ELECENG 3TP3

Lab 4: Signal Analysis Using Discrete Fourier Transform

Instructor: Dr. Kiruba

Jasmine Dosanjh – dosanj5 – 400531879

Warisha Noushad – noushadw – 400519903

December 19, 2025

Part I: Fast Fourier Transform Analysis

Part 2:

This part of the Lab involves listing to an audio file `tones.wav` that consists of a signal that is a combination of sinusoidal tones. This audio file plays a constant high-pitched frequency noise for 10 seconds.

Part 3:

The MATLAB code below displays the first 5 ms of the waveform in the file.

Listing 1. Part 3

```
% Read in the signal from the audio file
[signal, Fs] = audioread("tones.wav");
L = length(signal);
T = 1/Fs;
t = [0:L-1]*T;

% Plot the signal for t_plot msec
t_plot = 5;
msec_per_sec = 1000;
numSamples = t_plot*Fs/msec_per_sec; % takes 5 ms of audio

plot(msec_per_sec*t(1:numSamples), signal(1:numSamples))
title('Plot of Input Signal');
xlabel('Time (ms)');
ylabel('Amplitude');
grid('minor')

% Export the graph
text(3.5, 0.56, 'Jasmine Dosanjh - 400531879', 'FontSize', 8)
text(3.5, 0.52, 'Warisha Noushad - 400519903', 'FontSize', 8)
exportgraphics(gcf, "P1_Q3.jpg")
```

The generated plot is shown in Figure 1.

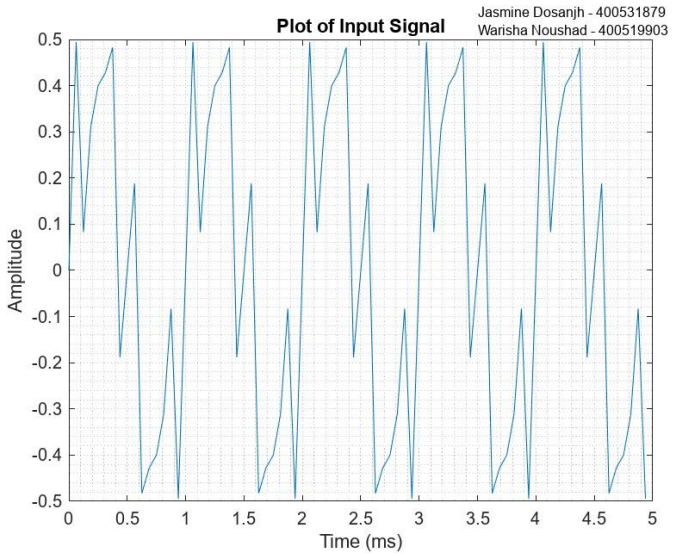


Figure 1. Plot of Input Signal for 5 ms

Part 4:

Based on Figure 1 from Part 3, we can estimate that more than one sinusoid exists, as a single sinusoid should be perfectly smooth, unlike our plot with sharp corners. The waveform is periodic, repeating approximately every 1 ms, thus $T_0 \approx 1\ ms$. In this duration, 3 peaks occur, implying that there are 3 frequency components. The fundamental frequency is $\omega_0 = 1/T_0 \approx 1\ kHz$. Since every frequency in a periodic signal is an integer multiple n of ω_0, the 3 sinusoids will have an individual frequency of $n \cdot \omega_0$, where $n = n_1, n_2, n_3$.

Part 5:

Using the fast Fourier transform (FFT), the discrete Fourier transform (DFT) of the audio signal can be found using the code in Listing 2.

Listing 2. Finding DFT of Audio Signal

```
% Read in the signal from the audio file
[signal, Fs] = audioread("tones.wav");
L = length(signal);
T = 1/Fs;
t = [0:L-1]*T;

% Compute DFT
X = fft(signal); % X[k] in default order: | 0 | + | + | + | - | - |
X_shift = fftshift(X); % shift to: | - | - | 0 | + | + |

% Make the freq x-axis
N = length(X); % N integers: -N/2 ... -1 0 1 ... N/2-1
k = -N/2 : (N/2)-1;
f = (k/N) * Fs;  % frequency in Hz

% Plot magnitude spectrum
stem(f, abs(X_shift))
title('Discrete FT of Input Signal');
xlabel('Frequency f (Hz)');
ylabel('|X(f)|')

%Export the graph
text(4500, 37000, 'Jasmine Dosanjh - 400531879', 'FontSize', 8)
text(4500, 36000, 'Warisha Noushad - 400519903', 'FontSize', 8)
exportgraphics(gcf, "P1_Q5.jpg")
```

The generated stem plots of the magnitude vs frequency of the vectors obtained from the MATLAB `fft` and `fftshift` functions are shown in Figure 2.

These functions are related such that `fft` finds the Discrete Fourier Transform, and `fftshift` shifts the output so the zero-frequency component is centered, with negative frequencies appearing on the left and positive frequencies on the right.

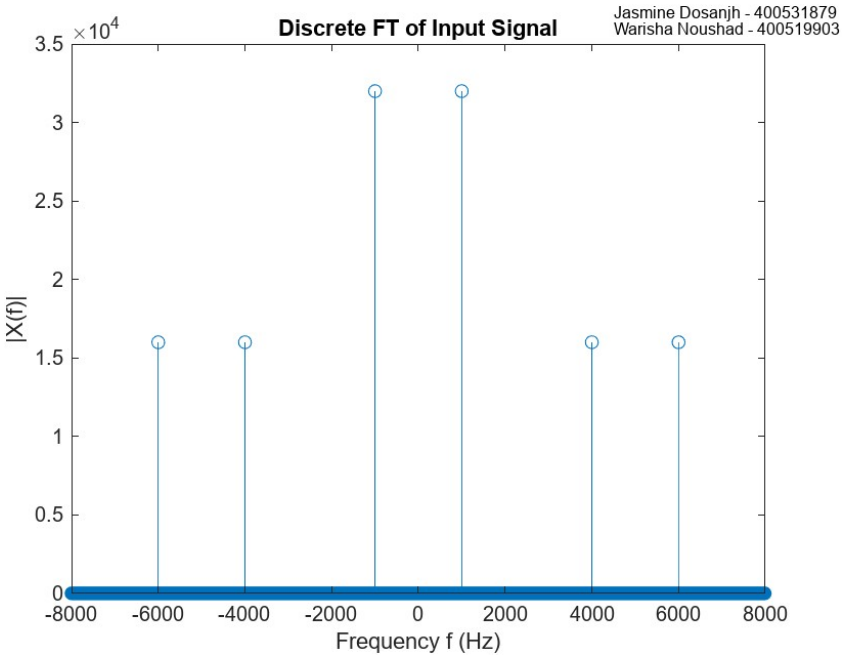


Figure 2. Discrete FT of Input Signal

Part 6:

Figure 3 includes a plot of the single-sided magnitude spectrum of the audio signal. The code in Listing 3 was used to generate this.

Listing 3. Creating Single-Sided Magnitude spectrum

```
% Read in the signal from the audio file
[signal, Fs] = audioread("tones.wav");
L = length(signal);
T = 1/Fs;
t = [0:L-1]*T;

% Take the DFT
Y = fft(signal)/L;
f = Fs/2*linspace(0,1,L/2+1);

% Plot the single-sided magnitude spectrum.
plot(f,2*abs(Y(1:L/2+1)));
title('Single-Sided Magnitude Spectrum')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')
axis([0 Fs/2 0 .5]);
grid('minor');

% Export the graph
text(4500, 0.48, 'Jasmine Dosanjh - 400531879', 'FontSize', 8)
text(4500, 0.45, 'Warisha Noushad - 400519903', 'FontSize', 8)
exportgraphics(gcf, "P1_Q6.jpg")
```

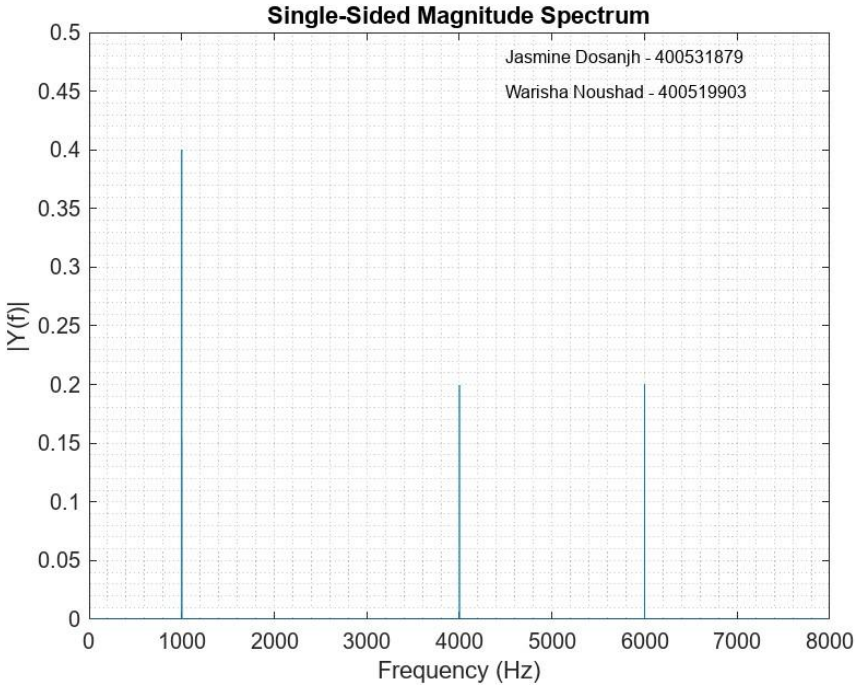


Figure 3. Single-Sided Magnitude Spectrum

Part 7:

Using the results from Part 6, we can see from Figure 3 that the audio signal is made up of three sinusoids.

The frequencies and magnitudes of the sinusoids are:

$f_1 = 1000\ Hz, |X(f_1)| = 0.4$

$f_2 = 4000\ Hz, |X(f_2)| = 0.2$

$f_3 = 6000\ Hz, |X(f_3)| = 0.2$

This corresponds which the hypothesis in Part 4, every sinusoid has a frequency that is an integer multiple n of the fundamental frequency $\omega_0 = 1000\ Hz$, where $n = n_1 = 1, n_2 = 4, n_3 = 6$.

Part 8:

Using the results from Part 7, Listing 4 shows the MATLAB code used to generate the signal that was used to create the audio file.

Listing 4. Generating Original Audio Signal

```
1. % Read in the signal from the audio file
2. [signal, Fs] = audioread("tones.wav");
3. L = length(signal);
4. T = 1/Fs;
5. t = [0:L-1]*T;
6.
```

```
% Recreate og signal
signal_1 = 0.4*sin(2*pi*1000*t);
signal_2 = 0.2*sin(2*pi*4000*t);
signal_3 = 0.2*sin(2*pi*6000*t);
combinedSignal = signal_1 + signal_2 + signal_3;

% Plot the signal for t_plot msec
t_plot = 5;
msec_per_sec = 1000;
numSamples = t_plot*Fs/msec_per_sec; % takes 5 ms of audio

tiledlayout("vertical")
x = 0:0.1:5;
nexttile;
plot(msec_per_sec*t(1:numSamples), signal(1:numSamples))
title('Plot of Input Signal'); xlabel('Time (ms)'); ylabel('Amplitude'); grid('minor')
nexttile
plot(msec_per_sec*t(1:numSamples), combinedSignal(1:numSamples))
title('Reconstructed Signal'); xlabel('Time (ms)'); ylabel('Amplitude'); grid('minor')

% Export the graph
text(3.5, 0.6, 'Jasmine Dosanjh - 400531879', 'FontSize', 8)
text(3.5, 0.7, 'Warisha Noushad - 400519903', 'FontSize', 8)
exportgraphics(gcf, "P1_Q8.jpg")
```

The first 5 ms of the reconstructed signal are plotted, so that it can be compared to the original signal found in Part 3. There are no differences, the signals are identical.

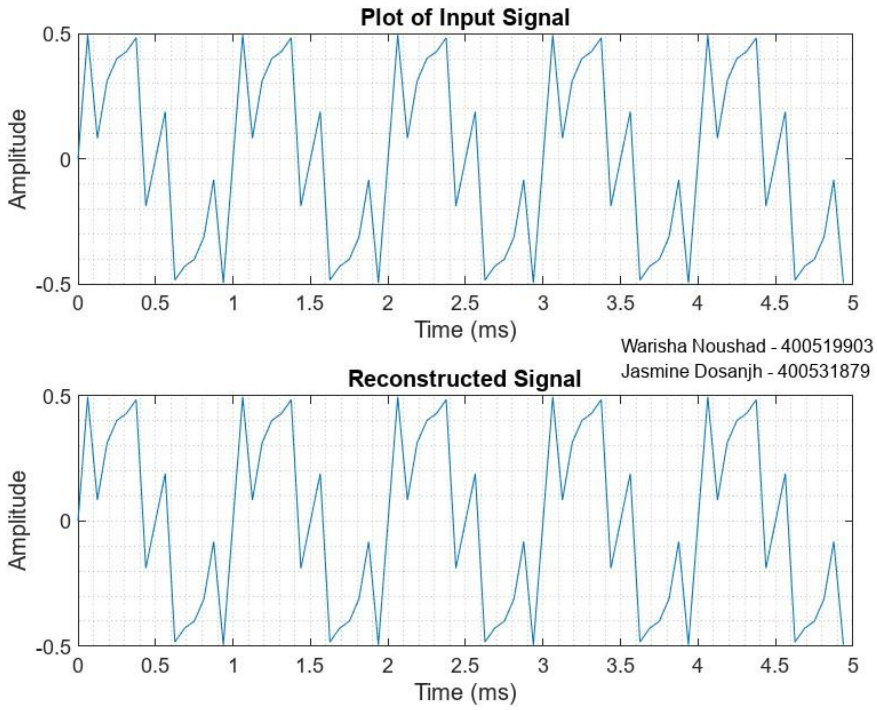


Figure 4. Reconstructed Signal vs Original Signal

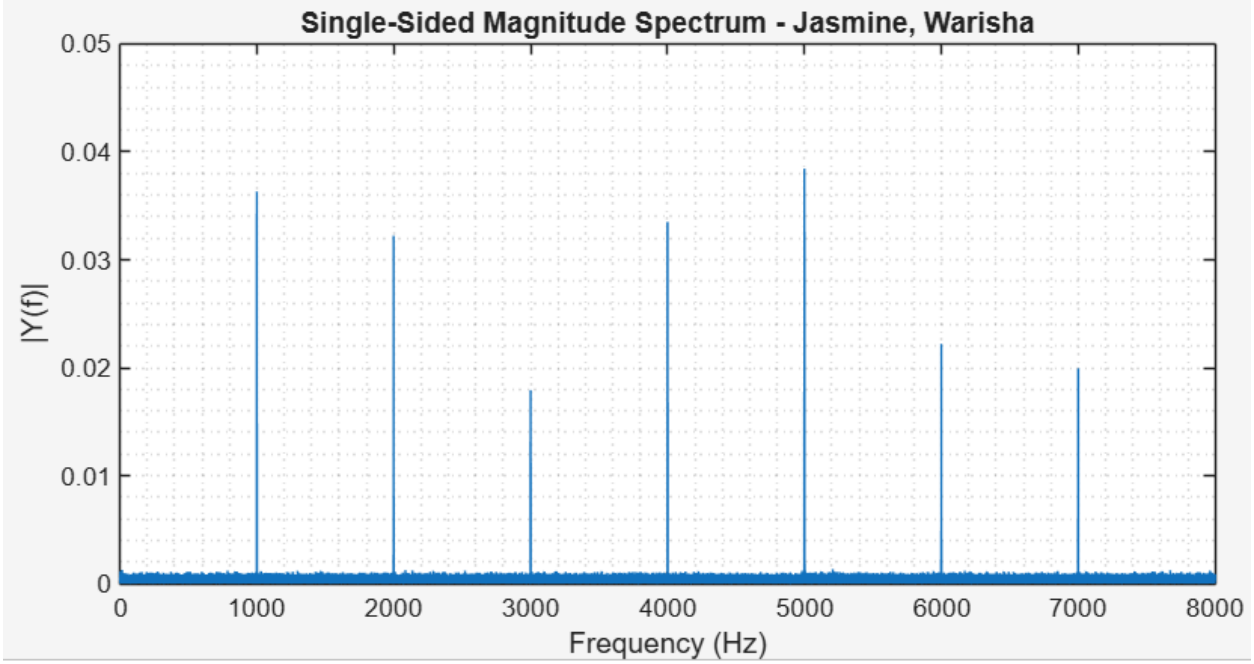
Part 2: Decoding Secret Message

Part 2:

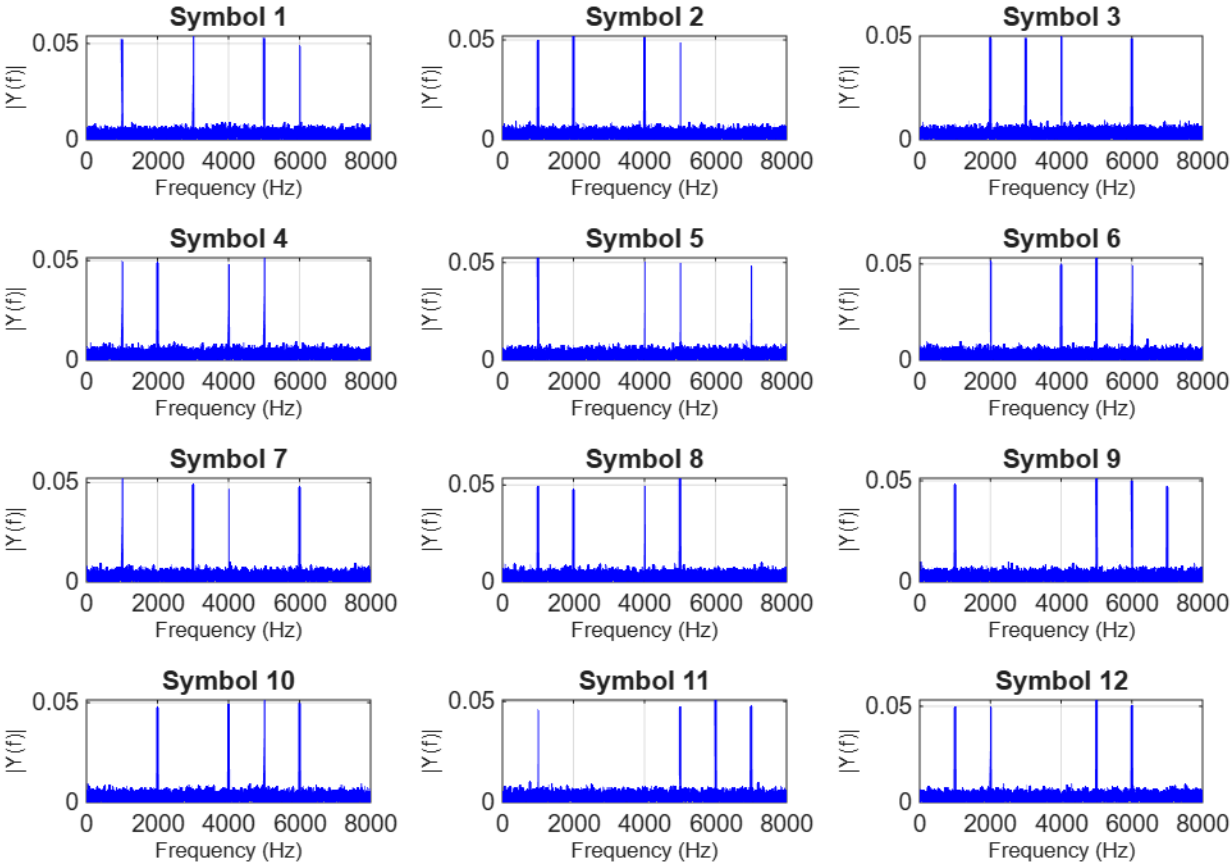
A high-pitched static noise with constant ringing in the background.

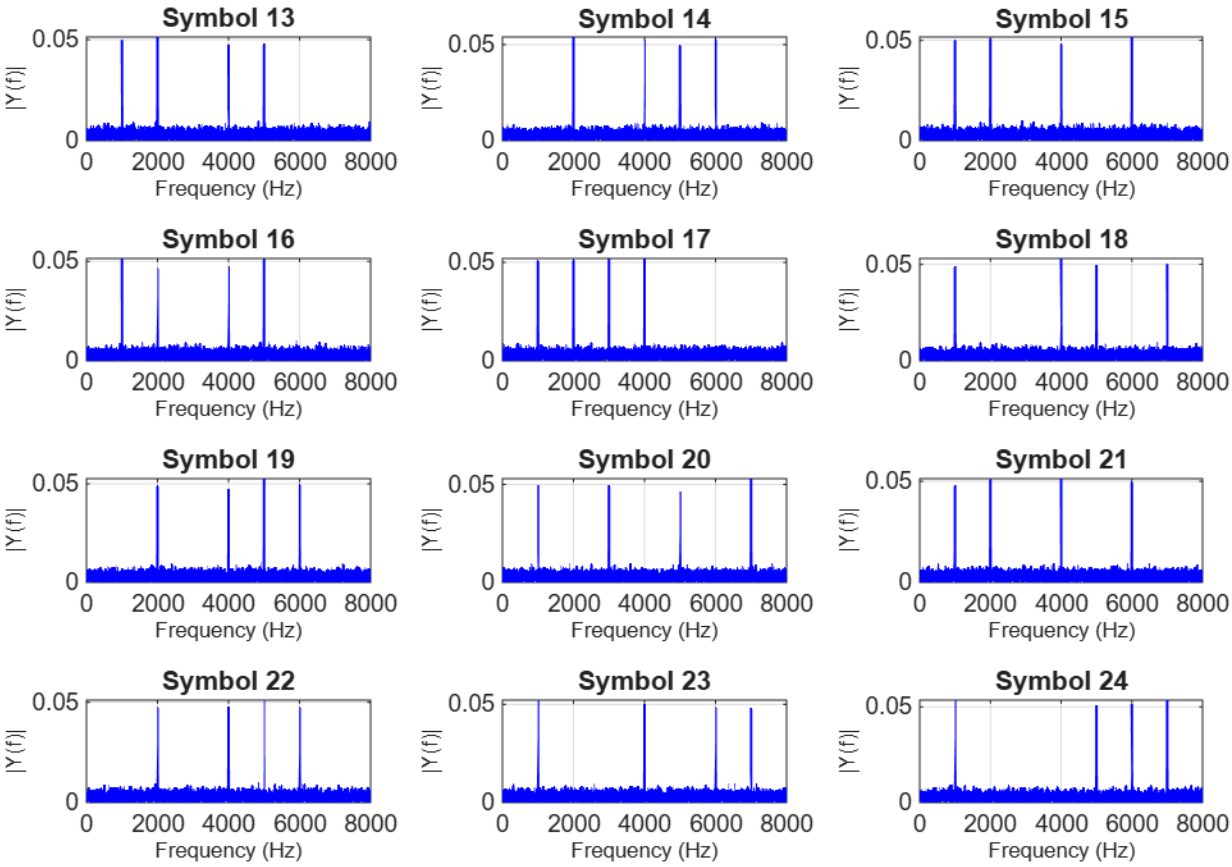
Part 3:

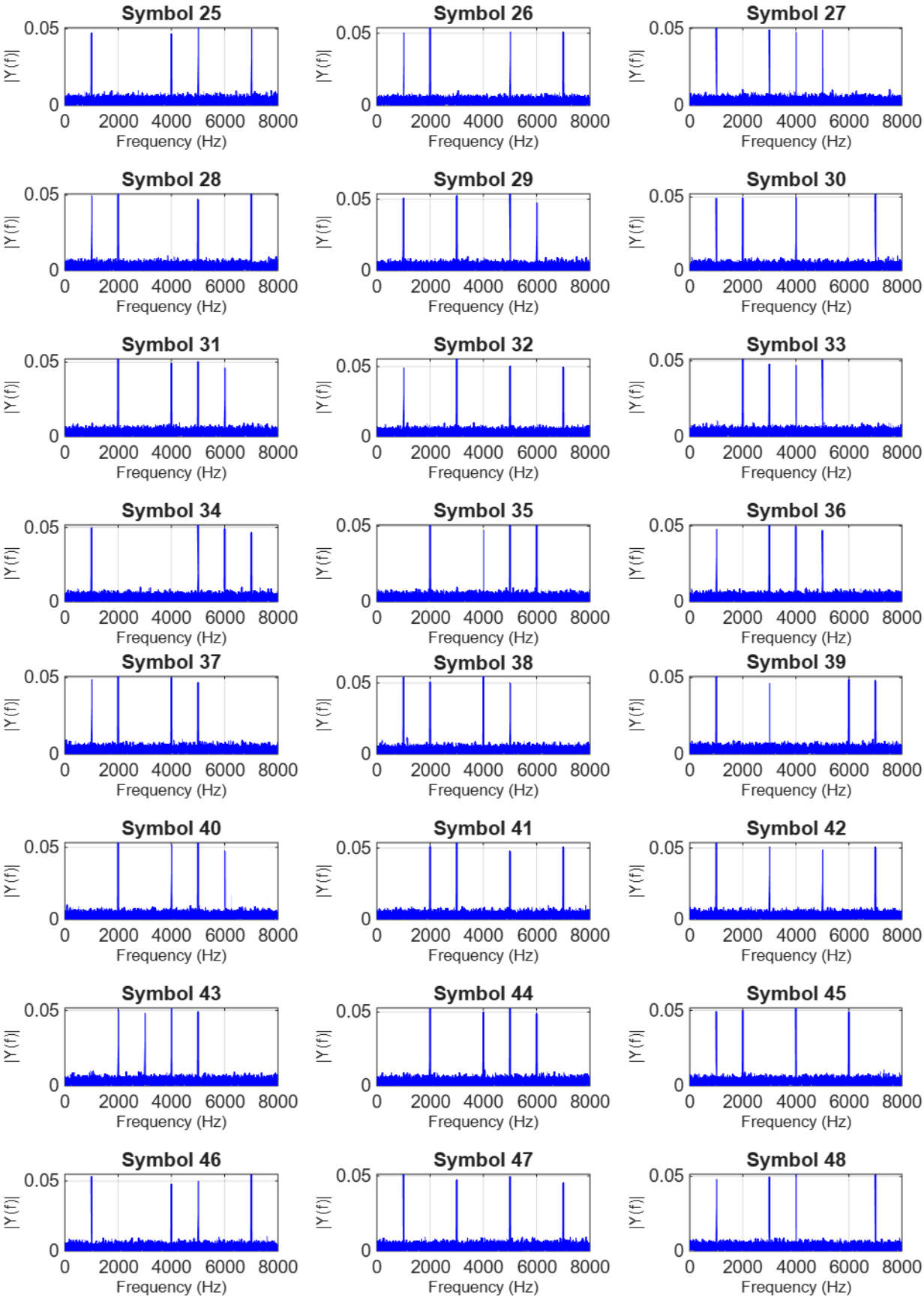
```
%%Lab 4
% Jasmine Dosanjh - dosanj5 - 400531879
% Warisha Noushad - noushadw - 400519903

%Read in the given audio signal
[signal,Fs] = audioread("SecretMessage.wav");
L = length(signal);
T = 1/Fs;
t = (0:L-1) * T; % Create a time vector based on the length of the signal

%take the DFT
Y = fft(signal)/L;
f = Fs/2 * linspace(0,1,L/2+1);

%Plotting
plot(f, 2*abs(Y(1:L/2+1)));
title ('Single-Sided Magnitude Spectrum - Jasmine, Warisha');
xlabel('Frequency (Hz)');
ylabel ('|Y(f)|');
axis ([0 Fs/2 0 0.05]);
grid('minor');

%Export data

exportgraphics(gcf, 'Part2_Q3.jpg');
```

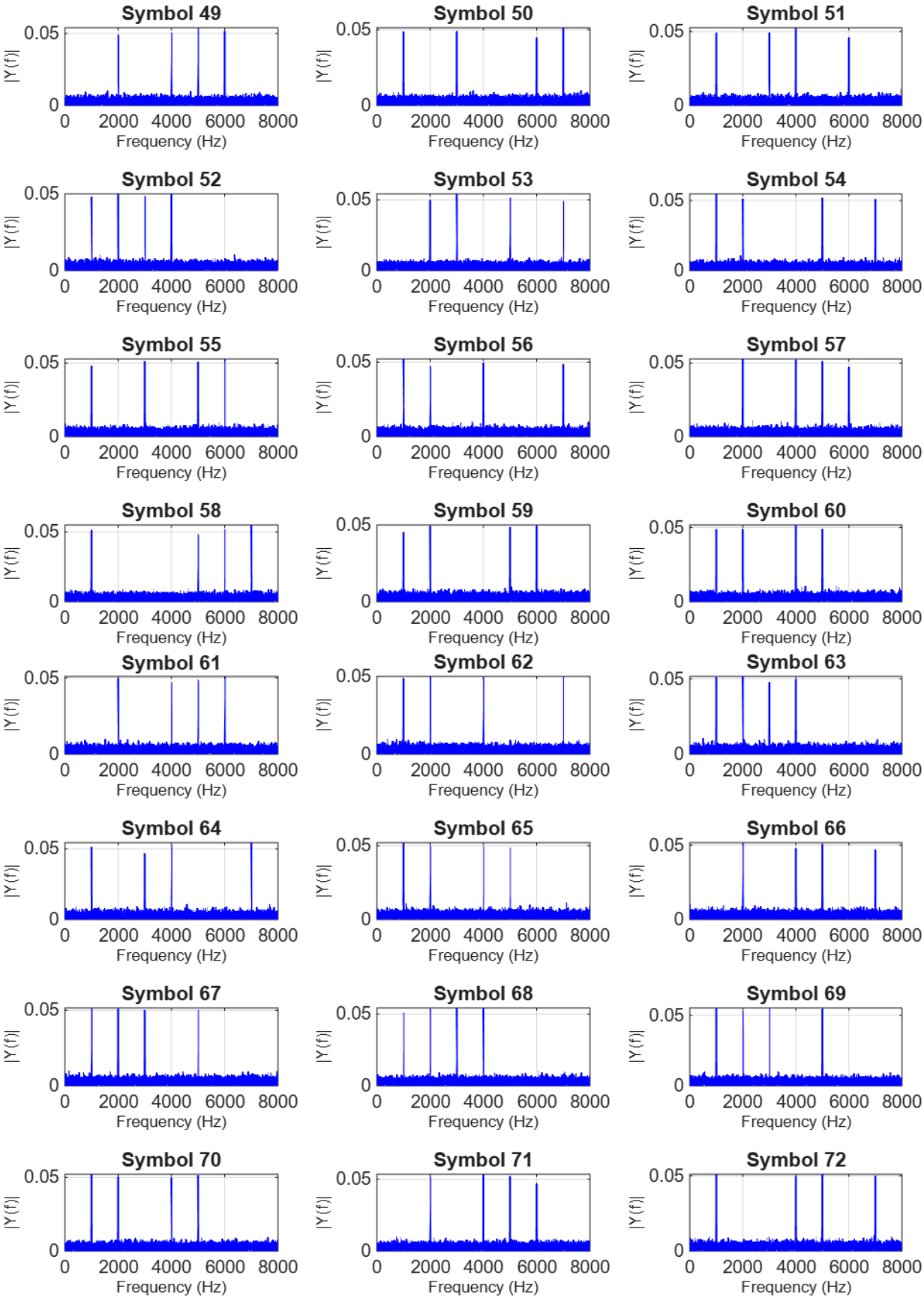

Single-Sided Magnitude Spectrum - Jasmine, Warisha
|Y(f)|
0.05
0.04
0.03
0.02
0.01
0
0
1000
2000
3000
4000
5000
6000
7000
8000
Frequency (Hz)

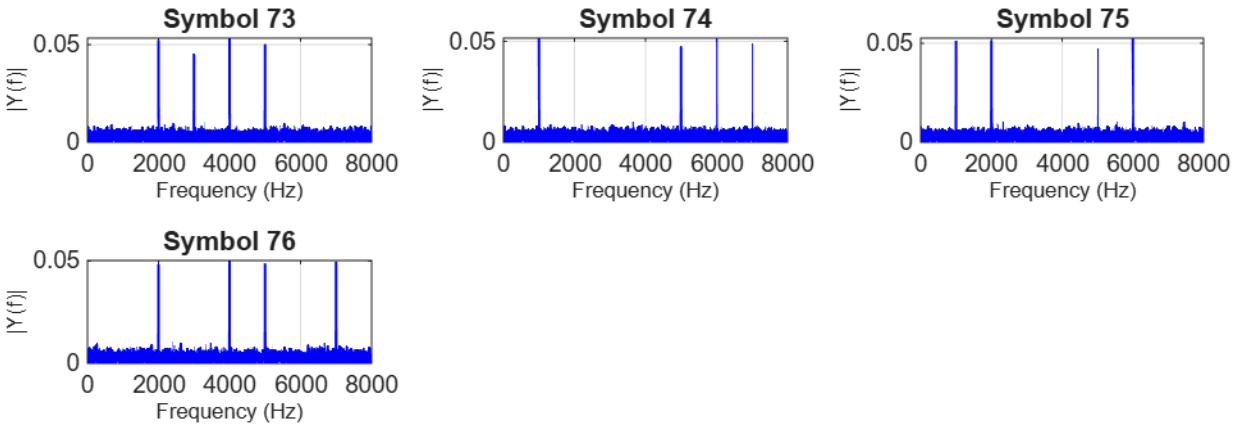
Part 4:


Symbol 1
Symbol 2
Symbol 3
Symbol 4
Symbol 5
Symbol 6
Symbol 7
Symbol 8
Symbol 9
Symbol 10
Symbol 11
Symbol 12
|Y(f)|
0.05
0
0
2000
4000
6000
8000
Frequency (Hz)


Symbol 13
Symbol 14
Symbol 15
Symbol 16
Symbol 17
Symbol 18
Symbol 19
Symbol 20
Symbol 21
Symbol 22
Symbol 23
Symbol 24
|Y(f)|
0.05
0
0 2000 4000 6000 8000
Frequency (Hz)


Symbol 25
Symbol 26
Symbol 27
Symbol 28
Symbol 29
Symbol 30
Symbol 31
Symbol 32
Symbol 33
Symbol 34
Symbol 35
Symbol 36
Symbol 37
Symbol 38
Symbol 39
Symbol 40
Symbol 41
Symbol 42
Symbol 43
Symbol 44
Symbol 45
Symbol 46
Symbol 47
Symbol 48
|Y(f)|
0.05
0
0 2000 4000 6000 8000
Frequency (Hz)


Symbol 49
Symbol 50
Symbol 51
Symbol 52
Symbol 53
Symbol 54
Symbol 55
Symbol 56
Symbol 57
Symbol 58
Symbol 59
Symbol 60
Symbol 61
Symbol 62
Symbol 63
Symbol 64
Symbol 65
Symbol 66
Symbol 67
Symbol 68
Symbol 69
Symbol 70
Symbol 71
Symbol 72
|Y(f)|
0.05
0
0 2000 4000 6000 8000
Frequency (Hz)



The decoded message is:

“NEVER LET THE FEAR OF STRIKING OUT KEEP YOU FROM PLAYING THE GAME.BABE RUTH.”

```
%%Question 4:
clc;
clear;
% Jasmine Dosanjh - dosanj5 - 400531879
% Warisha Noushad - noushadw - 400519903
[signal, Fs] = audioread("SecretMessage.wav");
L = length (signal);

period_sym = 1;  %1 second symbol duration
sample = Fs* period_sym; %samples/second
symbol = floor(L/sample);

symbolsperpage= 12; %number of plots on one page
pageCount = ceil(symbol/symbolsperpage);

for p = 1:pageCount
        figure;
        t = tiledlayout(4,3);
        t.TileSpacing = 'loose';
        t.Padding = 'loose';

        start = (p-1) * symbolsperpage + 1;
        endsim = min(p*symbolsperpage, symbol);

        for i = start:endsim
            %Extract Segment
            startIndex = (i-1)*sample+1;
            endIndex = i*sample;
            symbolSeg = signal(startIndex:endIndex);

            %FFT
            Y= fft(symbolSeg)/sample;
            f = Fs/2 * linspace(0, 1, sample/2 + 1);

            nexttile;
            plot(f, 2 * abs(Y(1:sample/2 + 1)), 'b-', 'LineWidth', 1.0);
            title(['Symbol ' num2str(i)], 'FontSize', 10);
            xlabel('Frequency (Hz)', 'FontSize',8);
            ylabel('|Y(f)|', 'FontSize', 8);
            xlim([0 , Fs/2]);
            grid on;
        end
        exportgraphics(gcf, sprintf('SymbolPlots_Page_%d.png',p));
end
```