THE AMERICAN UNIVERSITY IN CAIRO
الجامعة الأمريكية بالقاهرة

Project Report

By
Hadeel Mabrouk, 900-16-3213

To
Professor. Mahmoud Khalil
Digital Signal Processing (DSP)

Due date: May, 18th, 2020

For this project, I used some of the provided functions from the textbook to ease calculations and plotting, e.x. Ideal_lp, freqz_m, and Hr_type2. [**attached in the src directory**]
Vinay K. Ingle, John G. Proakis, *Digital Signal Processing Using Matlab. 4th edition*, Cengage Learning, 2016, ISBN: 978- 1305635128
https://www.mathworks.com/matlabcentral/fileexchange/2189-digital-signal-processing-using-matlab

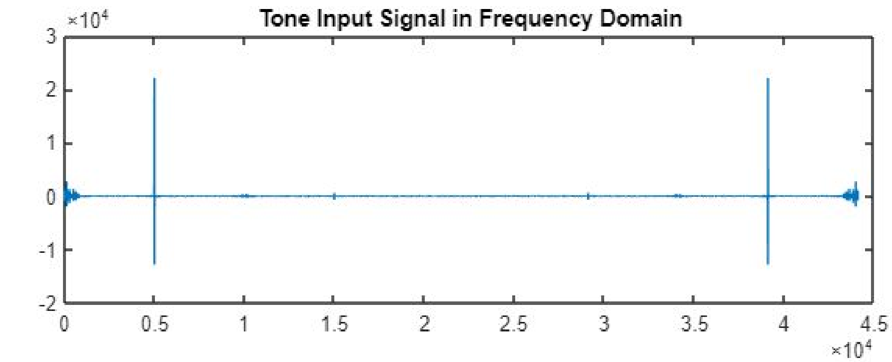
Question 3:

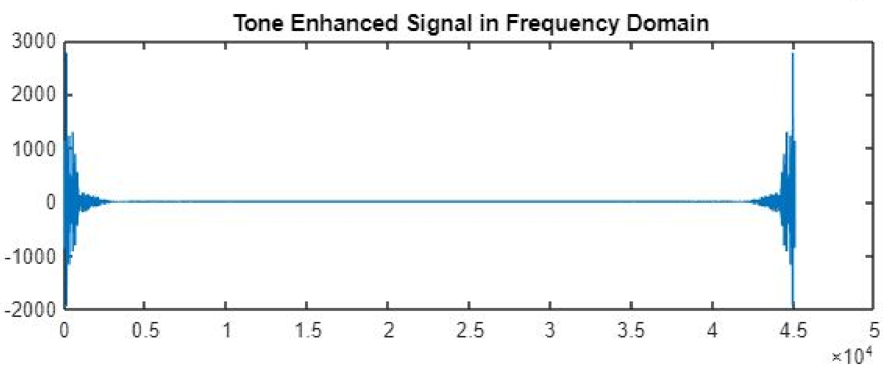
- **Tone Cancellation**
    - For this part, as we can notice from the input signal frequency plot, the tone has a unique very high frequency, so I used a simple low-pass filter to eliminate the tone noise as well as some background noises.
    - Also, after passing the signal through the filter, I tried to enhance the output signal a bit but enhancing the sample rate of the output signal.
    - **Code:**

```
%Question 3 - Tone Cancellation

%CODE
clc
clear
close all

[signalRead,sampleRate] = audioread("ILoveDSPtone.wav");
signalRead = signalRead';
[~,signalReadL] = size(signalRead);

%applying the low pass filter
w = lowpass(signalRead,900,sampleRate);
[~,outputLength] = size(w);
outputRate = (sampleRate*1.022);

%plots
%before enhancement
subplot(2,1,1);
f = sampleRate*(0:(signalReadL-1))/signalReadL;
plot(f,real(fft(signalRead)));
title("Tone Input Signal in Frequency Domain");
%after enhancement
subplot(2,1,2);
f = outputRate*(0:(outputLength-1))/outputLength;
plot(f,real(fft(w)));
title("Tone Enhanced Signal in Frequency Domain");

%saving the enhanced signal
```

```
sound(w,outputRate);
audiowrite("ToneOut.wav",w,outputRate);
```

- **Plots:**



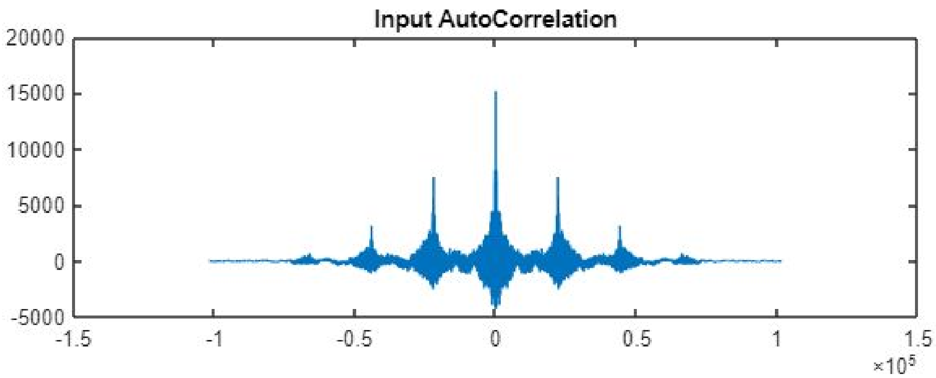


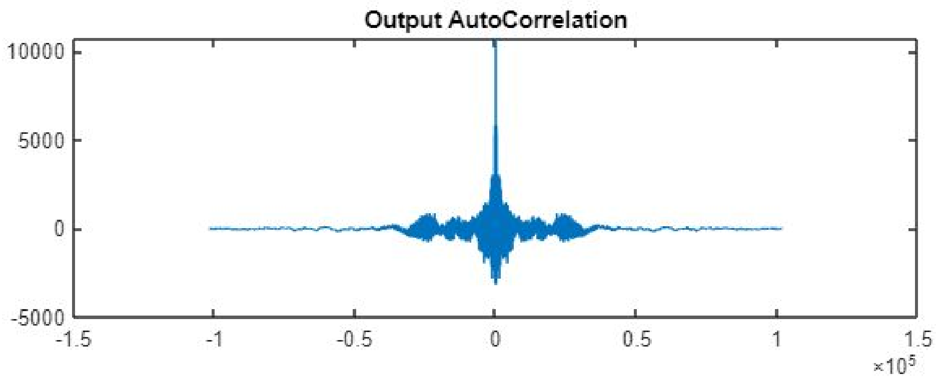
- **Echo Cancellation**
  - For the echo cancellation, I used the method described in the textbook *Digital Signal Processing Using MATLAB, 3rd ed.,* page 19, for echo removal using the command "filter(1,b,x)", where x is the input signal, and b is filter parameters.
  - For the filter parameters, we need to know two main features about the echo:
    - the amount of delay in samples *D*
    - the echo amplitude α , such that $0 < \alpha < 1$
  - To learn about those parameters from the input signal, I plotted the autocorrelation function that we used in Assignment 2, where the x coordinate for the peaks other than x = 0, represent the amount of delay *D*, and the ratio of the change in the amplitude of those peaks can give an estimation of the echo amplitude α.
  - For my filter, I designed it on three stages:
    - The first stage eliminates the first peak of the echo.

        - The second stage eliminates the second peak of the echo, by plotting the autocorrelation for the output signal from stage 1 and repeating the previous steps.
        - The final stage is a simple small low pass filter to cancel some remaining noise in the background.
    - From the plots, we can notice the big enhancement whether for the autocorrelation function, or in the frequency domain.
    - **Code:**

```
%Question 3 - Tone Cancellation

%CODE
clc
clear
close all

[signalRead,sampleRate] = audioread("ILoveDSPtone.wav");
signalRead = signalRead';
[~,signalReadL] = size(signalRead);

%applying the low pass filter
w = lowpass(signalRead,900,sampleRate);
[~,outputLength] = size(w);
outputRate = (sampleRate*1.022);

%plots
%before enhancement
subplot(2,1,1);
f = sampleRate*(0:(signalReadL-1))/signalReadL;
plot(f,real(fft(signalRead)));
title("Tone Input Signal in Frequency Domain");
%after enhancement
subplot(2,1,2);
f = outputRate*(0:(outputLength-1))/outputLength;
plot(f,real(fft(w)));
title("Tone Enhanced Signal in Frequency Domain");

%saving the enhanced signal
sound(w,outputRate);
audiowrite("ToneOut.wav",w,outputRate);
```

- **Plots:**
  - **Autocorrelation**





  - **Frequency Domain**