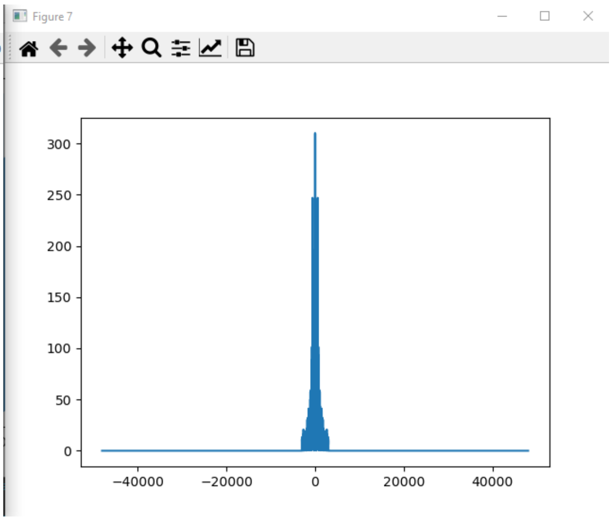




Stage 4: Lowpass Filtering

We only need the signal in the middle of the chart. So, I created a lowpass filter and used it.

```
lowpass_filter = np.where((frequencies >= -3000) & (frequencies <= 3000),1,0)
lowpass_filtered_record = np.multiply(lowpass_filter,bandpass_filtered_record_stage2_in_frequency)

plt.figure(6) # lowpass filter
plt.plot(frequencies, lowpass_filter)

plt.figure(7)
plt.plot(frequencies, np.abs(lowpass_filtered_record))
```




Stage 5: Converting signal to a voice file.

After all the operations above, I took the inverse Fourier transform of the signal and tried to understand the message.

```
sf.write("5k.wav", np.abs(ifft(lowpass_filtered_record)), 96000, subtype= 'PCM_16')
```