EEE391
Basics of Signals and Systems

MATLAB Assignment 1

Sabit Gökberk Karaca
21401862

Part 1: Voice Recording

In this part of the homework, I used a flute to generate pure musical notes. The note I chose was an A4 sound. While playing my instrument, I used MATLAB to record the audio. The code I used for that part is as follows;

```
recObj = audiorecorder(16000, 8, 1);
disp('Start recording.');
T = 3;
recordblocking(recObj, T);
disp('End of recording.');
```

Then, I played the sound I recorded and checked if it is the same as the record. After validating that, I extracted the data from sound record. I used the following code;

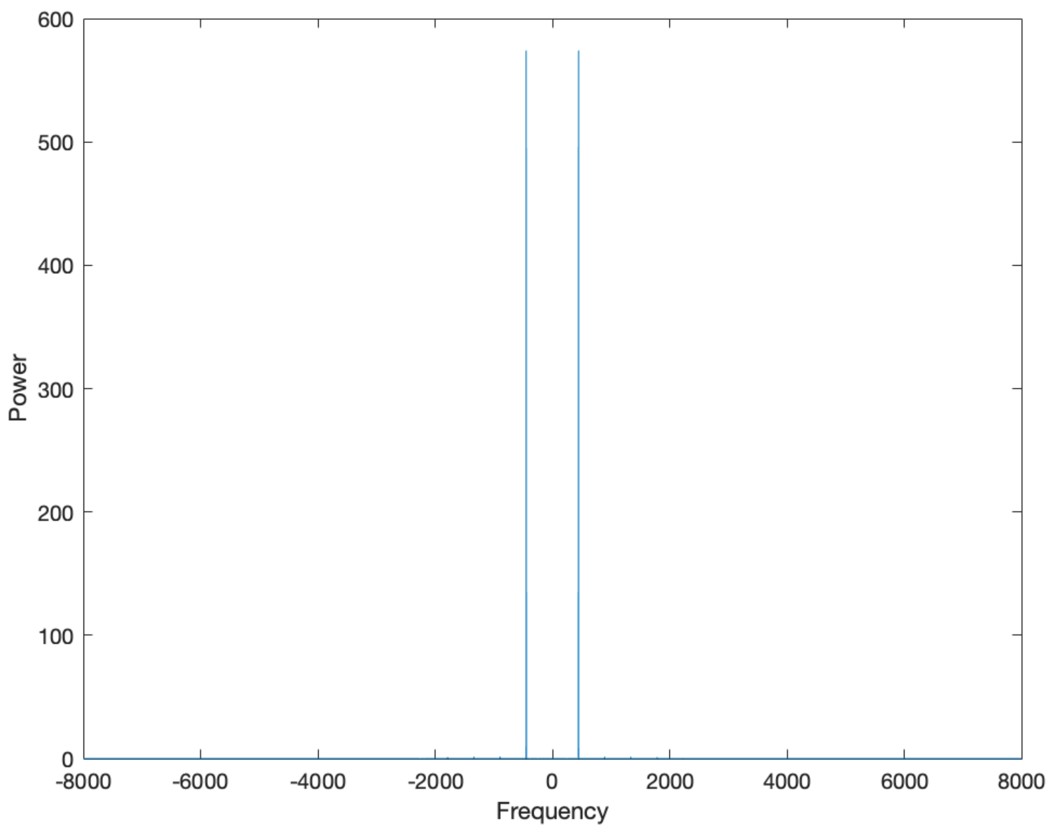
```
play(recObj);
soundArray = getaudiodata(recObj);
```

Part 2: Fundamental Period Calculation

After I recorded the sound and extracted data from the sound record, I needed to know the fundamental period of this signal. Since the note I played was an A4 sound, I checked online and learnt that A4 sound has the frequency 440Hz. Since I was going to use this knowledge in my calculations, I wanted to validate that the signal has 440Hz frequency. In order to do that, I chose Fast Fourier Transform method. I learnt how to use fft function of MATLAB from its documentation and applied the function to my data. The following code is taken from MATLAB documentation [1].

```
y = fft(soundArray);
fs = 16000;
ts = 1/fs;
n = length(soundArray);

y0 = fftshift(y);
f0 = (-n/2:n/2-1)*(fs/n);
power0 = abs(y0).^2/n;

plot(f0,power0)
xlabel('Frequency')
ylabel('Power')
```

Figure 1: Fast Fourier Transform Plot



The resulting graph shows that the frequency where the coefficient is not zero is 446.3Hz. Therefore theoretical knowledge about the frequency of A4 note is validated by using the data I have and Fast Fourier Transform.

After validating that the fundamental frequency is, I plotted soundArray and checked if the recording is normal or not. Although it looks normal, I took a part of it from the middle in order to make sure that there are no silent parts. The plots of of soundArray and partOfSoundArray signals can be found below. In the Figure 3, it is easier to see the periodic properties of the signal.

```
f0 = 446.3;
t0 = 1/f0;

t=linspace(0,3,length(soundArray));
plot(t, soundArray);
xlabel('Time (sec)')
ylabel('Amplitude')

numOfPeriods = 100;
numOfSamples = int64(numOfPeriods * fs / f0);
startingPoint = int64(1001);
endPoint = startingPoint + numOfSamples;
partOfSoundArray = soundArray(startingPoint:endPoint);
t = linspace(0, 100*t0, length(partOfSoundArray));
plot(t, partOfSoundArray);
```

Figure 2: soundArray Plot

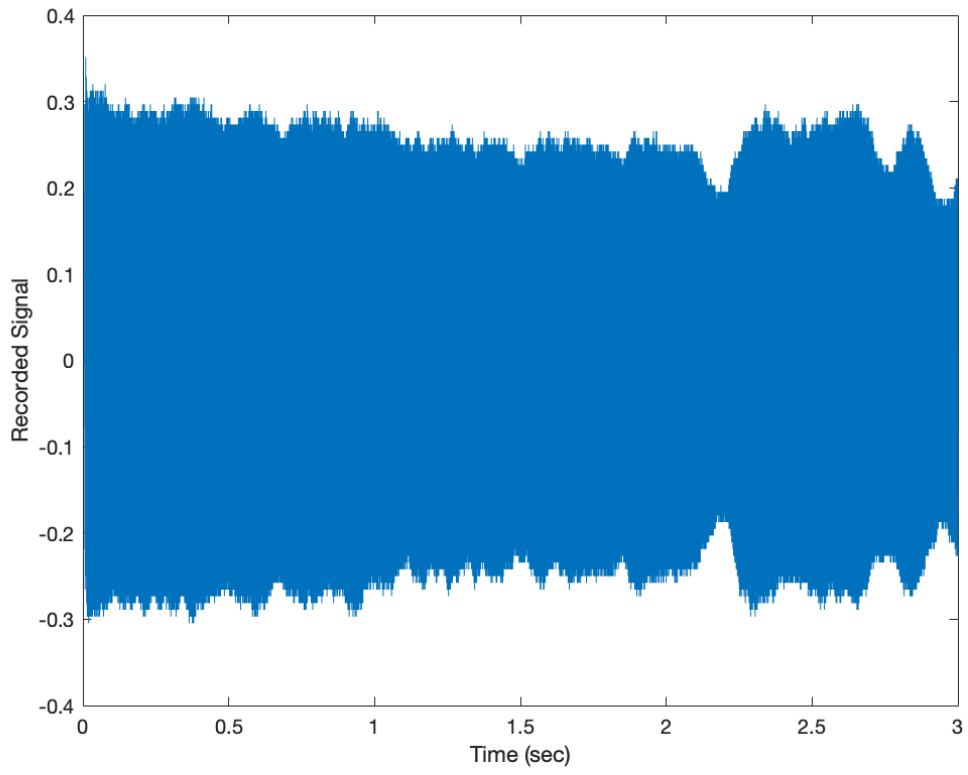


Figure 3: partOfSoundArray Plot

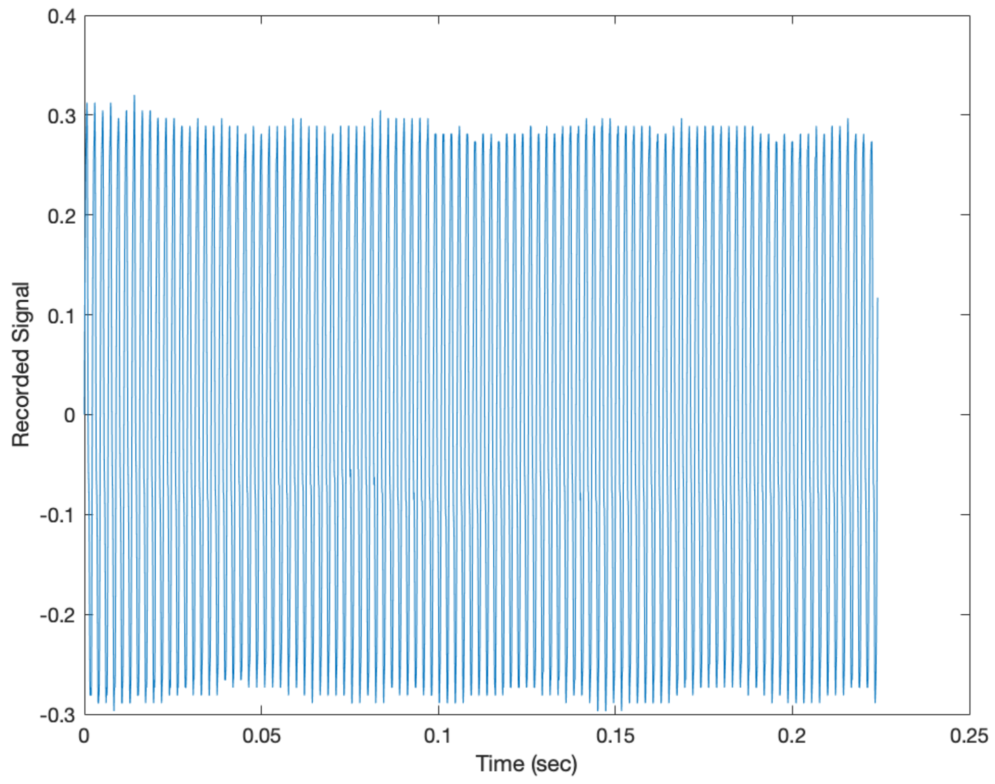
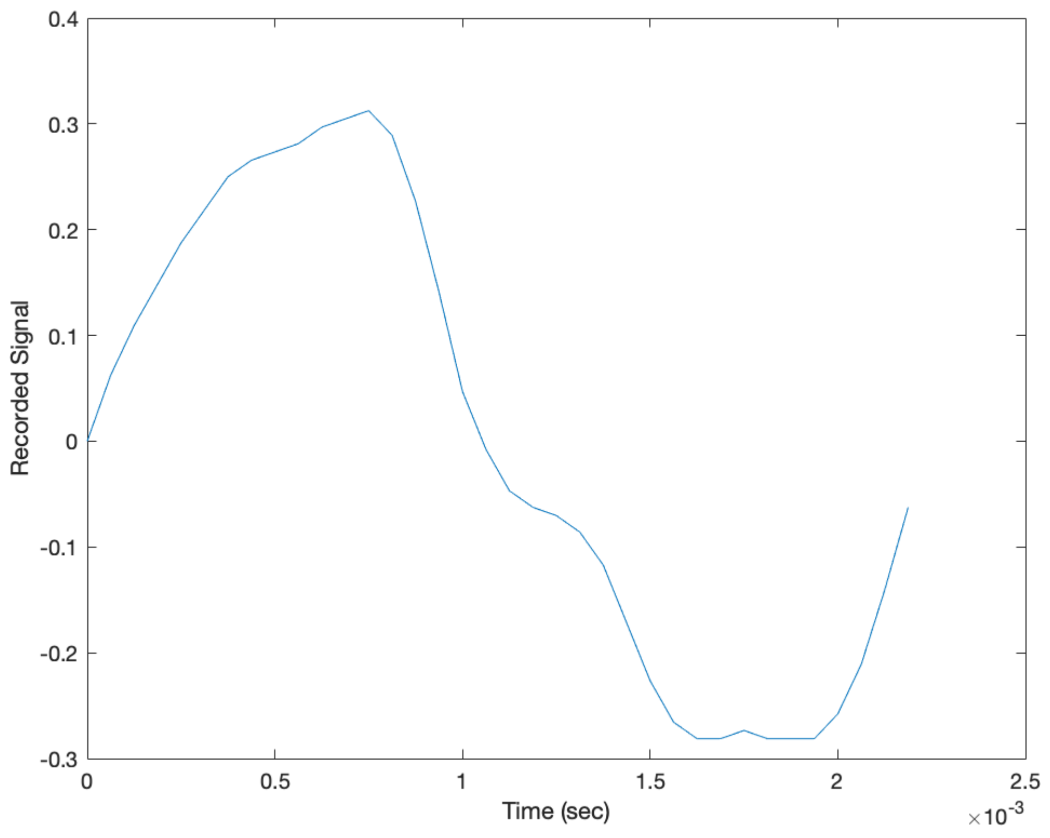


Figure 4: One period of recorded signal

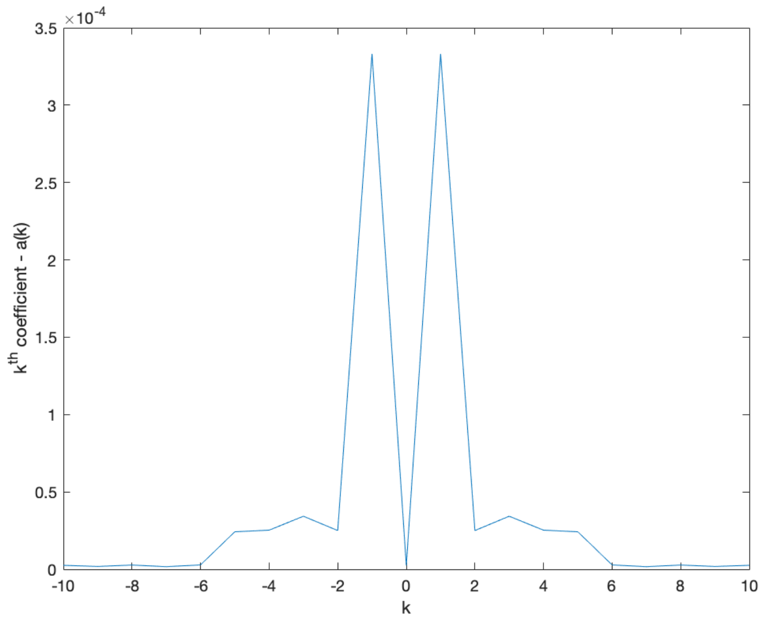


Part 3: Fourier Series Analysis

In that part, I extracted one period of the signal since the Fourier Series Analysis formula requires to integrate the function over one period. Then, I read the documentation of trapz function to learn about it. I also utilized from a similar example from the web [2]. After creating the firstPeriodArray and also the time vector t, I calculated a(k) for each k from -10 to 10. I found this interval for k experimentally. The code I wrote and the plot I obtained from the execution of that function can be found below;

```
firstPeriod = int64(fs/f0);
firstPeriodArray = partOfSoundArray(1:firstPeriod);

t = [0:length(firstPeriodArray)-1]*ts;
x = firstPeriodArray.';

for k=1:21
a(k)=trapz(t, x.*exp(-1i*2*pi*f0*(k-11)*t));
end
k=-10:10
plot(k, abs(a))
```

Figure 5: k – a(k) Plot



As it can be seen from the graph, the coefficients that are not 0 occurs at k = ± 1 and. Also there are some smaller coefficients at k = ± 2, ±3, ±4, ±5. Those values can be listed as;

a(±1) = 3.331e-04
a(±2) = 2.504e-05
a(±3) = 3.429e-05
a(±4) = 2.528e-05
a(±5) = 2.425e-05

Part 4: Fourier Series Synthesis

In that part, I used the coefficients from -5 to 5. Since others were quite small, I assumed that they are equal to zero. By using these coefficients, re-synthesized the signal by using Fourier Series Synthesis formula. After checking that one period of the synthesized signal is similar to one period of the original signal, I generated the 100 periods of that signal by adding the signals to each other since they are periodic. The code that I wrote for the synthesis can be found below, plots of the synthesized signal is also added to the report.

```
synthesized_x = 0;
for k = -5:5
    synthesized_x = synthesized_x + a(k+11) .* exp(-1i*2*pi*f0*k*t);
end
plot(t, synthesized_x)
ylabel('Synthesized Signal');
xlabel('time (sec)');

synthesized_100_periods_of_x = repmat(synthesized_x, 1, 100);
t = [0:length(synthesized_100_periods_of_x)-1]*ts;
plot(t, synthesized_100_periods_of_x);
ylabel('Synthesized Signal');
xlabel('time (sec)');
```

Figure 6: One period of synthesized signal

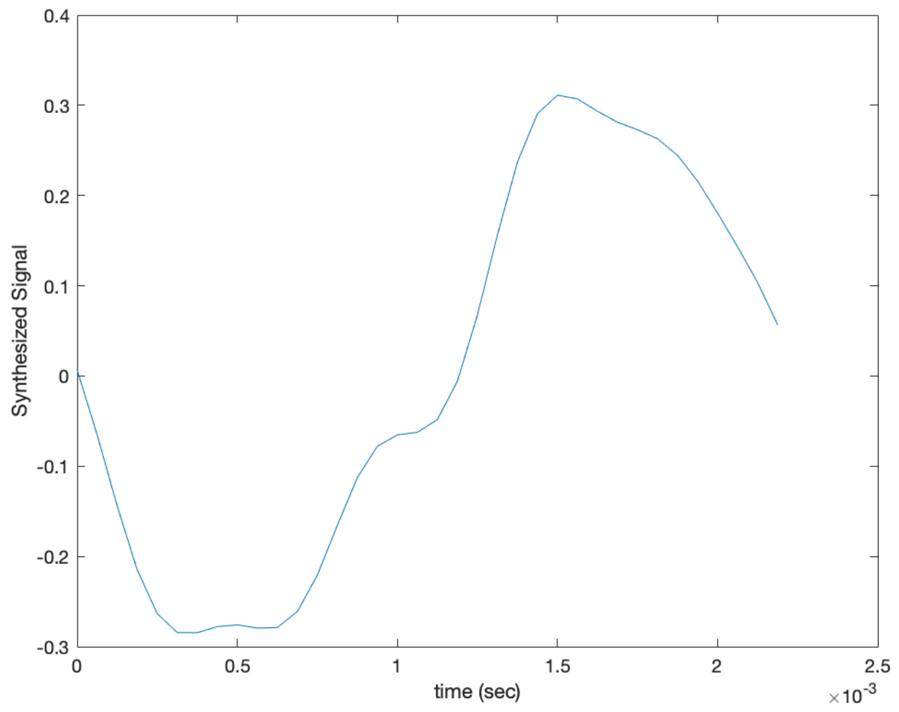
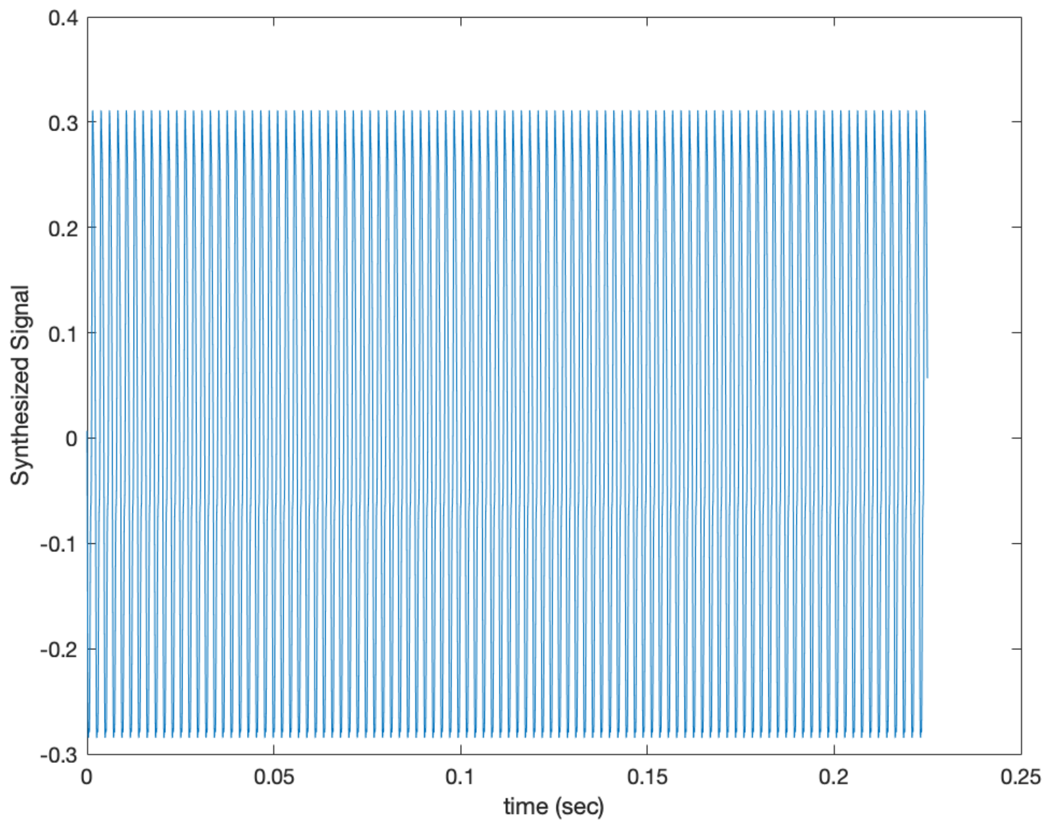


Figure 7: 100 periods of synthesized



Part 5: Voice File Generation

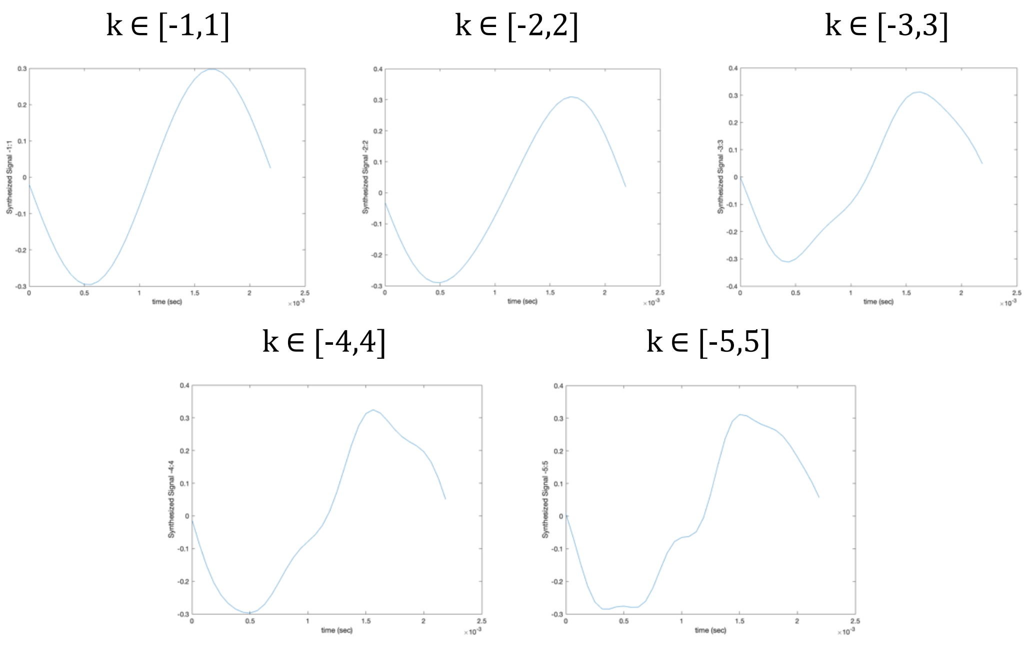
First, I studied the audiowrite function from MATLAB and generated a sound file from the data that I synthesized. Since the synthesized data has 100 periods of the signal, the sound file was quite short and hard to listen. Therefore, I generated one more signal with 1000 periods and generated a sound file from that signal too. Both sounds from 100 periods and 1000 periods signals will be included in the email to make it easier for you to listen.

```
filename = 'sytnhesized_signal_100_periods.wav';
audiowrite(filename, synthesized_100_periods_of_x, fs)

synthesized_1000_periods_of_x = repmat(synthesized_x, 1, 1000);
filename = 'sytnhesized_signal_1000_periods.wav';
audiowrite(filename, synthesized_1000_periods_of_x, fs)
```

Part 6: Voice Synthesis from FSCs

In that part, I repeated the synthesis process that I applied in Part 4 for different intervals of k. In that experiment, I observed that all of the generated signals have the same fundamental frequency but when I include more coefficients, the details in the signal become more evident. When I listened to the sounds of those signals, I realized that the signals including more coefficients sounds more similar to the originally recorded signal.

Figure 8: Signals with different intervals of k



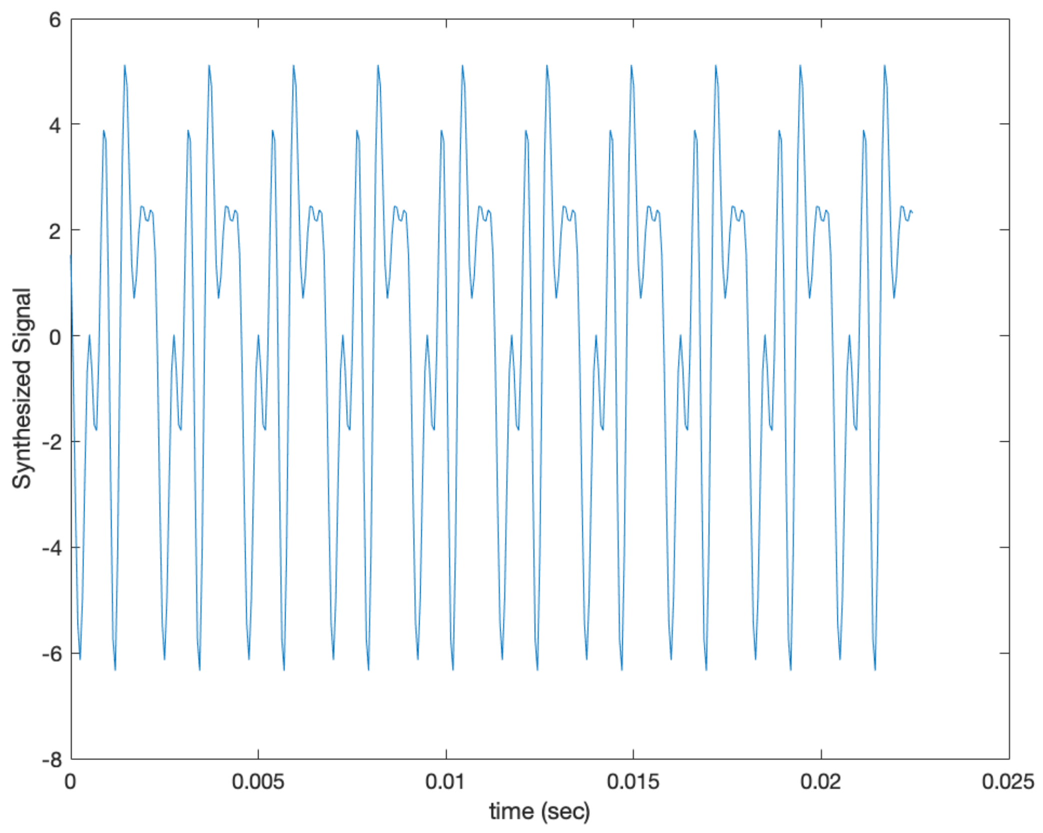
```
t = [0:length(firstPeriodArray)-1]*ts;
synthesized_x = 0;
for k = -1:1
    synthesized_x = synthesized_x + a(k+11) .* exp(-1i*2*pi*f0*k*t);
end
plot(t, synthesized_x)
ylabel('Synthesized Signal -1:1');
xlabel('time (sec)');

t = [0:length(firstPeriodArray)-1]*ts;
synthesized_x = 0;
for k = -2:2
    synthesized_x = synthesized_x + a(k+11) .* exp(-1i*2*pi*f0*k*t);
end
plot(t, synthesized_x)
ylabel('Synthesized Signal -2:2');
xlabel('time (sec)');

t = [0:length(firstPeriodArray)-1]*ts;
synthesized_x = 0;
for k = -3:3
    synthesized_x = synthesized_x + a(k+11) .* exp(-1i*2*pi*f0*k*t);
end
plot(t, synthesized_x)
ylabel('Synthesized Signal -3:3');
xlabel('time (sec)');

t = [0:length(firstPeriodArray)-1]*ts;
synthesized_x = 0;
for k = -4:4
    synthesized_x = synthesized_x + a(k+11) .* exp(-1i*2*pi*f0*k*t);
end
plot(t, synthesized_x)
ylabel('Synthesized Signal -4:4');
xlabel('time (sec)');

t = [0:length(firstPeriodArray)-1]*ts;
synthesized_x = 0;
for k = -5:5
    synthesized_x = synthesized_x + a(k+11) .* exp(-1i*2*pi*f0*k*t);
end
plot(t, synthesized_x)
ylabel('Synthesized Signal -5:5');
xlabel('time (sec)');
```

Part 7: Voice Synthesis from FSCs with Single Magnitude

Since I found that coefficients [-5, 5] contribute to the signal synthesis, I changed these coefficients' magnitude values to 1 and generated a new array b. By using the new coefficients, I synthesized the signal, checked its plot and listened to its sound. By looking at the generated plot, I found out that the fundamental frequency of that signal is also the same as the original signal. However, its components have different magnitudes and this situation results in different sounds and thus, two signals sound differently.

```
b = a(6:16);
for k = 1:11
    b(k) = 1.*exp(1i*angle(b(k)));
end
b(6) = a(11);
synthesized_x = 0;
for k = -5:5
    synthesized_x = synthesized_x + b(k+6) .* exp(-1i*2*pi*f0*k*t);
end
synthesized_10_periods_of_x = repmat(synthesized_x, 1, 10);
t = [0:length(synthesized_10_periods_of_x)-1]*ts;
plot(t, synthesized_10_periods_of_x)
ylabel('Synthesized Signal');
xlabel('time (sec)');
```

Figure 9: Synthesized Signal from FSCs with Single Magnitude

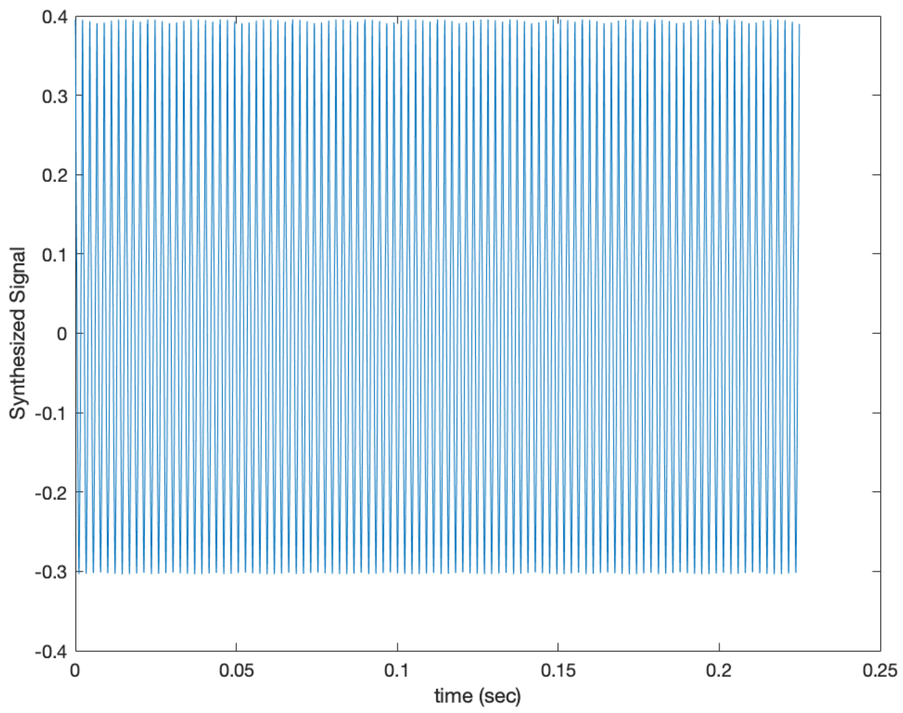


Part 8: Voice Synthesis from FSCs with Zero Phase

In that part, I did a very similar experiment to the previous one. Again, I created an array c and copied the coefficients a[-5, 5] to the new array. Then, I modified the signal by changing its coefficients. Instead of changing the magnitudes, I changed the phase of each coefficient this time. By doing these changes, I observed that magnitude of the generated signal is the same as the original one that is generated from a vector. On the other hand, frequency of the signal changed when I changed the phases of coefficients. Although I synthesized 10 periods from c vector, the generated signal is equivalent to 100 periods of the signal synthesized from a vector (Figure 7). Since these two signals are still periodic, they sound the same.

```
c = a(6:16);
for k = 1:11
    c(k) = abs(c(k)).*exp(0);
end
c(6) = a(11);
synthesized_x = 0;
for k = -5:5
    synthesized_x = synthesized_x + c(k+6) .* exp(-1i*2*pi*f0*k*t);
end
synthesized_10_periods_of_x = repmat(synthesized_x, 1, 10);
t = [0:length(synthesized_10_periods_of_x)-1]*ts;
plot(t, synthesized_10_periods_of_x)
ylabel('Synthesized Signal');
xlabel('time (sec)');
```

Figure 9: Synthesized Signal from FSCs with Zero Phase



References:

[1] “fft,” *Reconstructing an Image from Projection Data - MATLAB & Simulink Example*. [Online]. Available: https://www.mathworks.com/help/matlab/math/basic-spectral-analysis.html. [Accessed: 25-Nov-2018].

[2] “ECE 460 – Introduction to Communication Systems.” [Online]. Available: https://www.csun.edu/~skatz/ece460/matlab_tut_two.pdf. [Accessed: 25-Nov-2018].