Exercise 7.1 (May 22, 2020)

-B9TB1707

Question:

1. Create digital data of sinusoidal waves of different frequencies, play them as audio signals, and plot them in the time and frequency domain
2. Add one of the created sinusoidal waves to the voice signal (test.wav) and plot the resulting signal in the time and frequency domain
3. Consider a method for eliminating the added sinusoidal wave from the signal of Q2 as much as possible, and show the result by plots of the signals in the time and frequency domain
4. Choose another sound clip you like (e.g. from soundbible.com). Compare time and frequency domains to that of test.wav and also apply steps 2 and 3.

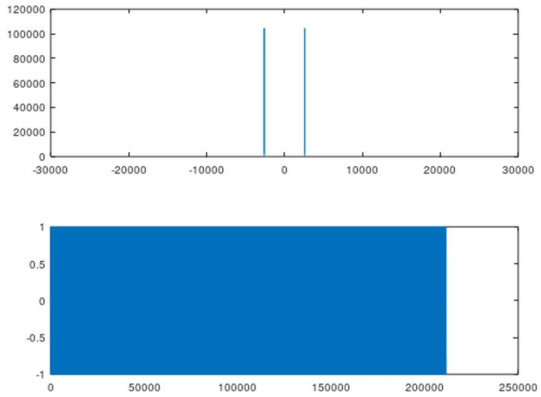
Solution:

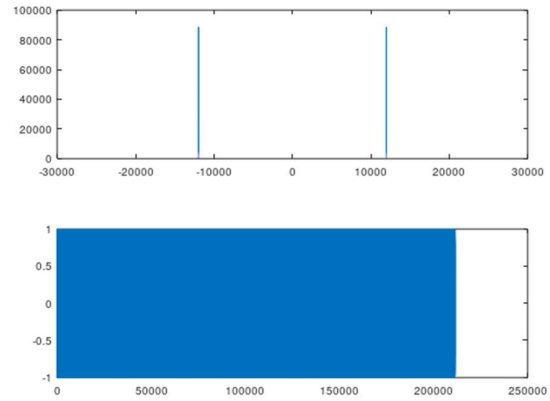
My code for the solution is as follows.

```
sfreq = 44100;
freq = randi([20,20000],1,3);
df = sfreq/length(linspace(0,4.7911,211289));
xi = -(sfreq/2):df:(sfreq/2)- df;
for i=1:3
mysound = sin(2*pi*freq(i)*linspace(0,4.7911,211289));
subplot(2,1,1)
plot(xi,abs(fftshift(fft(mysound))));
subplot(2,1,2);
plot(mysound);
sound(mysound, sfreq);
c = input("press enter to continue");
end
[f,sf]= audioread('D:\Eric\My CAPS homework\test.wav');
close;
ns = f + mysound';
subplot(2,2,1)
plot(f);
subplot(2,2,2);
plot(xi,abs(fftshift(fft(f))));
sound(f,sf);
c =input("This is the original sound, press enter to hear the noise induced version");
subplot(2,2,3)
plot(ns);
subplot(2,2,4);
plot(xi,abs(fftshift(fft(ns))));
sound(ns,sfreq);
c =input("This is the noise injected version, press enter to remove noise algorithmically");
close;
t = fftshift(fft(ns)).*((abs(xi) < 3000)');
subplot(2,2,1)
plot(ns);
subplot(2,2,2);
plot(xi,abs(fftshift(fft(ns))));
subplot(2,2,3);
plot(ifft(ifftshift(t)));
subplot(2,2,4);
plot(xi,abs(t));
sound(ifft(ifftshift(t),sfreq));
c =input("Press enter to try it with another sound");
[f2,sf2]= audioread('D:\Eric\My CAPS homework\CAPS_07_B9TB1707_sound.wav');
df2 = sf2/length(linspace(0,length(f2)/sf2,length(f2)));
xi2 = -(sf2/2):df2:(sf2/2)- df2;
noise2 = sin(2*pi*randi([20,20000])*linspace(0,length(f2)/sf2,length(f2)));
in2 = f2 + noise2';
subplot(2,2,1)
plot(f);
subplot(2,2,2);
plot(xi,abs(fftshift(fft(f))));
subplot(2,2,3)
plot(f2);
subplot(2,2,4);
plot(xi2,abs(fftshift(fft(f2))));
sound(f2,sf2);
c =input("This is the original sound, press enter to hear the noise induced version");
close;
subplot(2,2,1)
plot(f2);
subplot(2,2,2);
plot(xi2,abs(fftshift(fft(f2))));
subplot(2,2,3)
plot(in2);
subplot(2,2,4);
plot(xi2,abs(fftshift(fft(in2))));
sound(in2,sfreq);
c =input("This is the noise injected version, press enter to remove noise algorithmically");
close;
t2 = fftshift(fft(in2)).*((abs(xi2) < 3000)');
subplot(2,2,1)
plot(in2);
```

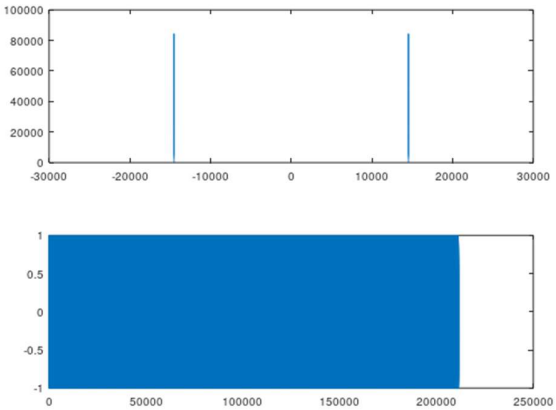
```
subplot(2,2,2);
plot(xi2,abs(fftshift(fft(in2))));
subplot(2,2,3);
plot(ifft(ifftshift(t2)));
subplot(2,2,4);
plot(xi2,abs(t2));
sound(ifft(ifftshift(t2),sf2));
```

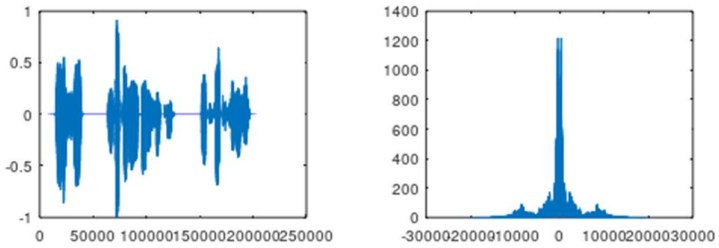
And my outputs are as follows:



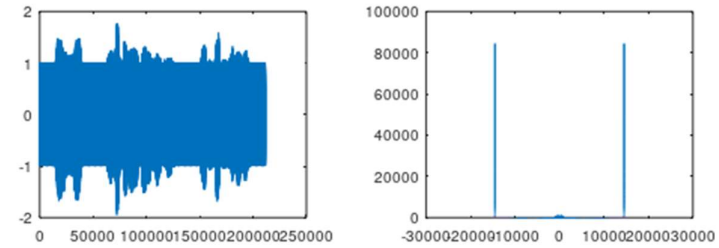


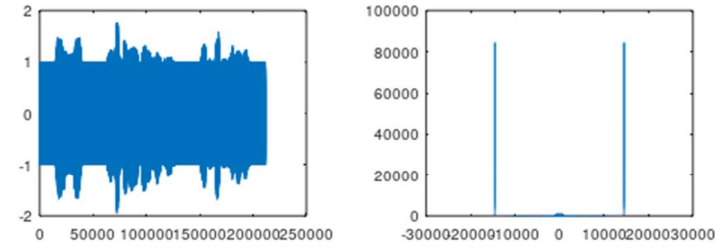
These are the plots of the sinusoidal sound waves



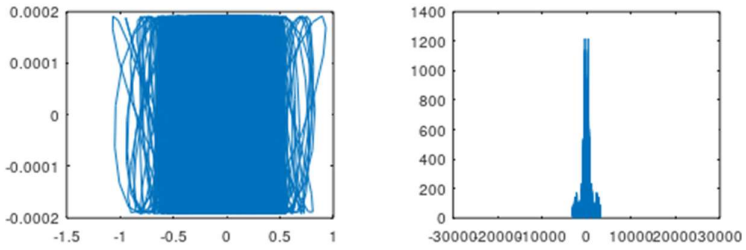


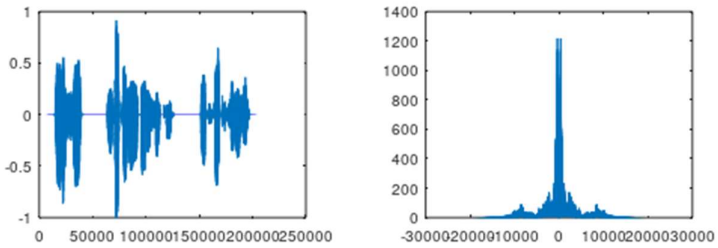
The First row is test.wav
And the second row shows the file with noise included.

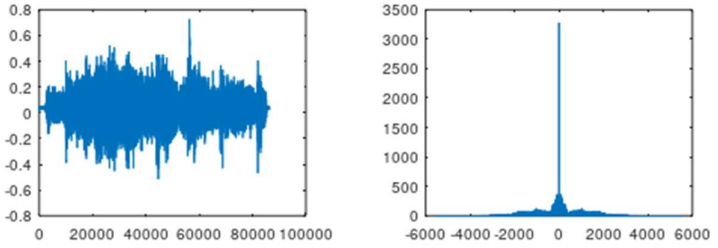




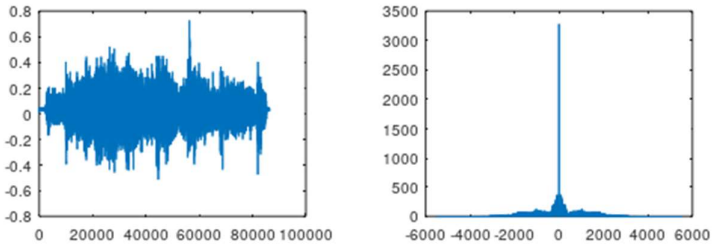
The first row shows the noisy version and the second row is de-noised version

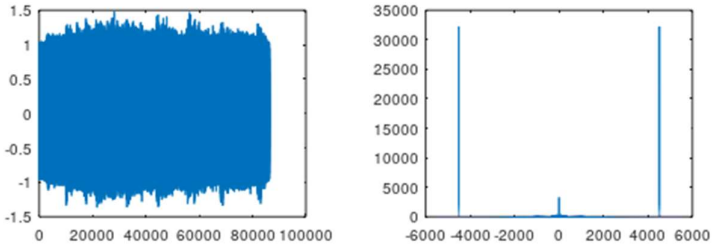




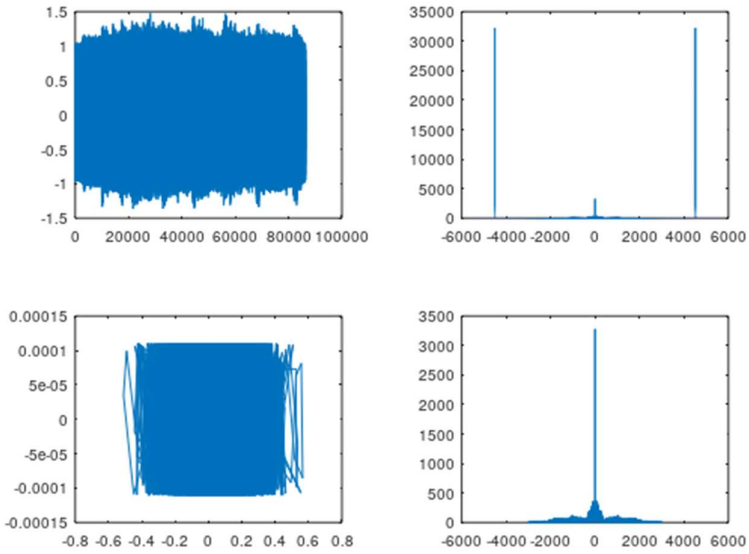


First row is test.wav
And the second row is another audio file





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