ELECENG 3TP3 Lab 4

Raeed Hassan
hassam41
McMaster University

December 12, 2020

Part 1

2. I hear a constant high frequency sound that plays for 10 seconds.

3. The MATLAB code used to generate a plot of the first 5 seconds of the waveform is shown in Listing 1. The plot of the first five seconds is shown in Figure 1.

Listing 1: Generating plot of waveform

```
[signal, Fs] = audioread('../tones2020.wav');
L = length(signal);
T = 1/Fs;
t = [0:L-1]*T;

t_plot = 5;
msec_per_sec = 1000;
numSamples = t_plot*Fs/msec_per_sec;
plot(msec_per_sec*t(1:numSamples), signal(1:numSamples));
title('Plot of Input Signal');
xlabel('time (ms)');
grid('minor');
exportgraphics(gcf, '../Figures/part1q3.png');
```

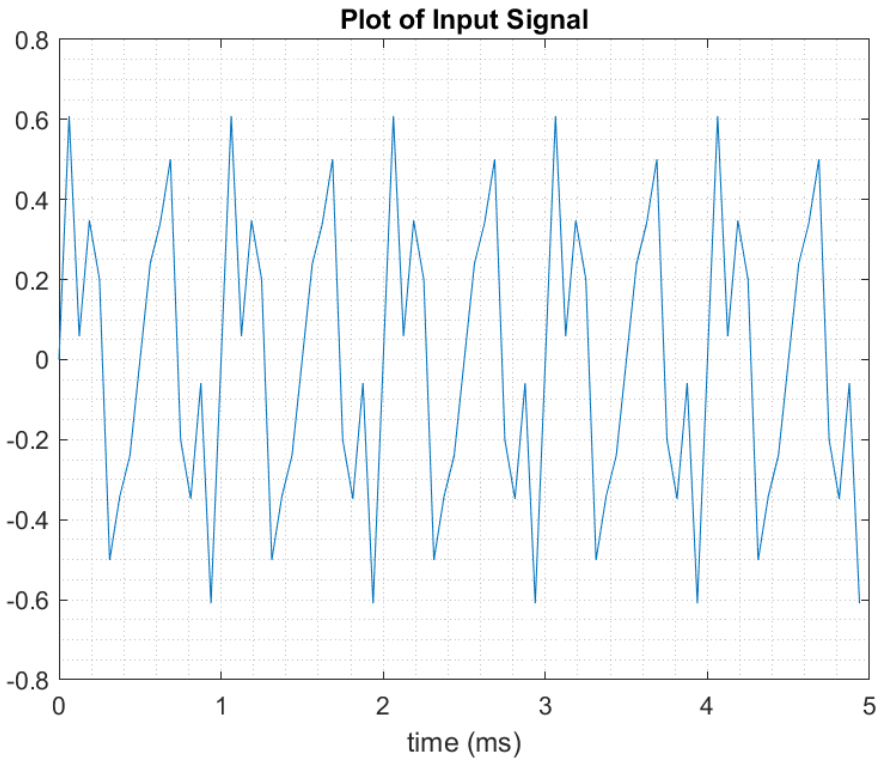

Figure 1: Plot of first 5 milliseconds of waveform for tones2020.wav

4. In the plot of the input signal, we can see that there is a periodic signal that repeats every 1 millisecond (frequency of 1000 Hz). It is difficult to tell how many sinusoids make up the signal and their frequencies from the plot of the input signal. The number of sinusoids that make up the signal and their frequencies can be determined by applying the Fourier transform on the input signal.

5. The plot of the DFT of the audio signal is shown in Figure 2. The MATLAB code used to generate the graphs is shown in Listing 2.

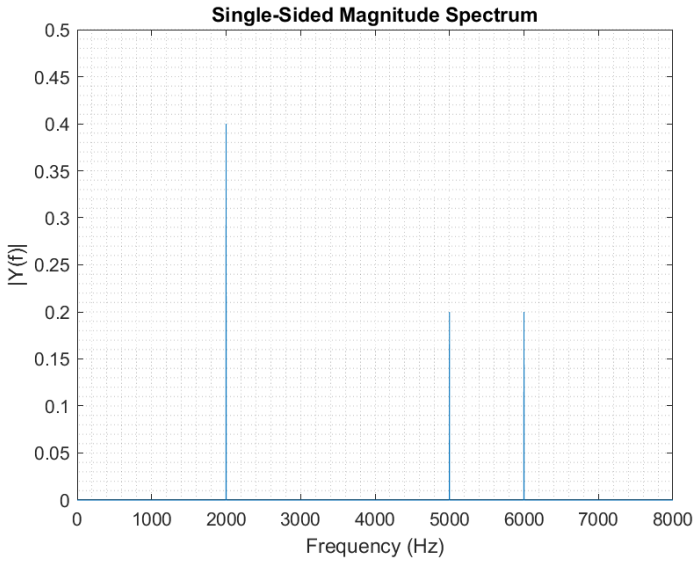

Figure 2: DFT of the audio signal

Listing 2: DFT of the audio signal

```
part1q3;

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

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');
exportgraphics(gcf, '../Figures/part1q5.png');
```

6. The audio signal is made up of three sinusoids, and the frequencies and magnitudes of the sinusoids are: 2000 Hz and 0.4, 5000 Hz and 0.2, and 6000 Hz and 0.2.

7. The MATLAB code to generate the signal, and compare the plots of the generated signal and the input signal is shown in Listing 3. The two plots are shown in Figure 3.

The plot of the generated signal is identical to the plot of the input signal found in Part 3, which is expected as there is no noise in the input signal (as seen in Part 5) and the input signal is made entirely of the sum of the three determined sinusoids.

Listing 3: Generating signal from sinuosoids

```
part1q3;

sin1 = 0.4*sin(2*pi*2000*t);
sin2 = 0.2*sin(2*pi*5000*t);
sin3 = 0.2*sin(2*pi*6000*t);

generated_signal = sin1+sin2+sin3;

tiledlayout('flow');
nexttile; plot(msec_per_sec*t(1:numSamples), signal(1:
   numSamples));
title('Plot of Input Signal'); xlabel('time (ms)'); grid('
   minor');
nexttile; plot(msec_per_sec*t(1:numSamples),
   generated_signal(1:numSamples));
title('Generated Signal'); xlabel('time (ms)'); grid('minor
   ');
exportgraphics(gcf, '../Figures/part1q7.png');
```

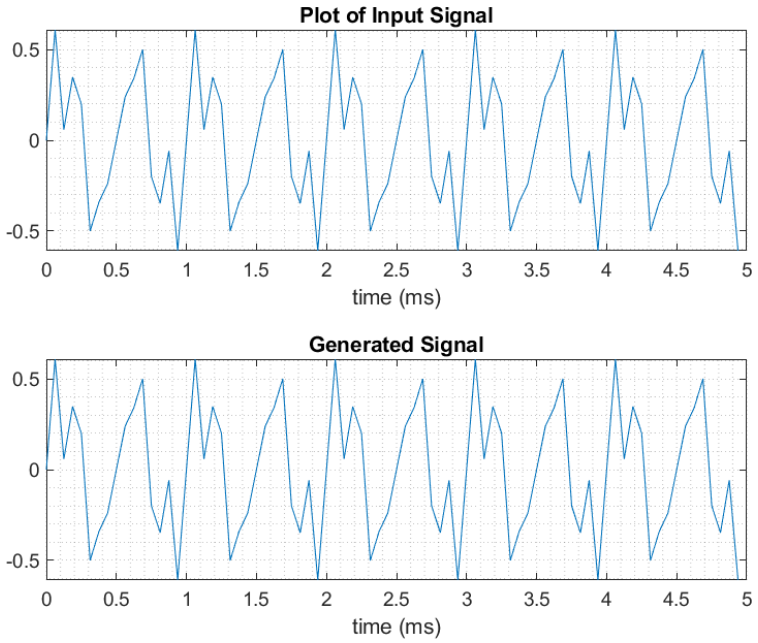

Figure 3: Comparison of two plots

Part 2

2. The signal is a very noisy sound, that changes to a slightly different tone every second. This behaviour of having a slightly difference tone every second persists for the entire signal.

3. The single-sided magnitude of the DFT was plotted using the same procedure described in Part I. We can determine the frequencies used in the signal by looking at the plot of the DFT shown in Figure 4. The frequencies used in the signal are: 1000 Hz, 2000 Hz, 3000 Hz, 4000 Hz, 5000 Hz, 6000 Hz, 7000 Hz, and 8000 Hz. The MATLAB code used to generate the DFT and save it to a file is shown in Listing 4.

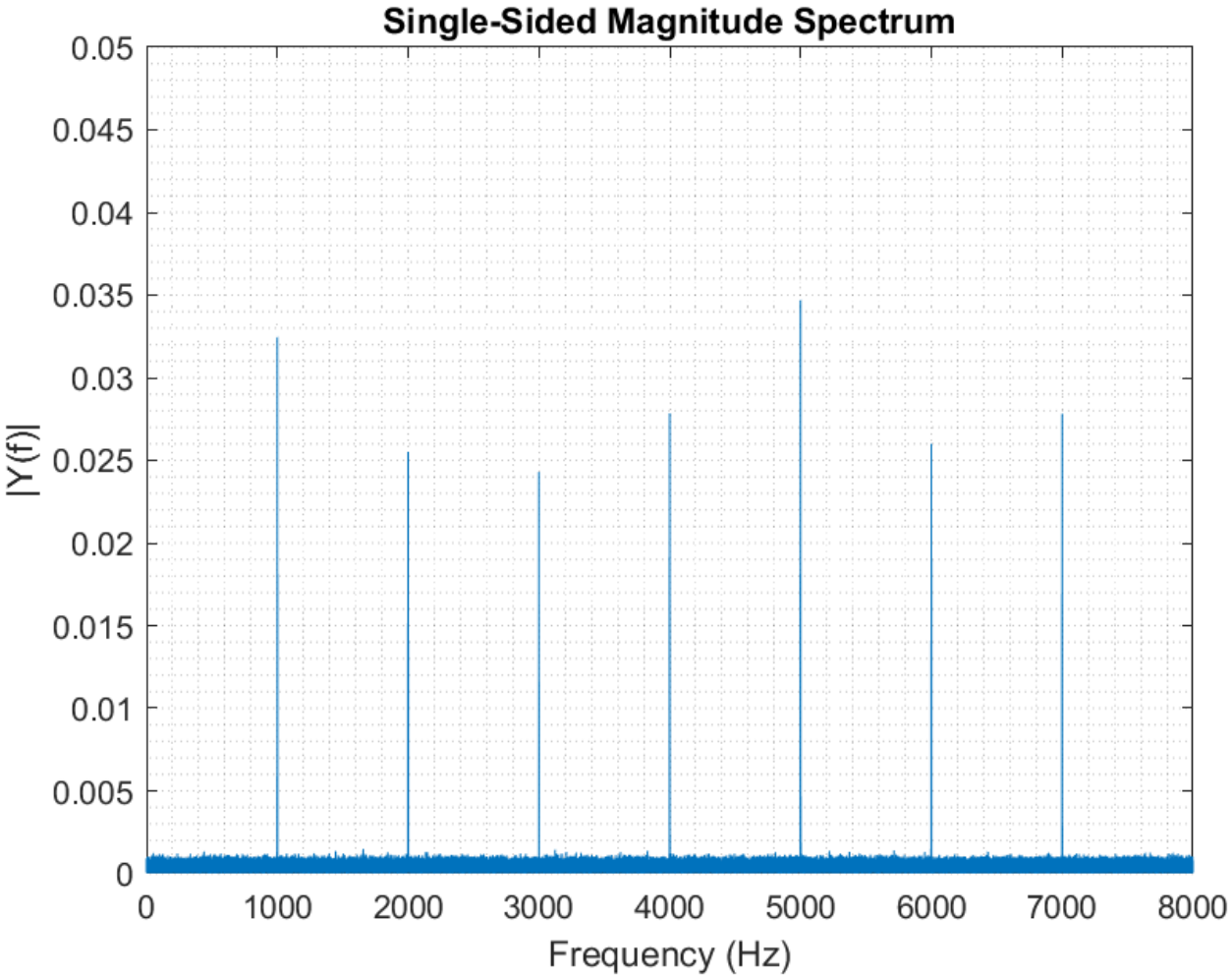

Figure 4: The DFT of the SecretMessage2020.wav file

Listing 4: Generate DFT of SecretMessage2020.wav file

```
1 [signal, Fs] = audioread('../SecretMessage2020.wav');
2 L = length(signal);
3 T = 1/Fs;
```

```
t = [0:L-1]*T;

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

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 .05]);
grid('minor');
exportgraphics(gcf, '../Figures/part2q3.png');
```

4. To decode the message, the signal was split up to each of its 1-second symbol periods in a loop, and the FFT of each symbol period was taken. The frequencies present in each period were determined using the maxk function, returning the indices (frequencies) of the four maximum values (maximum magnitudes) in the FTT. The frequencies were put into a 64 x 4 matrix, with each row containing the sorted 4 frequencies present in each of the 64 symbol periods. The MATLAB code used for this is shown in Listing 5.

Listing 5: Determing frequencies contained in each symbol period

```
[signal, Fs] = audioread('../SecretMessage2020.wav');
L = length(signal);
T = 1/Fs;
t = [0:L-1]*T;

freqs = zeros(64,4);

for i = 0:63
    Y = fft(signal(1+i*Fs:(i+1)*Fs))/(Fs);
    [B, I] = maxk(abs(Y(1:Fs/2+1)),4);
    freqs(i+1,:) = I-1;
end

freqs = sort(freqs,2);
```

The sorted frequencies are matched to the frequencies in the CodeBook.pdf file to decode and generate the message in the signal. A snippet of the MATLAB to perform this is shown in Listing 6. The decoded message is: "IT DOES NOT MATTER HOW SLOWLY YOU GO AS LONG AS YOU DO NOT STOP."

Listing 6: Matching frequency vectors with message characters to decode message

```
part2q4;

secret_message = "";
```

```
4
5  for i = 1:64
6      if freqs(i,:) == [1000 2000 3000 4000]
7          secret_message = secret_message + "A";
8      elseif freqs(i,:) == [1000 2000 3000 5000]
9          secret_message = secret_message + "B";
10     elseif freqs(i,:) == [1000 2000 3000 6000]
11         secret_message = secret_message + "C";
12     elseif freqs(i,:) == [1000 2000 3000 7000]
13         secret_message = secret_message + "D";
```