Lab Assignment

Fourier Analysis using MATLAB

Sahand Sabour

EEE204

Continuous and Discrete Time Signals and Systems II

Student ID

1614650

Xi'an Jiaotong Liverpool University

Section 1

The questions for this section were simulated in MATLAB for both the square waveform and the triangular waveform. The obtained results for each question are provided respectively below.

Questions

1. **plot a graph of the signal $x(t)$ with T = 100ms, A = 1, sampling interval of 0.1 ms, and the observation window is (W_b, W_c) = (-0.2, 0.2)s, using the functions "square_wave_fun.m" and "triangular_wave_fun.m"**

   The code for plotting the square waveform is as follows:

```
% Plotting the square wave
A = 1;
T = 100e-3; % s
T_s = 0.1e-3; % s
W_b = -0.2; % s
W_e = 0.2; % s
[x, T_s_vct] = square_wave_fun(T, A, T_s, W_b, W_e);
plot(T_s_vct, x), xlabel('time'), ylabel('magnitude'),
grid on
ylim([-1.5, 1.5]);
```

   The result of executing the above code is displayed in the following figure (Figure 1).

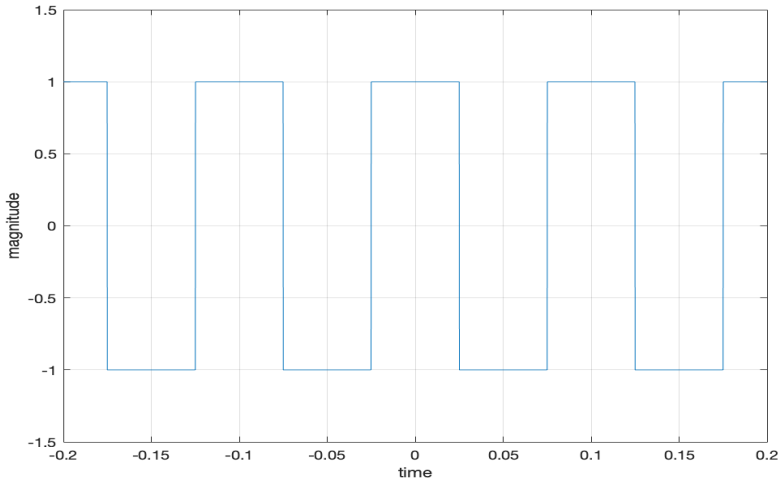

Figure 1: The graph of square waveform

The code for plotting the triangular waveform is as follows:

```
% Plotting the triangular wave
A = 1;
T = 100e-3; % s
T_s = 0.1e-3; % s
W_b = -0.2; % s
W_e = 0.2; % s
[x, T_s_vct] = triangular_wave_fun(T, A, T_s, W_b, W_e);
plot(T_s_vct, x), xlabel('time'), ylabel('magnitude'),
grid on
ylim([-1.5, 1.5]);
```

The resulting triangular waveform is displayed in Figure 2.

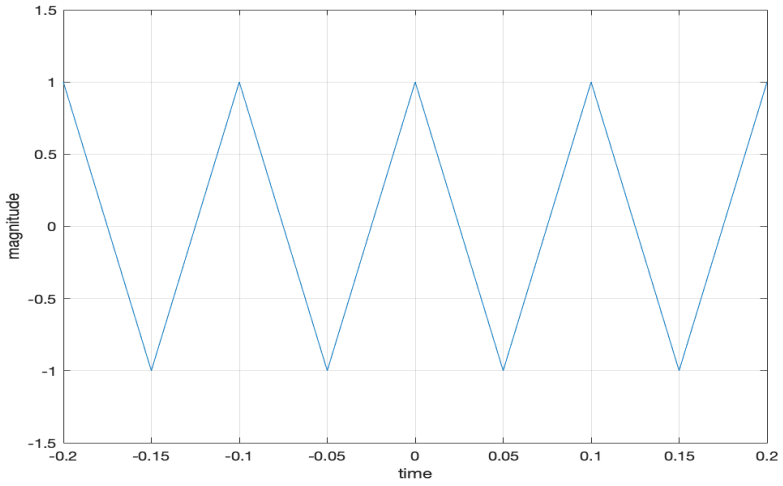

Figure 2: The graph of triangular waveform

2. **Plot a graph of the N-th partial sum $\tilde{x}(t)$ with T = 100ms, A = 1, and for N = [3,4,5,11], using the functions "square_wave_PFS _fun.m" and "triangular_wave_PFS_fun.m". Print the resulting waveforms and compare with the original signal $x(t)$, what do you observe about the behavior of the partial sum when N gets increased? what do you observe about the behavior of the partial sum of the discontinuities of $x(t)$? (compare the behavior of the triangular and square waveform)**

   The code for co-plotting the square waveforms corresponding to N, where N is chosen from [3,4,5,11], is as follows:

```
%Plotting the square waveform corresponding to N
A = 1;
T= 100e-3; % s
N=[3,4,5,11];
T_s = 0.1e-3; % s
W_b = -0.2; % s
W_e = 0.2; % s
index=1 % the value of N for which the graph is to be plotted
for i = N
[pfs_X, T_s_vct] = square_wave_PFS_fun(i, T, A, T_s, W_b, W_e);
subplot(2,2,index),
plot(T_s_vct, pfs_X), xlabel('time (s)'), ylabel('magnitude'),
legend(strcat('N = ', int2str(i))),
grid on
ylim([-1.5,1.5]);
index=index+1
end
```

The result of executing the above code displayed in the following figure:

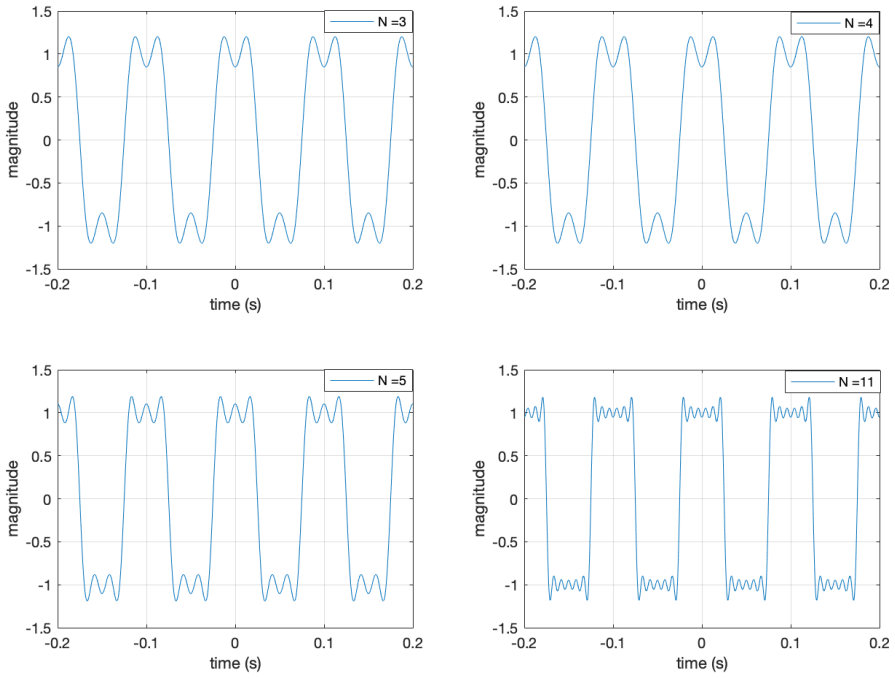

Figure 3: The graph of square waveform's N-th partial sum

The code for co-plotting the triangular waveforms corresponding to the same values of N is provided below.

```
%Plotting the triangular waveform corresponding to N
A = 1;
T= 100e-3; % s
N=[3,4,5,11];
T_s = 0.1e-3; % s
W_b = -0.2; % s
W_e = 0.2; % s
index=1 % the index of N for which the graph is to be plotted
for i = N
[pfs_X, T_s_vct]=triangular_wave_PFS_fun(i, T, A, T_s, W_b, W_e);
subplot(2,2,index),
plot(T_s_vct, pfs_X), xlabel('time (s)'), ylabel('magnitude'),
legend(strcat('N = ', int2str(i))),
grid on
ylim([-1.5,1.5]);
index=index+1
end
```

Figure 4 displays the obtained results for executing the above code.

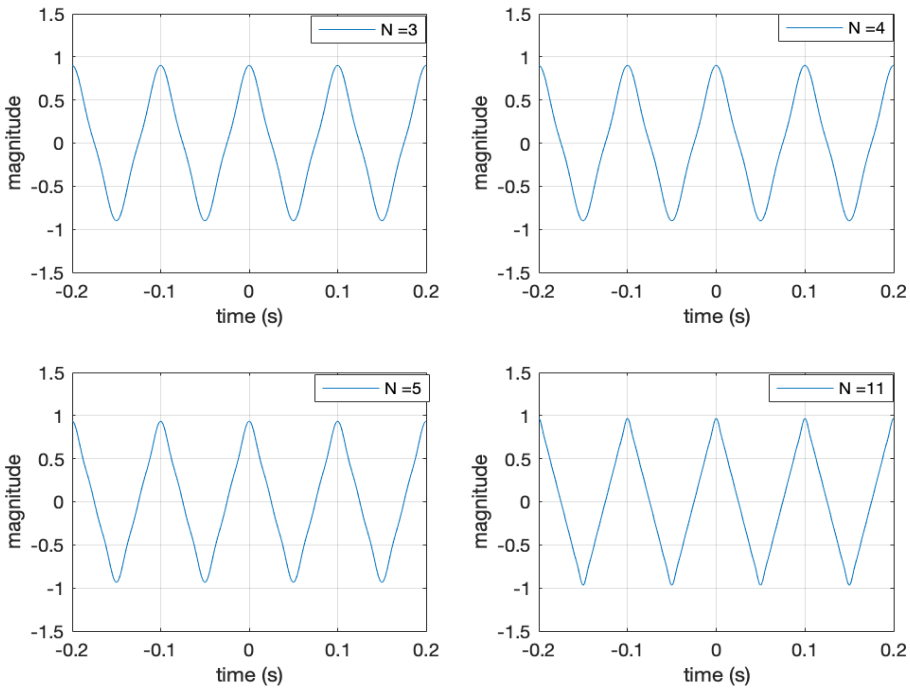

Figure 4: The graph of triangular waveform's N-th partial sum

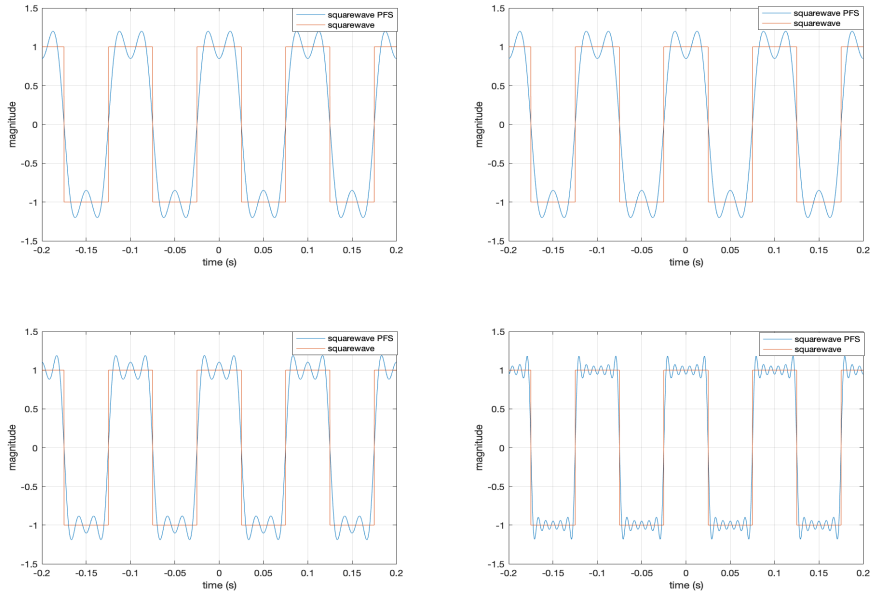

Figure 5: Comparison of $x(t)$ and $\tilde{x}(t)$ for square waveform

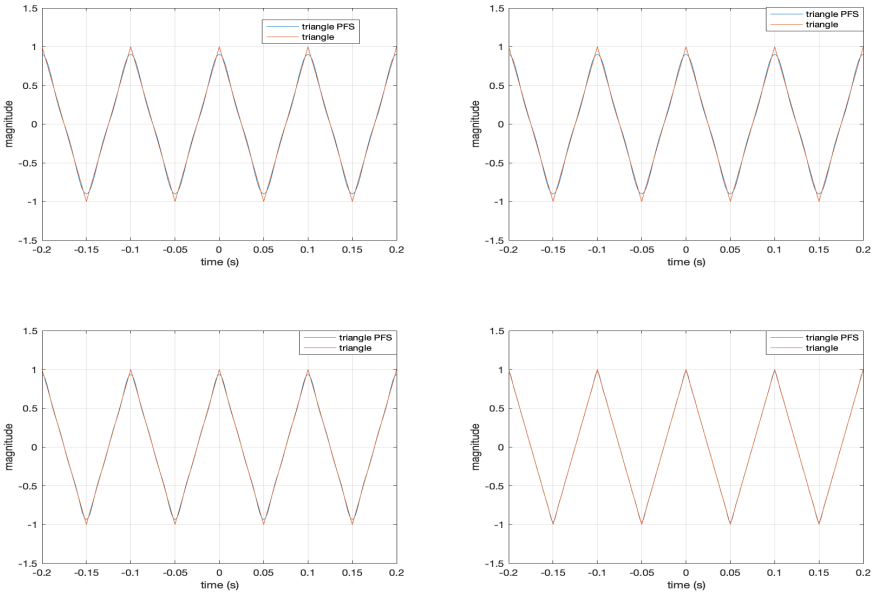

Figure 6: Comparison of $x(t)$ and $\tilde{x}(t)$ for triangular waveform

Based on the above figures (Figures 5 and 6), it can be noticed that the graph of the N-th partial sum $\tilde{x}(t)$ with the mentioned values of N resembles the original waveform $x(t)$ to an extent; however, this resemblance correlates with the value of N as it increases significantly with larger values of N.
Moreover, upon further investigation of the partial sum's behavior, it can be noticed that the mentioned behavior is slightly different in square and triangular waveforms near the discontinuity points of $x(t)$. As it is shown in Figure 3, the change in the magnitude of the waveform at the discontinuity points occurs instantly, whereas this change takes approximately 0.02 seconds to occur when N = 3 and is decreased with larger values of N.
However, that is not the case with the triangular waveform. As illustrated in Figure 4, the rise and decline in the waveform occurs at the approximately the same rate in the triangular waveform $x(t)$ and the N-th partial sum $\tilde{x}(t)$. In addition, comparison of the overshoot between the two waveforms shows much larger overshoot in the square waveform's N-th partial sum whereas there is approximately zero overshoot when a triangular waveform is implemented.

3. **Plot the signal $y(t)$ obtained by filtering the signal $x(t)$ (T = 100 ms and A =1) using the low-pass filter represented in Fig. 4. Print the resulting waveform and observe the large oscillations near the jump discontinuity (what is the name of this phenomena?). Evaluate the overshoot for the square waveform as a $= \tilde{x}(t_0^+) - \tilde{x}(t_0^-)$, with $\tilde{x}(t_0^+)(\tilde{x}(t_0^-))$ is the first maximum (minimum) value of $\tilde{x}$ to occur near the jump discontinuity point. Compare the evaluated overshoot value with the theoretical one.**

   Since N $= \frac{f_c}{f}$, where $f = \frac{1}{T}$, then the N to be used for this question would be N $= \frac{1250}{\frac{1}{0.1}} = \frac{1250}{10} = 125$. Accordingly, the code for the low pass filter of a square waveform is as follows:

```
%low pass filter for square waveform
N = 125;
A = 1;
T= 100e-3; % s
T_s = 0.1e-3; % s
W_b = -0.2; %s
W_e = 0.2; %s
[pfs_X, T_s_vct] = square_wave_PFS_fun(N, T, A, T_s, W_b, W_e);
sys= abs(max(pfs_X))-1;
disp(sys)
plot(T_s_vct, pfs_X), xlabel('time'), ylabel('magnitude'),
grid on ylim([-1.5, 1.5]);
```

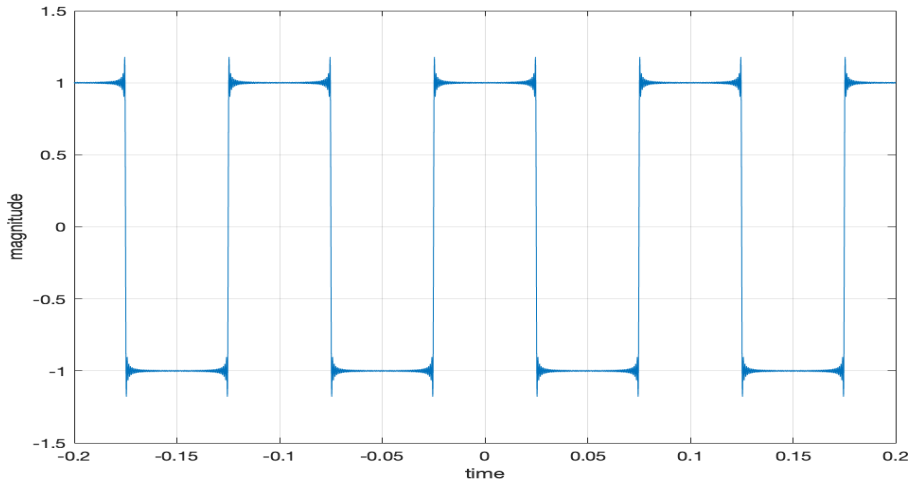

Figure 7: The graph of square waveform (low-pass filter)

The overshoot for this waveform was calculated by MATLAB as 0.1789.

As shown in the above figure (Figure 7), large oscillations near the jump discontinuity points can be observed. This phenomenon is referred to as the "Gibbs phenomenon", which states that the although the overshoot at the jump discontinuity point would be decreased with addition of more terms (larger values of N) but it will approach a finite limit and thus, will not be removed. In theory, the value of this finite limit ($\int_0^{\pi} \frac{sint}{t} dt$) is calculated to be approximately 1.851937051982; this is known as the Wilbraham–Gibbs constant. Consequently, as A = 1 in this experiment, the theoretical value for the overshoot would be $|\ \frac{1}{2}(1.851937051982) - 1\ | \approx 0.074$, which is significantly close to the evaluated overshoot value (=0.1789)

Moreover, the following is the code for plotting the triangular waveform through a low-pass filter.

```
%low pass filter for triangular waveform
N = 125;
A = 1;
T= 100e-3; % s
T_s = 0.1e-3; % s
W_b = -0.2; %s
W_e = 0.2; %s

[pfs_X, T_s_vct]=triangular_wave_PFS_fun(N, T, A, T_s, W_b, W_e);
sys= abs(max(pfs_X))-1;
disp(sys)
```

```
plot(T_s_vct, pfs_X), xlabel('time'), ylabel('magnitude'),
grid on
ylim([-1.5, 1.5]);
```

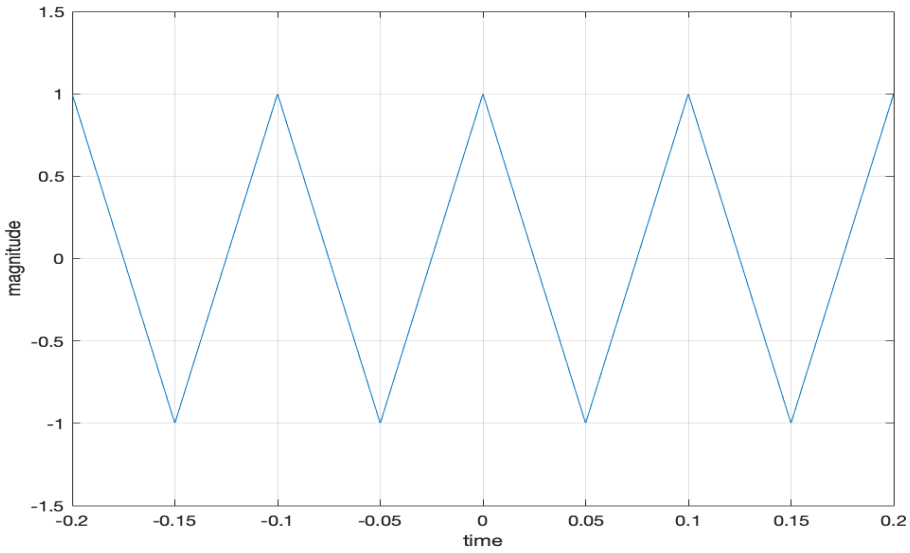

Figure 8: The graph of triangular waveform (low-pass filter)

The overshoot for this waveform was calculated by MATLAB as 0.0032. According to the mentioned phenomenon, the theoretical value of triangular waveform's overshoot is 0. Hence, the obtained results are verified.

4. **Plot the amplitude and phase of the harmonics of $x(t)$, in the frequency range (-1250, 1250) Hz, with T = 100ms, A =1. Print the resulting spectrum. How many harmonics are there in the plotted range of frequency? Note the higher frequency content of the square waveform.**

   Based on the given frequency range, FS would be set to 1250-(-1250)Hz = 2500Hz. The following code is used to plot the phase and amplitude spectrum of a square waveform.

```
N = 125;
A = 1;
T= 100e-3; % s
T_s = 0.1e-3; % s
W_b = -0.2; %s
W_e = 0.2; %s
```

```
FS= 2500;
[pfs_X, T_s_vct] = square_wave_PFS_fun(N, T, A, T_s, W_b, W_e);
getSpectrum(fft(pfs_X), FS)
```

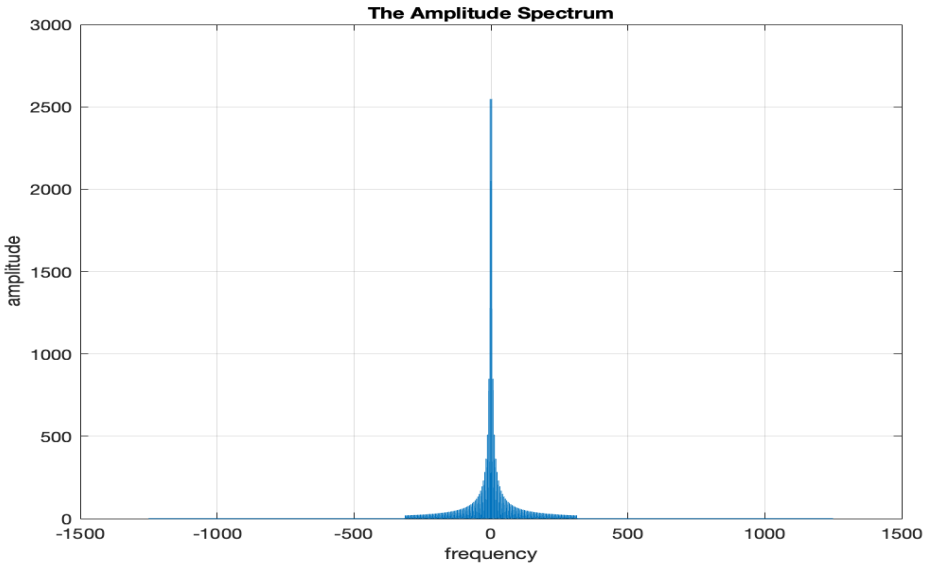

Figure 9: The amplitude spectrum of square waveform

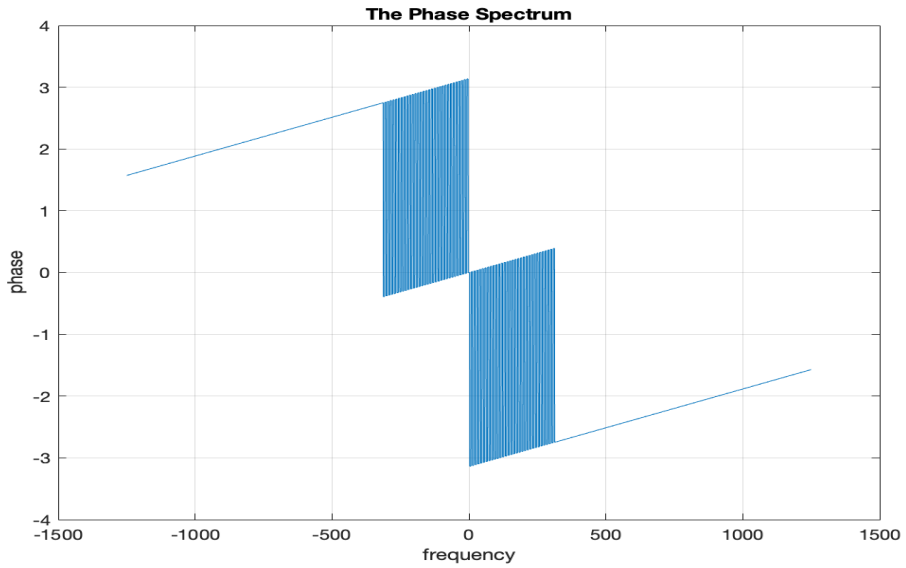

Figure 10: The phase spectrum of square waveform

Moreover, the following could used to plot the triangular waveform's phase and amplitude spectrum.

```
N = 125;
A = 1;
T= 100e-3; % s
T_s = 0.1e-3; % s
W_b = -0.2; %s
W_e = 0.2; %s
FS= 2500;
[pfs_X, T_s_vct]=triangular_wave_PFS_fun(N, T, A, T_s, W_b, W_e);
getSpectrum(fft(pfs_X), FS)
```

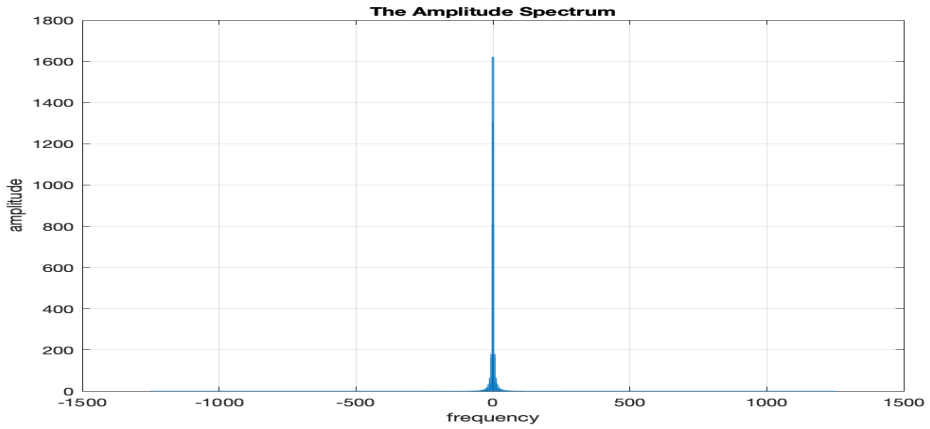

Figure 11: The amplitude spectrum of triangular waveform

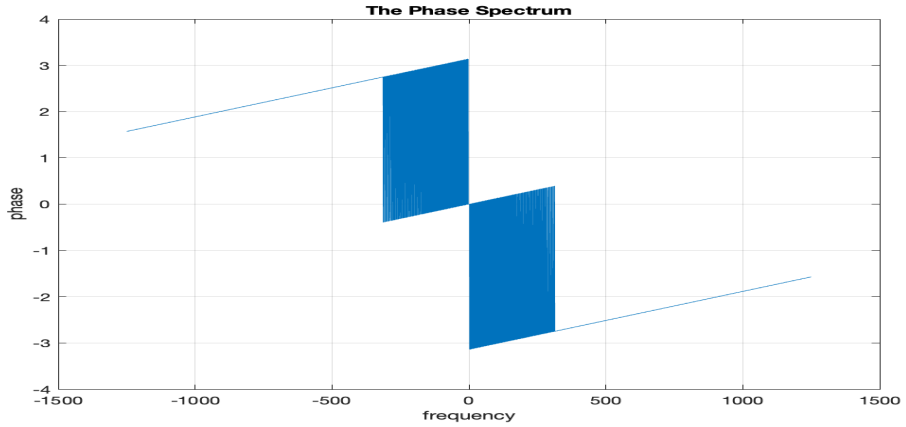

Figure 12: The phase spectrum of triangular waveform

Based on the previous calculation of N (N=125), there are N+1 = 126 harmonics in the plotted range of frequency.

Section 2

After loading the audio signal "music.wav" into the MATLAB environment, by using the *audioread* function. Consequently, a variable $x_{20}[n]$ containing the first 20 seconds of the loaded sound data was created. The obtained results for each question are provided respectively below.

Questions

1. **Use the command *audioplayer* and *play* to play the variable $x_{20}[n]$.**

```
% Code for loading the sound file and playing x20[n]
[y, Fs]=audioread('music.wav');
x_20= [1, 20*Fs];
[y, Fs]=audioread('music.wav', x_20);
player= audioplayer(y, Fs);
play(player);
```

2. **use the command *fft* to evaluate the DFT of the signal $x_{20}[n]$, plot the amplitude and phase spectrum using the command *plotSpectrum (estimator)* or the provided function *getSpectrum (f, FS)*. Print the amplitude spectrum of $x_{20}[n]$ and note that the low frequency components (up to 4000 Hz) carry most of signal energy.**

   The following code would plot the amplitude and phase spectrum of $x_{20}[n]$.

```
[y, Fs]=audioread('music.wav');
x_20= [1, 20*Fs];
[y, Fs]=audioread('music.wav', x_20);
getSpectrum(fft(y), Fs)
```

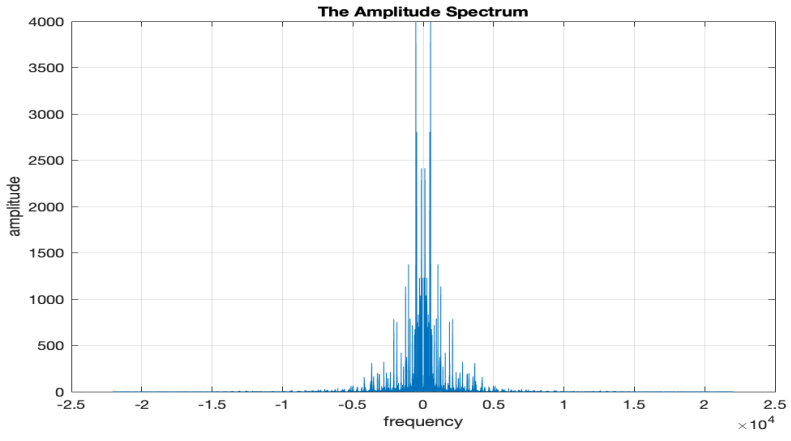

Figure 13: The amplitude spectrum of $x_{20}[n]$

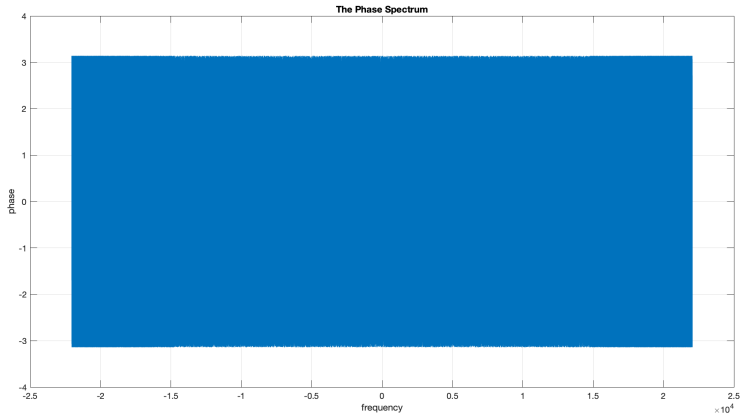

Figure 14: The phase spectrum of $x_{20}[n]$

3. **Scaling property (contracting and stretching signal): Generate a vector $x_d[n]$ by down-sampling the vector $x_{20}[n]$ by a factor of four. Using the original value of the sampling frequency, play the content of x_d by the command *play*, what did you notice? Use the command *fft* to evaluate the DFT of $x_d[n]$, plot the amplitude and phase spectrum using the command *plotSpectrum(estimator)* or *getSpectrum(f, FS)*. Print the amplitude spectrum of $x_d[n]$ and compare it with the amplitude spectrum of the original signal. What will happen to the signal spectrum when it is played faster?**

```
[y, Fs]=audioread('music.wav');
x_20= [1, 20*Fs];
[y, Fs]=audioread('music.wav', x_20);
y_1= downsample(y, 4);
getSpectrum(fft(y_1), Fs);
player_1= audioplayer(y_1, Fs);
play(player_1);
```

When playing the modified version of the audio file (the down-sampled version), it can be noticed the audio is being played at a much faster pace and higher frequency. The amplitude and phase spectrum of the modified audio signal are displayed respectively in the following two figures.

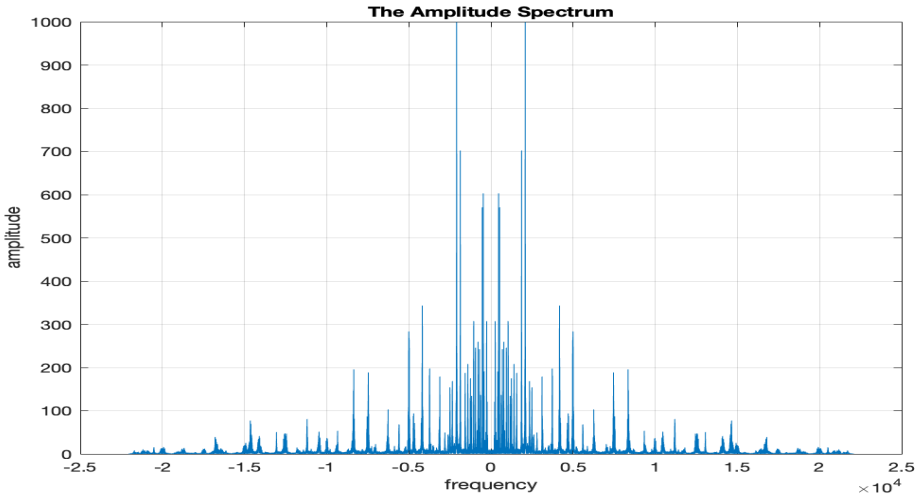

Figure 15: The amplitude spectrum of $x_{20}[n]$

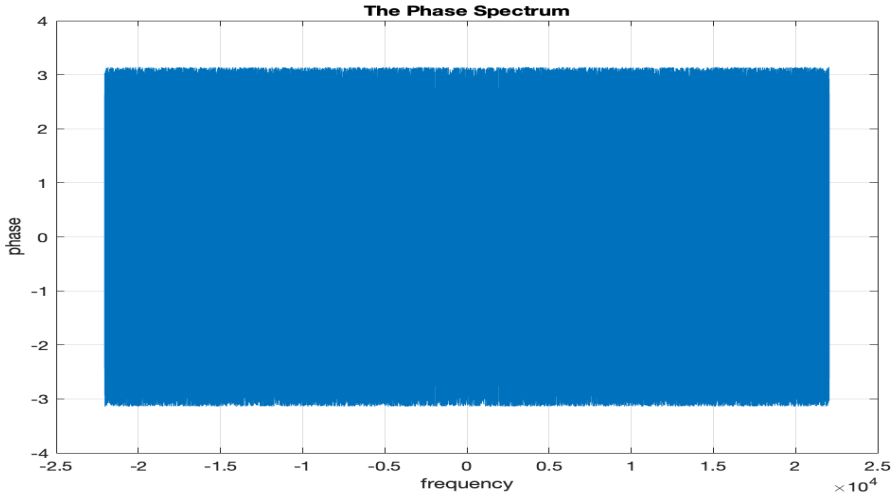

Figure 16: The phase spectrum of $x_{20}[n]$

By comparison between figures 13 and 15, it can be noticed that by down sampling the the audio signal, its frequency has significantly increased while its amplitude has considerably decreased. However, it can also be noticed the signal phase has not changed.