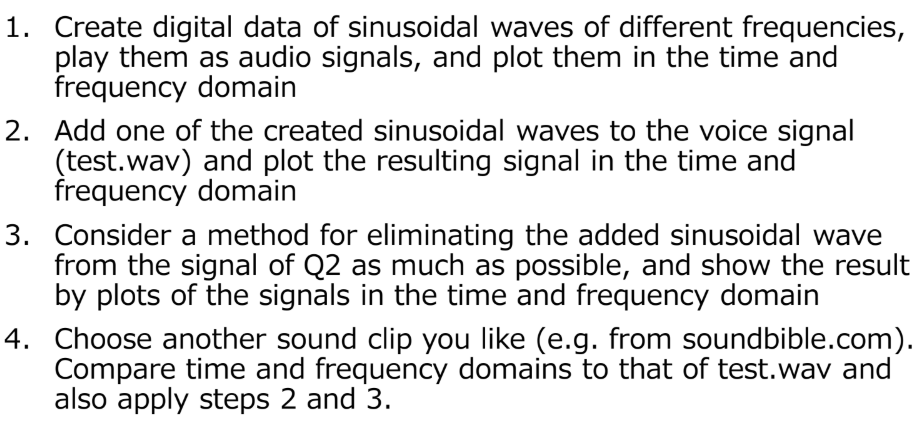
Exercise 7.1 (May 22, 2020)

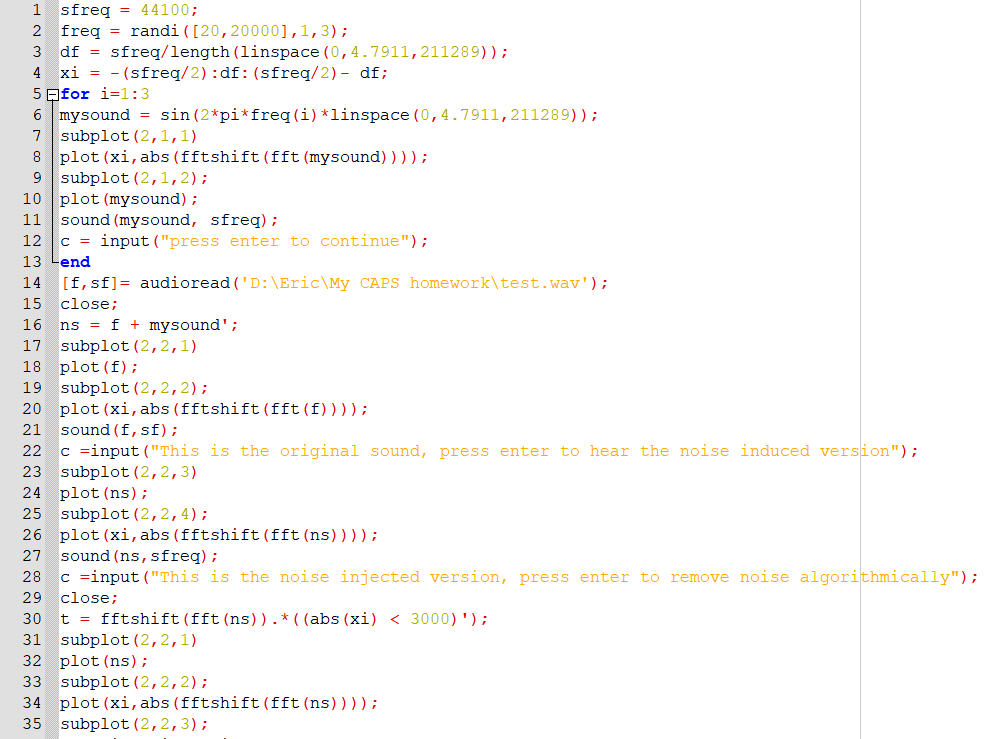
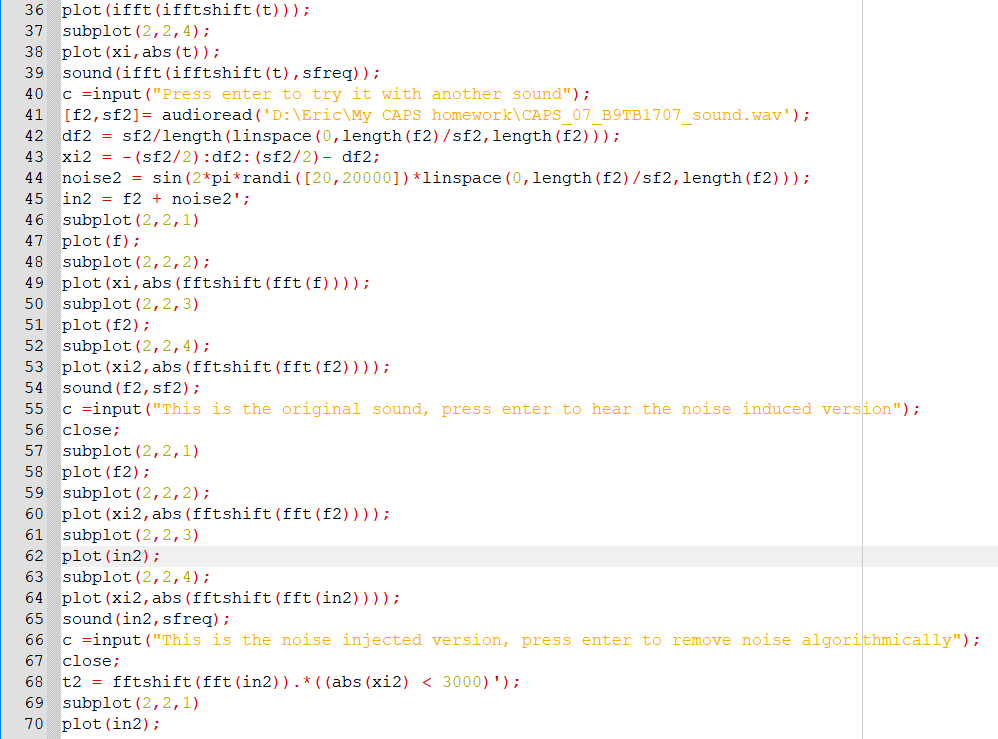
-B9TB1707

# Question:

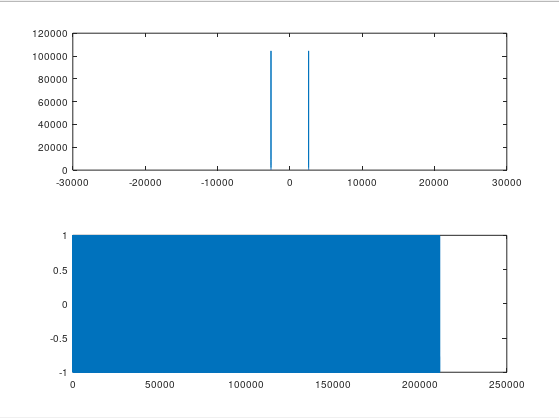


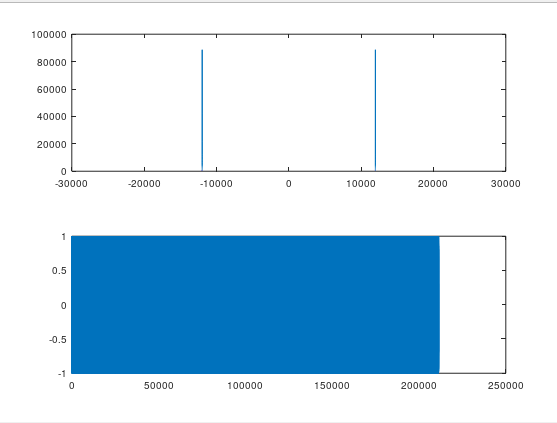
# Solution:

My code for the solution is as follows.

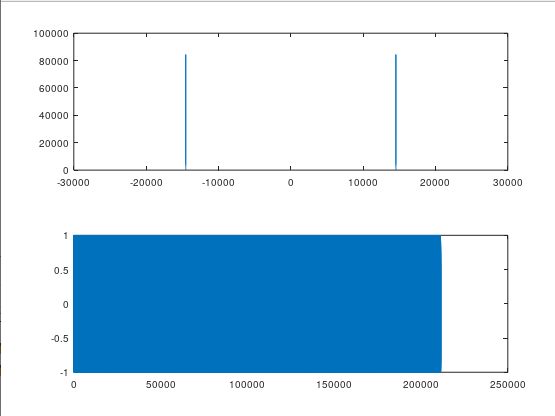
  

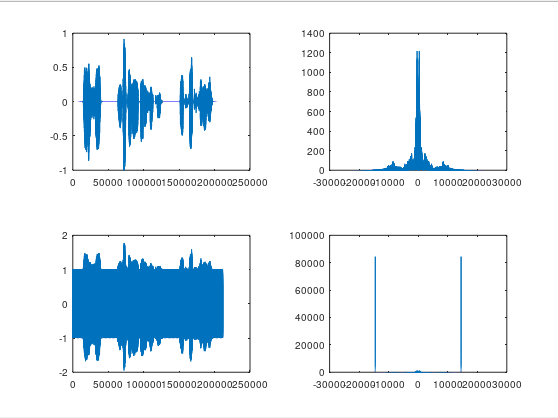
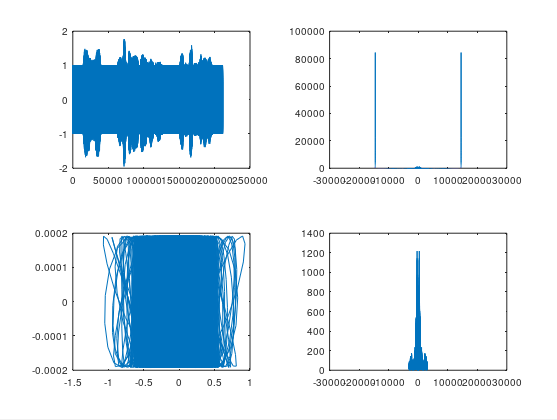
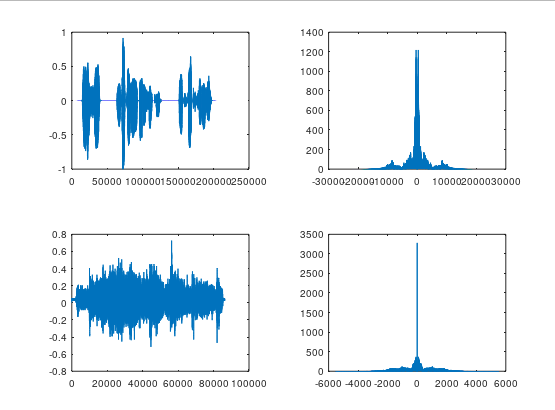
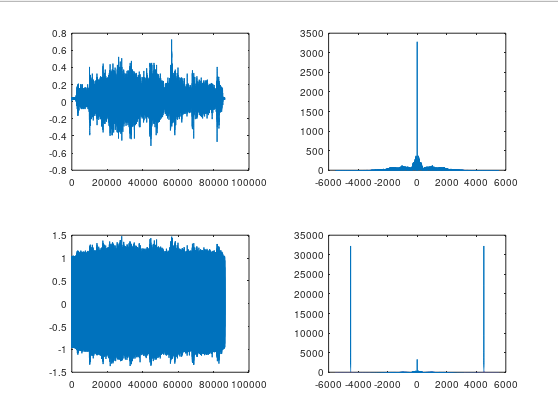
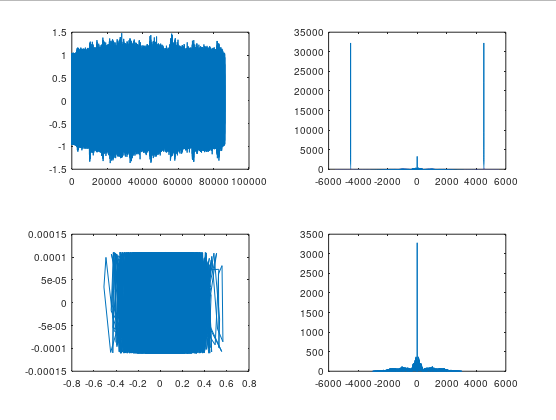
And my outputs are as follows:





These are the plots of the sinusoidial sound waves



The first row shows the noisy version and the second row is de-noised version

The First row is test.wav

And the second row shows the file with noise included.

First row is test.wav

And the second row is another audio file

The first row is the original and the second row is the noisy version

The first row depicts the noisy version and the second the de-noised version

##### How it works:

1. Line 1-4 are basic declarations of sampling frequency, randomized frequency of the sound wave, interva; increment and domain for the fourier plot.
2. Line 5-13 is a for block that generates thre different sound waves, and their time and frequency plots.
3. Line 6 generates a sound wave by by using the sin function, a randomized frequency, and a linespace contruct to store the value of the amplitude at every point the sound wave is sampled.
4. The subplot functions divides up the output screens and the plot function graph the result. The frequency plot is obtained by the methos specified in class( by using the fft and fftshift methods to implement the fourier transform).
5. I used an input function to let the user look at the results and press enter when comfortable.
6. Line 14 imports the test audio file.
7. Line 15 closes the previous plots.
8. Line 16 adds noise to the audio file.
9. Line 17-27 plots the original and noisy data.
10. Line 30 creates a sound with a filter applied. The filter discards any frequency with an absolute value greater than 3000
11. Line 31-40 output the data.
12. Line 40 to 77 repeat the process for another audio file.

# Conclusion:

Thus I have generated soundwaves and manipulated their time and frequency plots to add and remove noise by using Fourier transforms. I used the fft function, even though there are many other functions available (some more suited for this application like fftfilt which has built in filtration), since the filtration part wasn’t was very basic. I believe a better filtration for noise reduction is needed, since all we did was exclude higher frequencies. One way to do this would be to analyse the waveform, but that would be too complicated and require math that I currently don’t know how to do. For this reason, I think that the fft function is sufficient since we are using very rudimentary filtering techniques