Lab Three: Application of DFT/FFT to Analyze Signals, and Explore Transform Properties

Digital Signal Processing Laboratory

Matthew Bohr

Dr. Pearlstein

31 October 2022

Student	Analysis	Development	Coding	Results	Writing
Matthew Bohr	100%	100%	100%	100%	100%

Introduction

The DFT and FFT are powerful digital signal processing tools for analyzing the frequency component of discrete-time signals. In this lab the students needed to read in an audio signal and plot it using MATLAB. The FFT and Short-Time Fourier Transform (STFT) were performed on the signal and graphed as well. The FFT is an efficient algorithm that takes advantage of a divide-and-conquer method to compute the DFT. The STFT computes the DFT on a small segment of the original input signal before shifting the segment forward by a pre-determined offset upon which the DFT will be computed again on the new, shifted segment. This is repeated until the segment is shifted enough so that the DFT is computed on most of the information in the signal. Each DFT segment represents a sample of the overall STFT. Each individual DFT has a set of frequency coefficients, and we are computing the DFT a few hundred times, so the coefficients of the frequency components of the STFT will be stored in a 2D matrix. To plot the STFT means to produce a 3D graph. The x-axis would be the frequency, the y-axis would be the DFT sample, and the z-axis would be the magnitude of the DFT at a particular frequency at a particular STFT sample.

Procedure

The students first needed to read in a .wav file. After plotting the signal as well as plotting a portion of the signal, the students then needed to take the FFT of the signal and plot it as well.

After taking the FFT, the STFT was computed and graphed using two methods. The first method was a hardcoded STFT algorithm which computes the FFT on one portion of the overall signal at a time. The second method employed a built in MATLAB function called *spectrogram*.

Next, an impulse function was passed into a low pass filter. This was realized by convolving the impulse with an approximate sinc function. Graphs of the signals and their FFTs were produced as well. The final MATLAB script demonstrates that multiplication in the discrete time domain corresponds to circular convolution in the frequency domain.

Results

A .wav file was download and placed inside the directory with the MATLAB scripts. In the first part of the lab (code shown in Appendix A), the audio file was read into a vector x with a sampling frequency Fs. The signal was then stored in a column vector x_col from which various operations were performed upon it. Figure 1 shows a plot of the signal as well as a zoomed in portion.

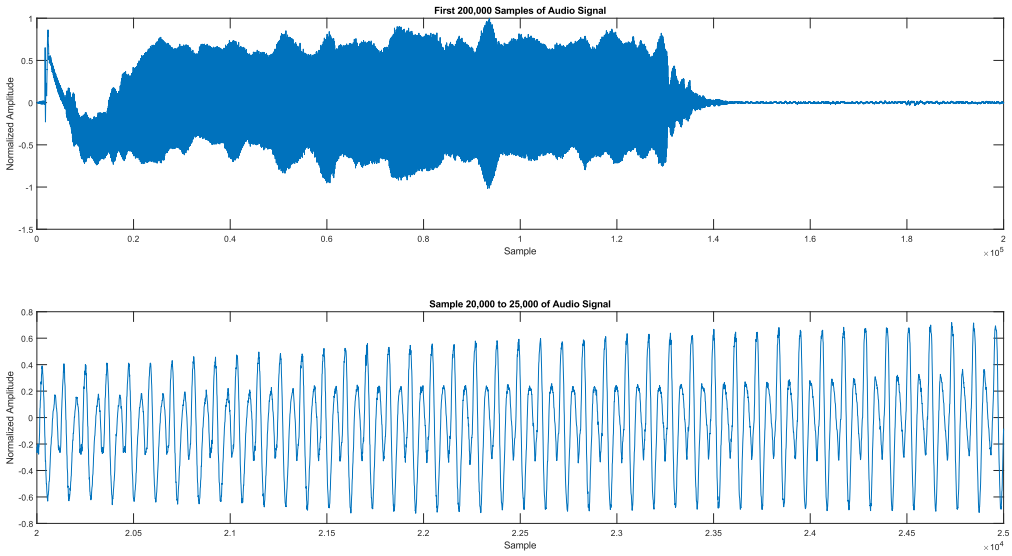

Figure 1: Plot of Audio Signal and Segment from Audio Signal

The next step was to use the MATLAB FFT function to view the frequency content of the audio signal. Figure 2 shows a plot of the normalized FFT as well as a zoomed in portion around the 891st frequency bin. This bin is significant since it is the first bin at which a peak occurs.

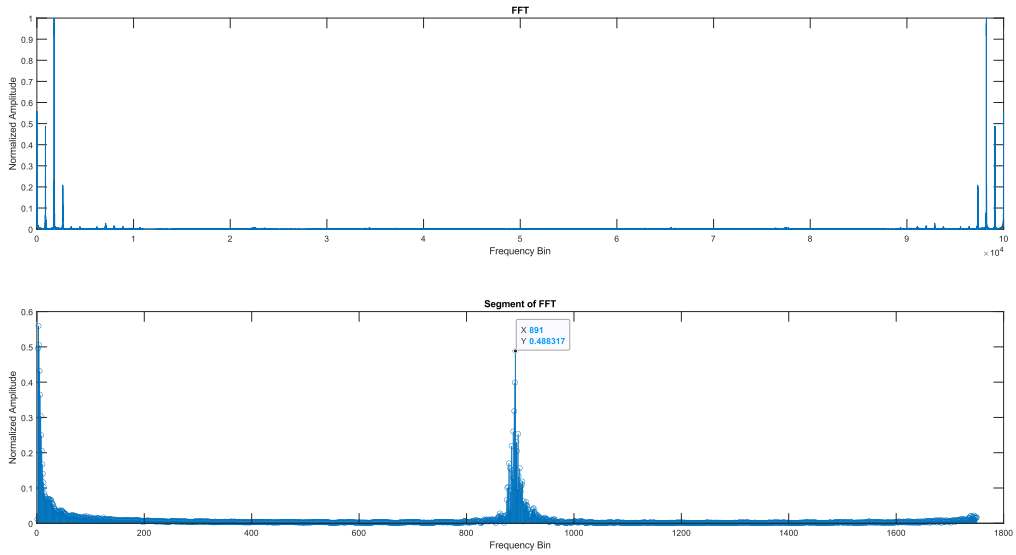

Figure 2: FFT and Segment of FFT of Audio Signal

The frequency of the audio sample can be calculated using the following equation:

$$f_c = \frac{k}{N} F_s = \frac{891}{100000}(48000) = 427.68\ Hz$$

This note corresponds to G# above middle C, which is extremely close to note A above middle C.

The students also performed the STFT on the signal. The STFT highlights how the frequency content of the signal changes at different segments in time. The STFT is realized by taking the FFT of a tiny segment of the signal, calculating the frequency coefficients, then shifting the segment forward in time by some amount (in our case it is 50% of the segment length so 0.5 * 1024 = 512), performing the FFT again, and then repeating this procedure until the end of the signal is reached. Figure 3 shows a horizontal view of the STFT.

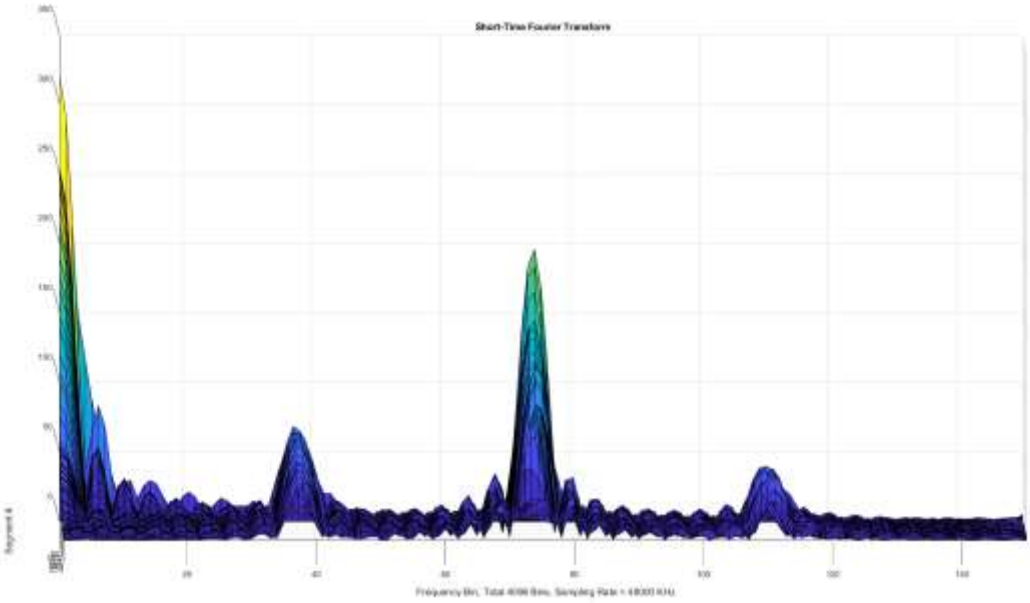

Figure 3: STFT Horizontal View

Figure 4 reveals the true nature of the STFT, highlighting the fact that each segment represents its own FFT. One thing that the STFT encapsulates that the FFT cannot is how the frequency content may change with time since we are reevaluating the FFT for each new segment.

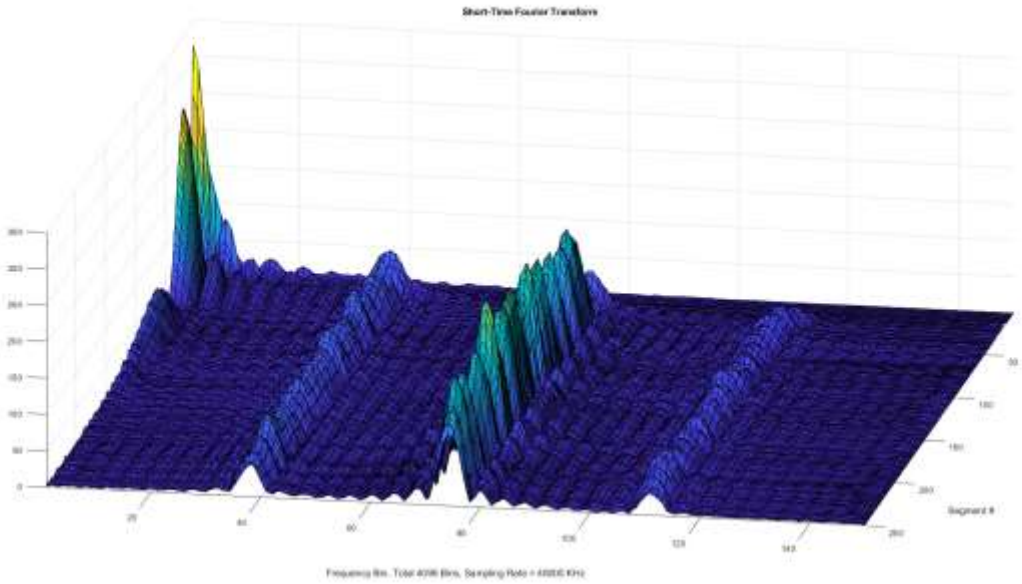

Figure 4: STFT 3/4th View

Figure 5 below shows a top view of the STFT.

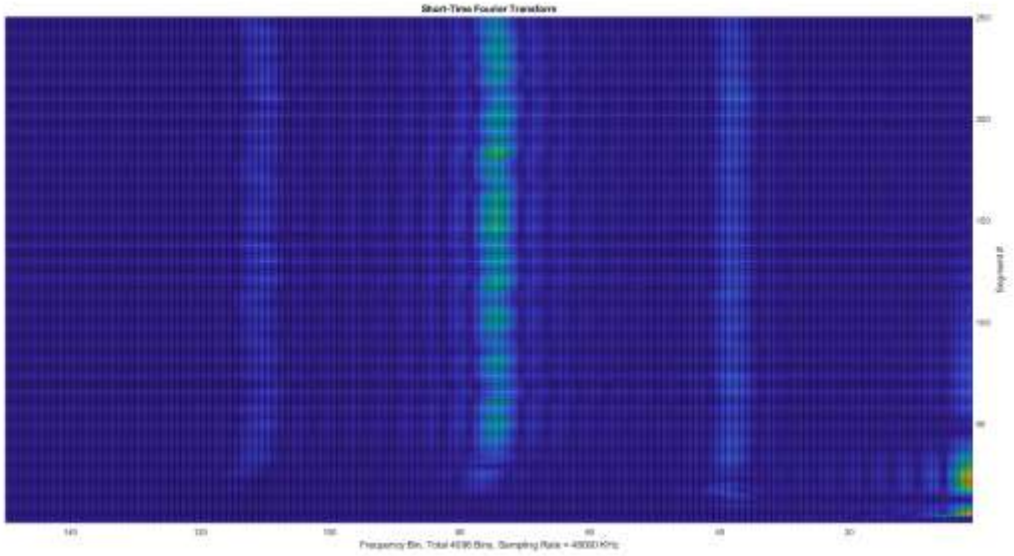

Figure 5: STFT Top View

Figures 6, 7, and 8 also show the result of using the inbuilt *spectrogram* MATLAB function. Note the similarities between the inbuilt spectrogram function and the hardcoded STFT.

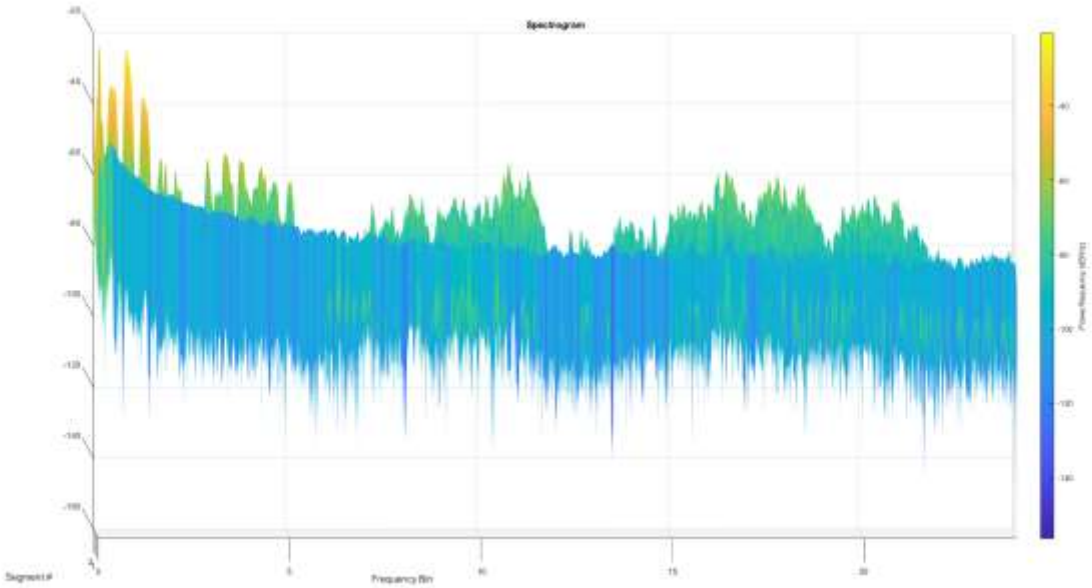

Figure 6: Spectrogram Horizontal View

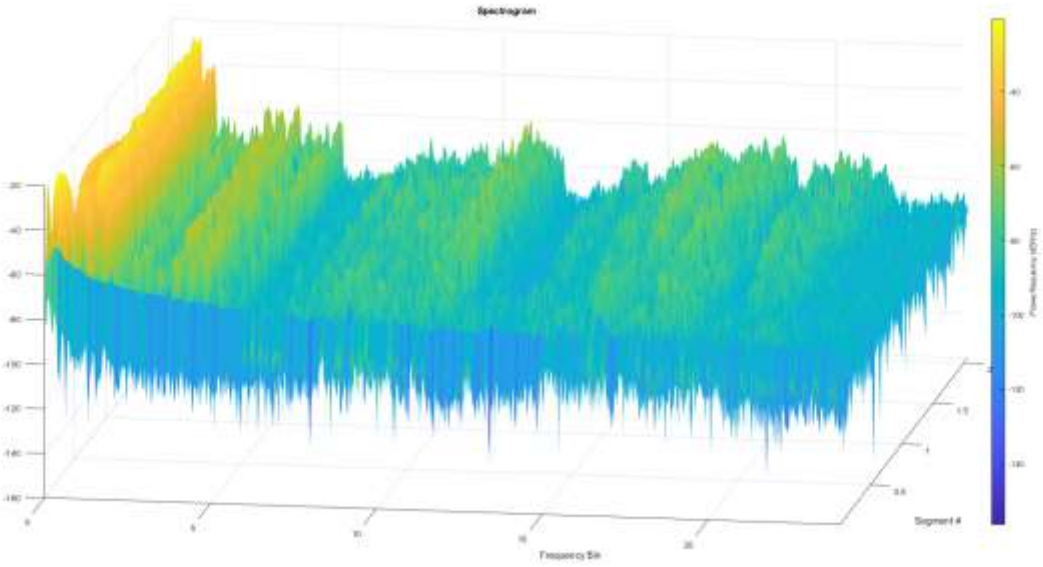

Figure 7: Spectrogram of Audio Signal 3/4th View

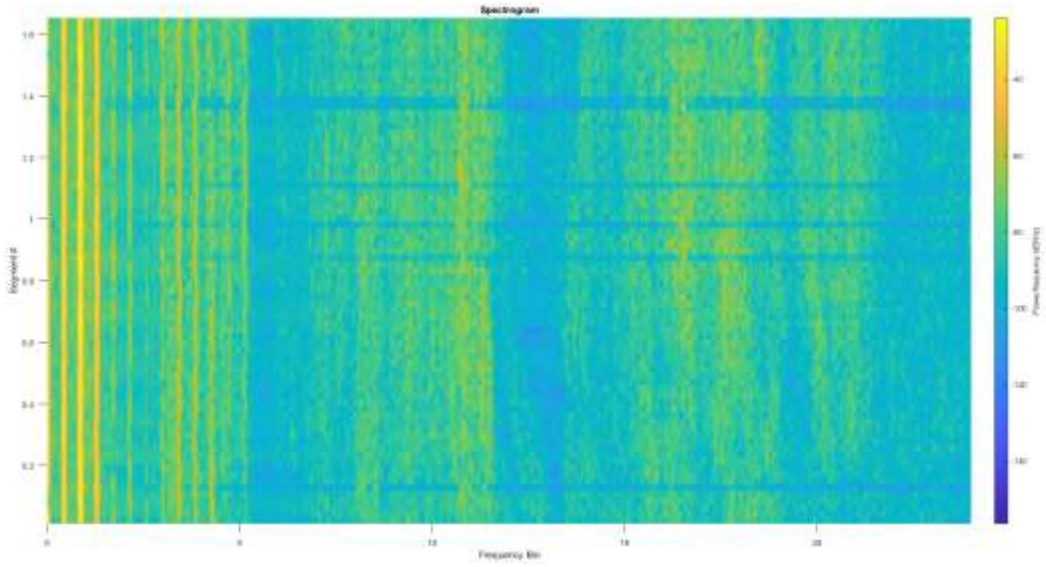

Figure 8: Spectrogram Top View

The students also verified that taking the DFT of the product of two signals is the same as performing circular convolution on the DFT of two signals. Two different examples are shown below in Figure 9. For both examples, the first row shows the two signals overlapped, the second row shows the result of taking the DFT of the signal produced when multiplying the two input signals together, and the third row shows the result of performing circular convolution on the DFT of the two signals. Notice how for each column, row 2 and row 3 match one another, indicating that the above statement is true.

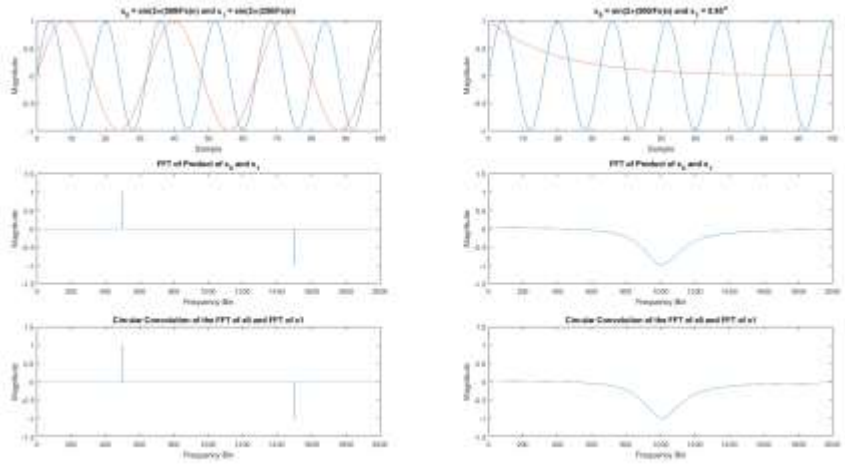

Figure 9: Comparing (a) the FFT of the Product of Two Signals and (b) Circular Convolution of the FFT of Two Signals

Moreover, an impulse was also passed into a low pass filter. The time domain representation of the low-pass filter takes on the form of an approximate sinc function since a LPF is a box in the

frequency domain. Therefore, when we convolve it with an impulse function, the output should also be an approximate sinc function. The first plot in Figure 10 shows how passing an impulse into a low pass filter produced an approximate sinc as expected. The magnitude and phase of the FFT are also shown.

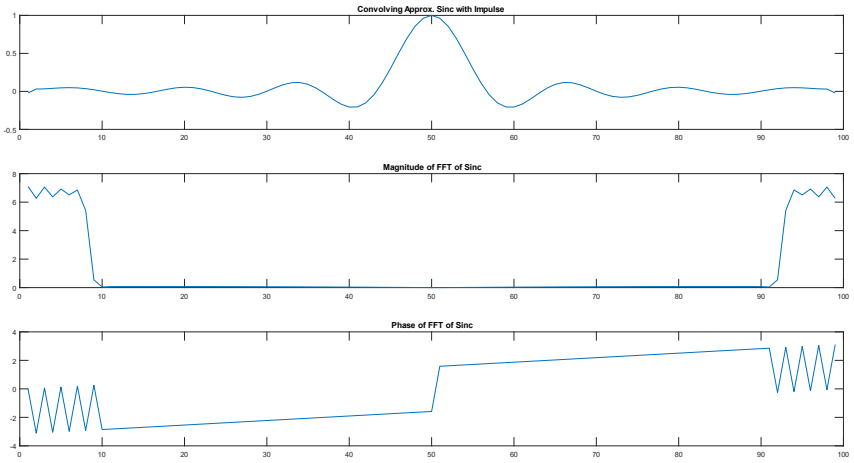

Figure 10: Convolving LPF with Impulse

Conclusion:

This lab exposed the students to the use of the DFT, FFT, and STFT for digital signal processing purposes. A strong emphasis is placed on the connection between the time domain and frequency domain and how useful information about the time-domain signal can be derived from its frequency domain representation. It was very cool to be able to determine the note of an audio signal by using the digital signal processing techniques we are learning in class and lab.

Appendix

Appendix A: lab3_part1.m

```
clear
close all

%reading audio file in x, audioread returns Fs
[x Fs] = audioread('la_la_land.mp4');

%storing x in row vector
x_col = x(:,1);
x_col = x_col / max(x_col);

%finding number of samples
N = size(x_col,1);

%creating variables to populate x
n = 0:1:N-1;

%plotting first 200,000 samples, as well as a segment from index 20k-25k
figure(1);
subplot(2, 1, 1);
plot(1:200000, x_col(1:200000));
title('First 200,000 Samples of Audio Signal');
xlabel('Sample');
ylabel('Normalized Amplitude');
subplot(2,1,2);
plot((20000:25000),x_col(20000:25000));
title('Sample 20,000 to 25,000 of Audio Signal');
xlabel('Sample');
ylabel('Normalized Amplitude');

%computing the FFT
xf = fft(x_col, N);
xf = xf / max(xf);

%plotting entire FFT and FFT from index 2300-2550
figure(2);
```

```
subplot(2,1,1);
plot(n, abs(xf(1:N)));
title('FFT');
xlabel('Frequency Bin');
ylabel('Normalized Amplitude');
xlim([0 N]);
subplot(2,1,2);
stem(2300:2550, abs(xf(2300:2550)));
title('Segment of FFT');
xlabel('Frequency Bin');
ylabel('Normalized Amplitude');

%playing the audio signal aloud
sound(x, Fs);

%computing short short time fourier transform
SEGMENT_LENGTH = 1024;
OFFSET_PER_SEGMENT = 0.5*SEGMENT_LENGTH;
fft_LENGTH = 4096;

%creating a matrix to store coeffs from STFT
coeffs = zeros(4096,267);

%loop that computes one FFT segment and assembles array of coeffs
for i = 0:512:N/2
    data = (x(i+1:1:i+SEGMENT_LENGTH));
    data_fft = abs(fft(data, fft_LENGTH));
    coeffs(:,i/512+1) = data_fft;
end

%using inbuilt spectrogram MATLAB function display STFT
figure(3)
segment = 1:100000;
spectrogram(x_col(segment), SEGMENT_LENGTH, 0, fft_LENGTH, Fs);
title('Spectrogram');
xlabel('Frequency Bin');
ylabel('Segment #');
```

```
figure(4);
surf(coeffs)
xlim([1 250])
ylim([1 150])
xlabel('Segment #')
ylabel(sprintf('Frequency Bin, Total %d Bins, Sampling Rate = %d KHz', fft_LENGTH, Fs));
title('Short-Time Fourier Transform');
```

Appendix B: lab3_part2.m

```
%get lowpass filt coeffs in Num
load('filt_3300_48000_coeffs.mat')

x = 1;

%convolving impulse with LPF
yf = conv(x, Num, 'full');
yf = yf / max(yf);

subplot(3,1,1)
plot((yf(1:99)))
title('Convolving Approx. Sinc with Impulse');

subplot(3,1,2)
y_fft = fft(y);
plot(abs(y_fft))
title('Magnitude of FFT of Sinc');

subplot(3,1,3)
phase = angle(y_fft);
plot(phase(1:99))
title('Phase of FFT of Sinc')
```

Appendix C: lab3_part2.m

```
clear
close all
```

```
%sampling frequency
Fs = 8000;

%frequnecy of two sine waves
f0 = 500;
f1 = 250;

%two seconds worth of samples
N = 2 * Fs;
n = 0:N-1;

%storing two sin waves with freq f0 and f1 in x0 and x1

x0 = cos(2*pi*(f0/Fs)*n);
x1 = cos(2*pi*(f1/Fs)*n);
x2 = 0.95.^n;

%takking the FFT signals
x0_fft = fft(x0);
x1_fft = fft(x1);
x2_fft = fft(x2);

%finding product of signals
x_prod0 = x0 .* x1;
x_prod1 = x0 .* x2;

%taking FFT of x_product
x_prod0_FFT = fft(x_prod0, N);
x_prod0_FFT = x_prod0_FFT / max(x_prod0_FFT);

x_prod1_FFT = fft(x_prod1, N);
x_prod1_FFT = x_prod1_FFT / max(x_prod1_FFT);

%circular convolution of FFTs of signals
cconv0 = cconv(x0_fft, x1_fft, N);
cconv0 = cconv0 / max(cconv0);

cconv1 = cconv(x0_fft, x2_fft, N);
```

```
cconv1 = cconv1 / max(cconv1);

PLOTS_ROWS = 3;
PLOTS_COLS = 2;

%-------------------------plotting-------------------------%
subplot(PLOTS_ROWS,PLOTS_COLS,1);
plot(n,x0);
hold on;
plot(n,x1);
hold
xlim([0 100])
title('x_0 = sin(2\pi(500/Fs)n) and x_1 = sin(2\pi(250/Fs)n)');
xlabel('Sample');
ylabel('Magnitude');

subplot(PLOTS_ROWS,PLOTS_COLS,3);
%plot(n,abs(x_sum_FFT));
plot(n,x_prod0_FFT);
xlim([0 2000])
ylim([-1.5 1.5])
title('FFT of Product of x_0 and x_1');
xlabel('Frequency Bin');
ylabel('Magnitude');

subplot(PLOTS_ROWS,PLOTS_COLS,5);
%plot(n,abs(sines_cconv));
plot(n,cconv0);
xlim([0 2000])
ylim([-1.5 1.5])
title('Circular Convolution of the FFT of x0 and FFT of x1');
xlabel('Frequency Bin');
ylabel('Magnitude');

subplot(PLOTS_ROWS,PLOTS_COLS,2);
plot(n,x0);
hold on;
plot(n,x2);
hold off;
xlim([0 100]);
```

```
title('x_0 = sin(2\pi(500/Fs)n) and x_1 = 0.95^n');
xlabel('Sample');
ylabel('Magnitude');

subplot(PLOTS_ROWS,PLOTS_COLS,4);
%plot(n,abs(x_sum_FFT));
plot(n,x_prod1_FFT);
xlim([0 2000])
ylim([-1.5 1.5]);
title('FFT of Product of x_0 and x_1');
xlabel('Frequency Bin');
ylabel('Magnitude');

subplot(PLOTS_ROWS,PLOTS_COLS,6);
%plot(n,abs(sines_cconv));
plot(n,cconv1);
xlim([0 2000]);
ylim([-1.5 1.5]);
title('Circular Convolution of the FFT of x0 and FFT of x1');
xlabel('Frequency Bin');
ylabel('Magnitude');
```