Signals and Systems

(EE-231)

DE-43 Mechatronics

Syndicate – A

PROJECT REPORT

Group Number 18

Date: 23-12-2022

Title: MATLAB Signal Sampling and Filtering Designing

Names of Group Members

1. NC Syed Muhammad Daniyal Gillani
   Reg # 375785
   Email: s.daniyalgellani@gmail.com
2. NC Muhammad Abdullah Khalid
   Reg # 367534
   Email: abdullah_boobak@hotmail.com

Submitted to: Prof Dr. Sir Zaki-uddin

TABLE OF CONTENTS

ABSTRACT:

The project involves the study of various signals in continuous and discrete form and studying the effects of various operations on signals. These operations include down sampling, amplitude scaling, filtering various frequency components of signals, zero padding signals etc. The signals under consideration are then output in frequency and time domain and the graphs and plots are analyzed. The signal used for this analysis was a sound signal and the analysis was done in Matrix Laboratory **MATLAB.** The analysis showed various outputs according to the operation being performed such as signal quality dropping, frequency components being filtered out, sound intensity and pitch improving and degrading.

INTRODUCTION:

MATLAB stands for Matrix laboratory. It is a programming language and numeric computing environment. It is used for matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and many more things. The purpose of this project was to learn about different manipulation techniques that can be applied to signals in MATLAB. In this report we focus on the down-sampling of functions, applying zero padding, designing an audio equalizer, and manipulation of signals in the frequency domain and its impact on the time-domain signal. In the first task down-sampling and zero padding was applied to given functions. In the second task, the audio signal was imported and manipulated in the software. In the third task, the filters were designed.

EXPERIMENT:

MATERIALS USED:

- **Software:** MATLAB ver. R2022b
- **Audio file**: (See reference)
    - **File format:** .wav format
    - **File size:** 1.69mb
    - **Audio duration:** 9 seconds.

METHODOLOGY:

The following Commands were used in MATLAB for the manipulation of signals:

Sr.	Function	Sr.	Function
1	fft	**6**	Audioread
2	Fftshift	**7**	Audiowrite
3	Plot	**8**	Abs
4	Subplot	**9**	tempname
5	stem	**10**	zeros

FUNCTION DETAILS:

1. **fft:**

   A fast Fourier transform (FFT) is a highly optimized implementation of the discrete Fourier transform (DFT), which converts discrete signals from the time domain to the frequency domain. FFT computations provide information about the frequency content, phase, and other properties of the signal.

   Syntax:

   ```
   Y = fft (X)
   Y = fft (X, n)
   Y = fft (X, n, dim)
   ```

Here 'X' is a vector, 'n' is the trailing of zeros to length 'n' and 'dim' is the dimension.

2. **fftshift:**

   fftshift rearranges a Fourier transform X by shifting the zero-frequency component to the centre of the array.

   Syntax:

   ```
   Y = fftshift (X)
   Y = fftshift (X, dim)
   ```

Here X is the 'vector' and 'dim' is the dimension.

3. **Subplot:**

subplot(m, n, p) divides the current figure into an m-by-n grid and creates axes in the position specified by p. MATLAB numbers subplot positions by row. The first subplot is the first column of the first row, the second subplot is the second column of the first row, and so on. If axes exist in the specified position, then this command makes the axes the current axes.

Syntax:

```
subplot (m, n, p)
```

Here 'm' and 'n' are grid parameters and 'p' is the position.

4. **Plot:**

Plot (X, Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

Syntax:

```
plot (X, Y)
```

5. Zeros:

zeros return the scalar 0.

Syntax:

zeros(n)

Here 'n' is the number of zeros that could be added to the function.

6. Stem:

Stem(Y) plots the data sequence, Y, as stems that extend from a baseline along the X-axis. The data values are indicated by circles terminating each stem.

Syntax:

stem (X, Y)

Here X and Y are vectors.

7. Audioread:

Audioread (filename) reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

Syntax:

[y, Fs] = audioread (filename, samples)

Here 'filename' is the name and address of the file and 'samples' is the numbers of samples of the function.

8. Abs:

Abs (X) returns the absolute value of each element in array X. If X is complex, abs (X) return the complex magnitude.

Syntax:

abs (X)

Here X is an array.

9. Sound:

Sound (y) sends audio signal y to the speaker at the default sample rate of 8192 Hz.

Syntax:

sound(y)

Here y is the function that is to be played.

10. Audiowrite:

Audiowrite (filename, y, Fs) writes a matrix of audio data, y, with sample rate Fs to a file called filename. The filename input also specifies the output file format. The output data type depends on the output file format and the data type of the audio data, y.

Syntax:

audiowrite (filename, y, Fs)

Here ‘filename’ is the name and address of the file, ‘y’ is the matrix and ‘Fs’ is the sample rate.

ADD-ONS USED:

- **Signal Analyzer**
- **Filter Designer**

PROCEDURE:

Task: 1

Signal given:

$$x_1[n] = e^{\frac{j2\pi n(10\times(K+1))}{100}} + e^{\frac{j2\pi n 33}{100}}$$

$$x_2[n] = \cos\left(\frac{2\pi n(10\times(K+1))}{100}\right) + \frac{1}{2}\cos\left(\frac{2\pi n(10\times(L+1))}{100}\right)$$

Figure1: Task 1 Signals

Values taken:

L=3 and K=18

Sub Tasks:

Task i:

Plot the length-N DFT magnitude of signals x1[n] and x2[n] with respect to frequency fr, where $fr = r/N$ and N is the number of samples. You can choose 10 samples for the above case and display the signal in the range ($-1/2 \leq fr < 1/2$).

Task ii:

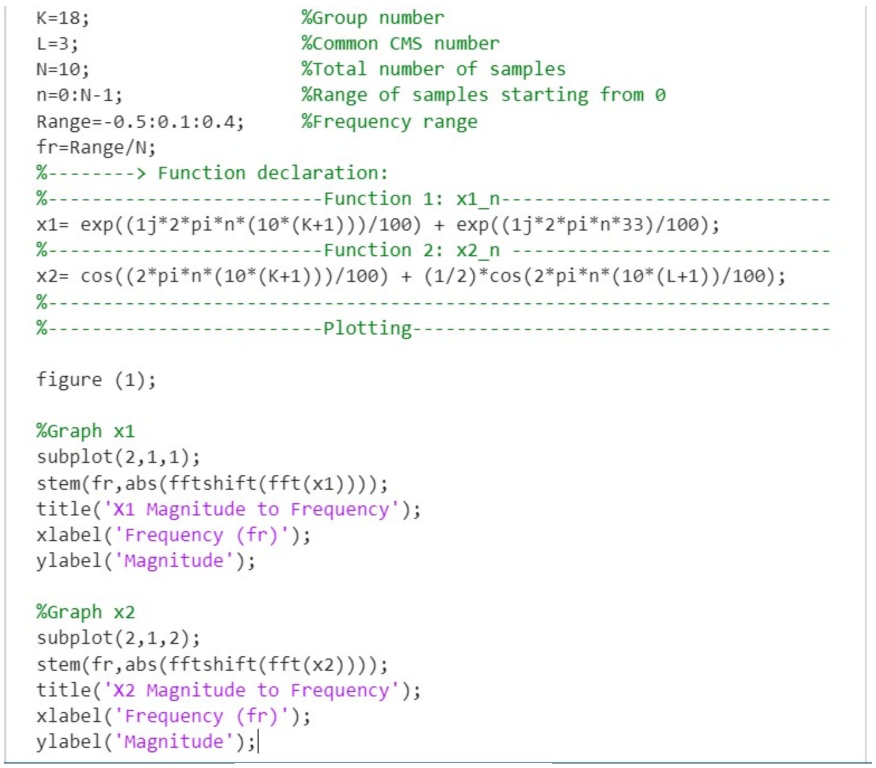
Zero-pad the signals x1[n] and x2[n] with 490 zeros, then compute and plot the length-500 DFT of both signals.

Task iii:

Repeat part (1) by taking 100 samples from each signal

Procedure (i):

- Open **MATLAB**, create a new script by pressing **Shift+N** on keyboard and write the following code in **Editor window.**



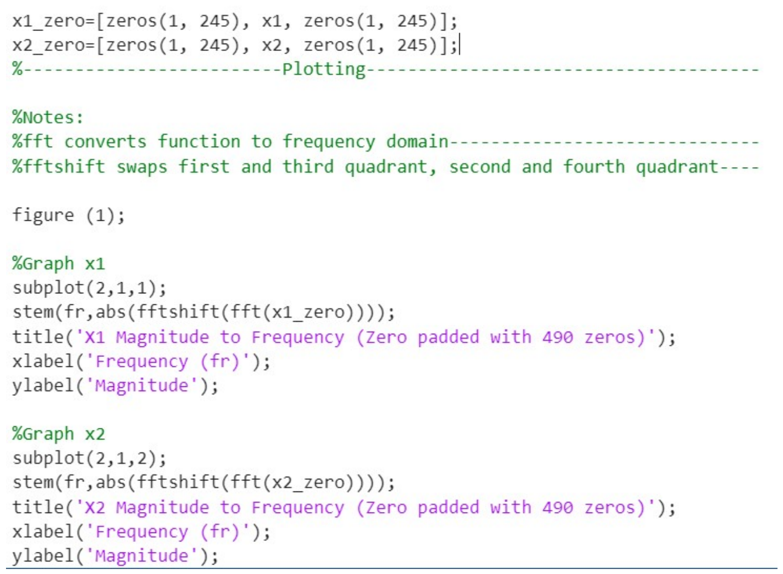
```
K=18;                    %Group number
L=3;                     %Common CMS number
N=10;                    %Total number of samples
n=0:N-1;                 %Range of samples starting from 0
Range=-0.5:0.1:0.4;      %Frequency range
fr=Range/N;
%--------> Function declaration:
%------------------------Function 1: x1_n-----------------------------
x1= exp((1j*2*pi*n*(10*(K+1)))/100) + exp((1j*2*pi*n*33)/100);
%------------------------Function 2: x2_n ----------------------------
x2= cos((2*pi*n*(10*(K+1)))/100) + (1/2)*cos(2*pi*n*(10*(L+1))/100);
%--------------------------------------------------------------------
%------------------------Plotting------------------------------------

figure (1);

%Graph x1
subplot(2,1,1);
stem(fr,abs(fftshift(fft(x1))));
title('X1 Magnitude to Frequency');
xlabel('Frequency (fr)');
ylabel('Magnitude');

%Graph x2
subplot(2,1,2);
stem(fr,abs(fftshift(fft(x2))));
title('X2 Magnitude to Frequency');
xlabel('Frequency (fr)');
ylabel('Magnitude');
```

Figure 2: Task 1 basic code MATLAB.

- From the **Editor tab** click on **Run** and analyze the graph.

Procedure ii:

- Include the following lines of code (figure: 3) in the same above code and **Run** it.



```
x1_zero=[zeros(1, 245), x1, zeros(1, 245)];
x2_zero=[zeros(1, 245), x2, zeros(1, 245)];
%------------------------Plotting-------------------------------------

%Notes:
%fft converts function to frequency domain-----------------------------
%fftshift swaps first and third quadrant, second and fourth quadrant----

figure (1);

%Graph x1
subplot(2,1,1);
stem(fr,abs(fftshift(fft(x1_zero))));
title('X1 Magnitude to Frequency (Zero padded with 490 zeros)');
xlabel('Frequency (fr)');
ylabel('Magnitude');

%Graph x2
subplot(2,1,2);
stem(fr,abs(fftshift(fft(x2_zero))));
title('X2 Magnitude to Frequency (Zero padded with 490 zeros)');
xlabel('Frequency (fr)');
ylabel('Magnitude');
```

Figure 3: Code for zero padding.

- Analyze the graphs and note down the results.

Procedure iii:

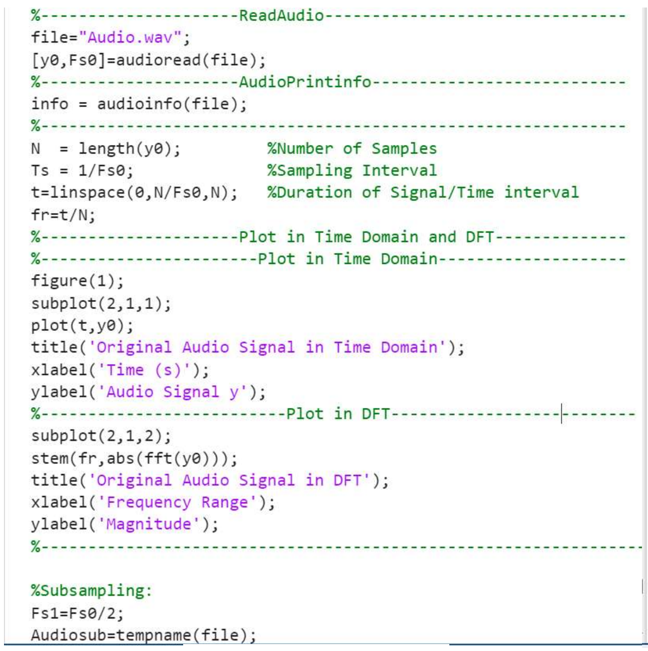
- Use the same code as used in procedure 1.
- Change the value of N to 100 and click on **Run** and observe the graphs.

Task: 2

1) Insert an audio signal and perform the following operations:
2) Find the number of samples (N0), the duration of the signal (T0), and the sampling interval (T). Plot signal y with respect to the time.
3) Compute and plot the DFT of signal y.
4) Generate the subsampled signal y1 from signal y by subsampling with rate 2. Find the number of samples (N0), the duration of the signal (T0), and the sampling interval (T).
5) Plot signal y1 with respect to time.
6) Compute and plot the DFT of y1.
7) Change the sampling to 5 and observe the output.

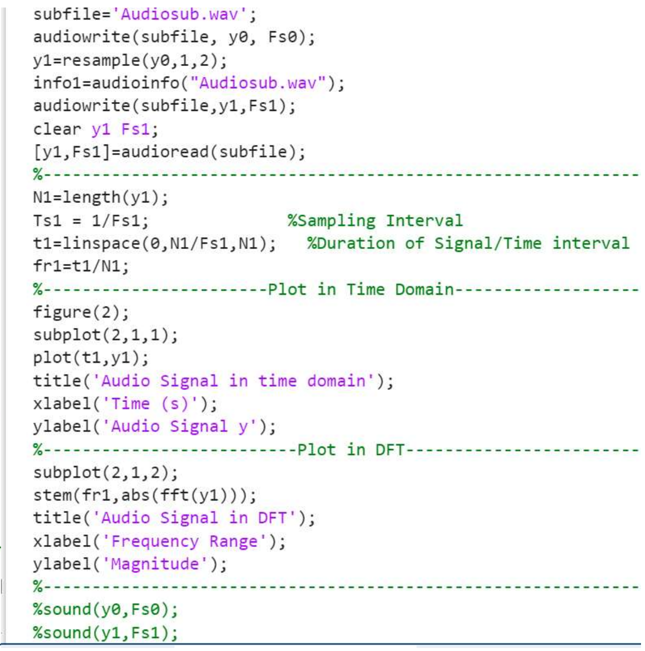
Procedure:

- Type the following code in MATLAB (see fig).
- **Run** the code and observe the output.
- To subsample the signal by a rate of 5. Simply change the values in line 28 and line 32 from 2 to 5.
- To find information about the signal use the audioinfo() function.



```
%---------------------ReadAudio-------------------------------
file="Audio.wav";
[y0,Fs0]=audioread(file);
%--------------------AudioPrintinfo---------------------------
info = audioinfo(file);
%-------------------------------------------------------------
N  = length(y0);         %Number of Samples
Ts = 1/Fs0;              %Sampling Interval
t=linspace(0,N/Fs0,N);   %Duration of Signal/Time interval
fr=t/N;
%--------------------Plot in Time Domain and DFT-------------
%----------------------Plot in Time Domain--------------------
figure(1);
subplot(2,1,1);
plot(t,y0);
title('Original Audio Signal in Time Domain');
xlabel('Time (s)');
ylabel('Audio Signal y');
%-------------------------Plot in DFT-------------------------
subplot(2,1,2);
stem(fr,abs(fft(y0)));
title('Original Audio Signal in DFT');
xlabel('Frequency Range');
ylabel('Magnitude');
%--------------------------------------------------------------

%Subsampling:
Fs1=Fs0/2;
Audiosub=tempname(file);
```

Figure 4: Task 2Code part 1.



```
subfile='Audiosub.wav';
audiowrite(subfile, y0, Fs0);
y1=resample(y0,1,2);
info1=audioinfo("Audiosub.wav");
audiowrite(subfile,y1,Fs1);
clear y1 Fs1;
[y1,Fs1]=audioread(subfile);
%-------------------------------------------------------------
N1=length(y1);
Ts1 = 1/Fs1;              %Sampling Interval
t1=linspace(0,N1/Fs1,N1);   %Duration of Signal/Time interval
fr1=t1/N1;
%----------------------Plot in Time Domain-------------------
figure(2);
subplot(2,1,1);
plot(t1,y1);
title('Audio Signal in time domain');
xlabel('Time (s)');
ylabel('Audio Signal y');
%-------------------------Plot in DFT-----------------------
subplot(2,1,2);
stem(fr1,abs(fft(y1)));
title('Audio Signal in DFT');
xlabel('Frequency Range');
ylabel('Magnitude');
%-------------------------------------------------------------
%sound(y0,Fs0);
%sound(y1,Fs1);
```

Figure 5: Task 2 Code part 2.

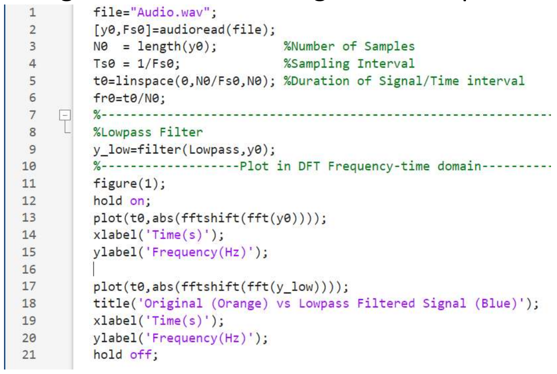
- To play the sound, use sound() function.

Task: 3

1) Read an audio file in MATLAB and apply the following operations:
2) Create a rect filter that only passes frequencies less than 2000. Apply this filter to the signal and plot the response in the time and frequency domain.
3) Use command play and listen to how the audio signal has changed.
4) Design a filter that cuts the bass frequencies (e.g., frequencies between 16 and 256 Hz). Plot the frequency spectrum and listen to the sound.
5) Design a filter that amplifies treble frequencies (e.g., frequencies between 2048 and 16384 Hz at the higher end of human hearing.) by 25%. Plot the frequency spectrum and listen to the sound.

Procedure:

- Write the following code as shown in figure for lowpass filter:



```
file="Audio.wav";
[y0,Fs0]=audioread(file);
N0  = length(y0);         %Number of Samples
Ts0 = 1/Fs0;              %Sampling Interval
t0=linspace(0,N0/Fs0,N0); %Duration of Signal/Time interval
fr0=t0/N0;
%-------------------------------------------------------------
%Lowpass Filter
y_low=filter(Lowpass,y0);
%------------------Plot in DFT Frequency-time domain---------
figure(1);
hold on;
plot(t0,abs(fftshift(fft(y0))));
xlabel('Time(s)');
ylabel('Frequency(Hz)');

plot(t0,abs(fftshift(fft(y_low))));
title('Original (Orange) vs Lowpass Filtered Signal (Blue)');
xlabel('Time(s)');
ylabel('Frequency(Hz)');
hold off;
```

Figure 6: Task 2: Lowpass filter Code

- From the **APPS** tab click on **Filter Design**

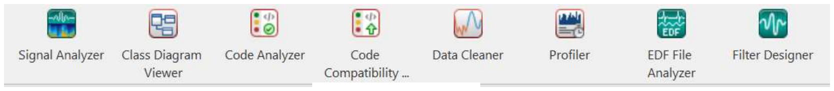
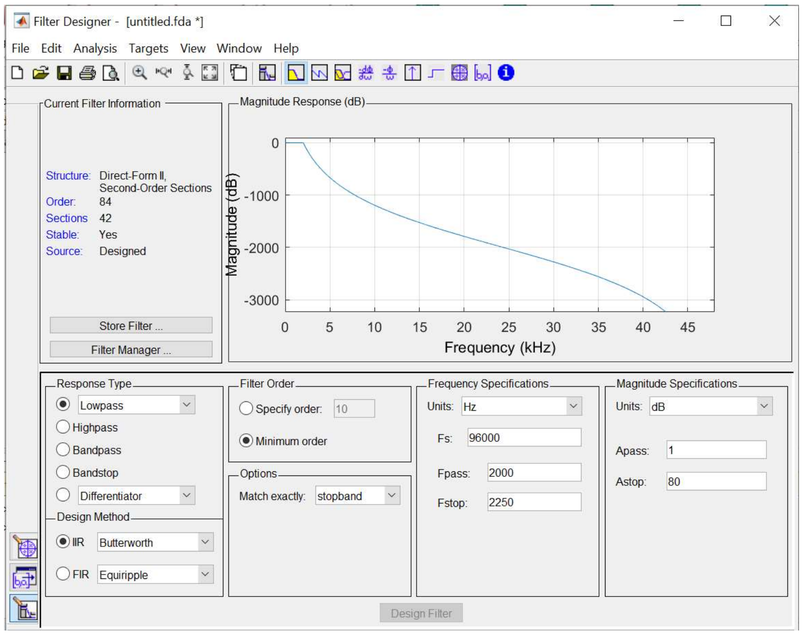


Figure 7: APPS tab

- Apply the following settings.

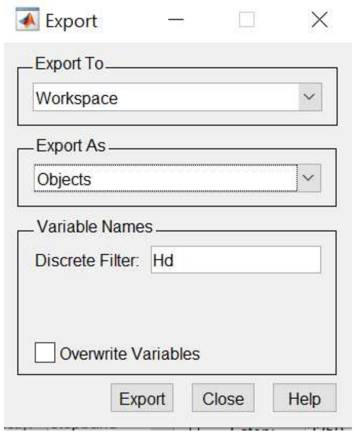

Filter Designer - [untitled.fda *]
File Edit Analysis Targets View Window Help
Current Filter Information
Magnitude Response (dB)
Magnitude (dB)
Frequency (kHz)
Store Filter ...
Filter Manager ...
Design Filter

Structure: Direct-Form II, Second-Order Sections
Order: 84
Sections: 42
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator
Design Method: [✓ IIR] Butterworth; FIR Equiripple
Filter Order: Specify order: 10; [✓ Minimum order]
Options: Match exactly: stopband
Frequency Specifications: Units: Hz; Fs: 96000; Fpass: 2000; Fstop: 2250
Magnitude Specifications: Units: dB; Apass: 1; Astop: 80

Figure 8: Filter Design tab, setting for Lowpass filter

- Go to file-> click on Export and write the name of the filter variable as per required.

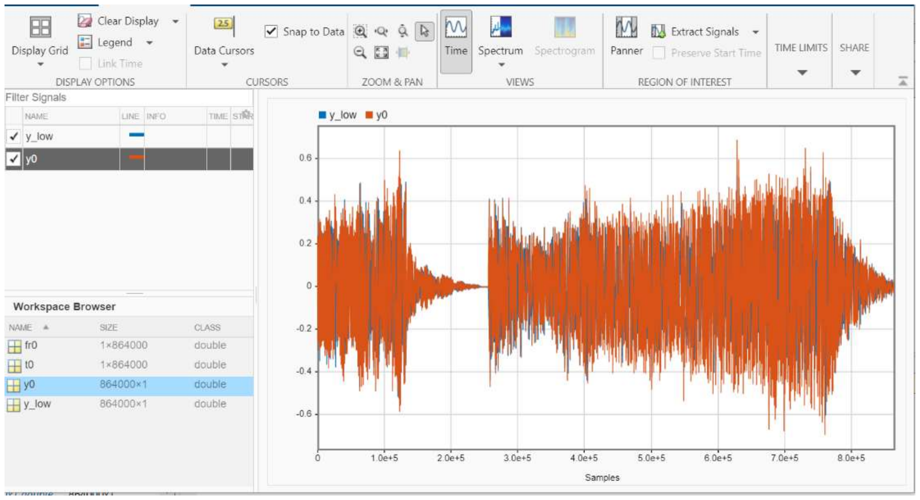


Export To: Workspace
Export As: Objects
Variable Names
Discrete Filter: Hd
Overwrite Variables

Figure 9: Export tab

- Click on Export, the variable will come under **Workspace window.**

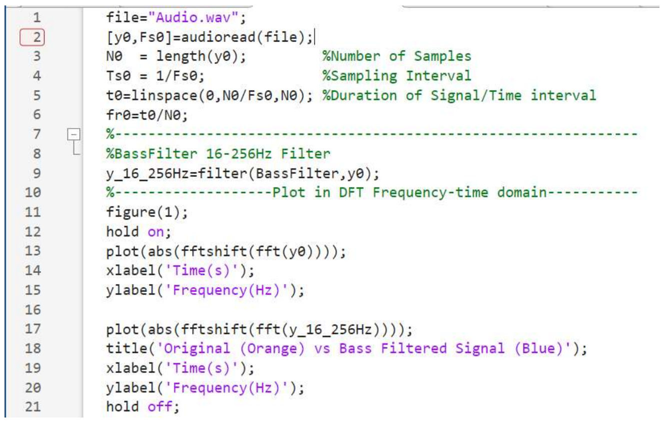
- To use the feature, use the following command:
  NewSignal_variable = **filter(**Name_of_Exported_variable,name_of_original_signal**);**
- From the **APPS** tab click on **Signal Analyzer.**
- In the **Signal Analyzer** window. Drag and drop the functions on the graph to create a time domain graph of the signals as shown:



NAME	SIZE	CLASS
fr0	1×864000	double
t0	1×864000	double
y0	864000×1	double
y_low	864000×1	double

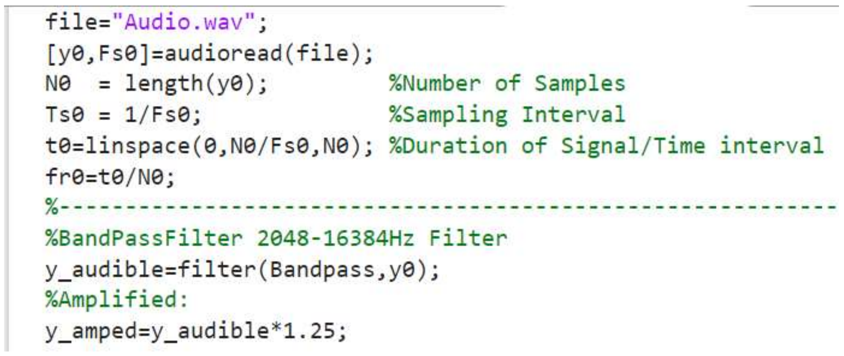
Figure 10: Signal Analyzer Lowpass filter graph

- Click on **Spectrum** from **Views** tab to add a **Frequency Spectrum** graph.
- Repeat the process for other sub tasks, by creating a bandpass, bandstop filter as per requirement.



```
file="Audio.wav";
[y0,Fs0]=audioread(file);
N0  = length(y0);         %Number of Samples
Ts0 = 1/Fs0;              %Sampling Interval
t0=linspace(0,N0/Fs0,N0); %Duration of Signal/Time interval
fr0=t0/N0;
%-------------------------------------------------------------
%BassFilter 16-256Hz Filter
y_16_256Hz=filter(BassFilter,y0);
%------------------Plot in DFT Frequency-time domain----------
figure(1);
hold on;
plot(abs(fftshift(fft(y0))));
xlabel('Time(s)');
ylabel('Frequency(Hz)');

plot(abs(fftshift(fft(y_16_256Hz))));
title('Original (Orange) vs Bass Filtered Signal (Blue)');
xlabel('Time(s)');
ylabel('Frequency(Hz)');
hold off;
```

Figure 11: Bass filter code.



```
file="Audio.wav";
[y0,Fs0]=audioread(file);
N0  = length(y0);         %Number of Samples
Ts0 = 1/Fs0;              %Sampling Interval
t0=linspace(0,N0/Fs0,N0); %Duration of Signal/Time interval
fr0=t0/N0;
%-------------------------------------------------------------
%BandPassFilter 2048-16384Hz Filter
y_audible=filter(Bandpass,y0);
%Amplified:
y_amped=y_audible*1.25;
```

Figure 12: Code for Bandpass filter.

RESULTS:

TASK1:

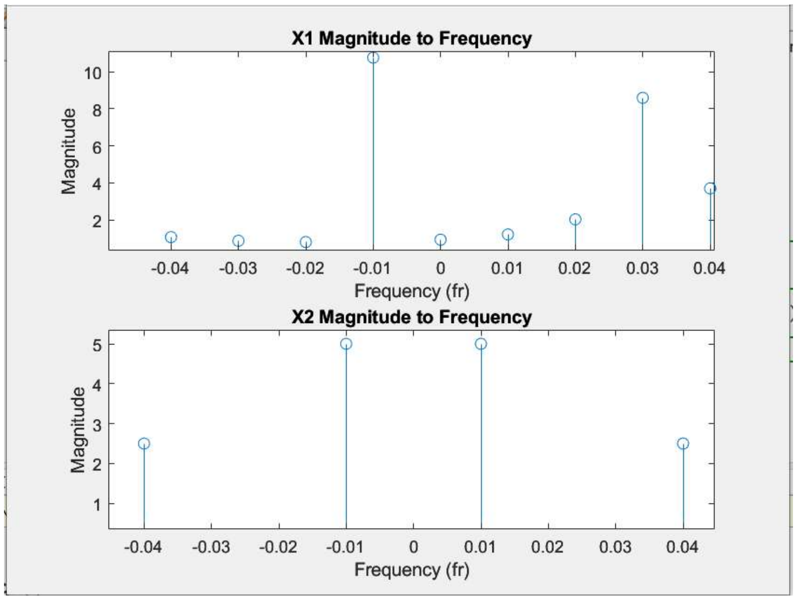


Figure 13: Plot of signals X1 and X2 for 10 samples.

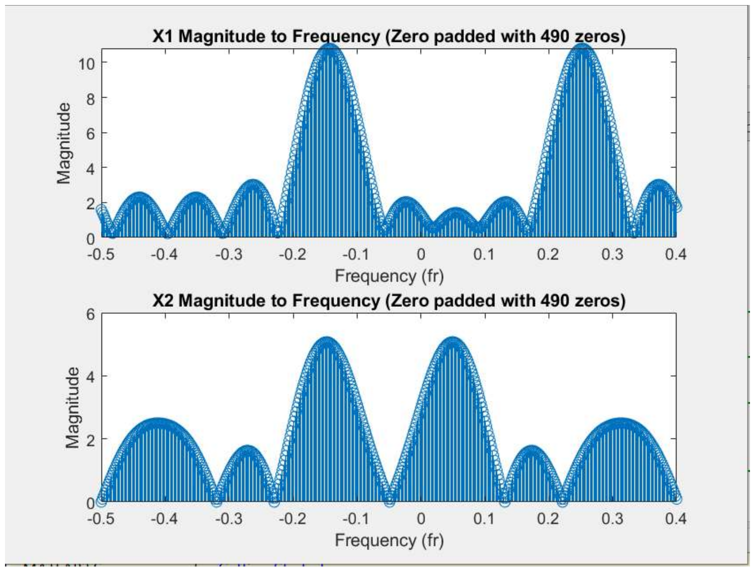


Figure 14: Zero padded signals (Task 1: part (2))

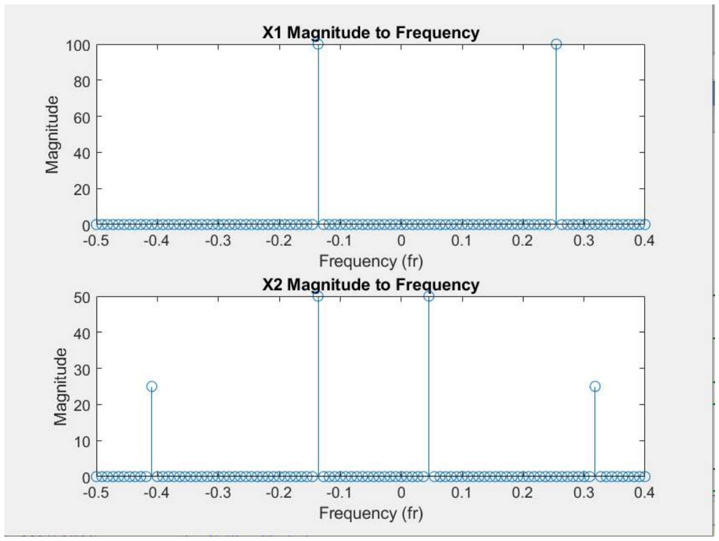


Figure 15: Plot of 100 samples (Task 1 (3))

Q1:

Plot the length-N DFT magnitude of signals x1[n] and x2[n] with respect to frequency fr, where $fr = r/N$ and N is the number of samples. You can choose 10 samples for the above case and display the signal in the range ($-1/2 \leq fr < 1/2$).

i. From your plot, can you explain which signal has a symmetric spectrum and why?
   - It is visible that X2 is symmetric as it appears to be an even function that is, it is uniform about the y-axis or its mirror images about the y-axis are same.

ii. Is it possible to distinguish both frequency components in the plots? Explain the reason.
   - Yes, it is possible to distinguish both frequency components in the graph. This can be done by looking at the frequency components of each as the graphs are in frequency domain.

iii. Explain why you see other frequency components in the plot for the DFT (x1[n]).
   - Other frequency components are visible in X1's graph as it spans a complex plane when absolute is not taken.

Q2:

Zero-pad the signals x1[n] and x2[n] with 490 zeros, then compute and plot the length-500

DFT of both signals. Check if there is any improvement.

- Zero padding has overall improved the signal by making it more stable. However, there is a chance that these zero padded signals distort the signal by producing unwanted noise.

Q3: Repeat part (1) by taking 100 samples from each signal. Plot the signal's spectrums and identify which signal has a symmetric spectrum and why?

- Ans3: Refer to Figure 15. After plotting the graph of 100 samples, none of the signals appear symmetric about y-axis.

TASK2:

1) Find the number of samples (N0), the duration of the signal (T0), and the sampling interval (T).
    - Number of samples (N0) = 864000
      The duration of the signal (T0) = 9 seconds
      Sampling Frequency: 96000

2) Generate the subsampled signal y1 from signal y by subsampling with rate 2. Find the number of samples (N0), the duration of the signal (T0), and the sampling interval (T).
    - Number of samples (N1) = 432000
      The duration of the signal (T1) = 9 seconds
      Sampling Frequency: 48000
      The sampling interval (T1) = 1/SamplingFrequency

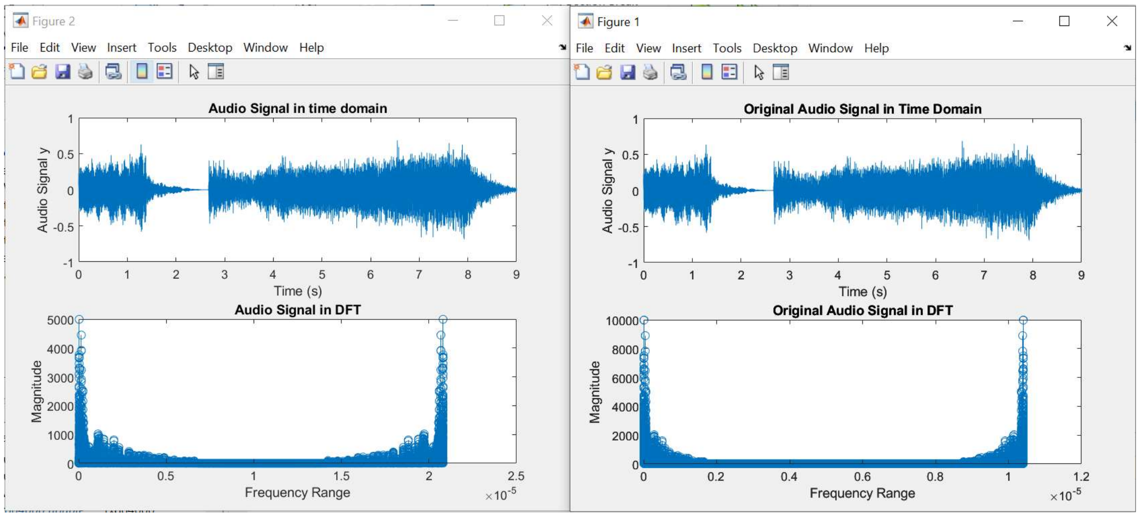
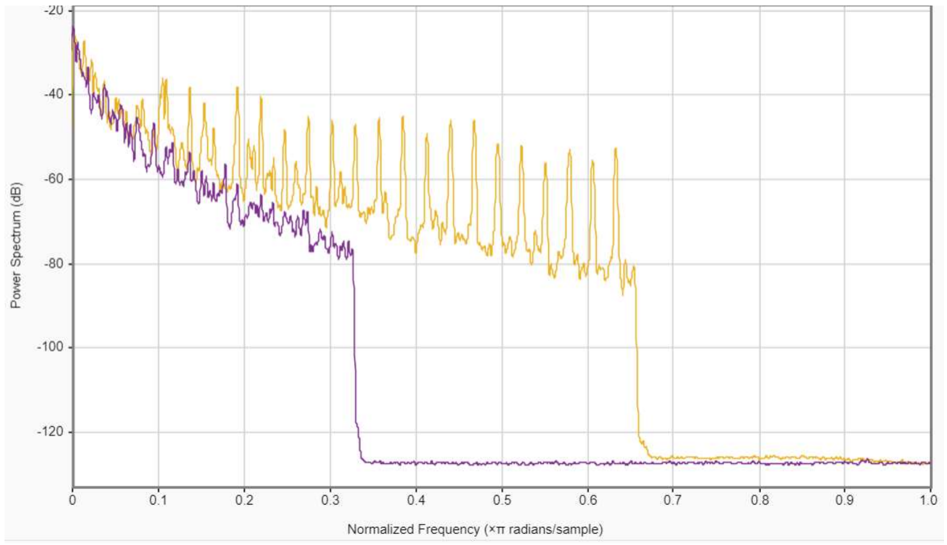


Figure 16: Plot of original signal y (Right) and plot of y1 subsampled (Left)

How has the signal y1 and its spectrum changed compared to the original signal? Enlarge a part of the plot for y and y1 for better visualization. Explain the reason.

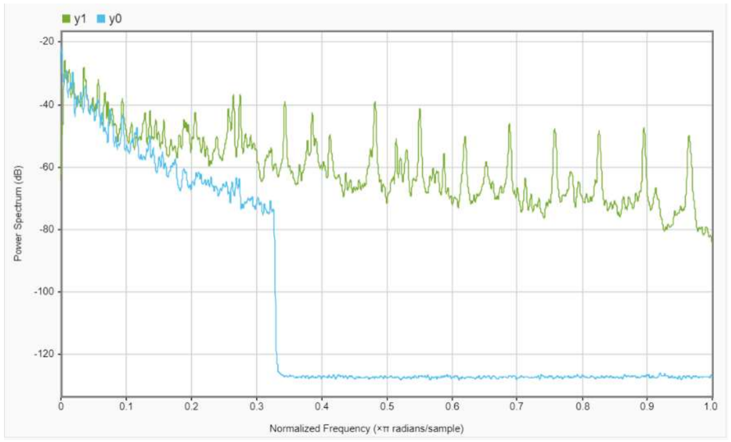
- Significant difference was observed in the spectrum of y1 compared to original signal.



3) To listen to the audio signal, use the command sounddevice.play(x, fs) . Play both signals andsee how the original audio signal has changed after subsampling.

   - Only the sound quality when listening to the sound was dropped.

4) Change the sampling rate in step (4) to 5 and explain how the audio signal and its spectrum change.



- Audio quality was further reduced, almost negligible.

TASK3:

1) Create a rect filter that only passes frequencies less than 2000. Apply this filter to the signal y and plot the response in the time and frequency domain.

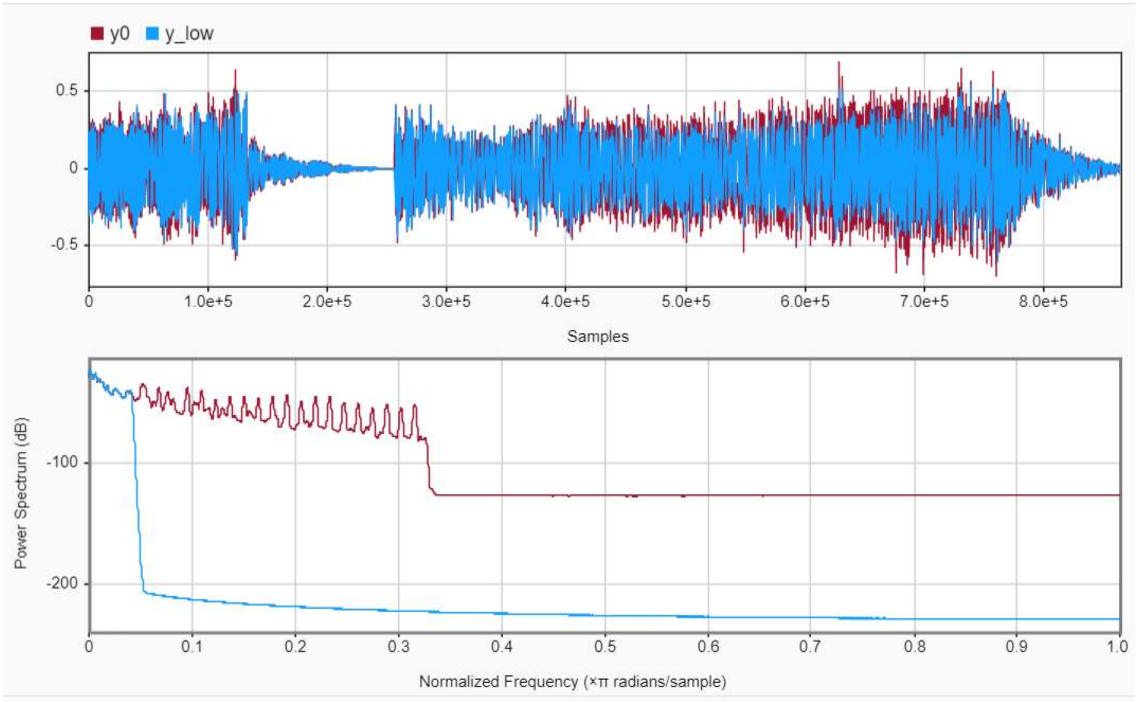


Figure 17: Time domain and Frequency spectrum plot of Lowpass filter.

2) Use command play and listen to how the audio signal has changed.
    - The audio became faint and higher pitch frequencies were no longer audible.

3) Design a filter that cuts the bass frequencies (e.g., frequencies between 16 and 256 Hz). Plot the frequency spectrum and listen to the sound. How has it changed?

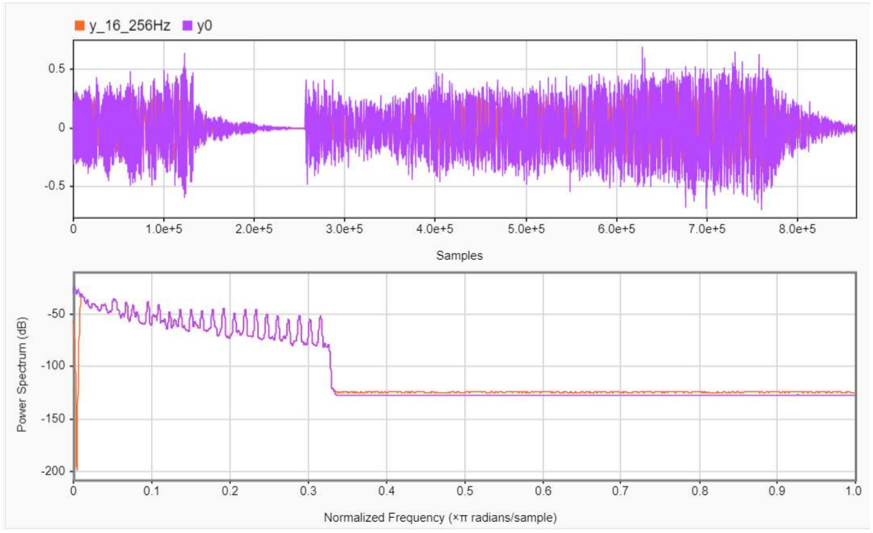


Figure 18: Frequency spectrum and time domain of Bandstop filtered signal and original signal.

- Not much difference was observed in the sound of the signal as the recording did not have much bass noise in it.

4) Design a filter that amplifies treble frequencies (e.g., frequencies between 2048 and 16384 Hz at the higher end of human hearing.) by 25%. Plot the frequency spectrum and listen to the sound. How has it changed?

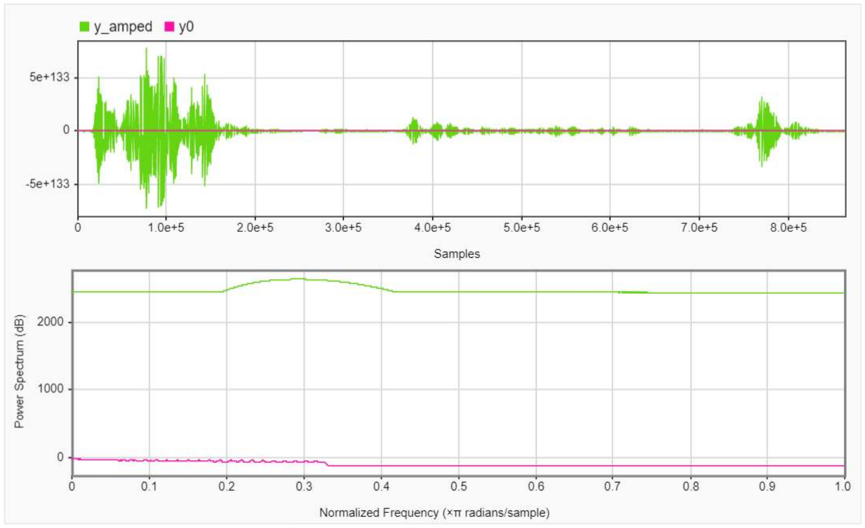


Figure 19: Amplified graph.

DISCUSSION:

The project was completed in a span of 5 days. Many difficulties were faced when trying to resample the signal as sometimes the original signal got overwritten causing errors in measurements. A few questions asked were difficult to answer but are still answered to the best of our knowledge. The signals were manipulated in various methods and various outcomes were observed and recorded.

CONCLUSION

In conclusion it was found out that a signal has a unique outcome when subject to various filters, change in frequency, change in number of samples etc. The signal mainly had a drop in quality when subject to filters and down samples. All tasks were performed using MATLAB and desired results were obtained.

REFERENCES:

https://www.mathworks.com/help/matlab/ref/

https://stackoverflow.com/