

AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

PROJECT REPORT

Course no :EEE 3218

Course Title :Digital Signal Processing(DSP)

Name Of The Project : Software Project

Date of Submission :23/02/23

Submitted by,

Name :MD Khorsheduzzman Sizan

ID :190205088

Section :B1

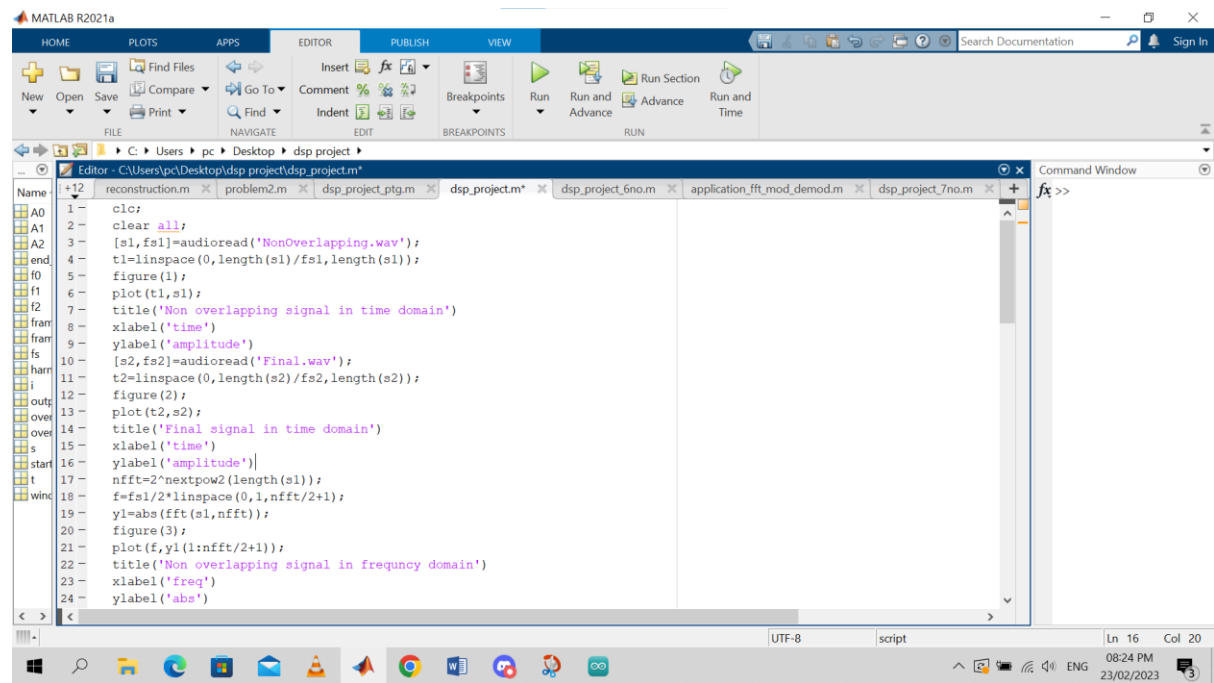
Year :3rd

Semester :2nd

Question 1: Find the spectrum of each of the instrument's sound from 'NonOverlapping.wav' file and determine the frequency ranges.

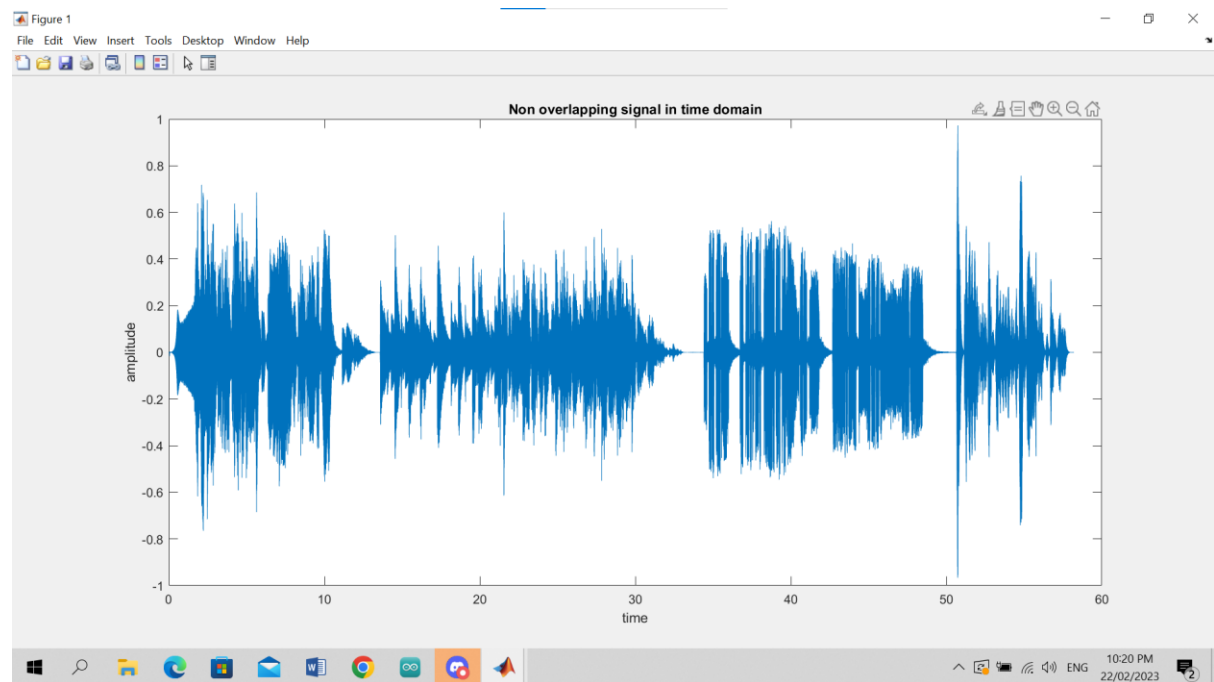
Ans:

Code:

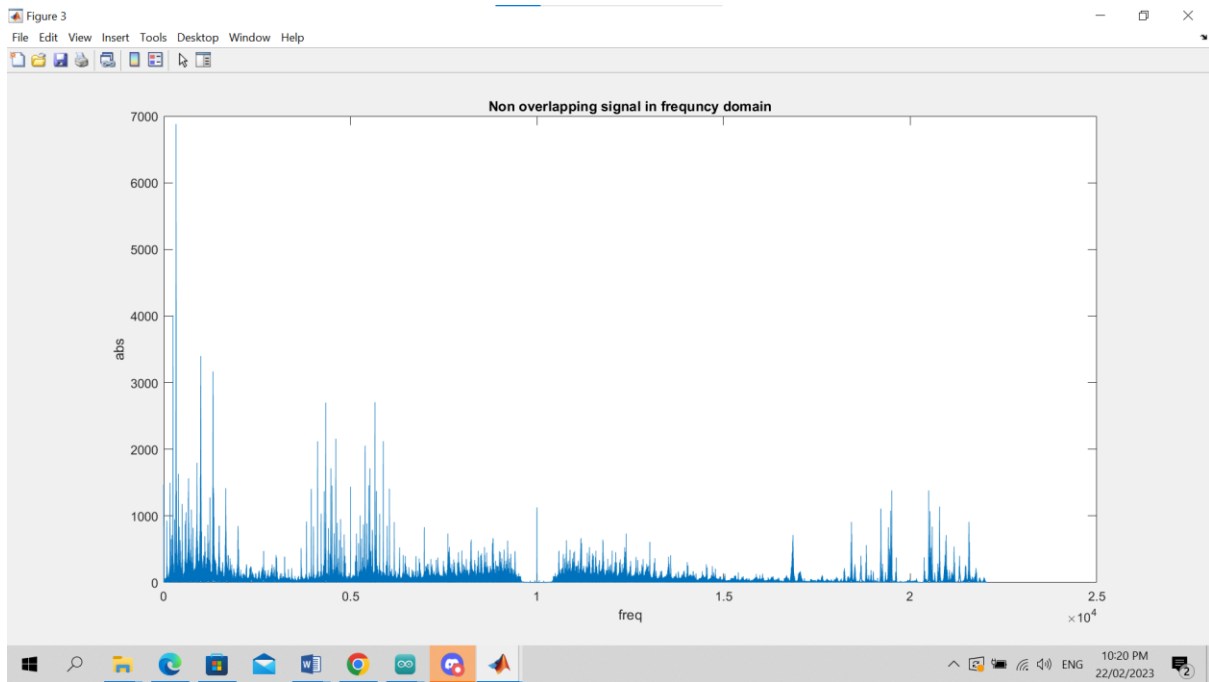


```
1 clc;
2 clear all;
3 [s1,fs1]=audioread('NonOverlapping.wav');
4 t1=linspace(0,length(s1)/fs1,length(s1));
5 figure(1);
6 plot(t1,s1);
7 title('Non overlapping signal in time domain')
8 xlabel('time')
9 ylabel('amplitude')
10 [s2,fs2]=audioread('Final.wav');
11 t2=linspace(0,length(s2)/fs2,length(s2));
12 figure(2);
13 plot(t2,s2);
14 title('Final signal in time domain')
15 xlabel('time')
16 ylabel('amplitude')
17 nfft=2*nextpow2(length(s1));
18 f=fs1/2*linspace(0,1,nfft/2+1);
19 y1=abs(fft(s1,nfft));
20 figure(3);
21 plot(f,y1(1:nfft/2+1));
22 title('Non overlapping signal in frequency domain')
23 xlabel('freq')
24 ylabel('abs')
```

Graph:



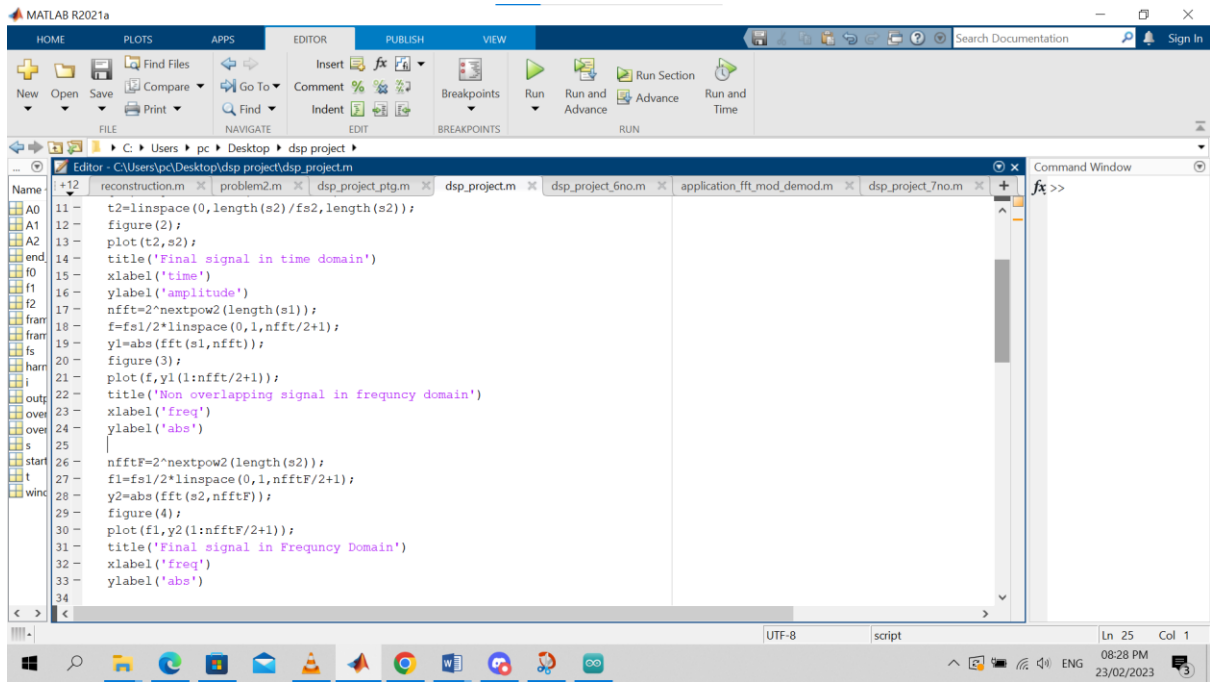
Here using audioread function the non overlapping audio signal is taken as an input in the project. Here the function gave the sampling frequency as well as the time domain expression. By plotting it is observed in time domain



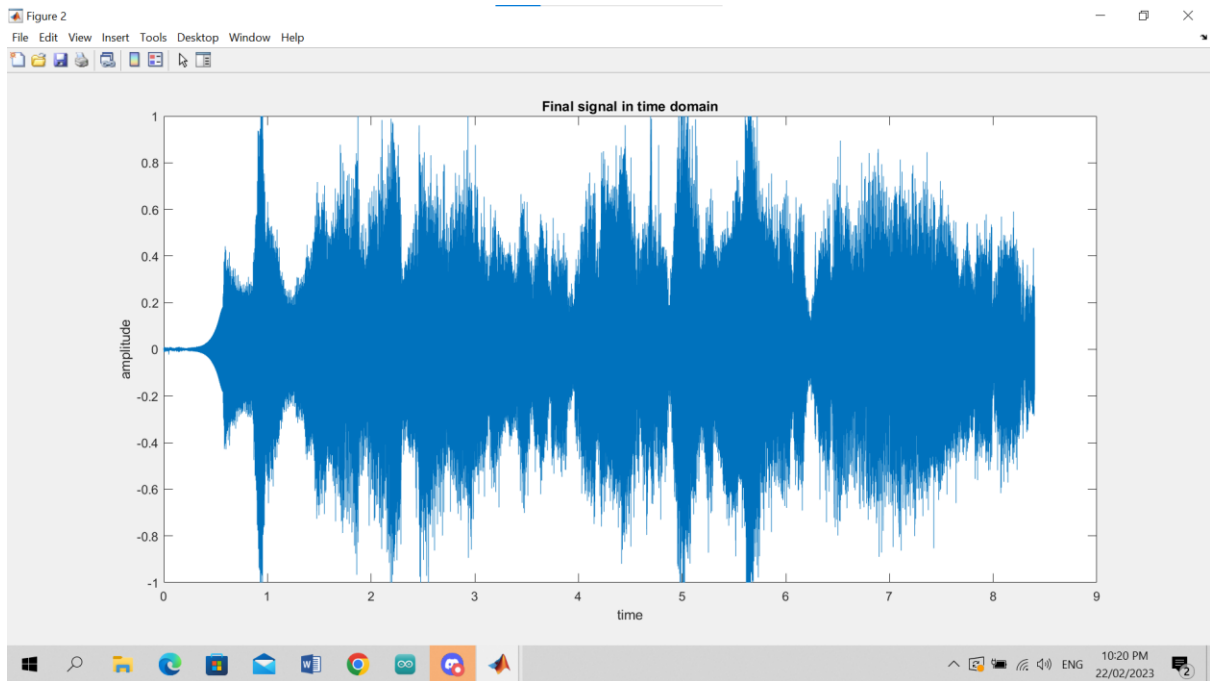
Here by using fft function the time domain signal is converted to frequency domain and the frequency spectrum of audio signal can be observed.

Question2: Find the spectrum of the combined sound signal from 'final.wav' file and determine the frequency range.

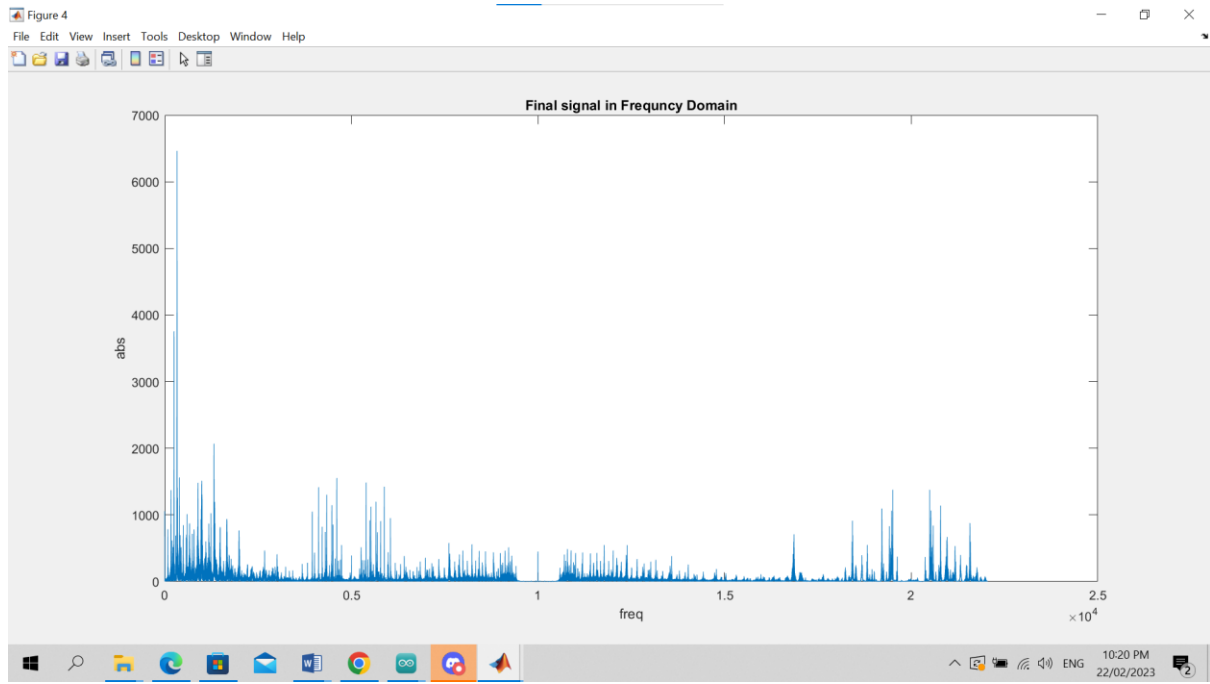
Codes:



Graphs



Here similar to question 1, the final.wav(audio signal) is taken as an input to matlab by using audioread function. The timedomain output can be observed here.



By using fft function ,the time domain signal is converted to frequency domain.Here 4 different frequency range can be observed.It can be observed that the file final.wave contains combined and overlapping modified sounds of the instruments - trumpet, piano, violin, and guitar.

Question 3: Design a digital filter that would separate the sound of each instrument from “final.wav” file. Mention the filter type, filter design methods, filter order, and the cutoff frequency of the filter.

Codes:

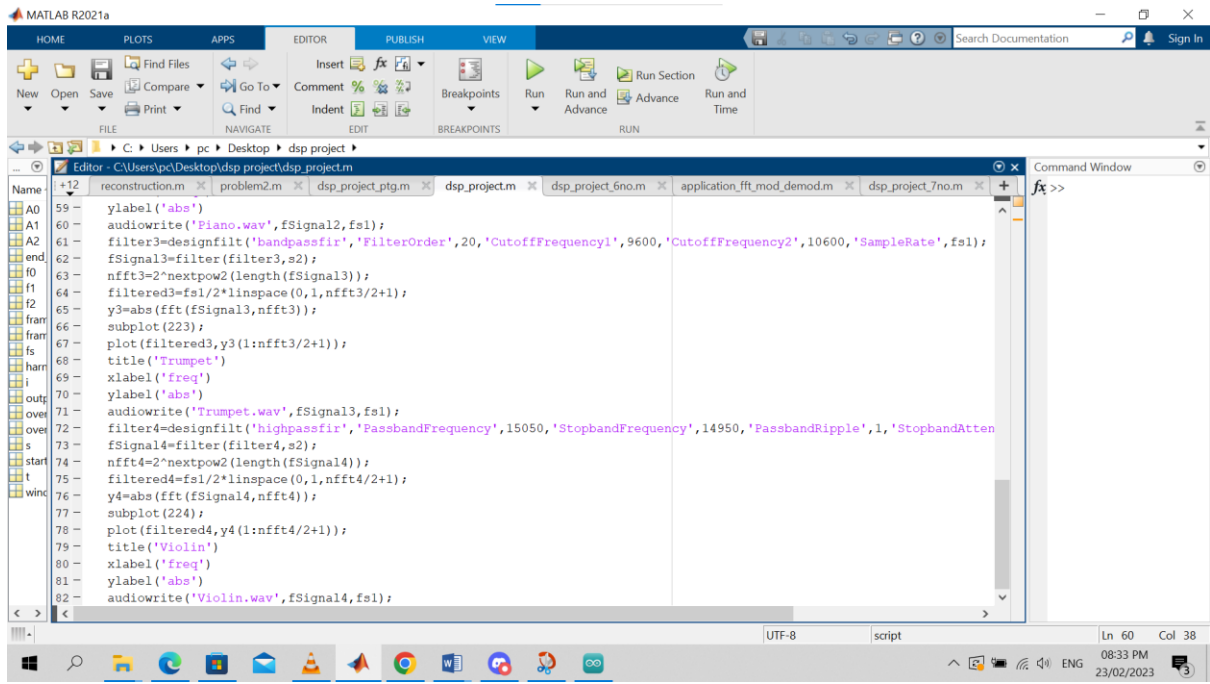
The image displays two screenshots of the MATLAB R2021a editor interface, showing code for digital filter design. The top screenshot shows the editor with a script file named 'dsp_project.m' open. The code defines a bandpass filter for piano and a highpass filter for trumpet. The bottom screenshot shows the same script file with additional code for a lowpass filter for guitar and a bandpass filter for piano.

```

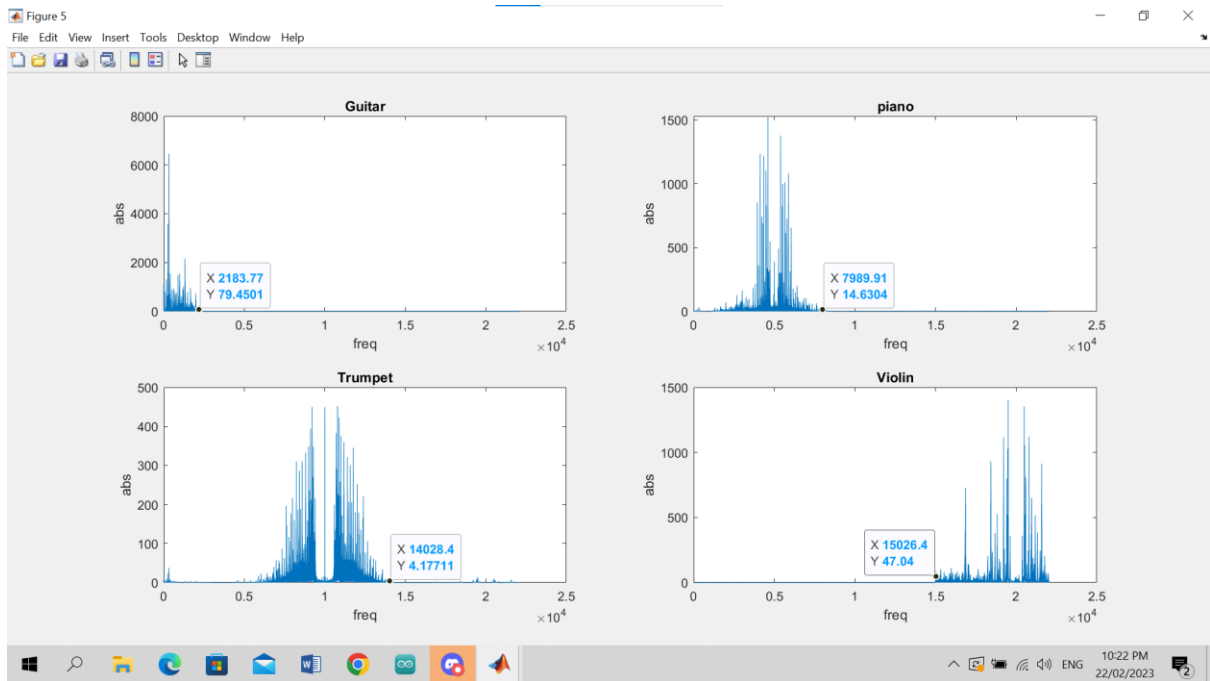
% Piano filter
title('piano')
xlabel('freq')
ylabel('abs')
audiowrite('Piano.wav', fSignal2, fs1);
filter3=designfilt('bandpassfir', 'FilterOrder', 20, 'CutoffFrequency1', 9600, 'CutoffFrequency2', 10600, 'SampleRate', fs1);
fSignal3=filter(filter3, s2);
nfft3=2*nextpow2(length(fSignal3));
filtered3=fsl/2*linspace(0,1,nfft3/2+1);
y3=abs(fft(fSignal3,nfft3));
subplot(223);
plot(filtered3,y3(1:nfft3/2+1));
title('Trumpet')
xlabel('freq')
ylabel('abs')
audiowrite('Trumpet.wav', fSignal3, fs1);
filter4=designfilt('highpassfir', 'PassbandFrequency', 15050, 'StopbandFrequency', 14950, 'PassbandRipple', 1, 'StopbandAttenuat
fSignal4=filter(filter4, s2);
nfft4=2*nextpow2(length(fSignal4));
filtered4=fsl/2*linspace(0,1,nfft4/2+1);
y4=abs(fft(fSignal4,nfft4));
subplot(224);
plot(filtered4,y4(1:nfft4/2+1));
title('Violin')
xlabel('freq')

% Guitar filter
filter1=designfilt('lowpassfir', 'PassbandFrequency', 2350, 'StopbandFrequency', 2450, 'PassbandRipple', 1, 'StopbandAttenuat
fSignal1=filter(filter1, s2);
nfft1=2*nextpow2(length(fSignal1));
filtered1=fsl/2*linspace(0,1,nfft1/2+1);
y3=abs(fft(fSignal1,nfft1));
figure(5);
subplot(221);
plot(filtered1,y3(1:nfft1/2+1));
title('Guitar')
xlabel('freq')
ylabel('abs')
audiowrite('Guitar.wav', fSignal1, fs1);

% Piano filter
filter2=designfilt('bandpassfir', 'FilterOrder', 30, 'CutoffFrequency1', 3500, 'CutoffFrequency2', 6200, 'SampleRate', fs1);
fSignal2=filter(filter2, s2);
nfft2=2*nextpow2(length(fSignal2));
filtered2=fsl/2*linspace(0,1,nfft2/2+1);
y2=abs(fft(fSignal2,nfft2));
subplot(222);
plot(filtered2,y2(1:nfft2/2+1));
title('piano')
  
```



Graphs:



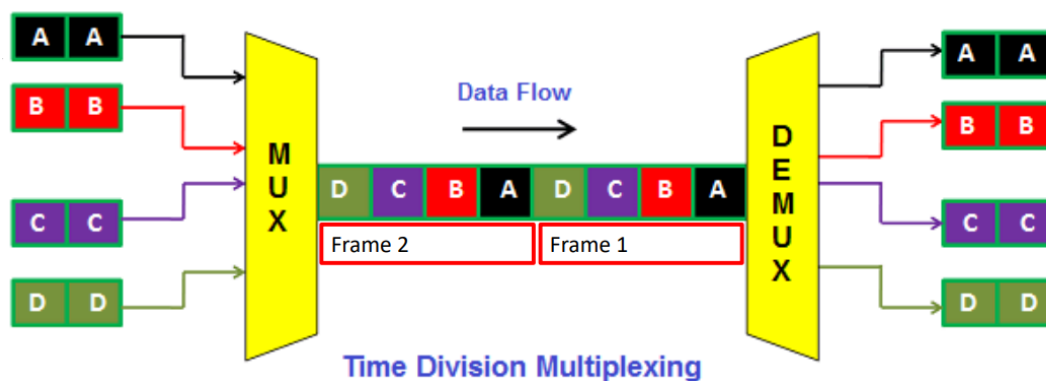
Here a lowpass, 2 bandpass and a highpass filter is designed to separate the sound of each instrument from “final.wav” file. As the the sound of guiter has low frequency range,a lowpass filter is used for it to separate the signal.On the other hand the band of piano and trumpet is in the middle.So bandpass filter is designed for them to separate from the given signal.Finally for the highest frequency range which is the band of violin ,is filtered through a highpass filter.Filter order and cutoff frequencies are mentioned.

Question 4: Extract each instrument sound in separate '.wav' file.

Ans:The audio files is given the zip file which was extracted.

Question 5: Closely observe the spectrum of all 4 separated wav-files individually. Can you suggest any way to pass the individual wav files separately through a channel of bandwidth 0 to 10 kHz?

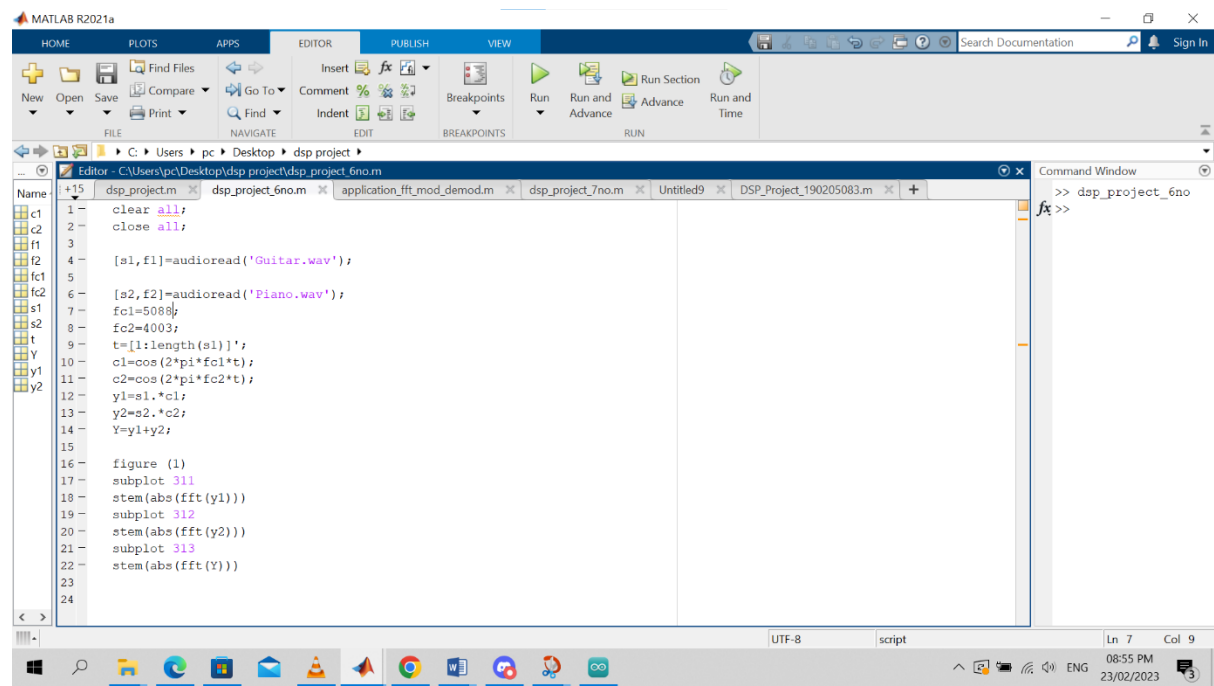
Ans: Yes. By using Time Division Multiplexing(TDM) we can pass the individual wav files separately through a channel of bandwidth 0 to 10kHz. We know in case of TDM ,the bandwidth or the frequency range remains same for all the signals. A frame of data which include bits from all the data wave is generated and passed through a single channel. Multiplexer and demultiplexer is used in this case.



Question 6: Send any 2 of the above 4 separated signals through a 2-channel FDM (frequency division multiplexed) link. Use a carrier of X Hz where X is the last 4 least significant digits of your 9 digit ID. Choose another carrier cleverly to optimize the FDM link bandwidth while keeping the signal fidelity as high as possible.

Ans:

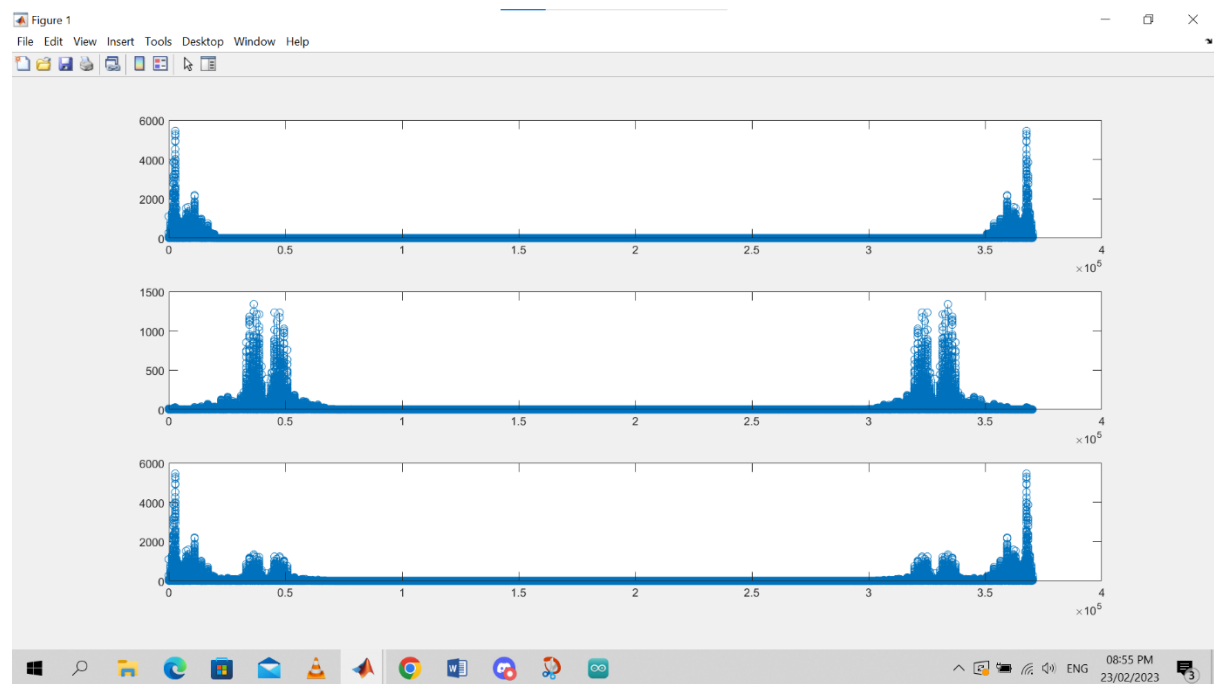
Codes:



```

1 clear all;
2 close all;
3
4 [s1,f1]=audioread('Guitar.wav');
5
6 [s2,f2]=audioread('Piano.wav');
7 fc1=5088;
8 fc2=4003;
9 t=[1:length(s1)]';
10 c1=cos(2*pi*fc1*t);
11 c2=cos(2*pi*fc2*t);
12 y1=s1.*c1;
13 y2=s2.*c2;
14 Y=y1+y2;
15
16 figure(1)
17 subplot(3,1)
18 stem(abs(fft(y1)))
19 subplot(3,1)
20 stem(abs(fft(y2)))
21 subplot(3,1)
22 stem(abs(fft(Y)))
23
24
  
```

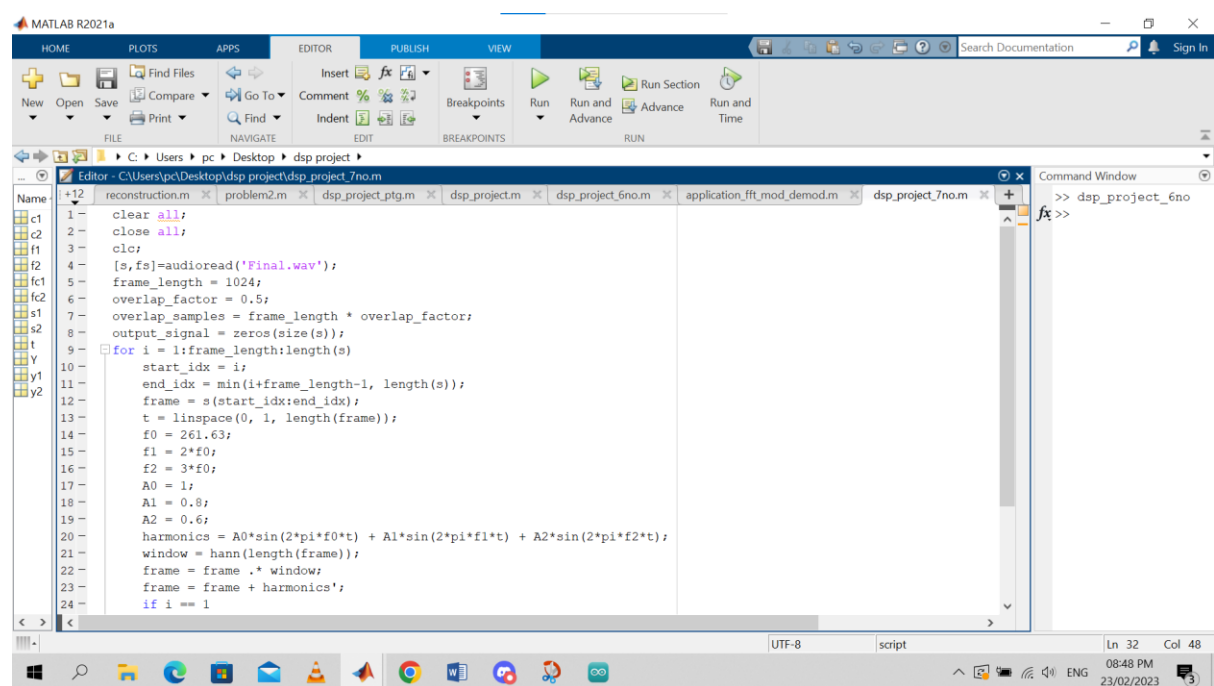
Graphs:



Here in question 6, 2 signals (the extracted signal of guitar and piano) is used as FDM inputs. The inputs are taken by audioread function. The message signals are modulated by 2 different frequencies. For time domain multiplication is done. For frequency domain fft is done. Finally the 2 signals are added and fft is done to see in frequency domain. Now this signal is ready to transmit to receiver side.

Question 7: The signal extracted from the 'final.wav' audio file lacks melody. Can you convert this into an overlapping yet melodious one with proper synchronization of octaves using MATLAB code?

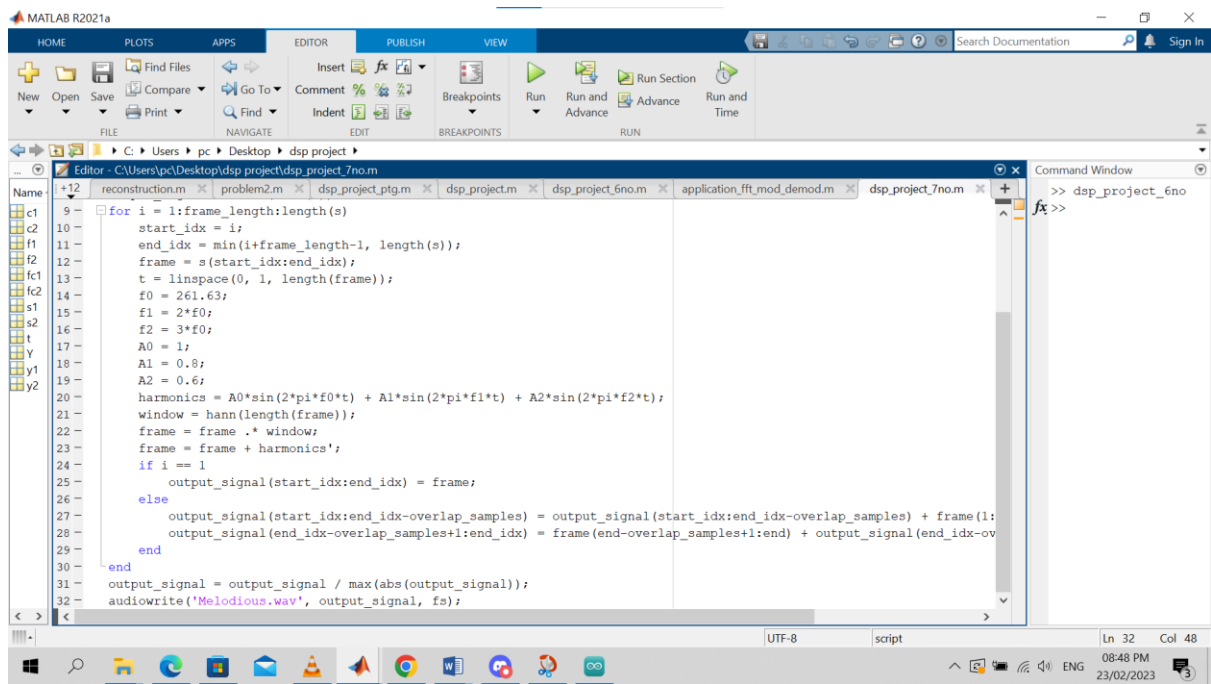
Code:



```
1 clear all;
2 close all;
3 clc;
4 [s,fs]=audioread('Final.wav');
5 frame_length = 1024;
6 overlap_factor = 0.5;
7 overlap_samples = frame_length * overlap_factor;
8 output_signal = zeros(size(s));
9 for i = 1:frame_length:length(s)
10     start_idx = i;
11     end_idx = min(i+frame_length-1, length(s));
12     frame = s(start_idx:end_idx);
13     t = linspace(0, 1, length(frame));
14     f0 = 261.63;
15     f1 = 2*f0;
16     f2 = 3*f0;
17     A0 = 1;
18     A1 = 0.8;
19     A2 = 0.6;
20     harmonics = A0*sin(2*pi*f0*t) + A1*sin(2*pi*f1*t) + A2*sin(2*pi*f2*t);
21     window = hann(length(frame));
22     frame = frame .* window;
23     frame = frame + harmonics;
24     if i == 1
```

Command Window

```
>> dsp_project_6no
```



The generated melodious signal is given in the zip file.