



Namal University, Mianwali

Department of Electrical Engineering

EE 345 (L) – Digital Signal Processing (Lab)

Lab – 9

Audio Signal Processing using MATLAB

Student Name	Student ID
Fahim Ur Rehman Shah	NIM-BSEE-2021-24

Instructor: Zulaikha Kiran

Lab Engineer: Faizan Ahmad

Version 1.0 – Muhammad Usman – 15/04/2021
Version 1.1 – Zulaikha Kiran – 05/04/2022
Version 1.2 – Husnain, Saiqa Dilawaiz – 06/04/2023
Version 1.3 – Faizan Ahmad – 26/02/2024

Introduction

The purpose of this lab is to enable the students to study audio signal synthesis and analysis using MATLAB.

Course Learning Outcomes

CLO1: Develop algorithms to perform signal processing techniques on digital signals using MATLAB and DSP Kit DSK6713

CLO3: Deliver a report/lab notes/presentation/viva, effectively communicating the design and analysis of the given problem

Equipment

- Software
 - MATLAB

Instructions

1. This is an individual lab. You will perform the tasks individually and submit a report.
2. Some of these tasks are for practice purposes only while others (marked as 'Exercise') have to be answered in the report.
3. When asked to display an image/ graph in the exercise, either save it as jpeg or take a screenshot, in order to insert it in the report.
4. The report should be submitted on the given template, including:
 - a. Code (copy and pasted, NOT a screenshot)
 - b. Output values (from command window, can be a screenshot)
 - c. Output figure/graph (as instructed in 3)
 - d. Explanation where required
5. The report should be properly formatted, with easy to read code and easy to see figures.
6. Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom. Multiple such incidents will result in disciplinary action being taken.

Background

Audio signal processing involves the manipulation, analysis, and enhancement of sound waves using digital techniques. Initially, analog audio signals are converted into digital format through analog-to-digital converters (ADCs). Once digitized, various processing techniques can be applied, including filtering, equalization, compression, and reverberation, among others. Filtering can remove unwanted frequencies or enhance specific ones, while equalization adjusts the frequency balance of the signal. Compression reduces the dynamic range of the signal, making it more uniform in volume, often used in music production and broadcasting. Reverberation adds simulated acoustic reflections to create a sense of space or ambiance. These techniques can be implemented using algorithms in software or hardware, providing flexibility and control over audio content in applications ranging from music production and mixing to telecommunications and entertainment. Additionally, advancements in machine learning and artificial intelligence have further expanded the capabilities of audio signal processing, enabling tasks such as noise reduction, source separation, and automatic audio tagging.

Exercise:**Task 1:**

1. Using this information create a few different notes in MatLab as follows:

```
f = sin(2*pi*174.61*t);
```

```
g = sin(2*pi*195.99*t);
```

```
a = sin(2*pi*220*t);
```

```
b = sin(2*pi*246.94*t);
```

2. Now to create a line of music use the following command:

Put the notes in the order you want them to play.

```
line1 = [a,b,g,f,f,b,b];
```

```
line2 = [a,b,b,f,f,g,g];
```

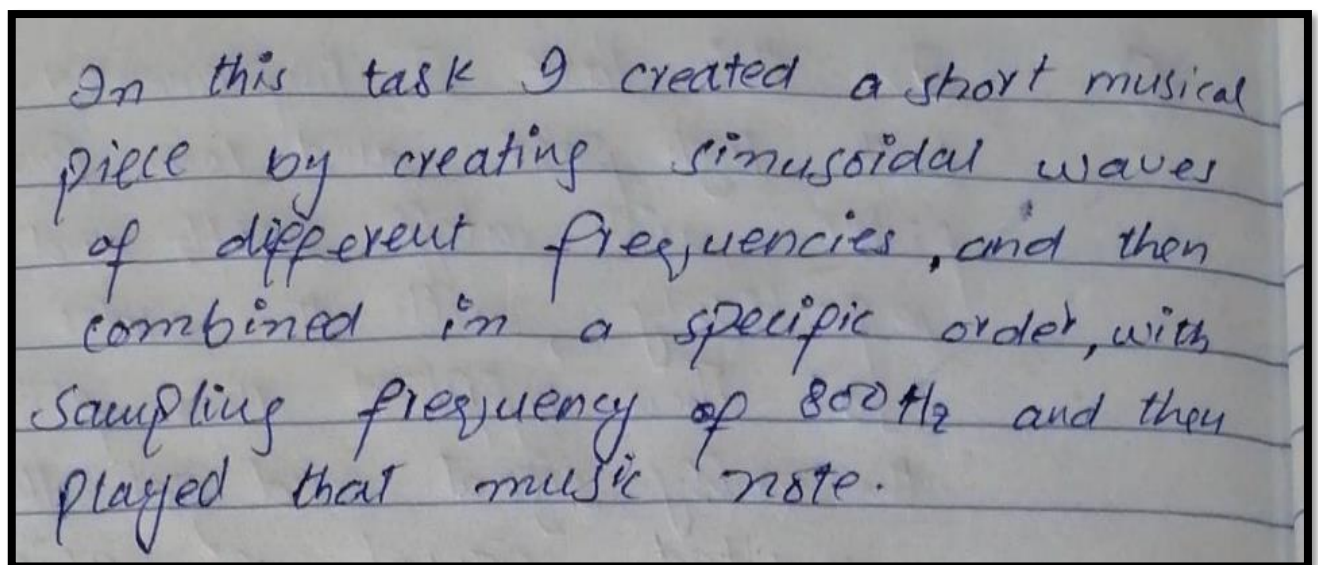
3. To create a music file use:

```
music = [line1, line2];
```

4. To play the file use 'sound(music)'

Code:

```
fs = 800; % Sampling frequency
t = 0:1/fs:1; % Time vector from 0 to 1 second with sampling rate 44100 Hz
f = sin(2*pi*174.61*t);
g = sin(2*pi*195.99*t);
a = sin(2*pi*220*t);
b = sin(2*pi*246.94*t);
line1 = [a, b, g, f, f, b, b];
line2 = [a, b, b, f, f, g, g];
music = [line1, line2];
sound(music)
```

Explanation:

In this task I created a short musical piece by creating sinusoidal waves of different frequencies, and then combined in a specific order, with sampling frequency of 800 Hz and then played that music note.

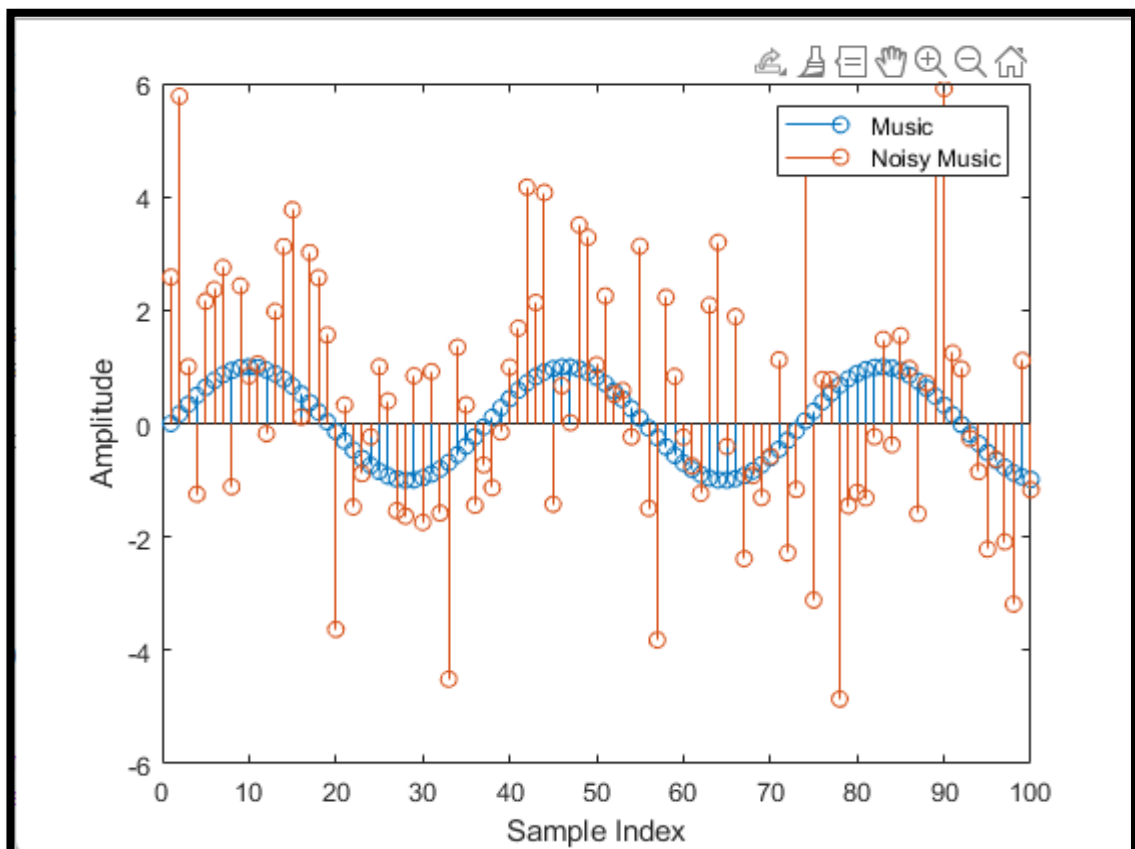
Task 2:

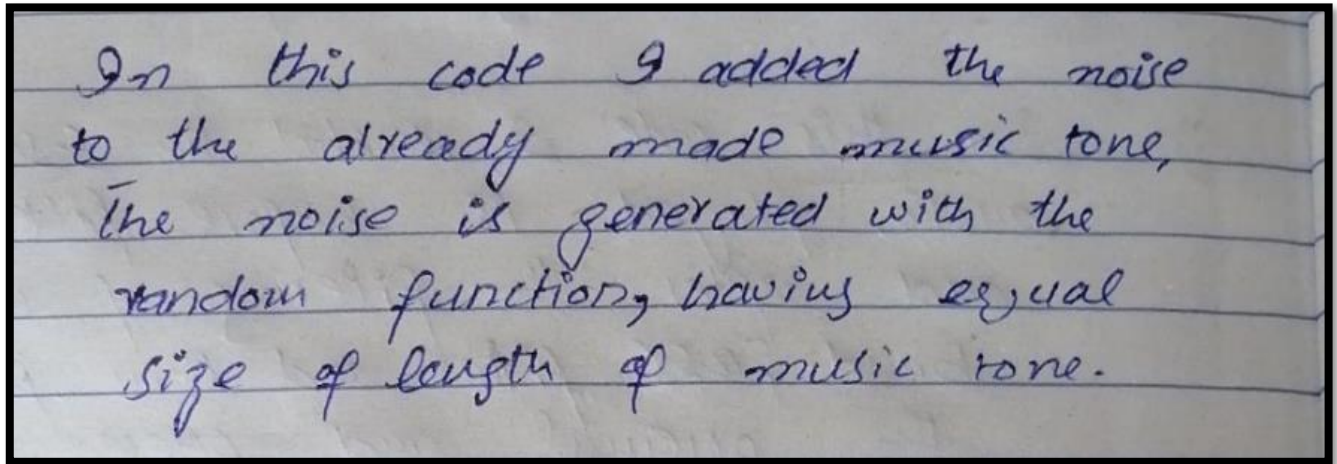
Add noise in the above music signal using `2*randn(size(t))` command, and take $F_s=8000$.

Code:

```
fs = 8000; % Sampling frequency
t = 0:1/fs:1; % Time vector from 0 to 1 second with sampling rate 8000 Hz
f = sin(2*pi*174.61*t);
g = sin(2*pi*195.99*t);
a = sin(2*pi*220*t);
b = sin(2*pi*246.94*t);
line1 = [a, b, g, f, f, b, b];
line2 = [a, b, b, f, f, g, g];
music = [line1, line2];
noise = 2*randn(size(music)); % Generate noise with the same size as music
noisy_music = music + noise; % Add noise to the music signal
sound(noisy_music, fs) % Play the noisy music signal
figure
stem(music)
xlim([0 100])
hold on
stem(noisy_music)
xlim([0 100])
xlabel('Sample Index') % Add x-axis label
ylabel('Amplitude') % Add y-axis label
legend('Music', 'Noisy Music') % Add legend
```

Graph:



Explanation:**Task 3:**

To remove noise, design a low-pass filter using commands below, and the cutoff frequency of filter should be 175Hz.

cutoff = f/(fs/2); order =

50;

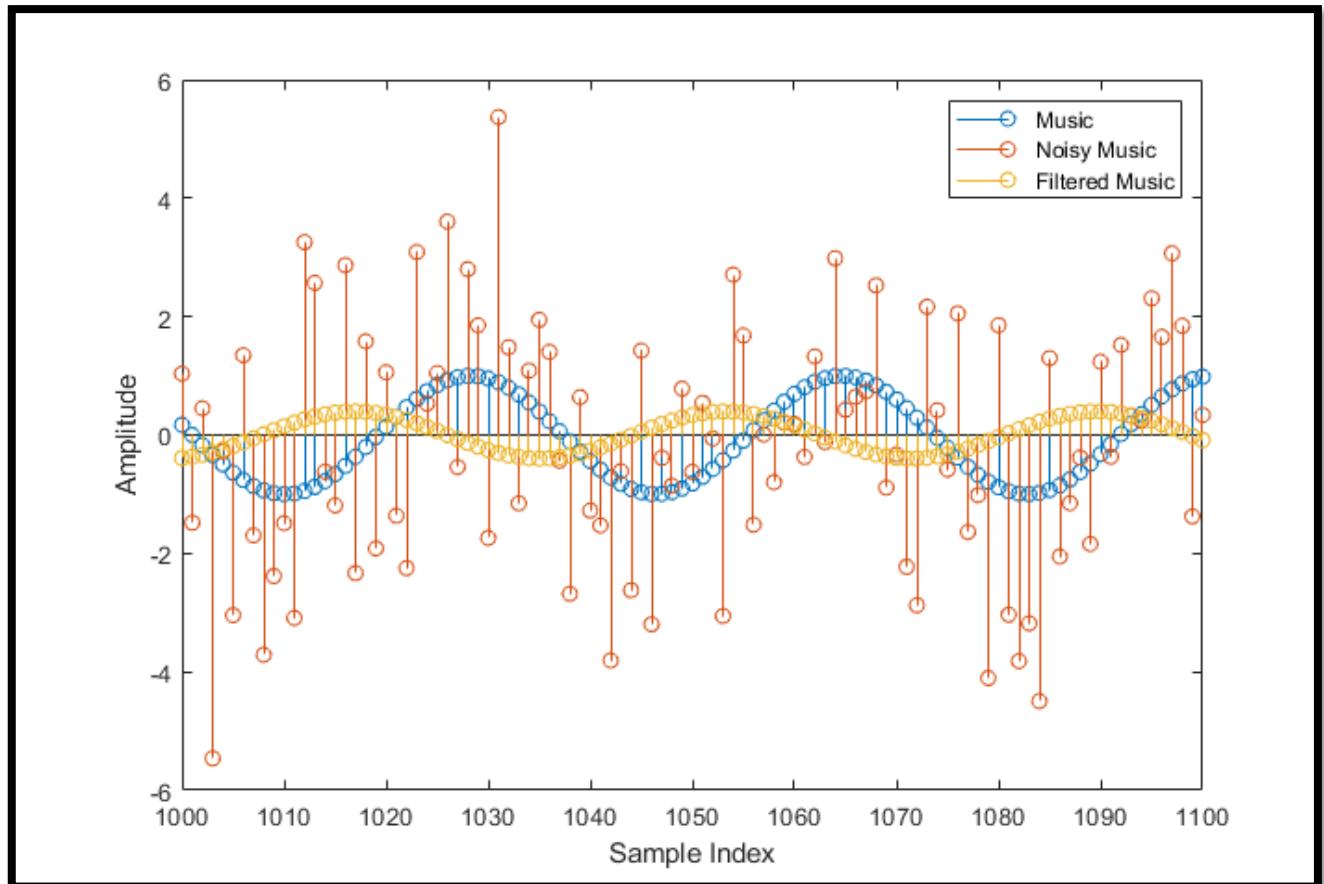
d = designfilt('lowpassfir','CutoffFrequency',cutoff,'FilterOrder',order); output=

filter(d,music);

Code:

```
fs = 8000; % Sampling frequency
t = 0:1/fs:1; % Time vector from 0 to 1 second with sampling rate 8000 Hz
f = sin(2*pi*174.61*t);
g = sin(2*pi*195.99*t);
a = sin(2*pi*220*t);
b = sin(2*pi*246.94*t);
line1 = [a, b, g, f, f, b, b];
line2 = [a, b, b, f, f, g, g];
music = [line1, line2];
noise = 2*randn(size(music)); % Generate noise with the same size as music
noisy_music = music + noise; % Add noise to the music signal
cutoff = 175/(fs/2); % Cutoff frequency of 175Hz
order = 50; % Filter order
d = designfilt('lowpassfir','CutoffFrequency',cutoff,'FilterOrder',order); % Design
the low-pass filter
output = filter(d,music); % Apply the filter to the music signal
sound(output, fs) % Play the filtered music signal
figure
stem(music)
xlim([1000 1100])
hold on
stem(noisy_music)
xlim([1000 1100])
hold on
stem(output)
xlim([1000 1100])
xlabel('Sample Index') % Add x-axis label
ylabel('Amplitude') % Add y-axis label
legend('Music', 'Noisy Music', 'Filtered Music') % Add legend
```

Graph:



Explanation:

In this code I designed a low-pass filter with cutoff frequency of 175 Hz and the order 50. Then I played the filtered output which was observed after the low-pass filter.

Task 4:

Design a high-pass filter having cutoff frequency 200Hz.

Code:

```
fs = 8000; % Sampling frequency
t = 0:1/fs:1; % Time vector from 0 to 1 second with sampling rate 8000 Hz
f = sin(2*pi*174.61*t);
g = sin(2*pi*195.99*t);
a = sin(2*pi*220*t);
b = sin(2*pi*246.94*t);
line1 = [a, b, g, f, f, b, b];
```

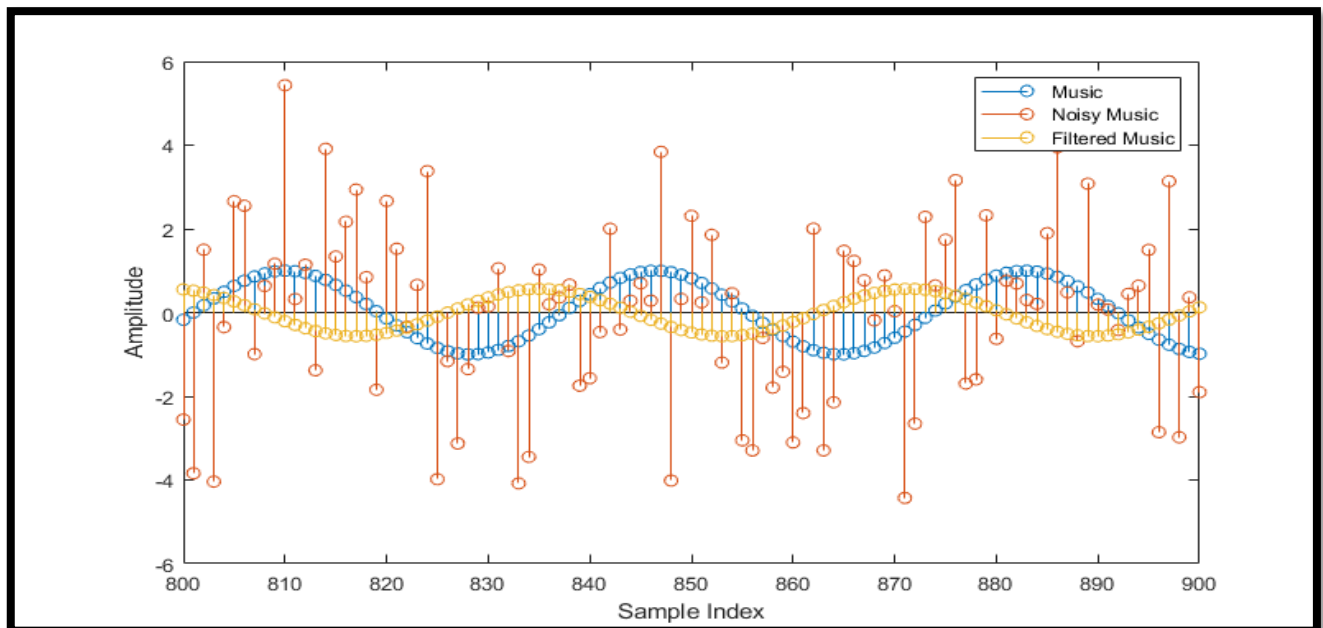


```

line2 = [a, b, b, f, f, g, g];
music = [line1, line2];
noise = 2*randn(size(music)); % Generate noise with the same size as music
noisy_music = music + noise; % Add noise to the music signal
cutoff = 200/(fs/2); % Cutoff frequency of 200Hz
order = 50; % Filter order
d = designfilt('highpassfir','CutoffFrequency',cutoff,'FilterOrder',order); % Design the high-pass filter
output = filter(d,music); % Apply the filter to the music signal
sound(output, fs) % Play the filtered music signal
figure
stem(music)
xlim([800 900])
hold on
stem(noisy_music)
xlim([800 900])
hold on
stem(output)
xlim([800 900])
xlabel('Sample Index') % Add x-axis label
ylabel('Amplitude') % Add y-axis label
legend('Music', 'Noisy Music','Filtered Music') % Add legend

```

Graph:



Explanation:

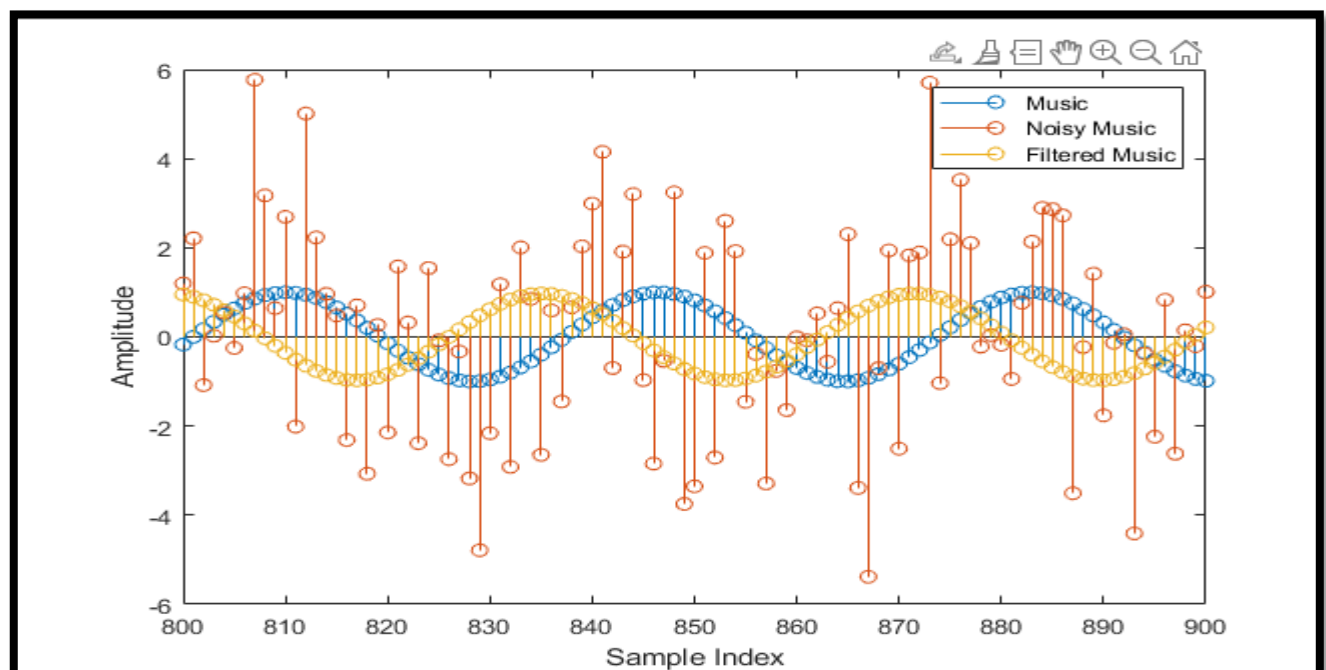
In this task I designed a high-pass filter for the noisy-music signal and then filtered the noise from signal and get the original music signal after filtering. The cut-off frequency of high-pass filter was 200 Hz.

Task 5:

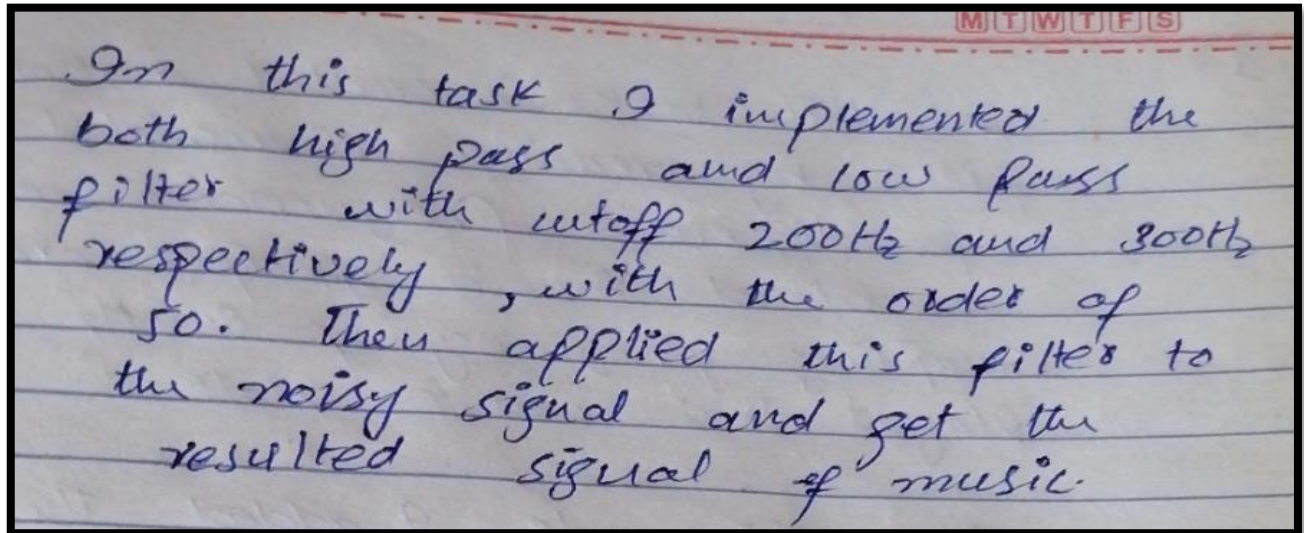
Design a band-pass filter which passes the range of frequency 200-300Hz.

Code:

```
fs = 8000; % Sampling frequency
t = 0:1/fs:1; % Time vector from 0 to 1 second with sampling rate 8000 Hz
f = sin(2*pi*174.61*t);
g = sin(2*pi*195.99*t);
a = sin(2*pi*220*t);
b = sin(2*pi*246.94*t);
line1 = [a, b, g, f, f, b, b];
line2 = [a, b, b, f, f, g, g];
music = [line1, line2];
noise = 2*randn(size(music)); % Generate noise with the same size as music
noisy_music = music + noise; % Add noise to the music signal
fcut1 = 200/(fs/2); % Lower cutoff frequency of 200Hz
fcut2 = 300/(fs/2); % Upper cutoff frequency of 300Hz
order = 50; % Filter order
d =
designfilt('bandpassfir','FilterOrder',order,'CutoffFrequency1',fcut1,'CutoffFrequency2',fcut2); % Design the band-pass filter
output = filter(d,music); % Apply the filter to the music signal
sound(output, fs) % Play the filtered music signal
figure
stem(music)
xlim([800 900])
hold on
stem(noisy_music)
xlim([800 900])
hold on
stem(output)
xlim([800 900])
xlabel('Sample Index') % Add x-axis label
ylabel('Amplitude') % Add y-axis label
legend('Music', 'Noisy Music','Filtered Music') % Add legend
```

Graph:

Explanation:



Task 6:

Record your own voice and observe the frequency spectrum?

```
myVoice = audiorecorder; disp('Start
speaking.') recordblocking(myVoice, 5)
disp('End of recording. Playing back ...')
play(myVoice)
```

Code:

```
fs = 8000; % Sampling frequency
myVoice = audiorecorder(fs, 16, 1); % Create an audiorecorder object with the desired
sampling frequency (fs), bit depth (16), and number of channels (1)
disp('Start speaking.')
recordblocking(myVoice, 5) % Record your voice for 5 seconds
disp('End of recording. Playing back ...')
play(myVoice) % Playback the recorded voice
voice = getaudiodata(myVoice); % Get the recorded voice data
fcut1 = 200/(fs/2); % Lower cutoff frequency of 200Hz
fcut2 = 300/(fs/2); % Upper cutoff frequency of 300Hz
order = 50; % Filter order
d=designfilt('bandpassfir', 'FilterOrder',order, 'CutoffFrequency1',fcut1, 'CutoffFrequency2',
fcut2); % Design the band-pass filter
output = filter(d,voice); % Apply the filter to the voice signal
sound(output, fs) % Play the filtered voice signal
figure
subplot(3,1,1)
plot(voice) % Plot the original voice signal
title('Original Voice')
subplot(3,1,2)
plot(output) % Plot the filtered voice signal
title('Filtered Voice')
subplot(3,1,3)
spectrogram(output,[],[],[],fs,'yaxis') % Plot the spectrogram of the filtered voice
title('Spectrogram')
```

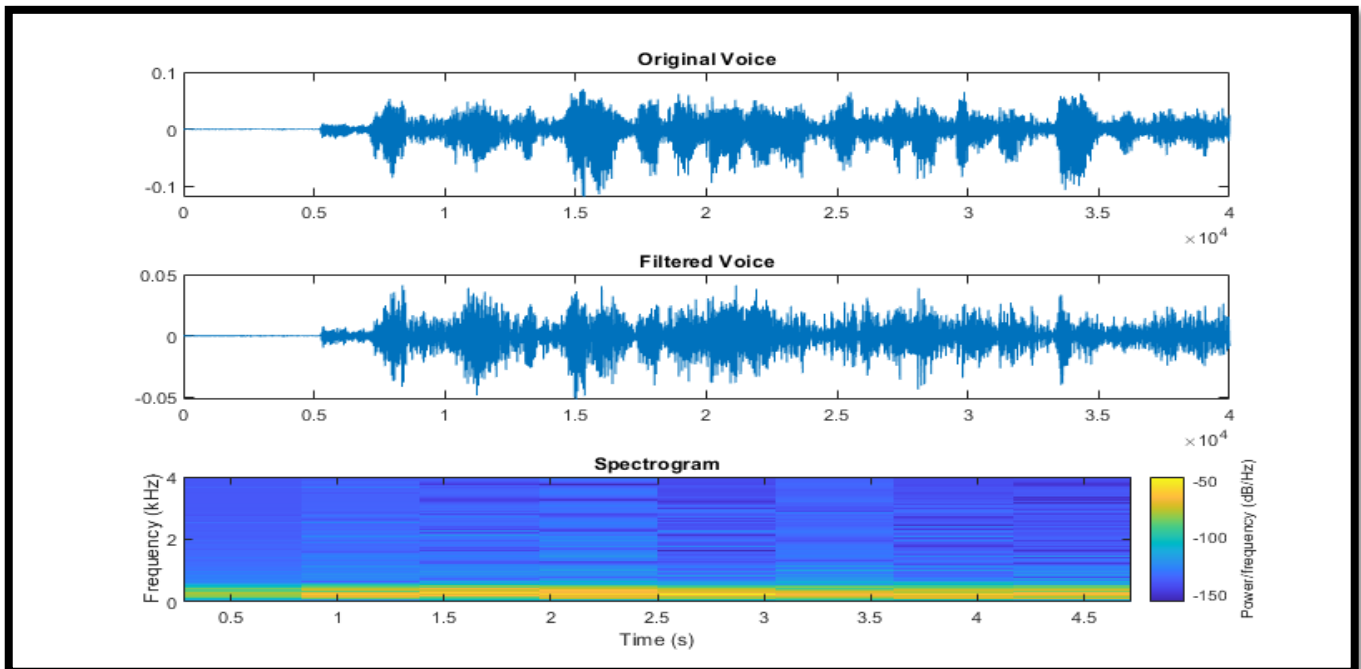
Output:

```
Start speaking.

End of recording. Playing back ...
ans =
    audioplayer with properties:

        SampleRate: 8000
        BitsPerSample: 16
        NumChannels: 1
        DeviceID: -1
        CurrentSample: 257
        TotalSamples: 40000
        Running: 'on'
        StartFcn: []
        StopFcn: []
        TimerFcn: []
        TimerPeriod: 0.0500
        Tag: ''
        UserData: []
        Type: 'audioplayer'
```

Graph:



Explanation:

In this code I recorded the 5-second audio, played it before filtering and after the filtering of low band-pass filter and then plotted the original and filtered signal and its frequency spectrum.

Conclusion:

In this lab I learned about the creation of random music signal, adding noise to it and then different filtering methods to get the original signal. I also learned about the creating audio signal and then plotted its spectrum.

Evaluation Rubric

- **Method of Evaluation:** In-lab marking by instructors, Report submitted by students
- **Measured Learning Outcomes:**

CLO1: Develop algorithms to perform signal processing techniques on digital signals using MATLAB and DSP Kit DSK6713

CLO3: Deliver a report/lab notes/presentation/viva, effectively communicating the design and analysis of the given problem

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Tasks (CLO1)	All tasks completed correctly. Correct code with proper comments.	Most tasks completed correctly.	Some tasks completed correctly.	Most tasks incomplete or incorrect.	All tasks incomplete or incorrect.	
Output (CLO1)	Output correctly shown with all Figures/Plots displayed as required and properly labelled	Most Output/Figures/Plots displayed with proper labels	Some Output/Figures/Plots displayed with proper labels OR Most Output/Figures/Plots displayed but without proper labels	Most of the required Output/Figures/Plots not displayed	Output/Figures/Plots not displayed	
Answers (CLO1)	Meaningful answers to all questions. Answers show the understanding of the student.	Meaningful answers to most questions.	Some correct/ meaningful answers with some irrelevant ones	Answers not understandable/ not relevant to questions	Not Written any Answer	
Report (CLO3)	Report submitted with proper grammar and punctuation with proper conclusions drawn and good formatting	Report submitted with proper conclusions drawn with good formatting but some grammar mistakes OR proper grammar but not very good formatting	Some correct/ meaningful conclusions. Some parts of the document not properly formatted or some grammar mistakes	Conclusions not based on results. Bad formatting with no proper grammar/punctuation	Report not submitted	
Total						