[CMPE 362 – Spring 2016]

[Homework 3]

[Boğaziçi University]

Enes Özipek
2012400162

In this homework, we are assigned to solve problems requiring operations in frequency domain. Therefore, I utilized fft mostly in this assignment.

Q1)

In this question, we are asked to combine 2 wav file and filter the mixed wav file and calculate SNR value. To do that I used the following algorithm:

Step 1: Read and create objects for 'Street.wav' and 'Mike.wav'.

Step 2: Sum those 2 wav files in time domain.

Step 3: Apply fft to combined wav file namely 'Street+mike'

Step 4: Filter mixed function by considering

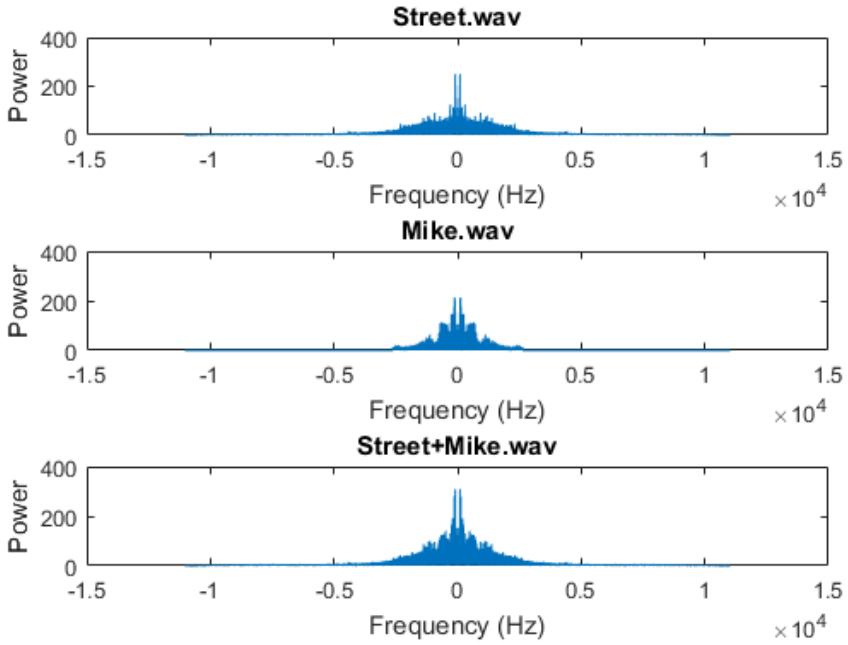
human voice band as 50-3000 Hz

Step 5: Apply ifft to filtered function to make it in time domain.

Step 6: Calculate SNR value for this signal.

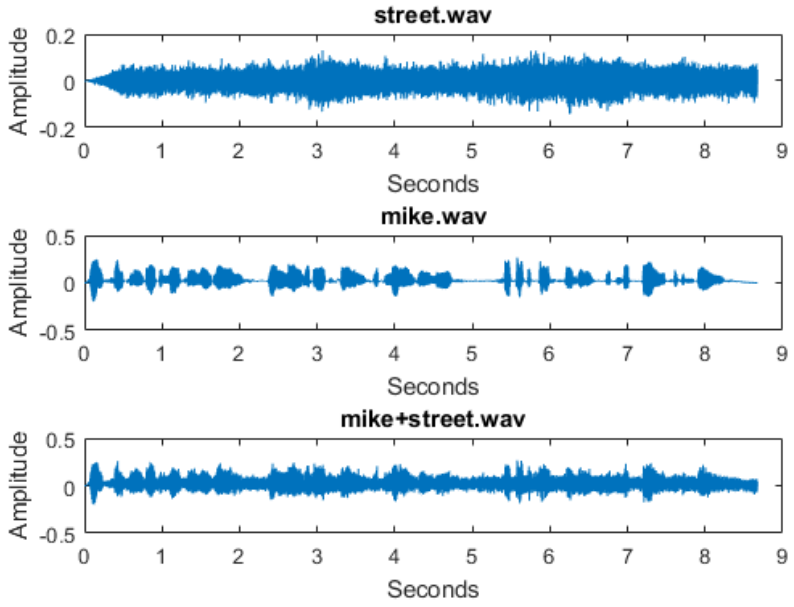
Global Task For Each Part: Plot the functions in desired domain as two sided by using fftshift. I'm going to explain in more detail.

1-) Frequency Domain Representation of Mike.wav, Street.wav, Mike+Street.wav



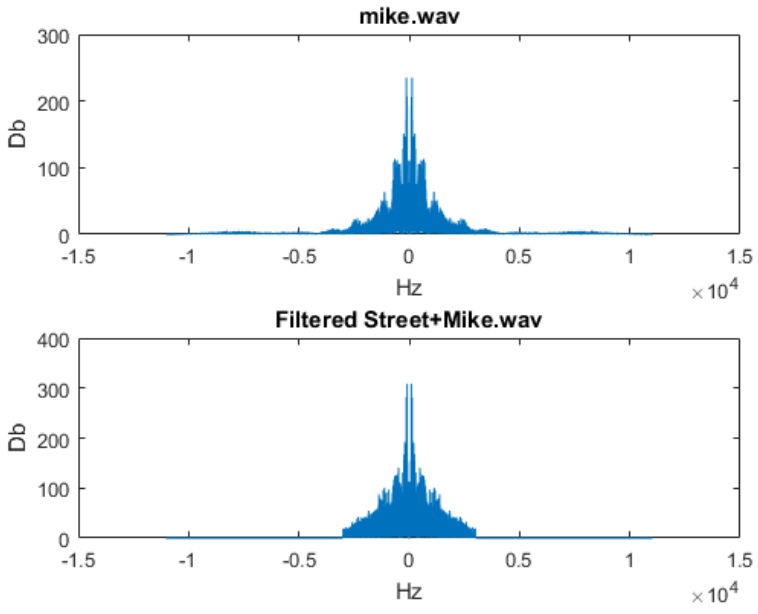
In this figure, I plot the functions written as title in frequency domain. Third plot is combined and non-filtered function. As we can see, there is an increase in power of mike signal. Because we are adding a noisy background.

2-) Time Domain Representation of Mike.wav, Street.wav, Mike+Street.wav



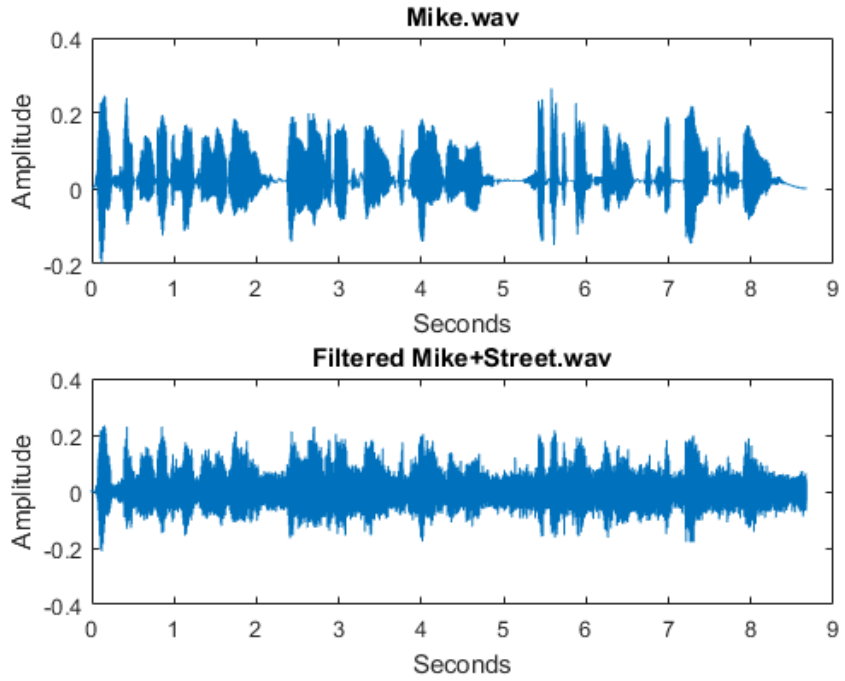
In this figure, I plot the signals in time domain. Again, by summing two functions (Street and Mike signals) in time domain we get combined function. Three functions have the same length in terms of duration.

3-) Frequency Domain Representation of Mike.wav, Filtered Mike+Street.wav



In this part, I plot the function of Mike and filtered Street+Mike function. To do that, I apply fft to the combined function and filter to that function. I set the human frequency band (50-3000 Hz) by observing from the internet. By setting every value which is out of this range to 0, I get and plot the filtered signal. By looking at the graph, we can see they are quite similar to each other.

4-) Time Domain Representation of Mike.wav, Filtered Mike+Street.wav



For previous figure, we have the filtered signal in frequency domain. To convert it to in time domain I used ifft function of Matlab. By looking at the function, we can see that there is a noisy background that was not completely cleared.

SNR VALUE CALCULATION

To calculate SNR I utilized the following algorithm:

```
signal = abs(y2);
noise_reduced_signal = abs(Y);
residual_noise = signal - noise_reduced_signal;
snr_after = mean( signal .^ 2 ) / mean( residual_noise .^ 2 );
snr_after_db = 10 * log10( snr_after );
fprintf('SNR value is %f \n',snr_after_db);
```

First of all, I take the pure Mike signal and filtered signal. Then, by applying above formula, I get the SNR value.

My SNR value is 0.999990. My interpretation for this result is that it is low since the filtered function still has a noisy background in it. Considering formula, we observe that the SNR value should go to infinity if we clear the noise perfectly. However, the technique I used is not efficient and practical. Therefore, the value is low.

Q2)

In this question we are asked to write Matlab code that differentiates the clap and snap sounds. My approach to solve this question is as follows:

Step 1: First read sound.
Step 2: Apply fft to this sound.
Step 3: Take abs of signal and then calculate mean of this signal.
Step 4: Set a threshold value not to take small values into consideration.
Step 5: Iterate over the signal and sum frequency values that are greater than value of threshold.
Step 6: Calculate the ratio of summation of frequencies to number of peak values. (i.e. mean of frequencies counted)

Step 7: If ratio/1000 is greater than 4 then it is snap sound.

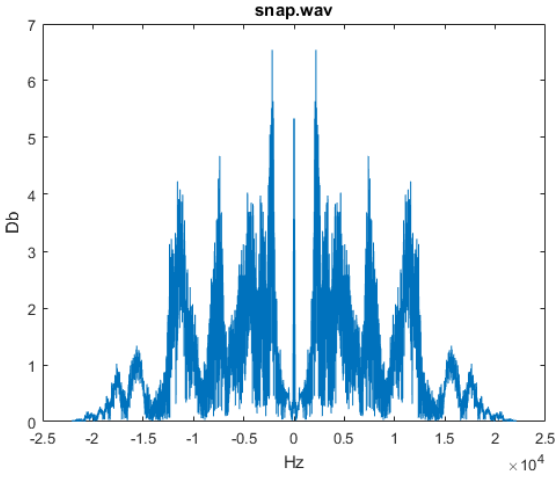
Step 8: Otherwise it is clap sound.

PS: I decided that the ratio value should be greater than 4 for a signal to be considered as snap sound after inspecting many sample wav files. And also I utilized the logic I used in previous homework.

Example:

What is the name of file (enter without .wav extension) snap

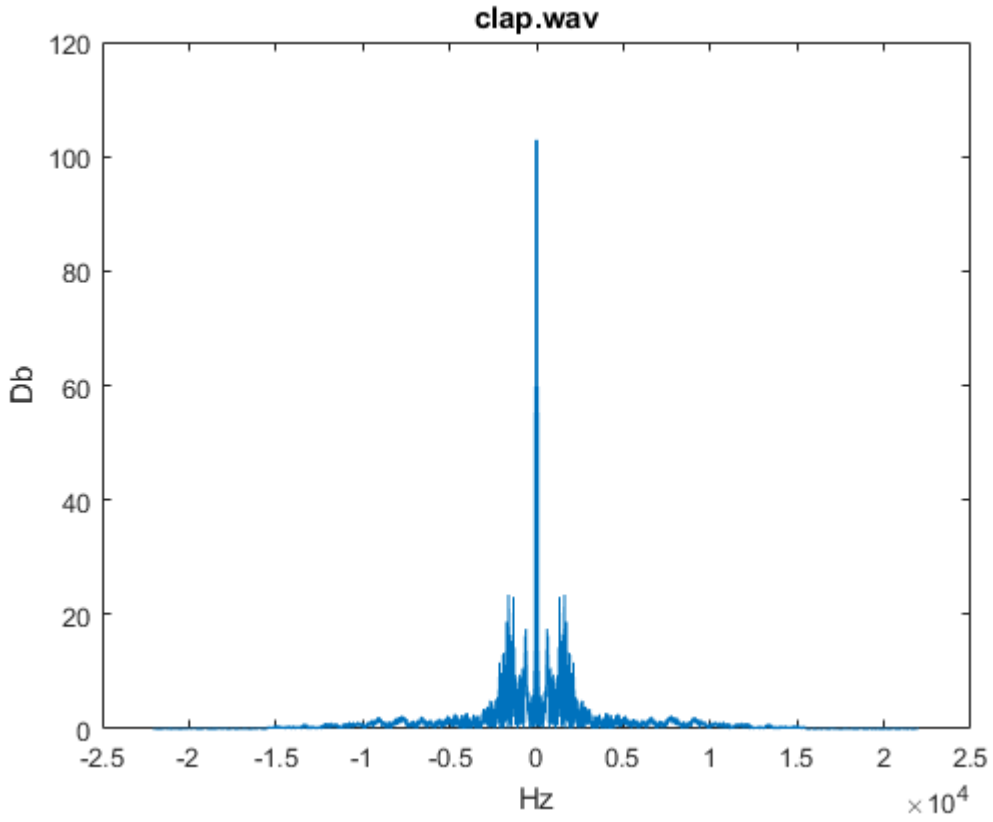
Snap sound detected



Example 2:

What is the name of file (enter without .wav extension) clap

Clap sound detected



Ratio for second example is: 2.207081933783999 (Clap)

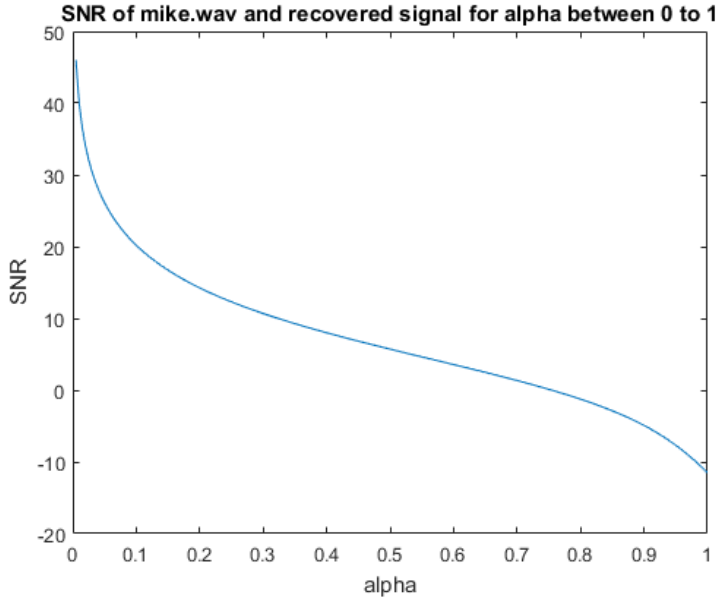
Ratio for first example is: 6.741563319929416 (Snap)

As we observe, the value for snap is greater than 4 while ratio for clap is less than 4.

Q3)

In this question, we are asked to write a program which uses 3 parameters namely N, K and alpha. By adjusting their values, we are supposed to plot pure Mike.wav and its corresponding recovered signal.

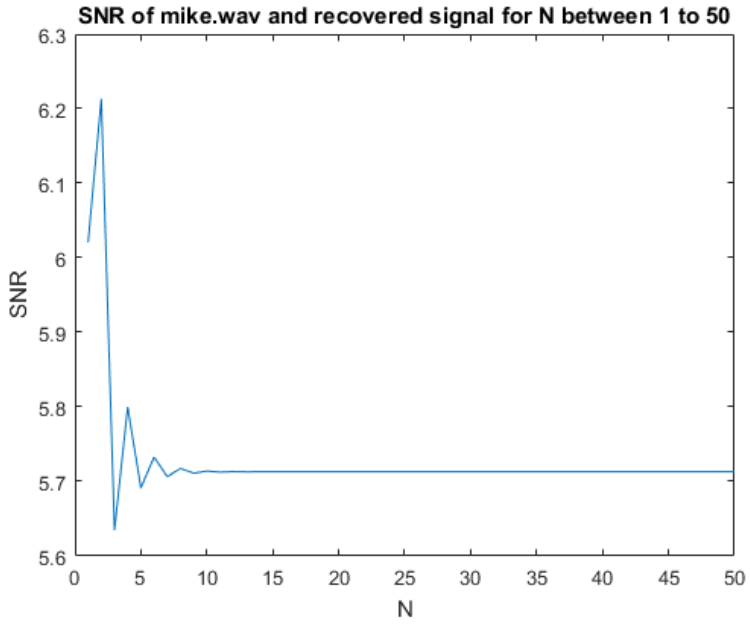
1-) Use constant N and K, change α from 0 to 1 and plot SNR of mike.wav and recovered signal.



K= 100 ms and N=20 while alpha is changing from 0 to 1.

As we see in the figure, when alpha increases, the value of SNR is decreasing. This shows that the coefficients we use in convolution affect the SNR adversely when they are increasing.

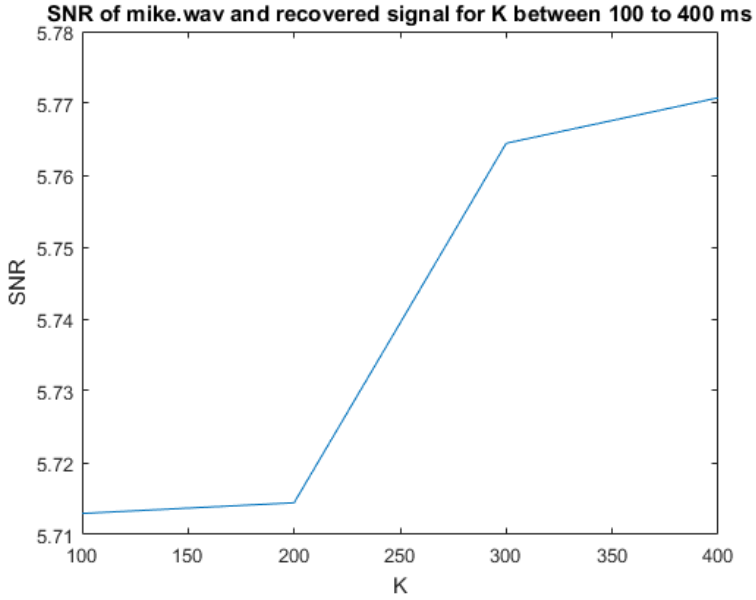
2-) Use constant α and K, change N from 1 to 50 and plot SNR of mike.wav and recovered signal.



K= 100 ms and alpha = 0.5 while N is changing from 1 to 50.

As we see in the figure, when N increases, the value of SNR increases and then decreases to some point. After that, it is stable. Since we are using N as the power of alpha like `(-alpha)^j` , after some value of N convolution coefficients will be very small.

3-) Use constant α and N, change K between 100,200,300,400 milliseconds and plot SNR of mike.wav and recovered signal



Alpha = 0.5 and N = 20 while K are changing between 100-400 ms

As we can observe from the figure, while everything is fixed and only delay time changes, the value of SNR won't be affected greatly. By incrementing K, actually we are increasing the number of coefficients used in convolution process. However, this increase won't be having huge impact on SNR value since its new values will be corresponding to last indexes of mike.wav. This makes not much effect.