Lab 6

Discrete Fourier Transform

EECS3451

Name: Chethana Wickramasinghe

Student Number: 214866511

Professor: Peter Lian

8th of April 2021

Name: Chethana Wickramasinghe   Student #: 214866511

1. Introduction:

Using MATLAB to answer questions provided. Questions leads to verifying the properties of DFT, using the convolution to calculate the output of the LTI DT system, and using DFT and IDFT to find the output of an LTI DT system. This report is mainly answering the provided questions using MATLAB. Results will demonstrate the use of MATLAB properly in analysing applications using Discrete Fourier Transform.

2. Equipment: MATLAB

3. Results and discussion:

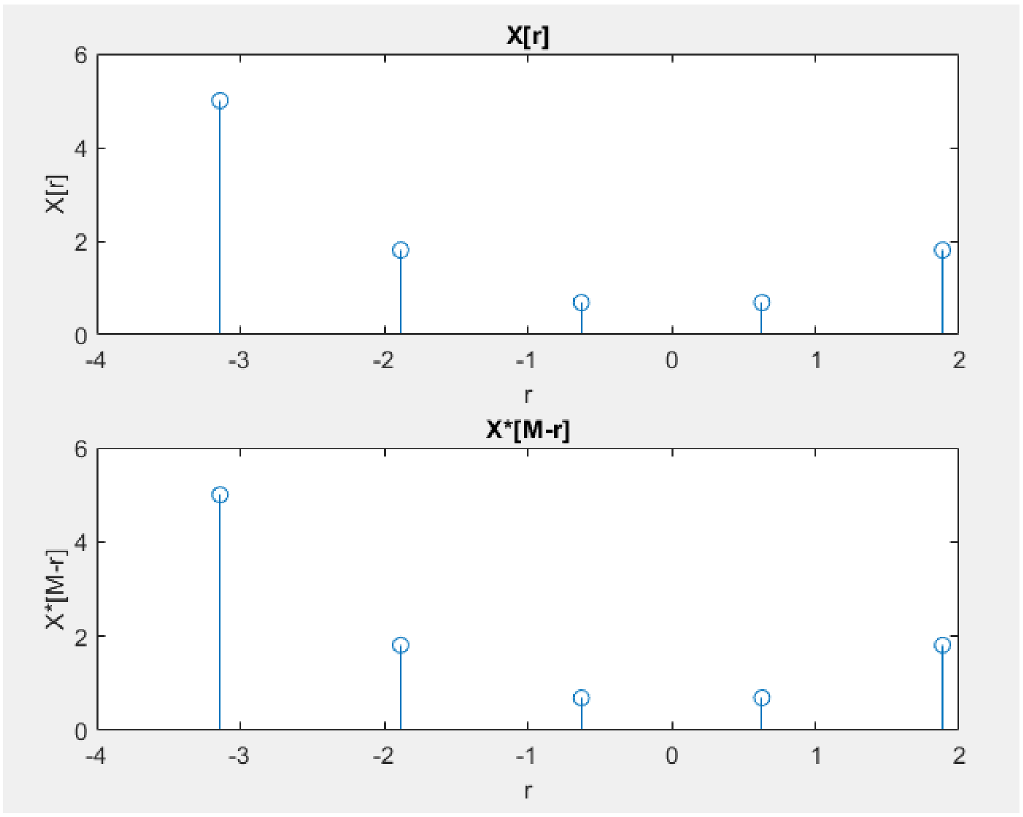
Q1. Given a DT sequence $x=\{2,1,1,0,1\}$, verify the following DFT properties. You can set the size of DFT the same as the DT sequence. (Hint: you may use Matlab commands fftshift, abs, angle, and stem.)

(a) $X[r]=X^*[M\text{-}r]$
(b) $|X[r]|=|X[M\text{-}r]|$
(c) $\angle X[r] = -\angle X[M-r]$

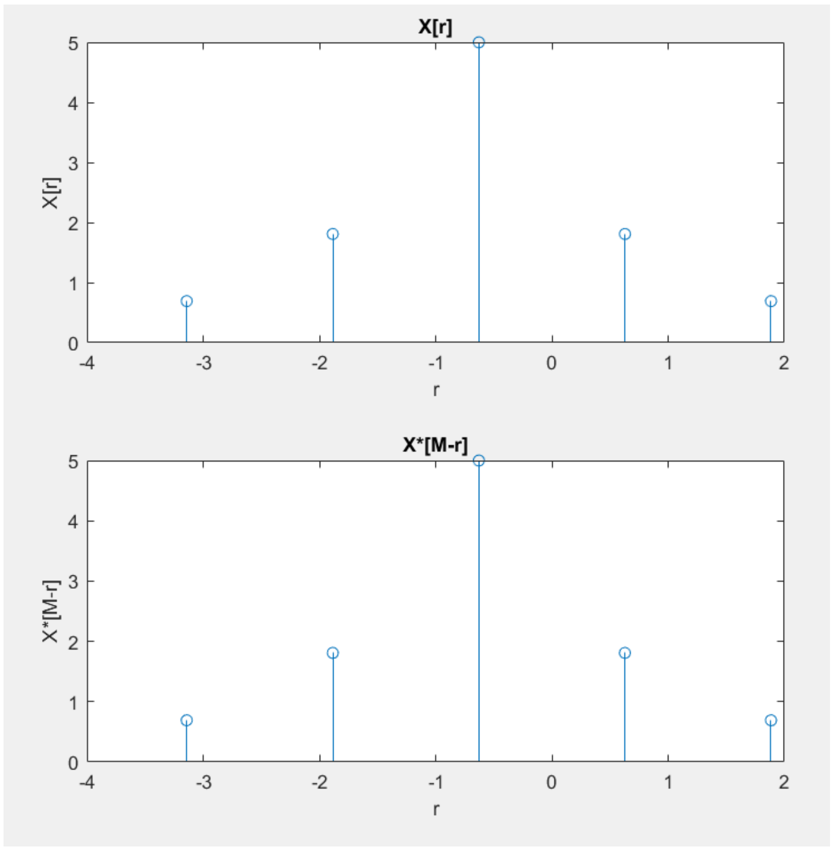
```
N=5; k = 0:N-1;
xk= [2 1 1 0 1];
```

a)

```
% X(r)
Xr = fft(xk);
w = -pi:2*pi/N:pi-2*pi/N;
subplot(2,1,1),stem(w, Xr);
xlabel('r');
ylabel('X[r]');
title('X[r]');
% X*[M-r] conjucate of complex x+yj -> x-yj and vise versa
cXr = conj(Xr);
subplot(2,1,2),stem(w, cXr);
xlabel('r');
ylabel('X*[M-r]');
title('X*[M-r]');
%they are the same
```

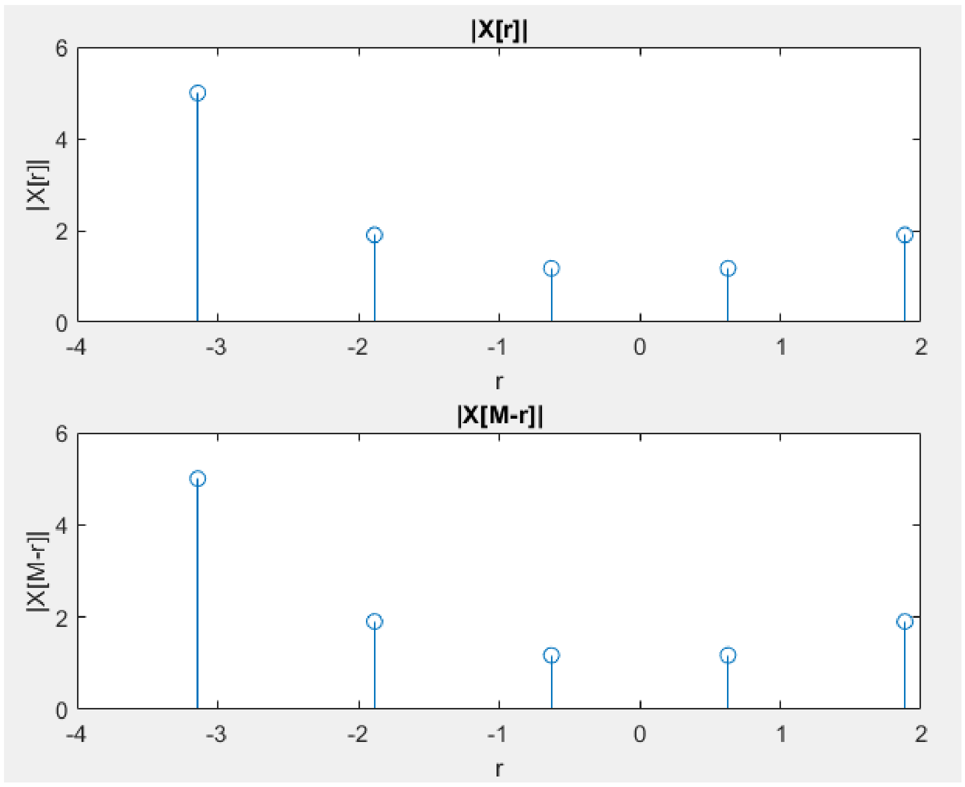


```
%Shifted for better observation purpose
Xr = fftshift(fft(xk));
w = -pi:2*pi/N:pi-2*pi/N;
subplot(2,1,1),stem(w, Xr);
xlabel('r');
ylabel('X[r]');
title('X[r]');
% X*[M-r] conjucate of complex x+yj -> x-yj and vise versa
cXr = conj(Xr);
subplot(2,1,2),stem(w, cXr);
xlabel('r');
ylabel('X*[M-r]');
title('X*[M-r]');
%they are the same
```



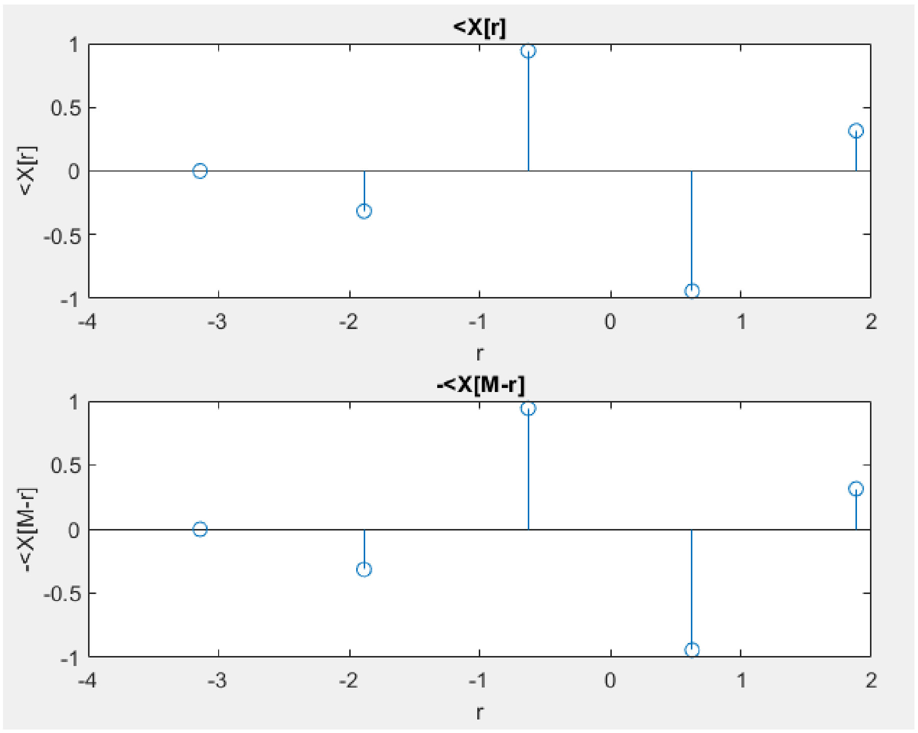
b)

```
subplot(2,1,1),stem(w, abs(Xr));
xlabel('r');
ylabel('|X[r]|');
title('|X[r]|');
subplot(2,1,2),stem(w, abs(cXr));
xlabel('r');
ylabel('|X[M-r]|');
title('|X[M-r]|');
%they are the same
```



c)

```
subplot(2,1,1),stem(w, angle(Xr));
xlabel('r');
ylabel('<X[r]');
title('<X[r]');
subplot(2,1,2),stem(w, -angle(cXr));
xlabel('r');
ylabel('-<X[M-r]');
title('-<X[M-r]');
%they are the same
```



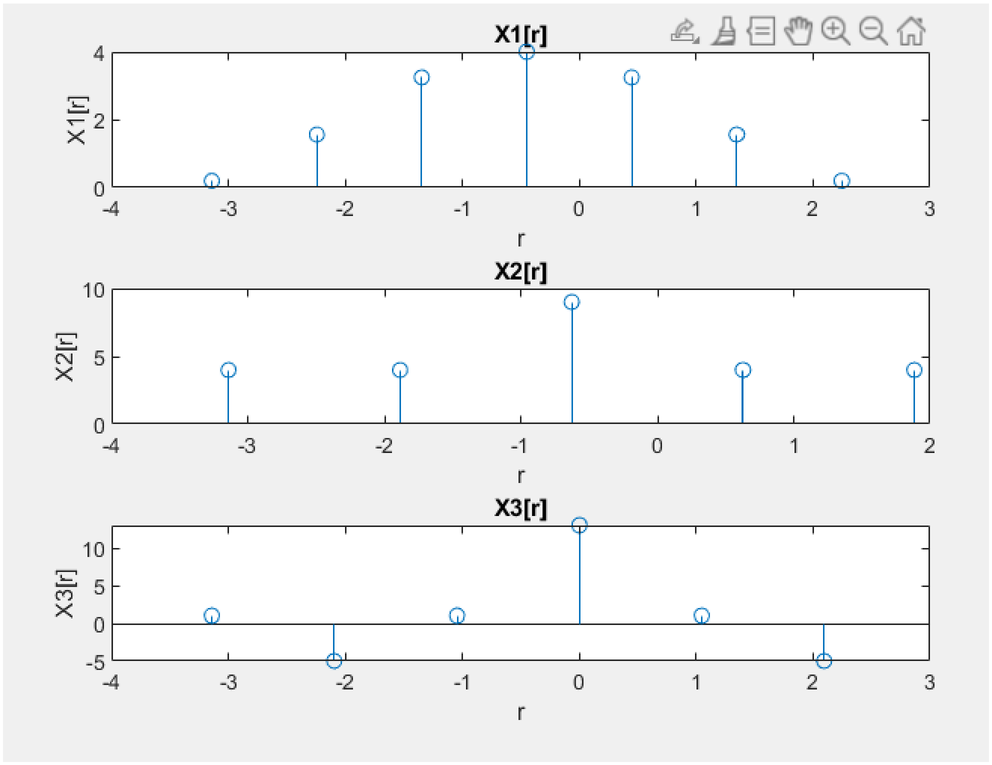
Q2. Use Matlab function FFT to find the DFT of following sequences. Set the number of points of DFT the same as DT sequences.

(a). $x1=\{2,1,0,0,0,0,1\}$
(b). $x2=\{5,1,1,1,1\}$
(c). $x3=\{1,3,3,0,3,3\}$

i) What is your observation on DFTs of x1, x2, and x3?

```
N1=7; k1=0:N1-1;
x1k = [2 1 0 0 0 0 1];
X1r = fft(x1k);
X1r = fftshift(X1r);
w1 = -pi:2*pi/N1:pi-2*pi/N1;
subplot(3,1,1),stem(w1, X1r);
xlabel('r');
ylabel('X1[r]');
title('X1[r]');

N2=5; k2=0:N2-1;
x2k = [5 1 1 1 1];
X2r = fft(x2k);
X2r = fftshift(X2r);
w2 = -pi:2*pi/N2:pi-2*pi/N2;
subplot(3,1,2),stem(w2, X2r);
```

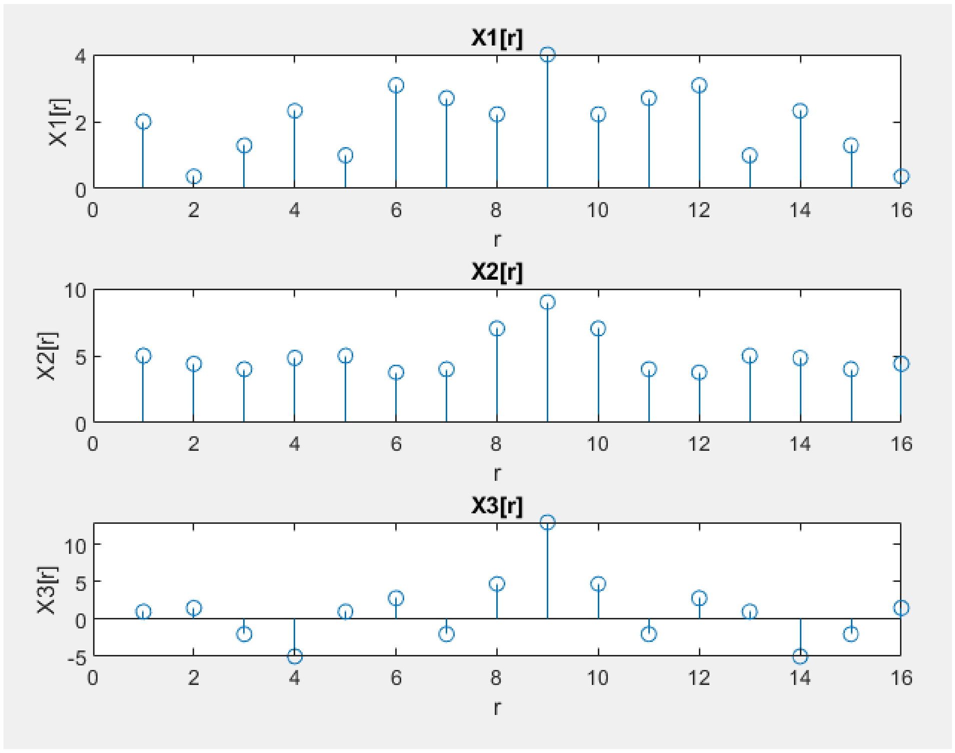
```
xlabel('r');
ylabel('X2[r]');
title('X2[r]');

N3=6; k3=0:N3-1;
x3k = [1 3 3 0 3 3];
X3r = fft(x3k);
X3r = fftshift(X3r);
w3 = -pi:2*pi/N3:pi-2*pi/N3;
subplot(3,1,3),stem(w3, X3r);
xlabel('r');
ylabel('X3[r]');
title('X3[r]');
```



Observation are as shown in the plots above. They have different number of points. X[3] has negative values.

ii) Change the number of points of DFT to 16, what is your observation on the DFTs of x1, x2, and x3? Explain your observations.

```
N1=7; k1=0:N1-1;
x1k = [2 1 0 0 0 0 1];
X1r = fft(x1k, 16);
X1r = fftshift(X1r);
w1 = -pi:2*pi/N1:pi-2*pi/N1;
subplot(3,1,1),stem(X1r);
xlabel('r');
ylabel('X1[r]');
title('X1[r]');

N2=5; k2=0:N2-1;
x2k = [5 1 1 1 1];
X2r = fft(x2k,16);
X2r = fftshift(X2r);
w2 = -pi:2*pi/N2:pi-2*pi/N2;
subplot(3,1,2),stem(X2r);
xlabel('r');
ylabel('X2[r]');
title('X2[r]');

N3=6; k3=0:N3-1;
x3k = [1 3 3 0 3 3];
X3r = fft(x3k,16);
X3r = fftshift(X3r);
w3 = -pi:2*pi/N3:pi-2*pi/N3;
subplot(3,1,3),stem(X3r);
xlabel('r');
ylabel('X3[r]');
title('X3[r]');
```



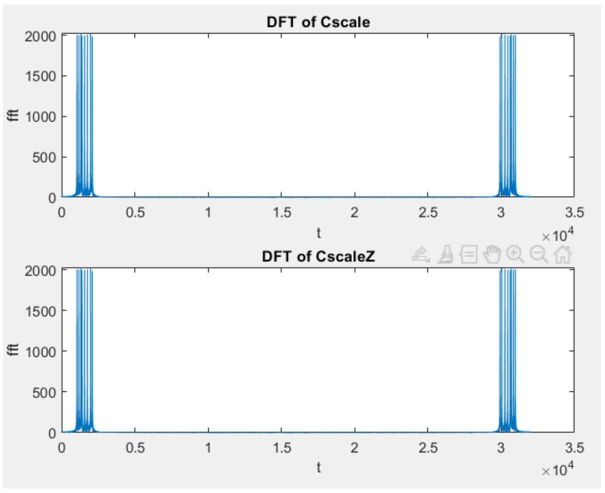
Observations are as shown in the plots above, more intermediate frequency values compared to DFT with same points as sequence. They all have 16 points because we extended the points of DFT to 16, but not the same length as the sequence in previous part of the question.

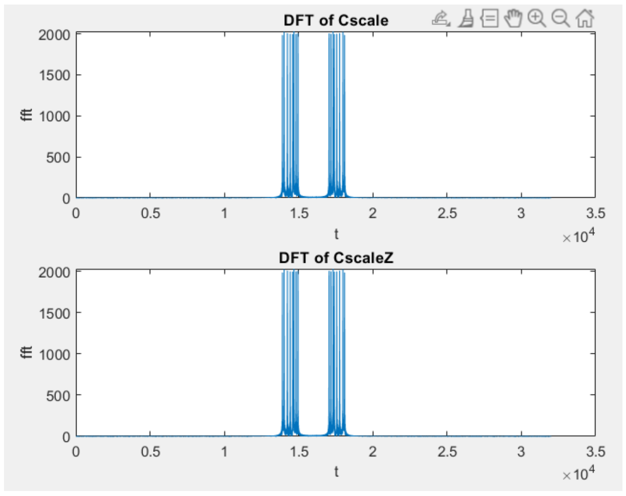
Q3. The DFT can only tell us which frequencies are present in a signal as you did in Lab5. But, it cannot tell you how a melody flows over time. If you want to see the frequency contents over time, what you would do is to compute the Fourier transform of several short successive sections of the original signal. A representation of this sort is called the short-time Fourier transform (STFT). This question is to show you how to do it.

(1) In Matlab, read two wav files you created in Lab5 Q2, i.e. CScale.wav and CScaleZ.wav. Perform the DFT on each signal and plot the magnitude of each signal. Can you tell the difference between these two signals based on the plots?

```
[cy1,fs1]=audioread('CScale.wav');
[cy2,fs2]=audioread('CScaleZ.wav');
subplot(2,1,1),plot(abs(fftshift(fft(cy1))));
xlabel('t');
ylabel('fft');
title('DFT of Cscale');
subplot(2,1,2),plot(abs(fftshift(fft(cy2))));
xlabel('t');
```

```
ylabel('fft');
title('DFT of CscaleZ');
```





Non Shifted

Shifted

There isn't an visible difference between DFT of Scale and DFT of CscaleZ.

(2) Let's perform short time Fourier transform on the audio signal from CScale.wav. Note that the signal read from CScale.wav consists of 8 "keys" in one vector. Recall that the duration of each key is 0.5 second and sampling frequency is 8000Hz when you create the signal in Lab5. How many points are there in each "key"?

4000 points because Sampling frequency was 8000 samples/seconds and each key has a duration of 0.5 seconds.

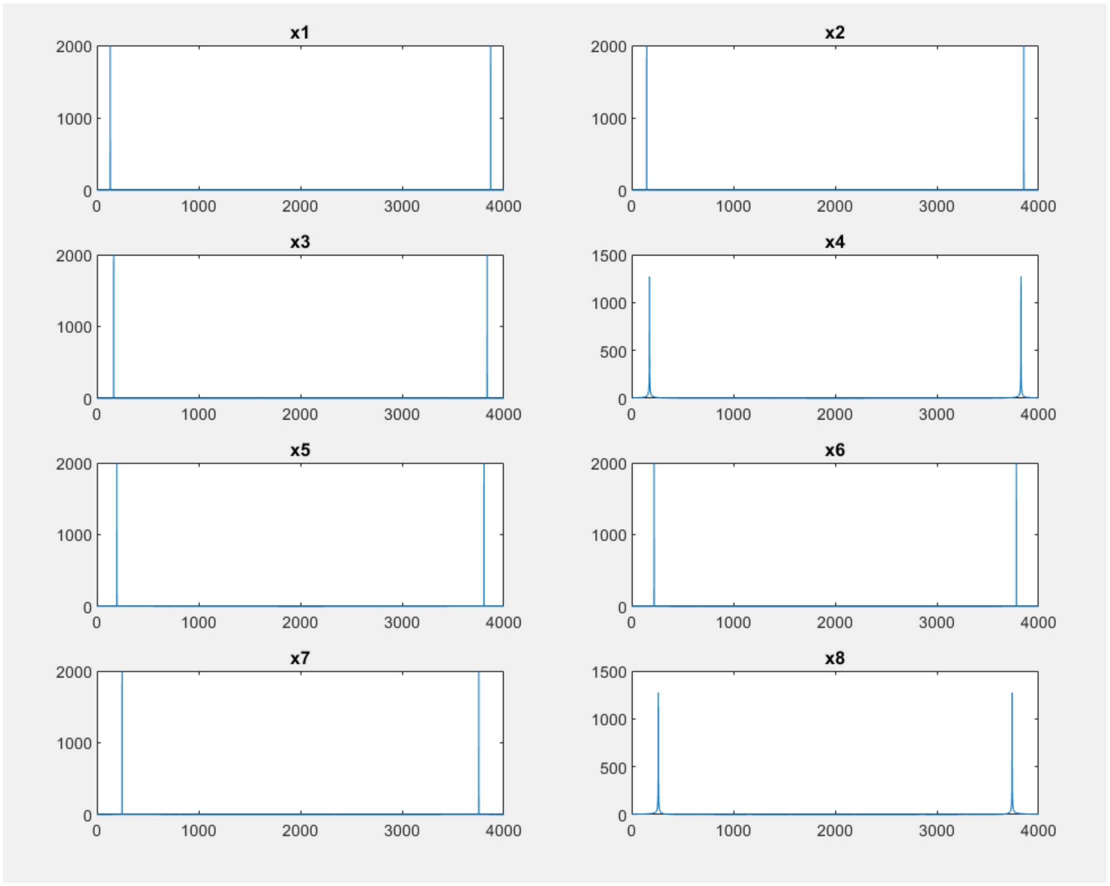
8000/2 = 4000 points

(3) In order for you to perform STFT, you need to split these 8 "keys" into 8 separate signals, e.g. *x*1, *x*2, …, *x*8. Write a Matlab program to split the C Scale signal into 8 vectors, each vector contains one "key". After the separation, plot the magnitude response of any one of the "keys" to verify you did it correctly. This process is referred as windowing, i.e. you cut a long sequence into several pieces, where the length of each piece is determined by the window. Note the window length is the number of points you determined in (2).

Window length = 4000

```
x1 = cy1(1:4000);
x2 = cy1(4001:8000);
x3 = cy1(8001:12000);
```

```
x4 = cy1(12001:16000);
x5 = cy1(16001:20000);
x6 = cy1(20001:24000);
x7 = cy1(24001:28000);
x8 = cy1(28001:32000);
subplot(4,2,1),plot(abs(fft(x1)))
title('x1');
subplot(4,2,2),plot(abs(fft(x2)))
title('x2');
subplot(4,2,3),plot(abs(fft(x3)))
title('x3');
subplot(4,2,4),plot(abs(fft(x4)))
title('x4');
subplot(4,2,5),plot(abs(fft(x5)))
title('x5');
subplot(4,2,6),plot(abs(fft(x6)))
title('x6');
subplot(4,2,7),plot(abs(fft(x7)))
title('x7');
subplot(4,2,8),plot(abs(fft(x8)))
title('x8');
```

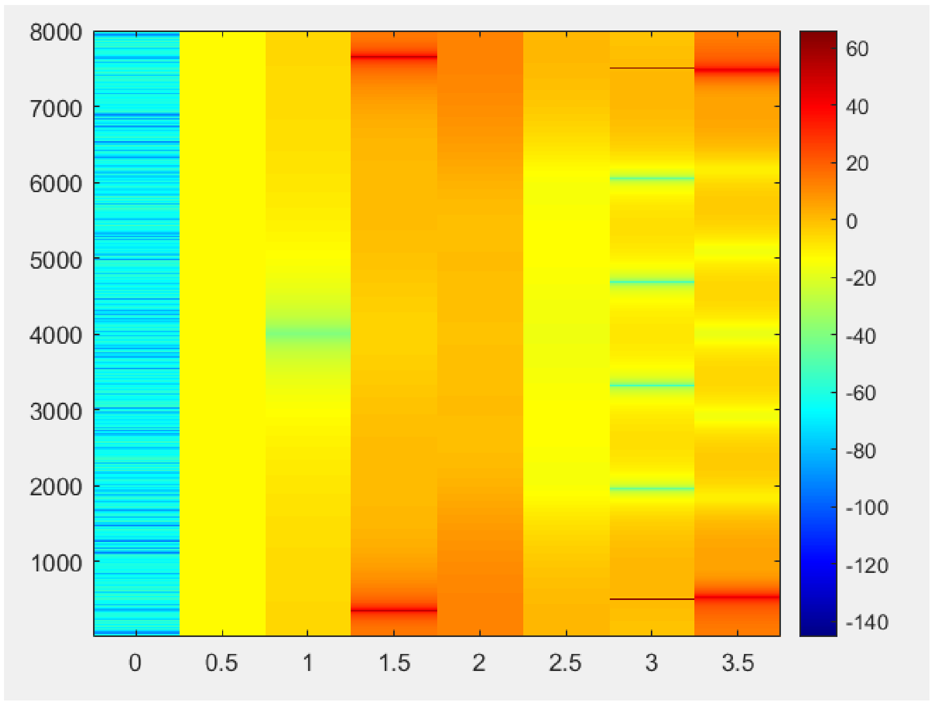


(4) Create a for loop, calculate the FFT of the windowed signal, i.e. $x1$, …, $x8$. The number point of DFT should be the same as the window length. Store FFT results to x1fft, …, x8fft.

```
names = {'x1fft' 'x2fft' 'x3fft' 'x3fft' 'x3fft' 'x3fft' 'x3fft'
'x3fft'};
for index = 1 : length(names)
    s.(names{index}) = fft(cy1((index-1)*4000+1:index*4000), 4000);
end
```

(5) Every fft produces a vector of complex numbers. Reshape them as a column vector. Put all these column vectors in a matrix. Call this matrix "spect". Note that successive frames of the original signal are now represented as successive columns of the original time function (rather than rows as they were before). The rows of "spect" represent different frequency components, and the columns represent different time segments.
Add the following lines to your Matlab script for display purpose.

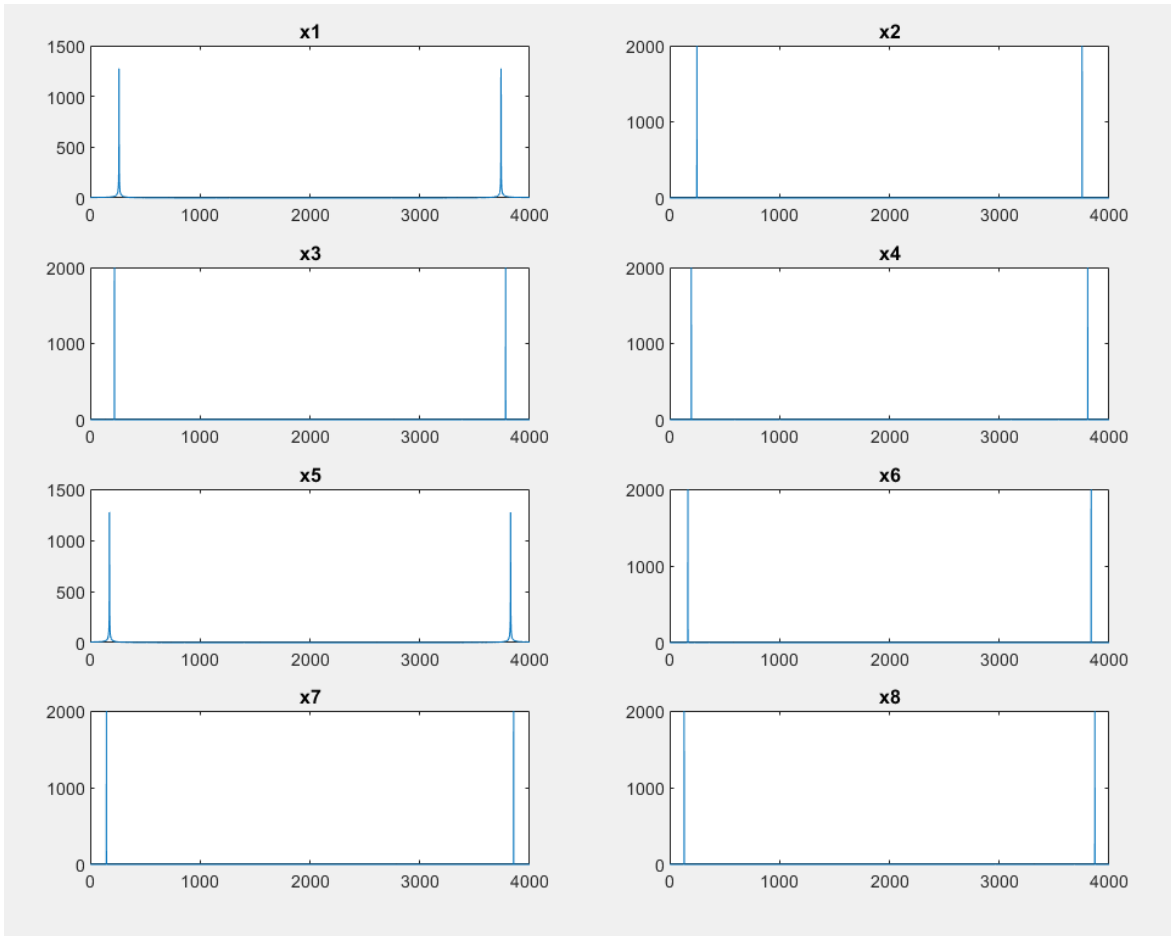
```
for i=1:8
     xfft = cy1((i-1)*4000+1:i*4000);
     xfft = fft(xfft, 4000);
     spect(:,i) =xfft;
end

window_length = 4000;

spect_mag=20*log10(abs(spect));
plot(spect_mag)
t=(0:window_length:(length(cy1)-window_length))/fs1;
f=(1:window_length)*fs1/window_length;
imagesc(t, f, spect_mag);
axis xy
colormap(jet)
colorbar
```



REPEAT Q3 part (3), (4), (5) for CscaleZ.wav

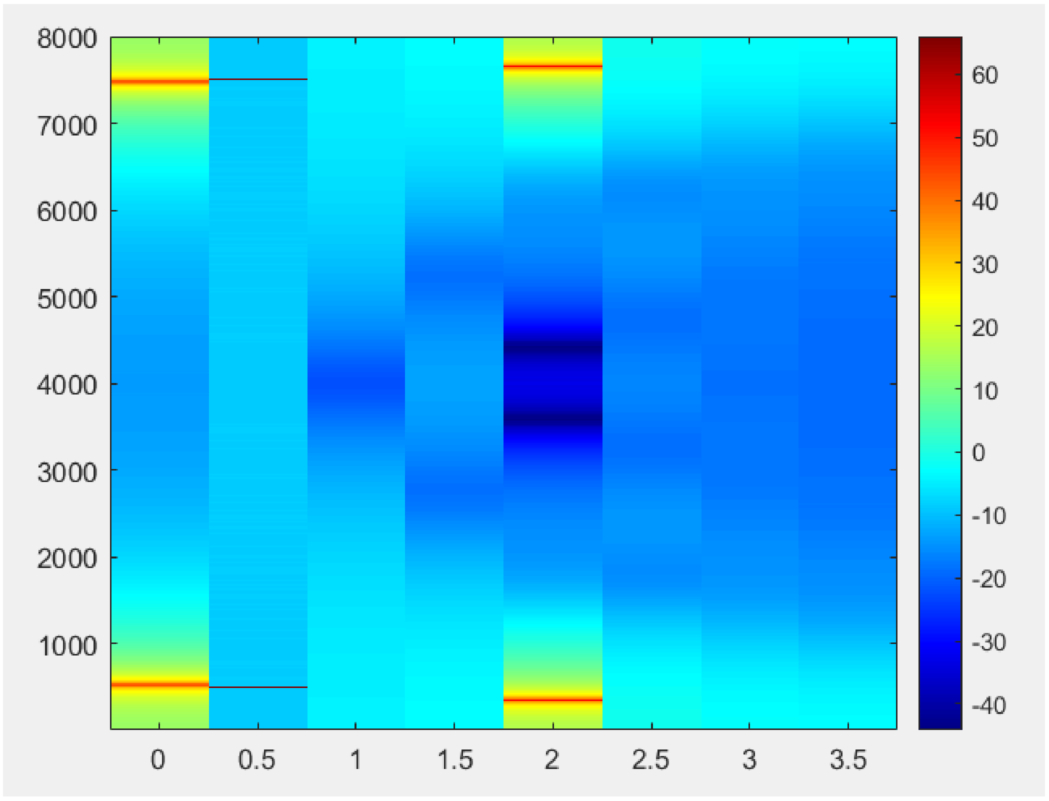
```
[cy2,fs2]=audioread('CScaleZ.wav');

% part (3)
x1 = cy2(1:4000);
x2 = cy2(4001:8000);
x3 = cy2(8001:12000);
x4 = cy2(12001:16000);
x5 = cy2(16001:20000);
x6 = cy2(20001:24000);
x7 = cy2(24001:28000);
x8 = cy2(28001:32000);
subplot(4,2,1),plot(abs(fft(x1)))
title('x1');
subplot(4,2,2),plot(abs(fft(x2)))
title('x2');
subplot(4,2,3),plot(abs(fft(x3)))
title('x3');
subplot(4,2,4),plot(abs(fft(x4)))
title('x4');
```

```
subplot(4,2,5),plot(abs(fft(x5)))
title('x5');
subplot(4,2,6),plot(abs(fft(x6)))
title('x6');
subplot(4,2,7),plot(abs(fft(x7)))
title('x7');
subplot(4,2,8),plot(abs(fft(x8)))
title('x8');
```



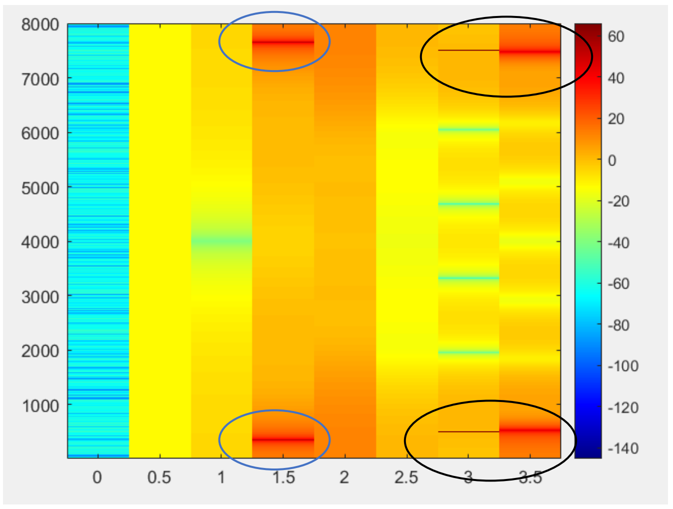
```
% part (4)
names = {'x1fft' 'x2fft' 'x3fft' 'x3fft' 'x3fft' 'x3fft' 'x3fft'
'x3fft'};
for index = 1 : length(names)
    s.(names{index}) = fft(cy2((index-1)*4000+1:index*4000), 4000);
end
```

```
% part (5)
for i=1:8
        xfft = cy2((i-1)*4000+1:i*4000);
        xfft = fft(xfft);
        spect(:,i) =xfft;
end

window_length = 4000;

spect_mag=20*log10(abs(spect));
plot(spect_mag)
t=(0:window_length:(length(cy2)-window_length))/fs2;
f=(1:window_length)*fs2/window_length;
imagesc(t, f, spect_mag);
axis xy
colormap(jet)
colorbar
```

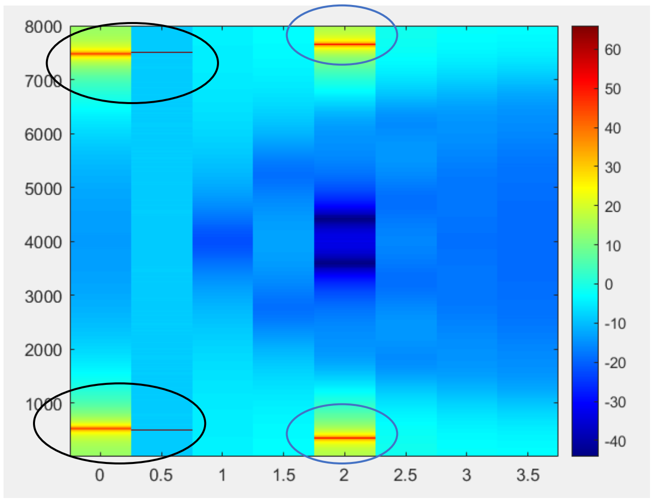


What are the differences between the spectrogram of CScale and CScaleZ?

Name: Chethana Wickramasinghe Student #: 214866511

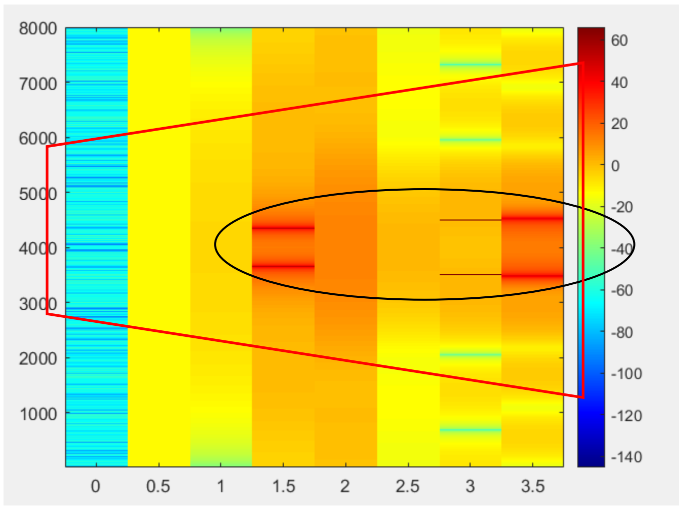


Cscale.wav

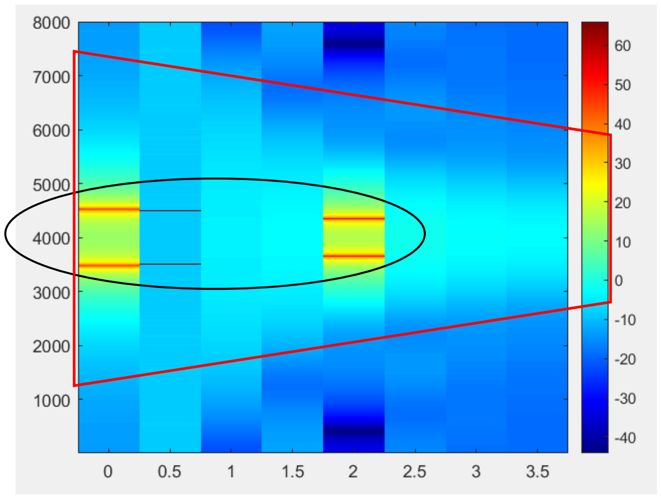


CscaleZ.wav

After shifting.



Cscale.wav

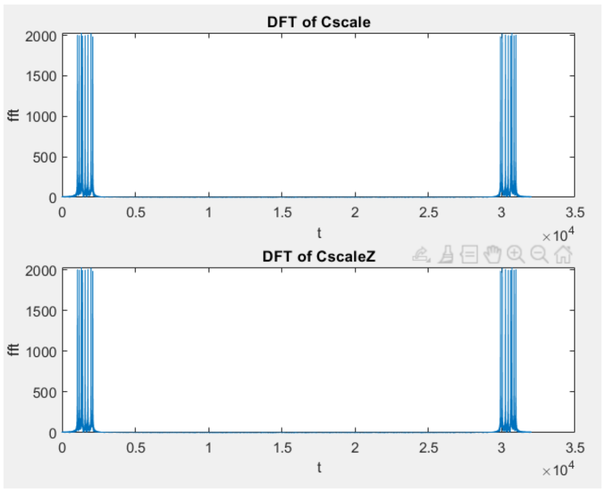


CscaleZ.wav

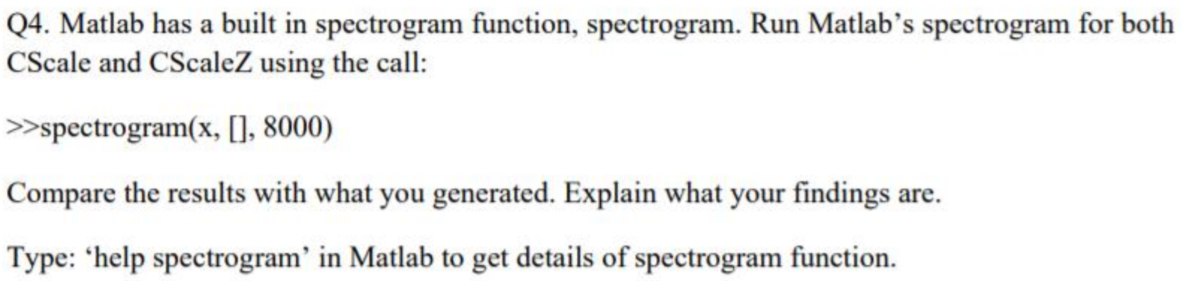
They both have 8 divisions.

Differences: color scales are different, the direction of sections seems to be opposite to each other, and Cscale.wav has an unusual section at 0 that is not visible in Cscale.wav diagram.

Compare it with the fft you did in (1).



There isn't any visible difference in part (1) solution compared to differences found in part (5) spectrogram solution.



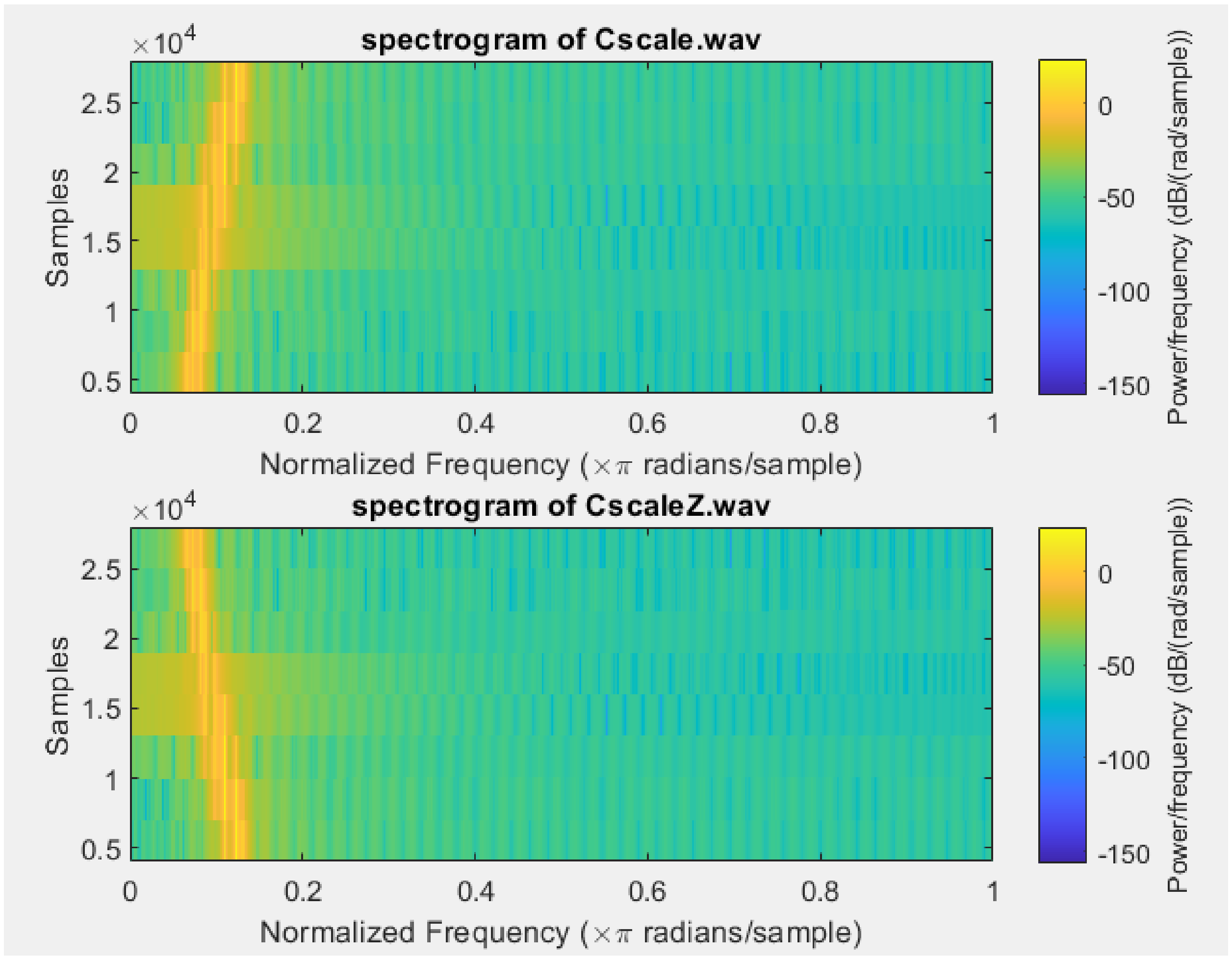
Q4. Matlab has a built in spectrogram function, spectrogram. Run Matlab's spectrogram for both CScale and CScaleZ using the call:

>>spectrogram(x, [], 8000)

Compare the results with what you generated. Explain what your findings are.

Type: 'help spectrogram' in Matlab to get details of spectrogram function.

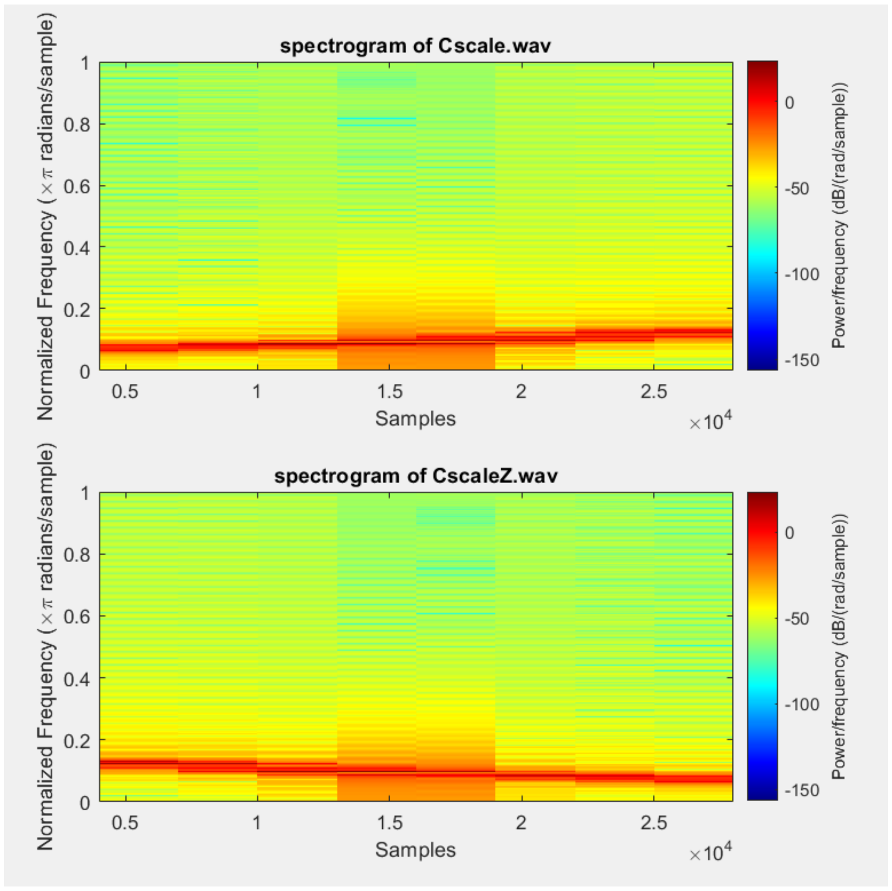
```
[cy1,fs1]=audioread('CScale.wav');
[cy2,fs2]=audioread('CScaleZ.wav');

subplot(2,1,1),spectrogram(cy1, [], fs1)
subplot(2,1,2),spectrogram(cy2, [], fs2)
```



Observations: a bright yellow line in spectrogram of Cscale.wav is flipped in spectrogram of CscaleZ.wav. They have opposite slops.

NOTE: the axis is flipped from what has been done in Q3. Therefore, it has been flipped in the plots below.

```
subplot(2,1,1),spectrogram(cy1, [], fs1, 'yaxis'),
title('spectrogram of Cscale.wav');
subplot(2,1,2),spectrogram(cy2, [], fs2, 'yaxis'),
title('spectrogram of CscaleZ.wav');
```



Observations: Red line in spectrogram Cscale.wav is increasing its gradient (frequency value increases), but the red line in the spectrogram of CscaleZ.wav is decreasing its gradient (frequency value decreased along the sample axis)

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

Learned how to verify the properties of DFT, how to use the convolution to calculate the output of the LTI DT system, and how to use DFT and IDFT to find the output of an LTI DT system.

All the questions were answered using MATLAB and was able to get a better understanding of how to use MATLAB properly in analysing applications using Discrete Fourier Transform.

Q3 was challenging and comparing the results were difficult because the color schemes seems to differ for the comparing sounds.