Lab 5

Sound Signals

EECS3451

Name: Chethana Wickramasinghe

Student Number: 214866511

Professor: Peter Lian

27th of March 2021

1. Introduction:

Using MATLAB to answer questions provided. Questions leads to reading, writing and playing sounds to get an understanding on signal sampling and the effect of sampling frequency. This report is mainly answering the provided questions using MATLAB. Results will demonstrate the use of MATLAB properly in reading, writing, and playing audio files.

2. Equipment: MATLAB

3. Results and discussion:

Q1. Study the following Matlab code:

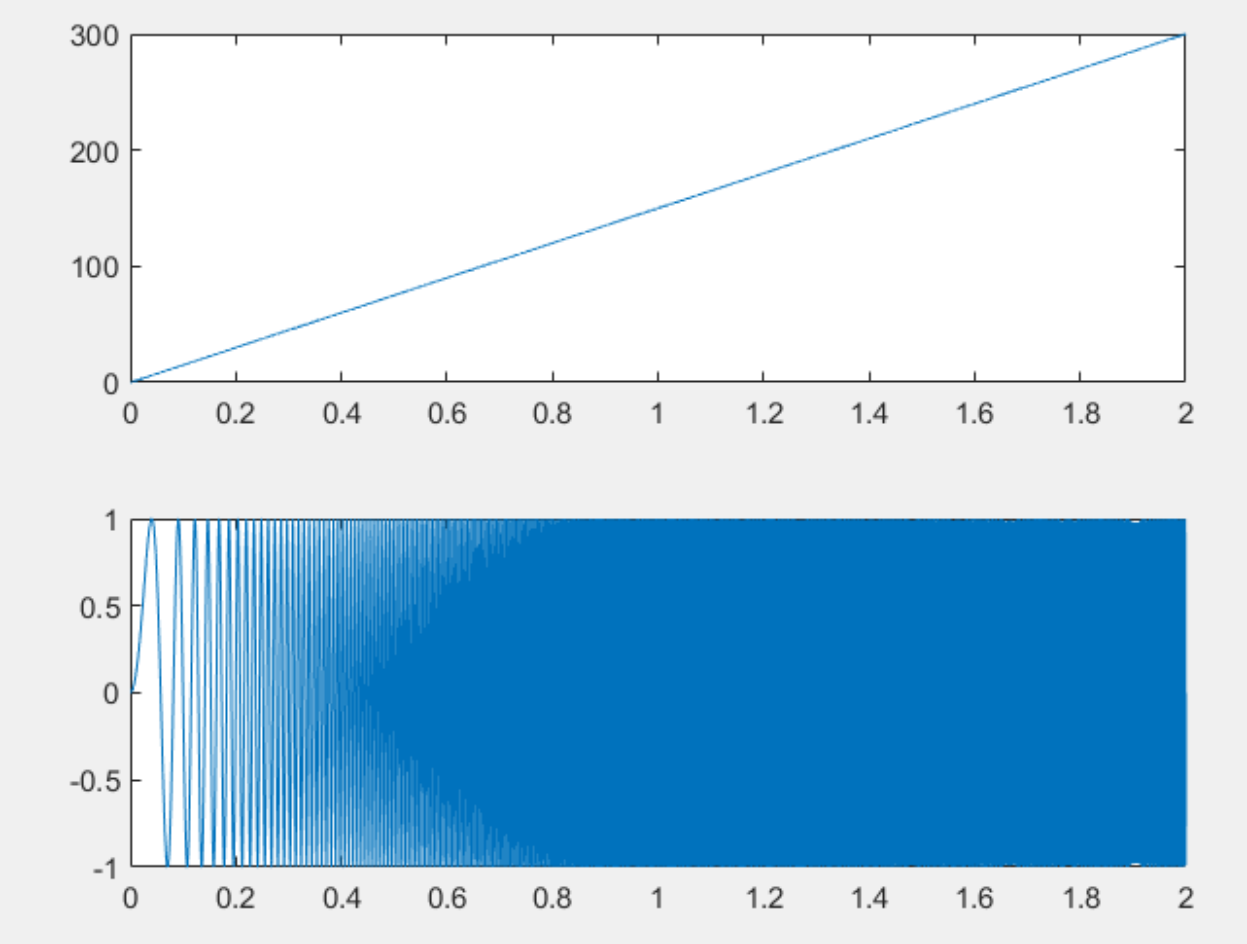
```
Clear all
a=0:(1/8000):2;
f=150*a;
x=sin(2*pi*f.*a);
subplot(2,1,1), plot(a,f)
subplot(2,1,2), plot(a,x)
audiowrite('sound1.wav',x,8000);
sound(x,8000);
```

(1) Based on your understanding of the code, how many samples are there in 1 second, or what is the sampling frequency?

8000Hz

(2) Listen to the generated sound. Explain why it sounds in the way it does

According to this plot



and the sound. It seems like the frequency of the input signal increase over time. In other words, the pitch of the sound increase.

Q2. The keys of a piano generate musical notes at given frequencies. For a list of these frequencies, check https://en.wikipedia.org/wiki/Piano_key_frequencies.

(1) Write a Matlab code to generate a sine wave over the C major scale (key numbers: 40,42,44,45,47,49,51,52), each note lasting **0.5 seconds**, with a sampling frequency of **8000Hz.** Save the data to a WAV file called Cscale.wav.

40 = 261.6256,42=293.6648,44=329.6276,45=349.2282,

47=391.9954,49=440.0000,51=493.8833,52=523.2511

```
t=0:(1/8000):0.5;
myVector = [sin(262*2*pi*t) sin(294*2*pi*t) sin(330*2*pi*t)
sin(349*2*pi*t) sin(392*2*pi*t) sin(440*2*pi*t)
sin(494*2*pi*t) sin(523*2*pi*t)];
```

```
audiowrite('Cscale.wav',myVector,8000);
sound(myVector,8000);
```

(2) Listen to the Cscale.wav to make sure you generate a correct C major scale.

It sound like the C major scale C D E F G A B C when I play it the keyboard.

(3) Read the C major scale to the vector Y. Listen to the C major scale with different sampling frequencies of 4000Hz, 8000Hz, and 16000Hz. Do they sound same? Why?

```
[y,fs]=audioread('Cscale.wav');
sound(y,fs);

% below sounds don't sound the same
sound(y,4000); % sound low pitch and slower rate than
fs=8000 because it collects less samples per sec and
captures less cycles per second
sound(y,16000); % sound high pitch and faster rate than
fs=8000 because it collects more samples per sec and
captures more cycles per second
```

(4) Read the C major scale to Y, and write a Matlab code to generate Z[k]=Y[-k] and write Z to a WAV file, CscaleZ.wav. Listen to CscaleZ.wav. How CscaleZ.wav different from Cscale.wav? (Please save Cscale.wav and CscaleZ.wav, these files are required in Lab6.)

Assuming this means reverse order of the scale

```
[y,fs]=audioread('Cscale.wav');
z = zeros(1,length(y));
z = flipud(y)
audiowrite('CscaleZ.wav',z,fs);

% reverse scale C B A G F E D C from high to low frequency
sound
CscaleZ.wav sound start from high and gets low, while
Cscale.wav sound start low and gets high
```

Q3. In Matlab, type "help audiowrite". You will find the following explanations about the valid range of the data in Y:

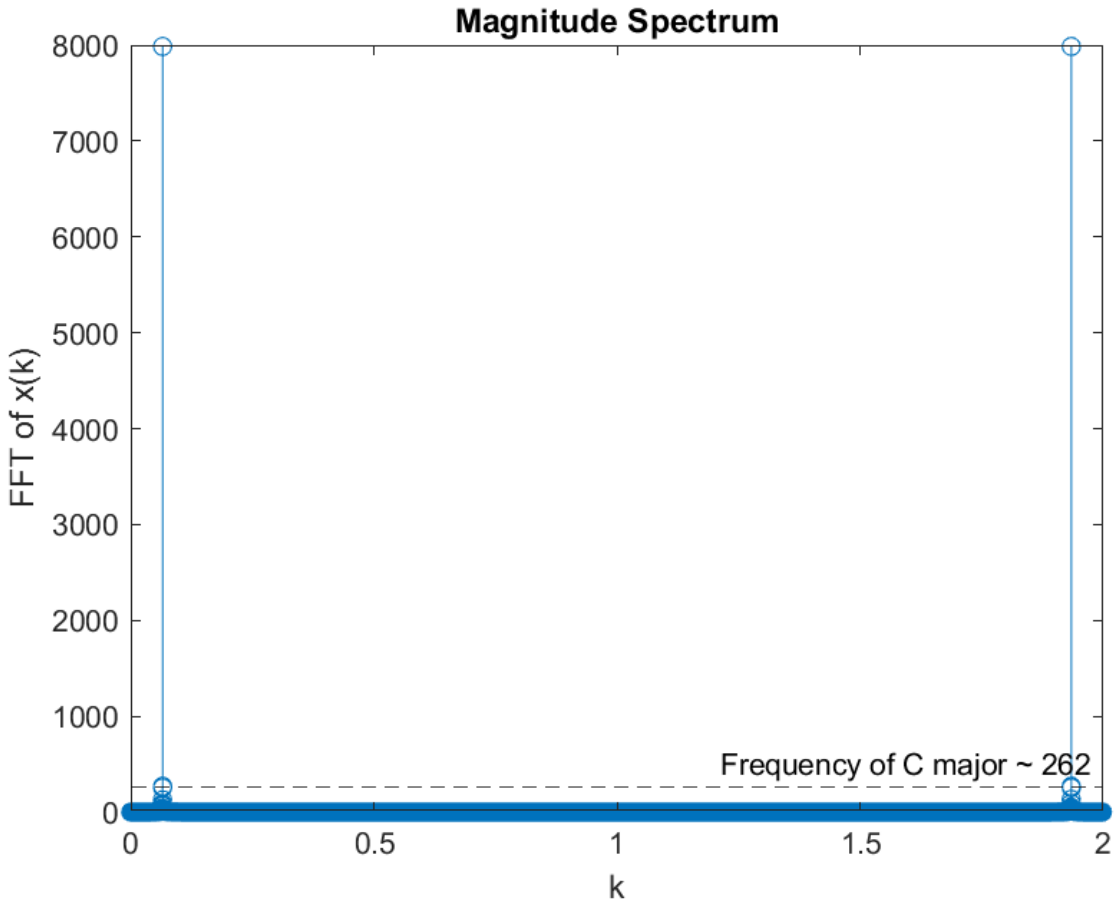
The valid range for the data in Y depends on the data type of Y. Supported data types are as follows:

Data Type of Y	Valid Range for Y
uint8	0 <= Y <= 255
int16	-32768 <= Y <= +32767
int32	-2^32 <= Y <= 2^32-1
single	-1.0 <= Y <= +1.0
double	-1.0 <= Y <= +1.0

Data beyond the valid range is clipped. If Y is single or double, then audio data in Y should be normalized to values in the range -1.0 and 1.0, inclusive.

(1) Write a Matlab code to generate a **C major signal (key number 40)** x(k) with the amplitude in the range **-1 to 1**, and sampling frequency of **8000Hz**. Save the data to a WAV file, **Cmajor1.wav**, take the FFT of x(k) and plot its magnitude response to verify its frequency is the same as the key number 40, play the signal x(k) to verify that it sounds like a C major.

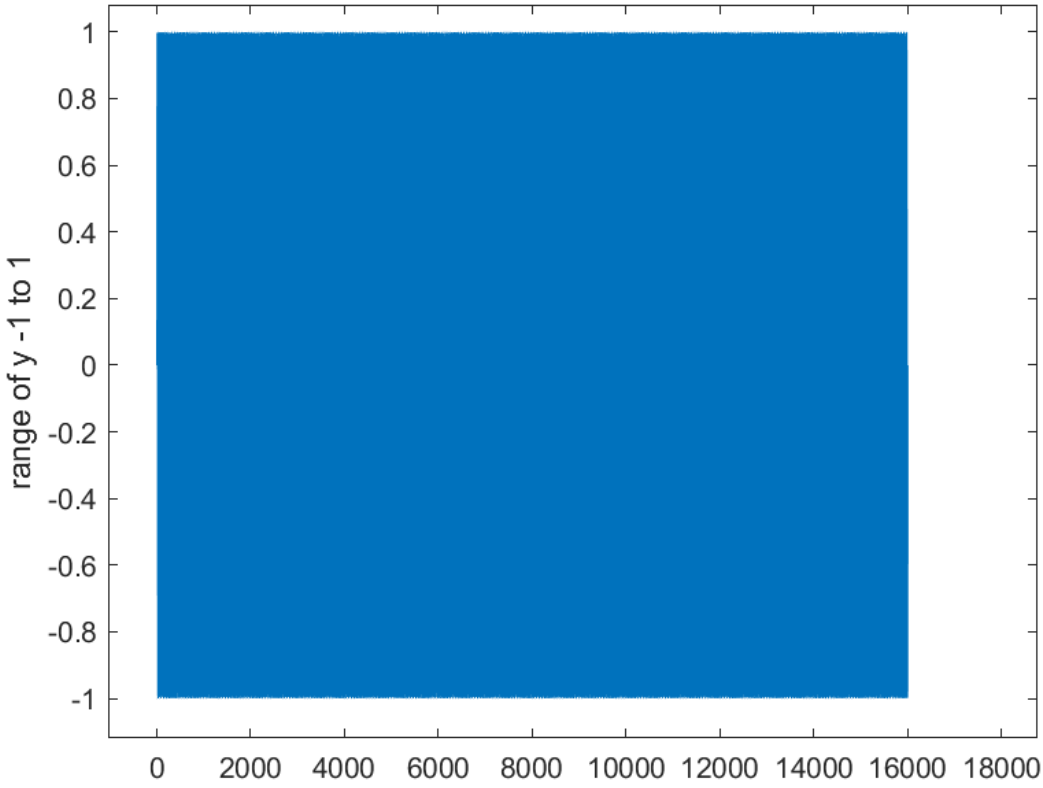
```
t=0:(1/8000):2;
f=262;
xk=sin(2*pi*f.*t);
subplot(2,1,1), plot(t,f)
subplot(2,1,2), plot(t,xk)
audiowrite('Cmajor1.wav',xk,8000);
sound(xk,8000);

% FFT of x(k)
fftxk = fft(xk);

tt =0:0.01:5;
% plot the magnitude response and verify the frequency
stem(t,abs(fftxk))
xlabel('k');
ylabel('FFT of x(k)');
title('Magnitude Spectrum');
yline(262,'--k','Frequency of C major')
```



(2) Read in the Cmajor1.wav file using audioread, examine what is the valid range of the data.

```
[y,fs]=audioread('Cmajor1.wav');
audioinfo('Cmajor1.wav')

% Y is single or double because they are normalized and in
the range of -1 and 1
% proof by
```

```
plot(y)
```



(3) Change the program in (1) to generate the same C major signal with peak magnitude of 20, i.e. in the range of **-20 to 20**, save the data to a WAV file, Cmajor2.wav. Do you receive any **warning** or error message?

```
t=0:(1/8000):2;
f=262;
xk=20*sin(2*pi*f.*t);
subplot(2,1,1), plot(t,f)
subplot(2,1,2), plot(t,xk)
audiowrite('Cmajor2.wav',xk,8000);


% % Yes warning
% Warning: Data clipped when writing file.
% > In audiowrite>clipInputData (line 470)
% In audiowrite (line 241)
```

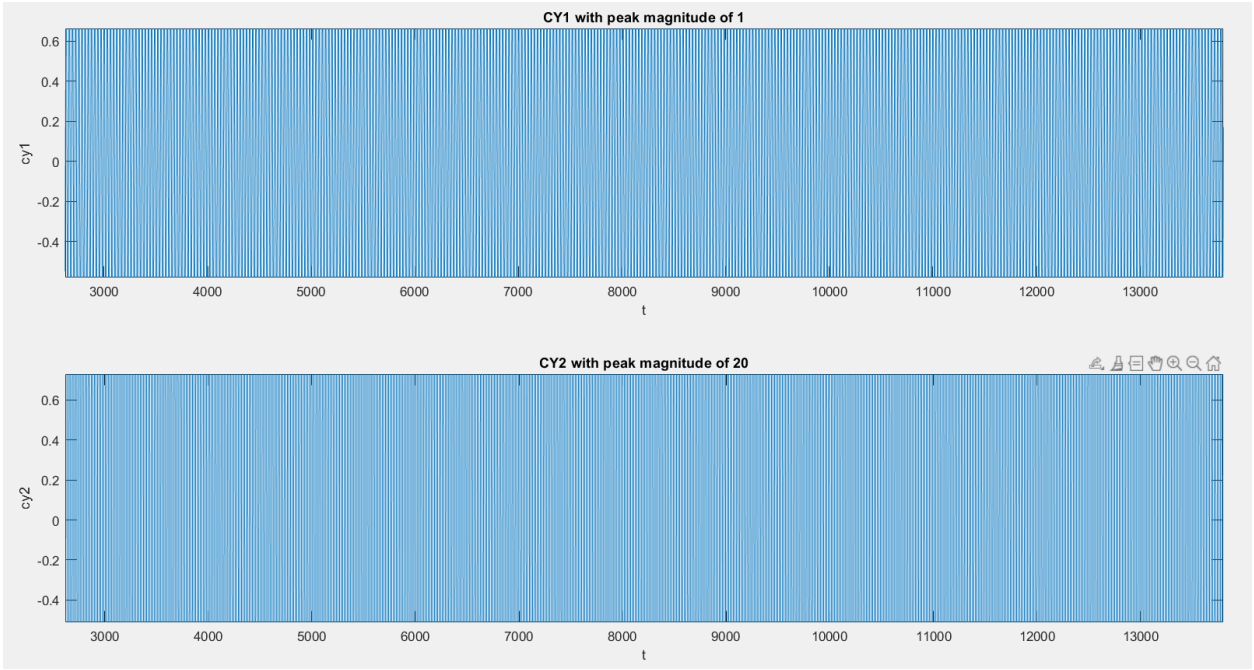
(4) Read Cmajor1.wav and Cmajor2.wav to vectors CY1 and CY2. Listen to both CY1 and CY2. Do they sound the same? Explain what happened to data CY2 by plotting both waveforms.

```
[cy1,fs1]=audioread('Cmajor1.wav');
```

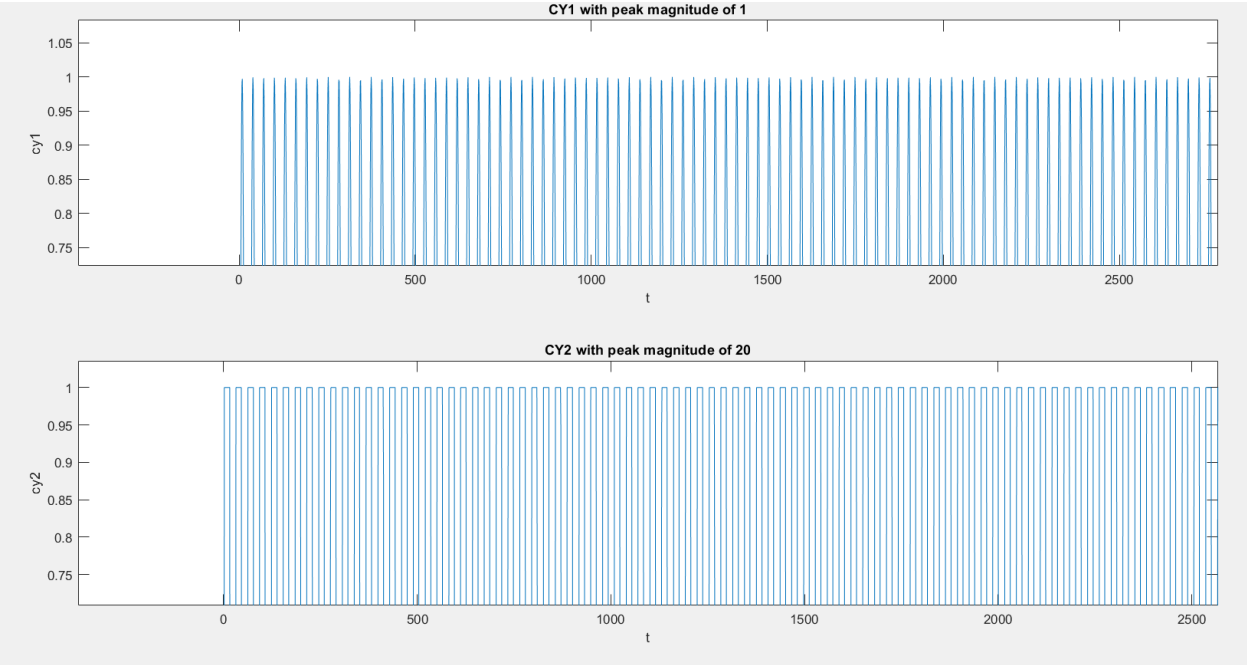
```
[cy2,fs2]=audioread('Cmajor2.wav');

sound(cy1,fs1);
sound(cy2,fs2);
```

Plotting

```
%-----------
t=0:(1/8000):2;
subplot(2,1,1), plot(cy1)
xlabel('t');
ylabel('cy1');
title('CY1 with peak magnitude of 1');
subplot(2,1,2), plot(cy2)
xlabel('t');
ylabel('cy2');
title('CY2 with peak magnitude of 20');
%-----------
% different sounds
% in cy2 all the values greater than 1 and less than -1 is
cut and that
% explained the warning given when saving into a wav file
```



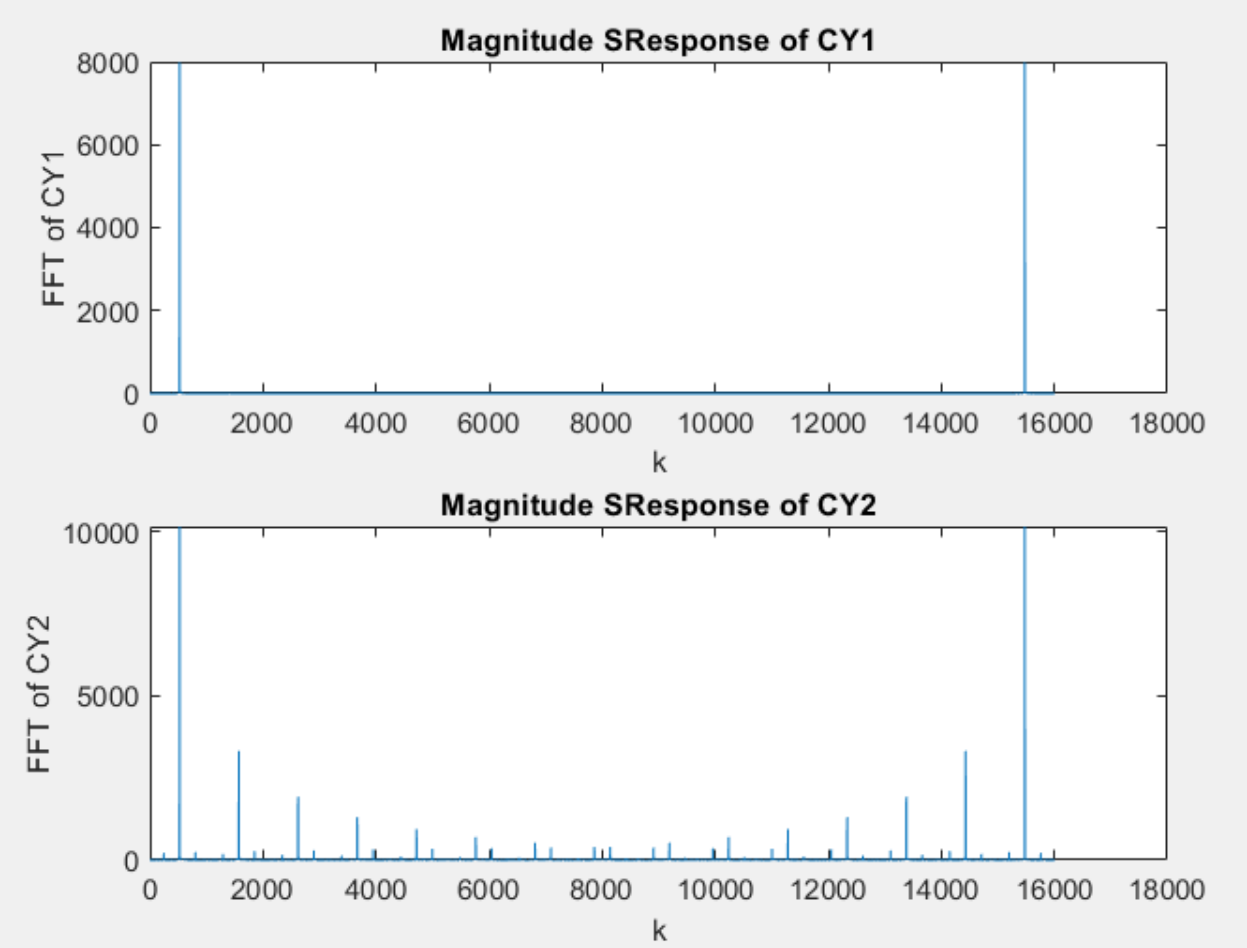
Zoomed in



Cy2 is almost like a square wave and we know square waves are generally made using many frequencies adding up together.

(5) Take the FFT of both CY1 and CY2 and plot the magnitude responses of CY1 and CY2. You can do this by using Matlab command: y=fft(x), and plot(abs(y)). What is your observation?

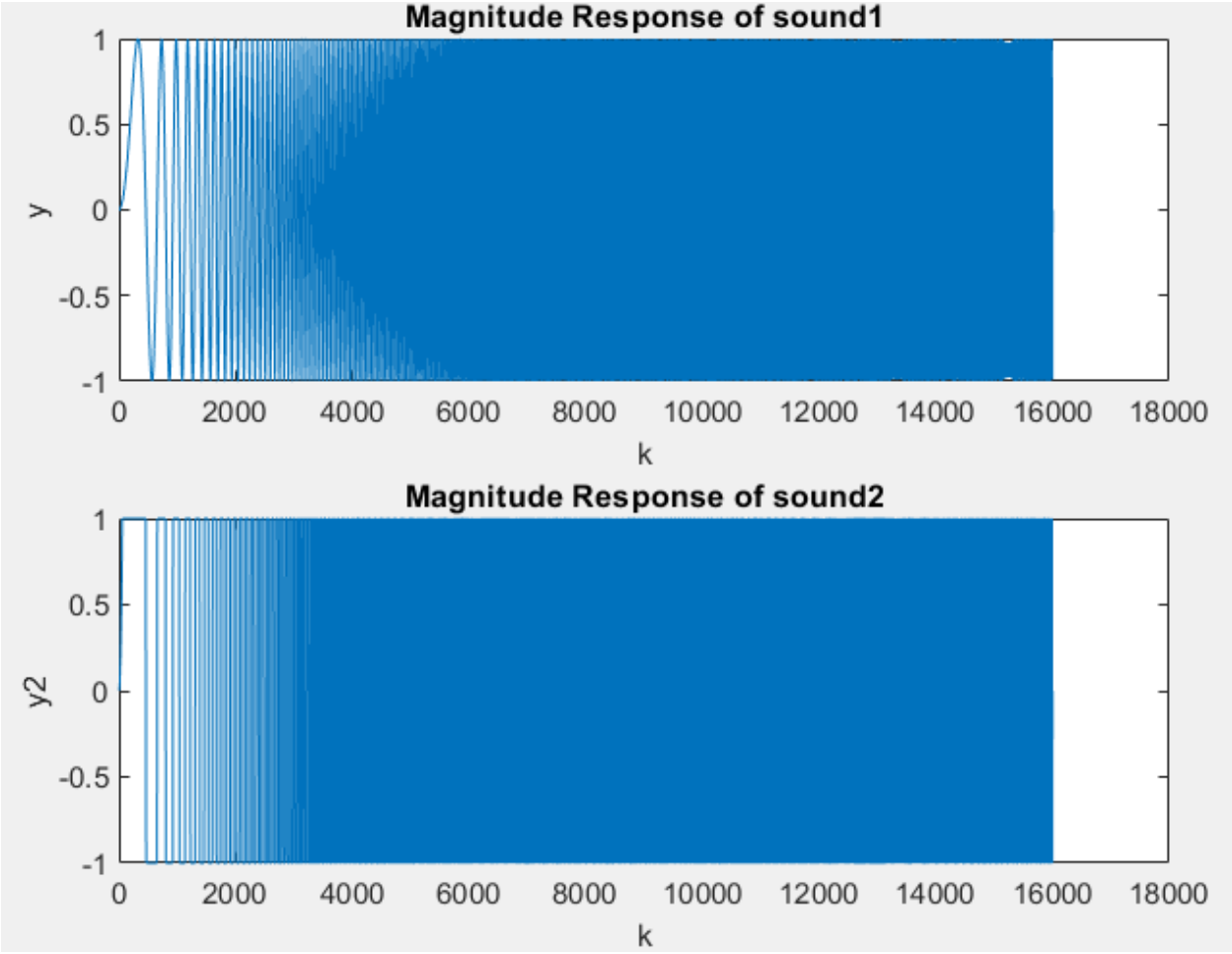
```
subplot(2,1,1),plot(abs(fft(cy1)))
xlabel('k');
ylabel('FFT of CY1');
title('Magnitude Response of CY1');

subplot(2,1,2),plot(abs(fft(cy2)))
xlabel('k');
ylabel('FFT of CY2');
title('Magnitude Response of CY2');

% % Observation : cy2 has multiple small spikes between  as
shown below
```



(6) Read in the sound1.wav in Q1, change the peak **magnitude to 20** and save it to sound2.wav. Do you receive any **warning** or error message? Listen to the both sound1.wav and sound2.wav. Do they sound the same? Combining the observation you made in (4) and (5), please explain what happened to sound2.wav

```
[y,fs]=audioread('sound1.wav');
audiowrite('sound2.wav',20*y,fs);

% % Yes, warning
% Warning: Data clipped when writing file.
% > In audiowrite>clipInputData (line 470)
% In audiowrite (line 241)

[y2,fs2]=audioread('sound2.wav');
sound(y,fs);
sound(y2,fs2);
```

```
% They don't sound the same. Sound2 seems to have a wavy
effect, where Sound1 is smooth

subplot(2,1,1),plot(y)
xlabel('k');
ylabel('y');
title('Magnitude Response of sound1');
subplot(2,1,2),plot(y2)
xlabel('k');
ylabel('y2');
title('Magnitude Response of sound2');
```



According to the observations from 4) and 5) the Sound2 behave as such because its almost like a square wave once its magnitude above 1 and below 1 is trimmed. By definition "A square wave can be expressed as a combination of a basic sine wave of same frequency plus other sine waves of higher frequencies of odd number"[1], therefore the wavy sound we hear in sound2.wav is because of multiple frequencies in it, where as sound1.wave consist of only 1 frequency and sound smooth through the time its played.

4. Conclusion: state what you learn from this lab, lab objectives you achieved, and any difficulties you met.

Learned how to use audiowrite(), audioread(), sound(), and audioinfo() commands in MATLAB to investigate signal sampling and the effect of sampling frequency.

All the questions were answered using MATLAB and was able to get a better understanding of signal sampling.

Had to spent time in understanding what the sampling does to sound2.wav to sound as it does.

This lab was one of the most interesting labs ever. I learned a lot on how signal sampling work and how they might be using in the industry.

References:

[1] “Frequency of square wave,” *Electrical Engineering Stack Exchange*, 01-Jul-1963. [Online]. Available: https://electronics.stackexchange.com/questions/130458/frequency-of-square-wave. [Accessed: 26-Mar-2021].