Lab Project Report

Course No: EEE 3218

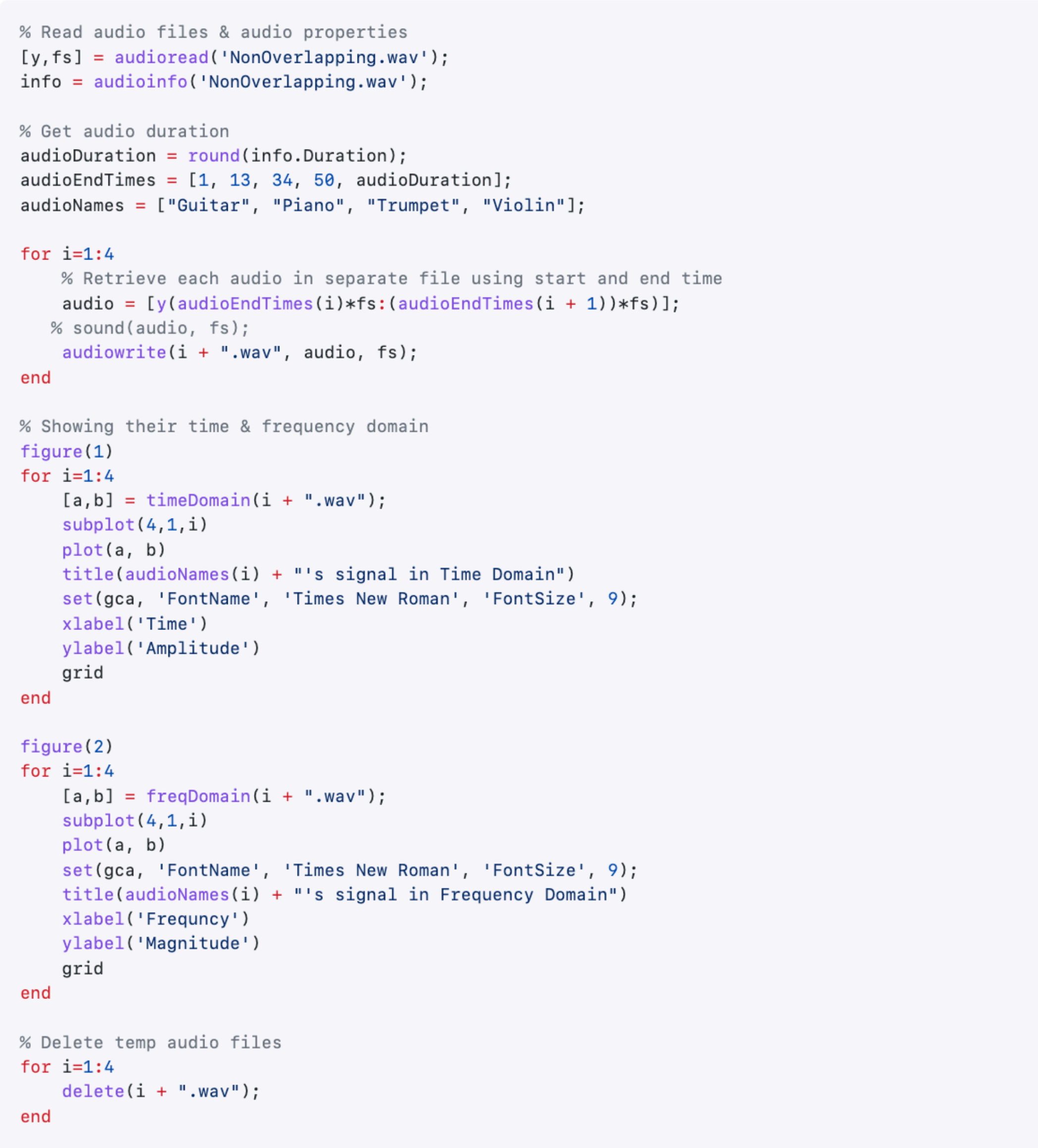
Name: Shahriar Nasim Nafi
ID: 190205154
Section: C
Group: C1
Year: 3rd **Semester**: 2nd

23 February, 2023

Github: https://github.com/SNNafi/dsp-project-eee-3218/

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

Answer: To get spectrum of each of the instrument's sound, we have first separated each instrument in its own *.wav* file.Then read them & plot their frequency & time domain using timeDomain & freqDomain function. These function are custom functions.



```
% Read audio files & audio properties
[y,fs] = audioread('NonOverlapping.wav');
info = audioinfo('NonOverlapping.wav');

% Get audio duration
audioDuration = round(info.Duration);
audioEndTimes = [1, 13, 34, 50, audioDuration];
audioNames = ["Guitar", "Piano", "Trumpet", "Violin"];

for i=1:4
    % Retrieve each audio in separate file using start and end time
    audio = [y(audioEndTimes(i)*fs:(audioEndTimes(i + 1))*fs)];
   % sound(audio, fs);
    audiowrite(i + ".wav", audio, fs);
end

% Showing their time & frequency domain
figure(1)
for i=1:4
    [a,b] = timeDomain(i + ".wav");
    subplot(4,1,i)
    plot(a, b)
    title(audioNames(i) + "'s signal in Time Domain")
    set(gca, 'FontName', 'Times New Roman', 'FontSize', 9);
    xlabel('Time')
    ylabel('Amplitude')
    grid
end

figure(2)
for i=1:4
    [a,b] = freqDomain(i + ".wav");
    subplot(4,1,i)
    plot(a, b)
    set(gca, 'FontName', 'Times New Roman', 'FontSize', 9);
    title(audioNames(i) + "'s signal in Frequency Domain")
    xlabel('Frequncy')
    ylabel('Magnitude')
    grid
end

% Delete temp audio files
for i=1:4
    delete(i + ".wav");
end
```

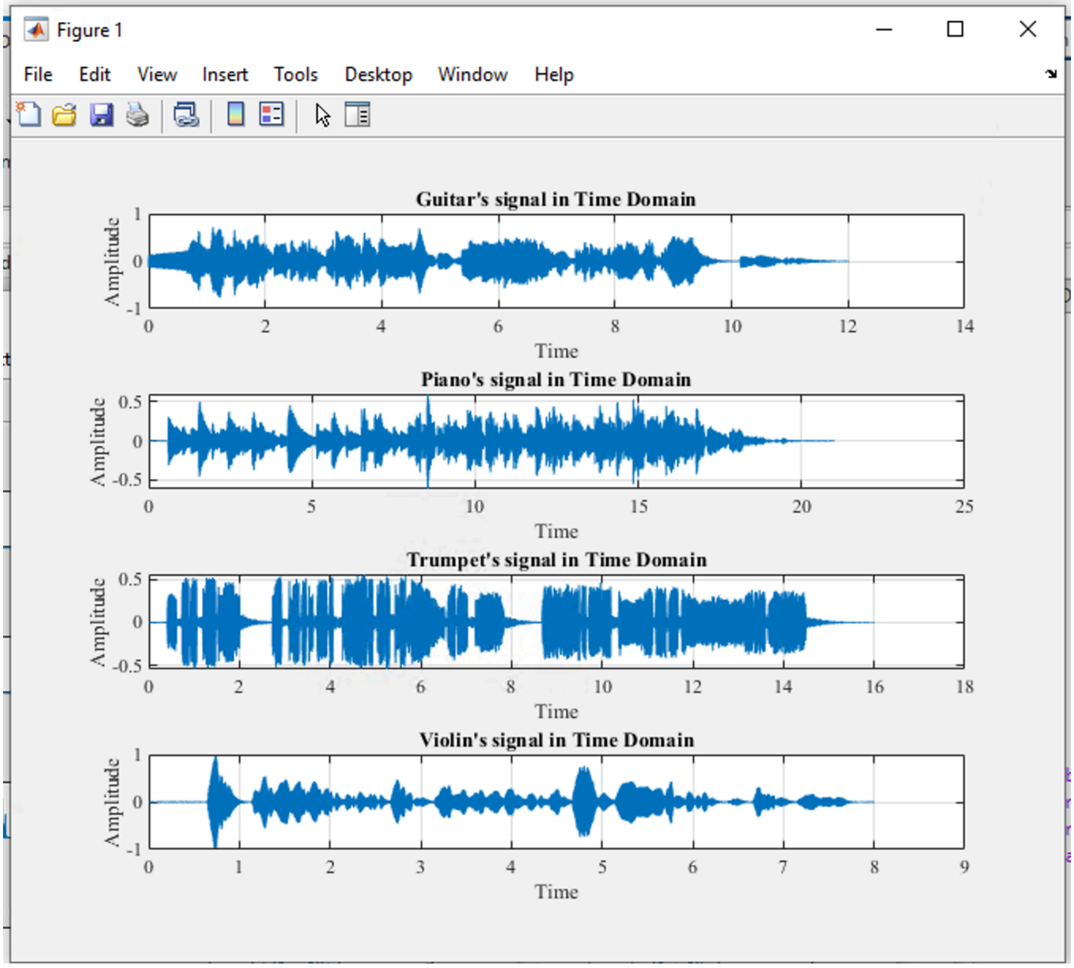


Figure 1
File Edit View Insert Tools Desktop Window Help
Guitar's signal in Time Domain
Amplitude
1
0
-1
0 2 4 6 8 10 12 14
Time
Piano's signal in Time Domain
Amplitude
0.5
0
-0.5
0 5 10 15 20 25
Time
Trumpet's signal in Time Domain
Amplitude
0.5
0
-0.5
0 2 4 6 8 10 12 14 16 18
Time
Violin's signal in Time Domain
Amplitude
1
0
-1
0 1 2 3 4 5 6 7 8 9
Time

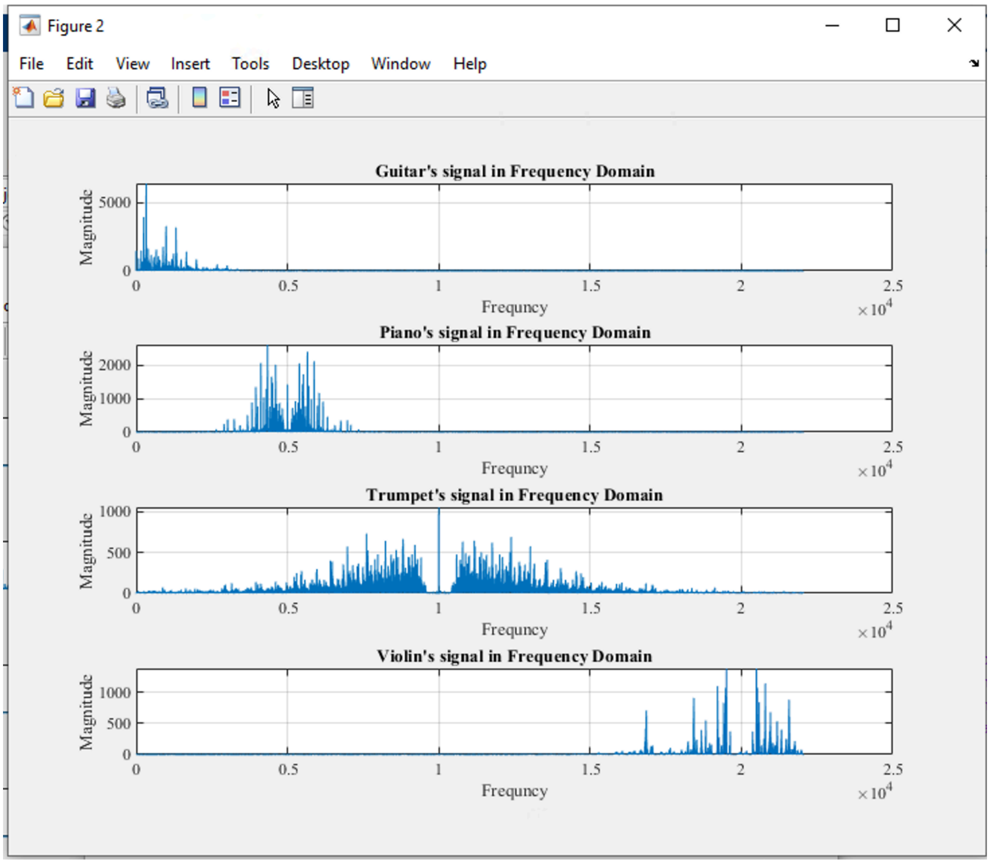
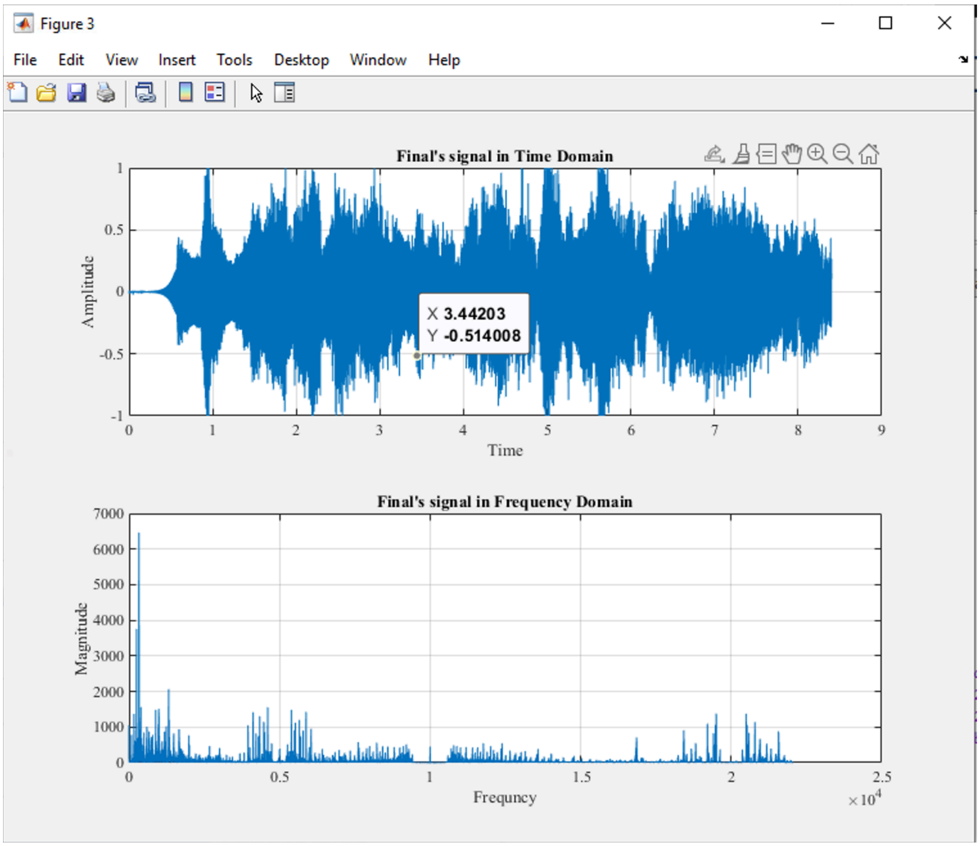


Figure 2
File Edit View Insert Tools Desktop Window Help
Guitar's signal in Frequency Domain
Magnitude
5000
0
0 0.5 1 1.5 2 2.5
Frequncy
$\times 10^4$
Piano's signal in Frequency Domain
Magnitude
2000
1000
0
0 0.5 1 1.5 2 2.5
Frequncy
$\times 10^4$
Trumpet's signal in Frequency Domain
Magnitude
1000
500
0
0 0.5 1 1.5 2 2.5
Frequncy
$\times 10^4$
Violin's signal in Frequency Domain
Magnitude
1000
500
0
0 0.5 1 1.5 2 2.5
Frequncy
$\times 10^4$

Question 2: Find the spectrum of the combined sound signal from ‘Final.wav’ file and determine the frequency range

Answer: To accomplish this, first we need to read audio by audioread function, them plot their frequency & time domain using timeDomain & freqDomain function. These function are custom functions.

```
[x1,y1] = timeDomain("Final.wav");
[x2,y2] = freqDomain("Final.wav");
figure(3)
subplot(2,1,1)
plot(x1, y1)
set(gca, 'FontName', 'Times New Roman', 'FontSize', 9);
title("Final's signal in Time Domain")
xlabel('Time')
ylabel('Amplitude')
grid
subplot(2,1,2)
plot(x2, y2)
title("Final's signal in Frequency Domain")
set(gca, 'FontName', 'Times New Roman', 'FontSize', 9);
xlabel('Frequncy')
ylabel('Magnitude')
grid
```



Question 3 & 4: 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. Extract each instrument sound in separate .wav file.

Answer: For doing we first need to design digital filter to get individual signal from the Final.wav.

These are four digital filter, has designed for extracting individual audio instrument's signal. There are two types of filter has used. Low Pass FilterBand Pass Filter.

Blackman has used as window for every filter.

Filter order has calculated using window.

For Guitar, have used low pass filter

Pass band frequency is 0

Stop band frequency is 2000.

So Transition width 2000 - 0 = 2000;

Cutoff frequency is (2000 + 2000) / 2 = 2000

For Piano, have used Band Pass filter. It has two cutoff frequency.

For first cutoff frequency,

Pass band frequency is 3500

Stop band frequency is 4800.

So Transition width 4800 - 3500 = 1300

And, Cutoff frequency is (3500 + 1300) / 2 = 2400

For second cutoff frequency,

Pass band frequency is 5100

Stop band frequency is 6300.

So Transition width 6300 - 5100 = 1200

And, Cutoff frequency is (5100 + 1200) / 2 = 3150

For Trumpet, have used Band Pass filter. It has two cutoff frequency.

For first cutoff frequency,

Pass band frequency is 7300

Stop band frequency is 9500.

So Transition width 9500 - 7300 = 2200

And, Cutoff frequency is (7300 + 2200) / 2 = 4750

For second cutoff frequency,

Pass band frequency is 10500

Stop band frequency is 14000.

So Transition width 14000 - 10500 = 3500

And, Cutoff frequency is (10500 + 3500) / 2 = 7000

For Violin, have used Band Pass filter. It has two cutoff frequency.

For first cutoff frequency,

Pass band frequency is 18000

Stop band frequency is 19500.

So Transition width 19500 - 18000 = 1500

And, Cutoff frequency is (18000 + 1500) / 2 = 9750

For second cutoff frequency,

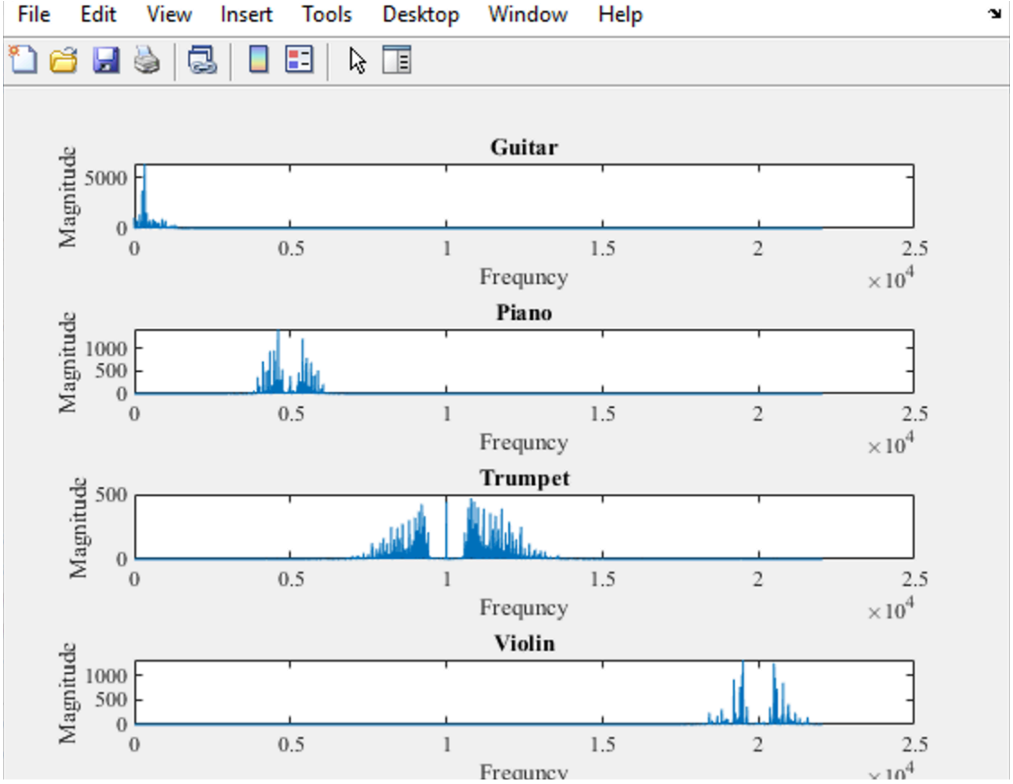
Pass band frequency is 20400

Stop band frequency is 21800.

So Transition width 21800 - 20400 = 1400

And, Cutoff frequency is (10500 + 1400) / 2 = 10900

The frequency domains of filtered audio signals'

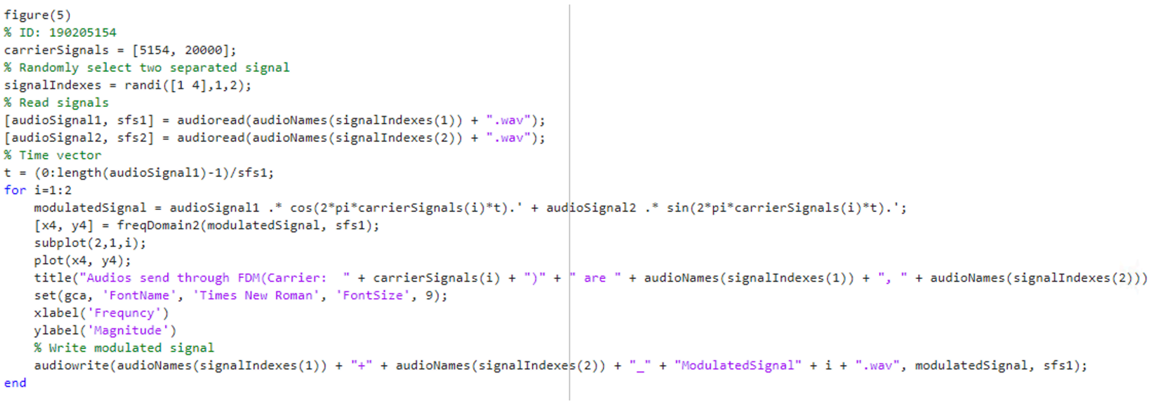


Question 5:Closely observe the spectrum of all 4 separated wav-files individually. Can you suggest any way to pass the individual wav files separately through a channel of bandwidth 0 to 10 kHz?

Answer: : Guitar, Piano and Trumpet can be passed through a channel of bandwidth **0** to **10** kHz. **BUT**, *Violin* can not pass because of its *high frequency*. By using **FDM** it can be passed by this channel.

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.

Answer:



```
figure(5)
% ID: 190205154
carrierSignals = [5154, 20000];
% Randomly select two separated signal
signalIndexes = randi([1 4],1,2);
% Read signals
[audioSignal1, sfs1] = audioread(audioNames(signalIndexes(1)) + ".wav");
[audioSignal2, sfs2] = audioread(audioNames(signalIndexes(2)) + ".wav");
% Time vector
t = (0:length(audioSignal1)-1)/sfs1;
for i=1:2
    modulatedSignal = audioSignal1 .* cos(2*pi*carrierSignals(i)*t).' + audioSignal2 .* sin(2*pi*carrierSignals(i)*t).';
    [x4, y4] = freqDomain2(modulatedSignal, sfs1);
    subplot(2,1,i);
    plot(x4, y4);
    title("Audios send through FDM(Carrier:  " + carrierSignals(i) + ")" + " are " + audioNames(signalIndexes(1)) + ", " + audioNames(signalIndexes(2)))
    set(gca, 'FontName', 'Times New Roman', 'FontSize', 9);
    xlabel('Frequncy')
    ylabel('Magnitude')
    % Write modulated signal
    audiowrite(audioNames(signalIndexes(1)) + "+" + audioNames(signalIndexes(2)) + "_" + "ModulatedSignal" + i + ".wav", modulatedSignal, sfs1);
end
```

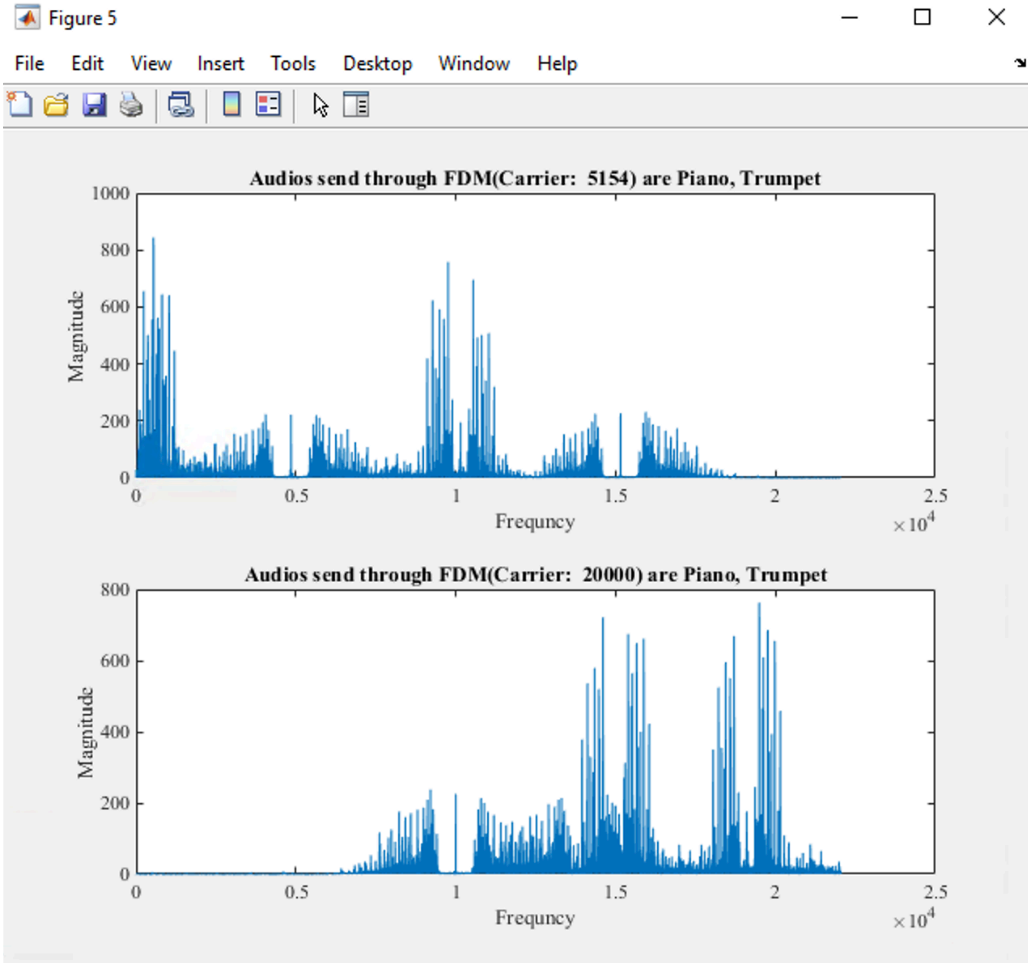
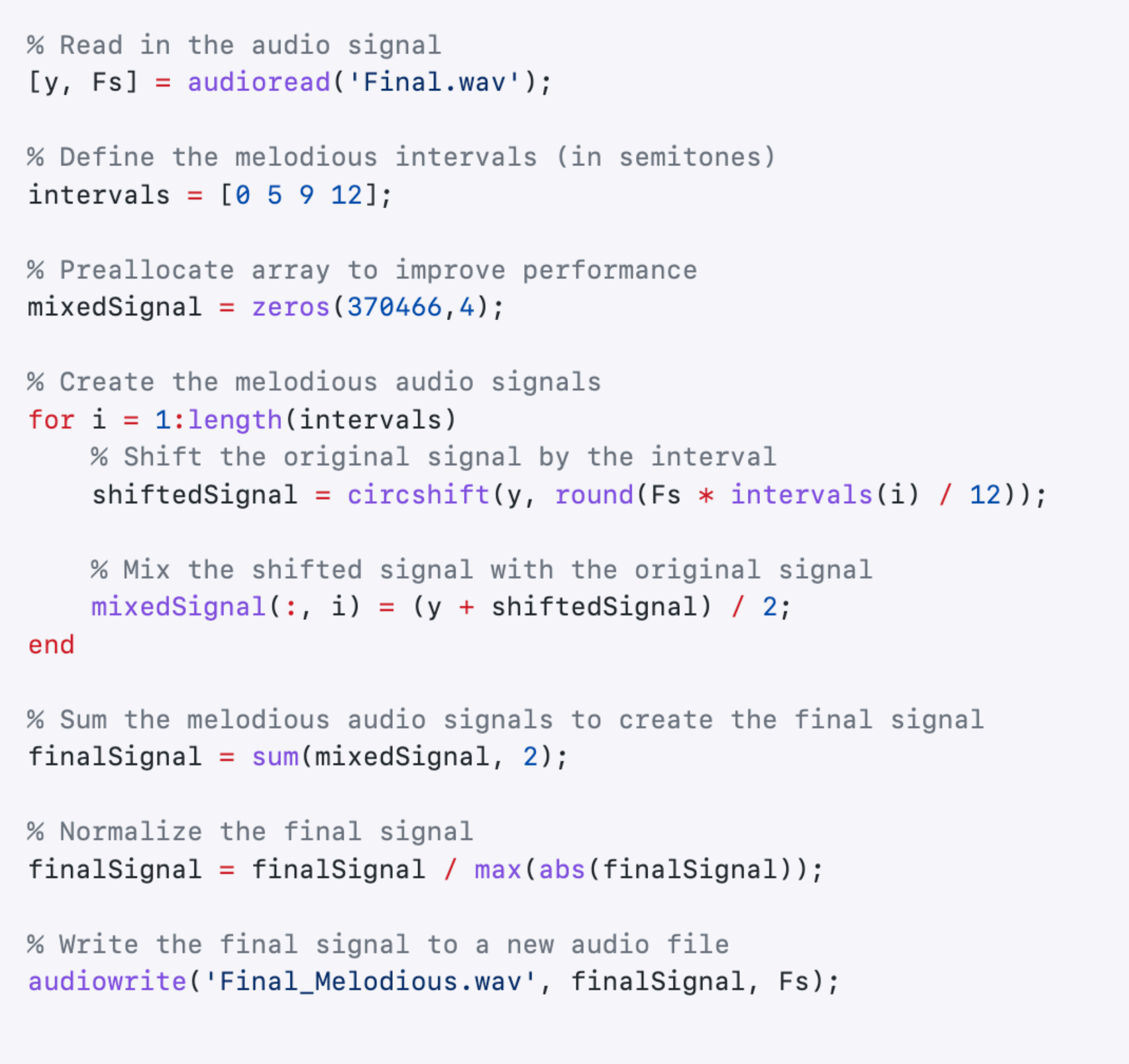


Figure 5
File Edit View Insert Tools Desktop Window Help
Audios send through FDM(Carrier: 5154) are Piano, Trumpet
Magnitude
1000
800
600
400
200
0
0
0.5
1
1.5
2
2.5
Frequncy
$\times 10^4$
Audios send through FDM(Carrier: 20000) are Piano, Trumpet
Magnitude
800
600
400
200
0
0
0.5
1
1.5
2
2.5
Frequncy
$\times 10^4$

Question 7: 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.

Answer:



```
% Read in the audio signal
[y, Fs] = audioread('Final.wav');

% Define the melodious intervals (in semitones)
intervals = [0 5 9 12];

% Preallocate array to improve performance
mixedSignal = zeros(370466,4);

% Create the melodious audio signals
for i = 1:length(intervals)
    % Shift the original signal by the interval
    shiftedSignal = circshift(y, round(Fs * intervals(i) / 12));

    % Mix the shifted signal with the original signal
    mixedSignal(:, i) = (y + shiftedSignal) / 2;
end

% Sum the melodious audio signals to create the final signal
finalSignal = sum(mixedSignal, 2);

% Normalize the final signal
finalSignal = finalSignal / max(abs(finalSignal));

% Write the final signal to a new audio file
audiowrite('Final_Melodious.wav', finalSignal, Fs);
```