EEE 391
MATLAB Assignment 1

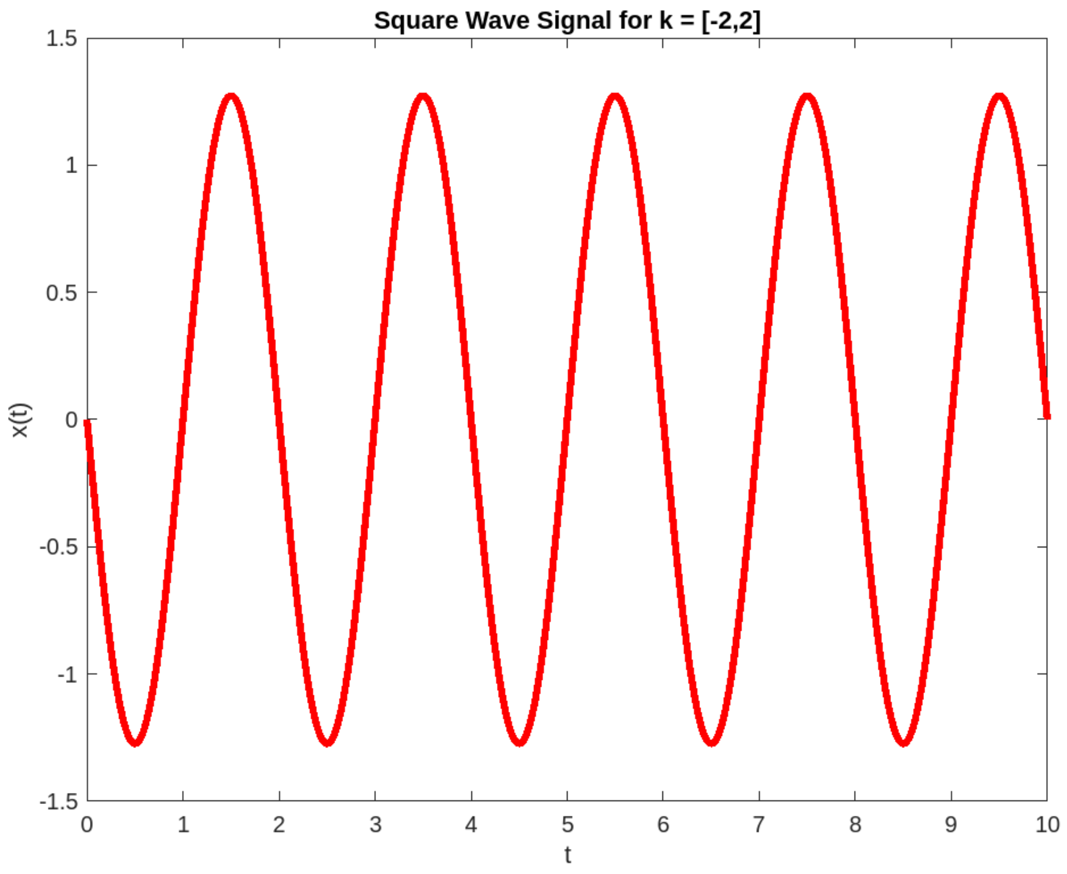
Full Name: Arda İynem
ID: 22002717
Section: 2

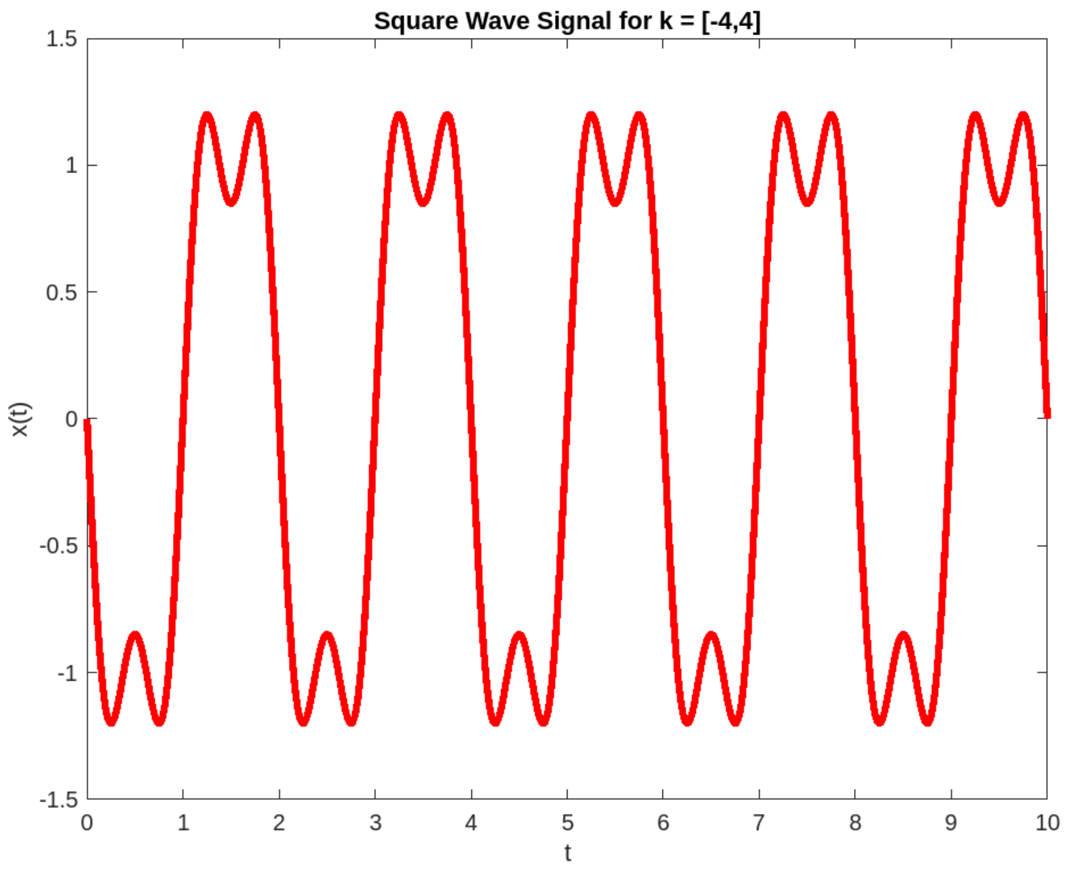
Part 1 (a)

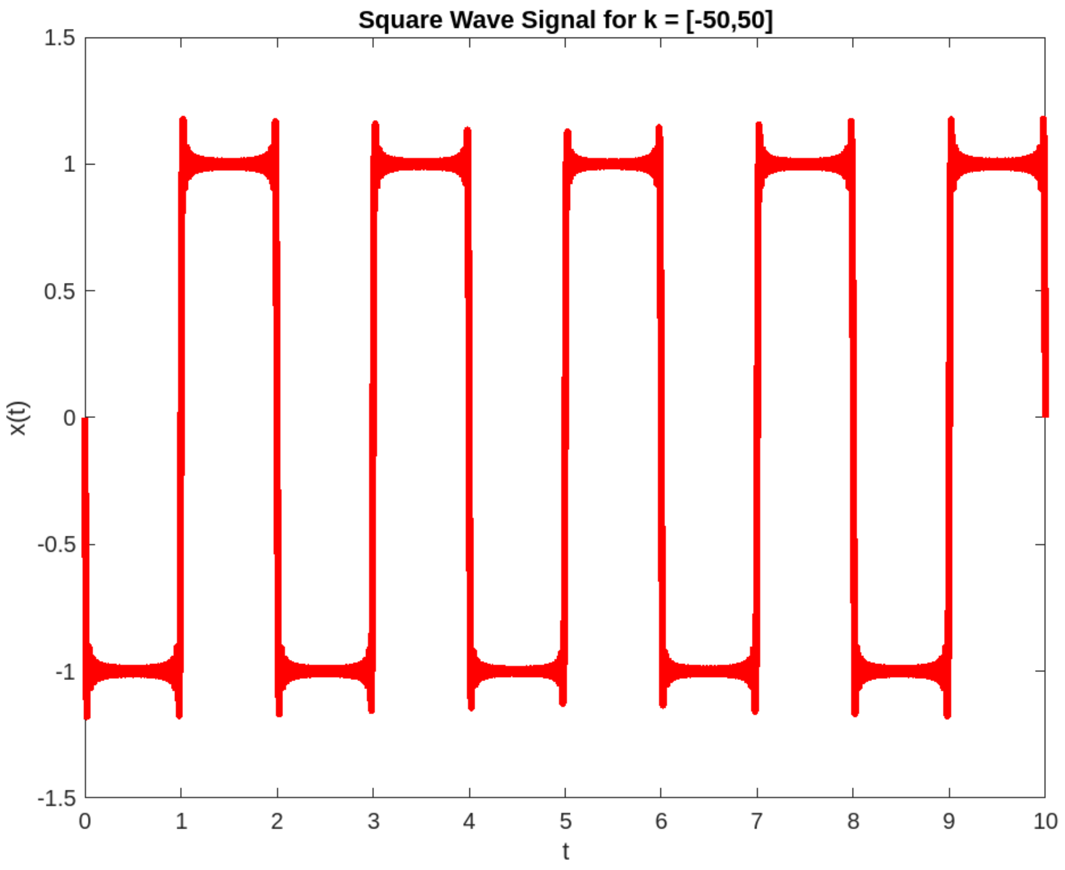
MATLAB Code of Square Wave Signal

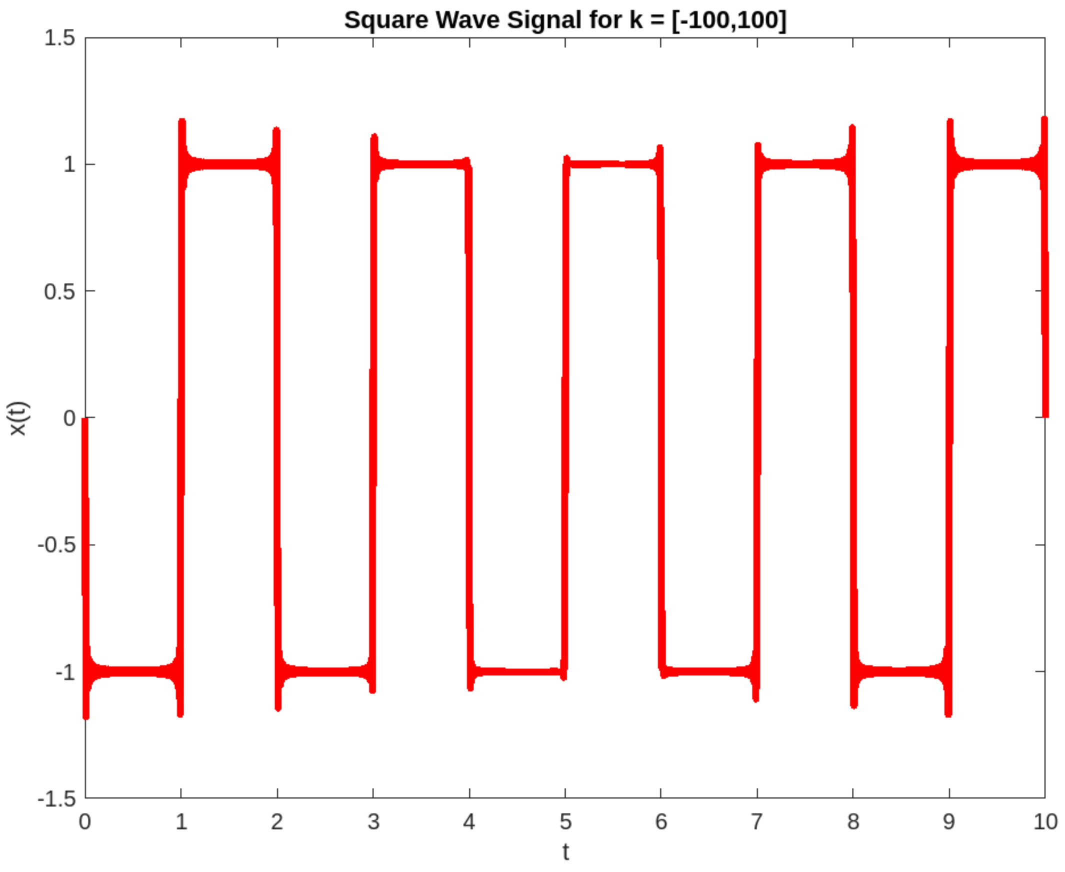
```
period = 2;
frequency = 1 / period;
K = 2;
coefAmount = 2 * K + 1;
ak = zeros(1, coefAmount);

for k = -K: K
   fourierFunc = @(t) ( -1 * (t > 0) + 1 * (t <= 0) ) .*
exp(-1j * 2 * pi * frequency * k * t);
   ak(k + (K + 1)) = integral(fourierFunc, -period / 2, period
/ 2) / period;
end

time_plot = linspace(0, 10, 1000);
reconstructedSignal = zeros(size(time_plot));

for k = -K: K
   reconstructedSignal = reconstructedSignal + ak(k + (K + 1))
.* exp(1j * 2 * pi * frequency * k * time_plot);
end

figure;
plot(time_plot, real(reconstructedSignal), 'r-', 'LineWidth',
3);
xlabel("t");
ylabel("x(t)")
title(['Square Wave Signal for k = [' num2str(-K) ','
num2str(K) ']']);
```

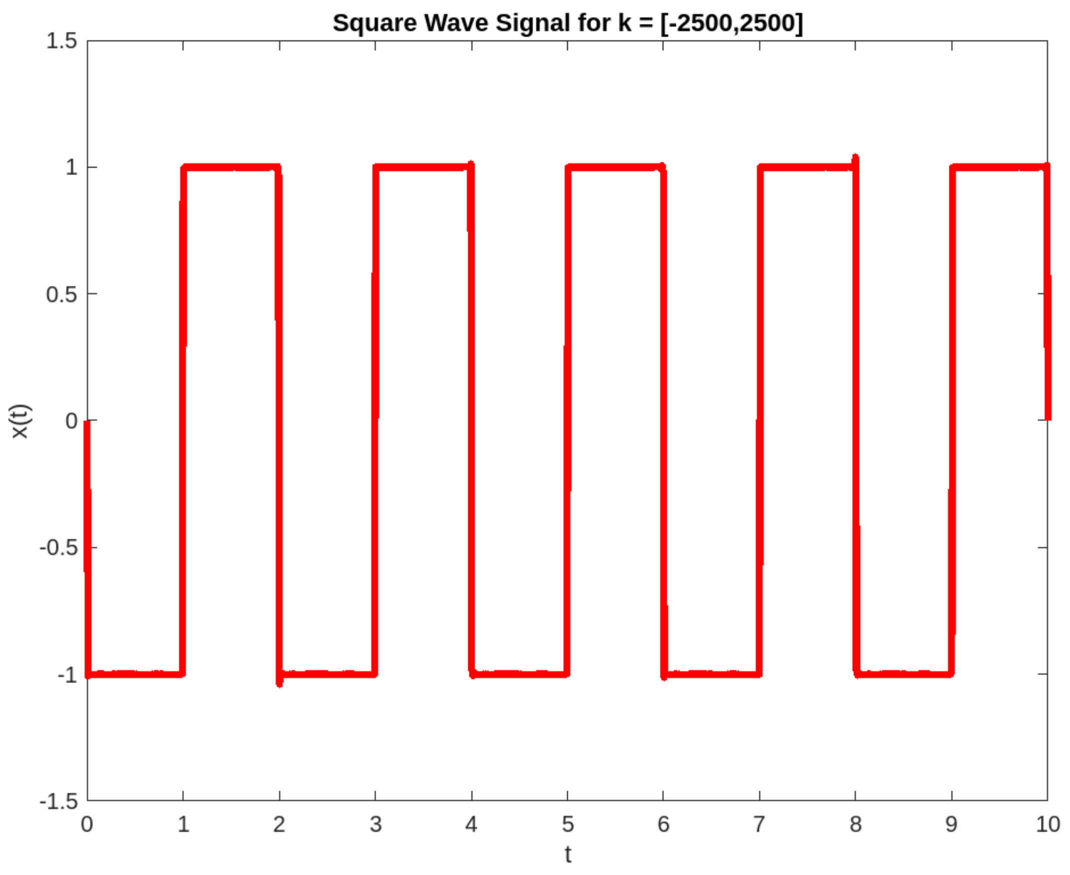
Figures of Square Wave Signal






Square Wave Signal for k = [-50,50]
1.5
1
0.5
0
-0.5
-1
-1.5
x(t)
0
1
2
3
4
5
6
7
8
9
10
t


Square Wave Signal for k = [-100,100]
1.5
1
0.5
0
-0.5
-1
-1.5
x(t)
0
1
2
3
4
5
6
7
8
9
10
t



Outcomes of Square Wave Signal Experiment

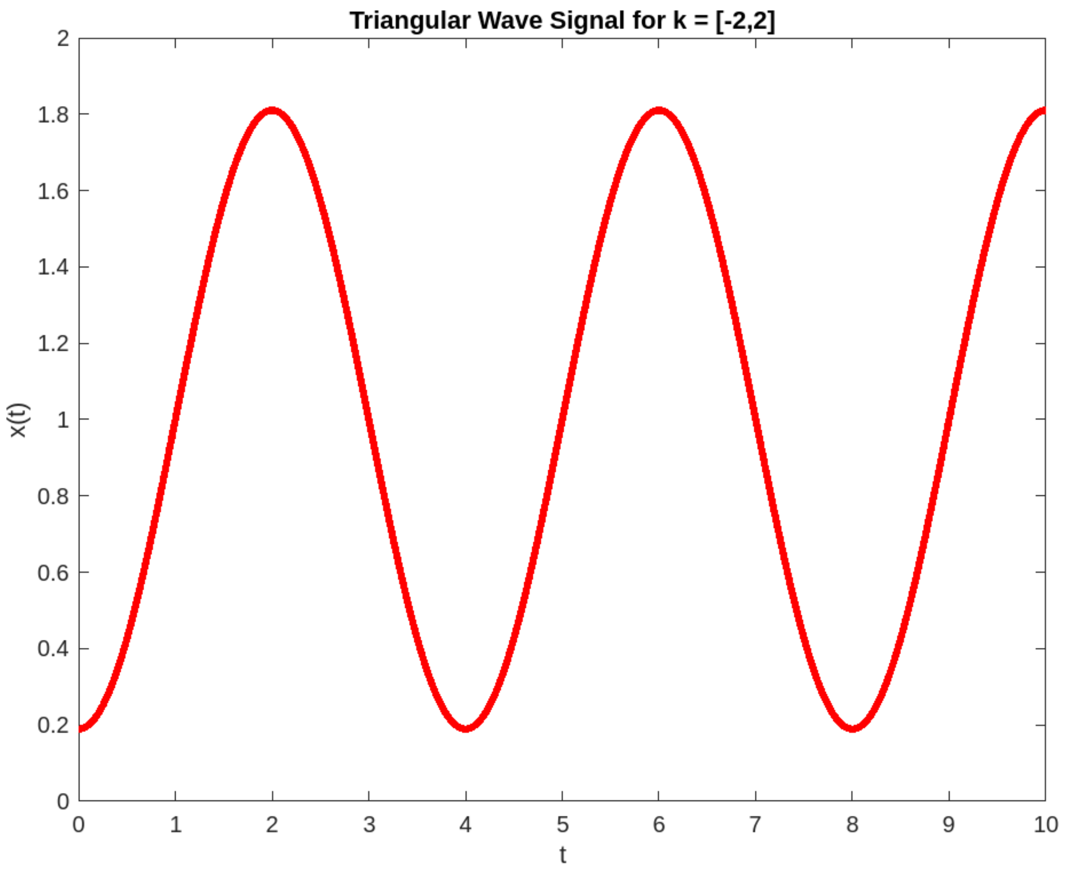
As the figures clearly depict, small values for k results in a signal wave that looks like a sinusoidal wave more than a square wave. When the value of k slightly increases the resulting signal is still a periodic function that does not clearly depict the characteristics of the square function. Only when the value of k becomes significantly larger relative to the prior values, the signal starts to look like a square signal wave. This outcome is consistent with the claims of the Fourier as his formula mentions that any periodic signal can be constructed with an infinite amount of signals with the frequency that is integer multiple of the fundamental frequency. Therefore, as k increases, the approximation error decreases; and when k goes to infinity, the resulting signal will be equivalent to the original square wave signal.

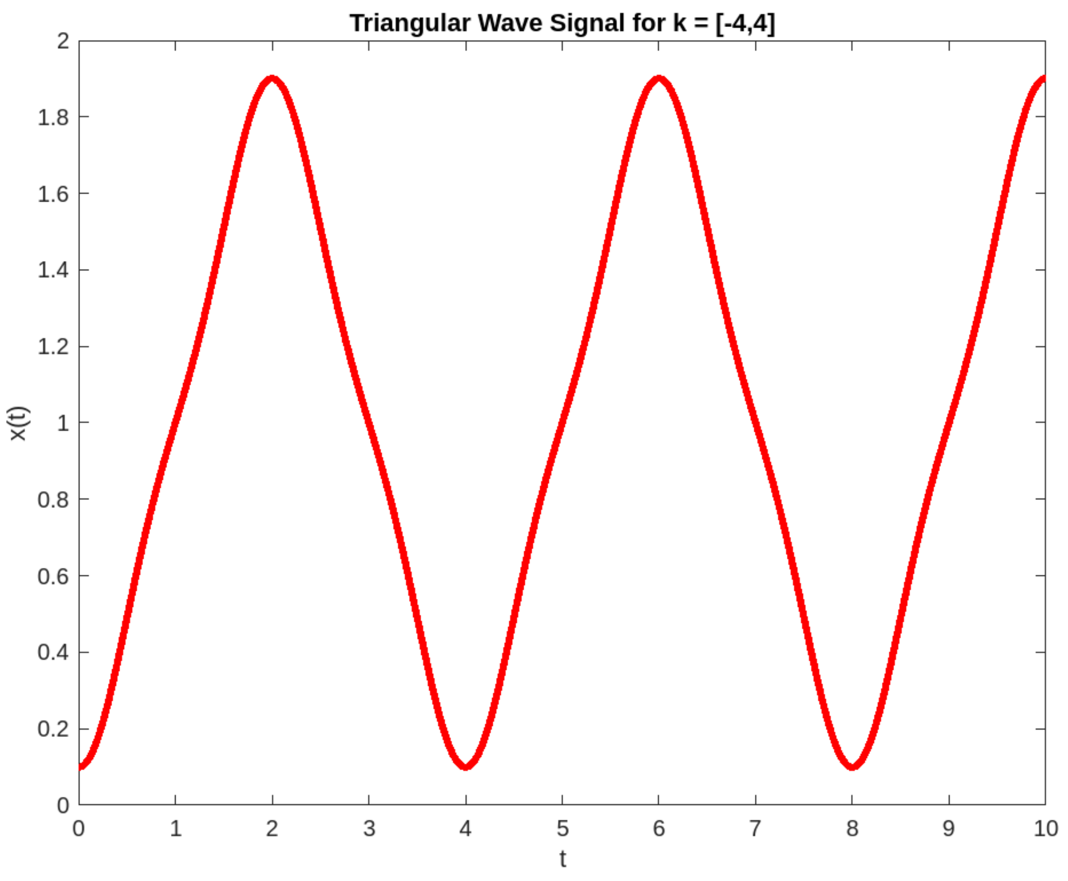
Part 1 (b)

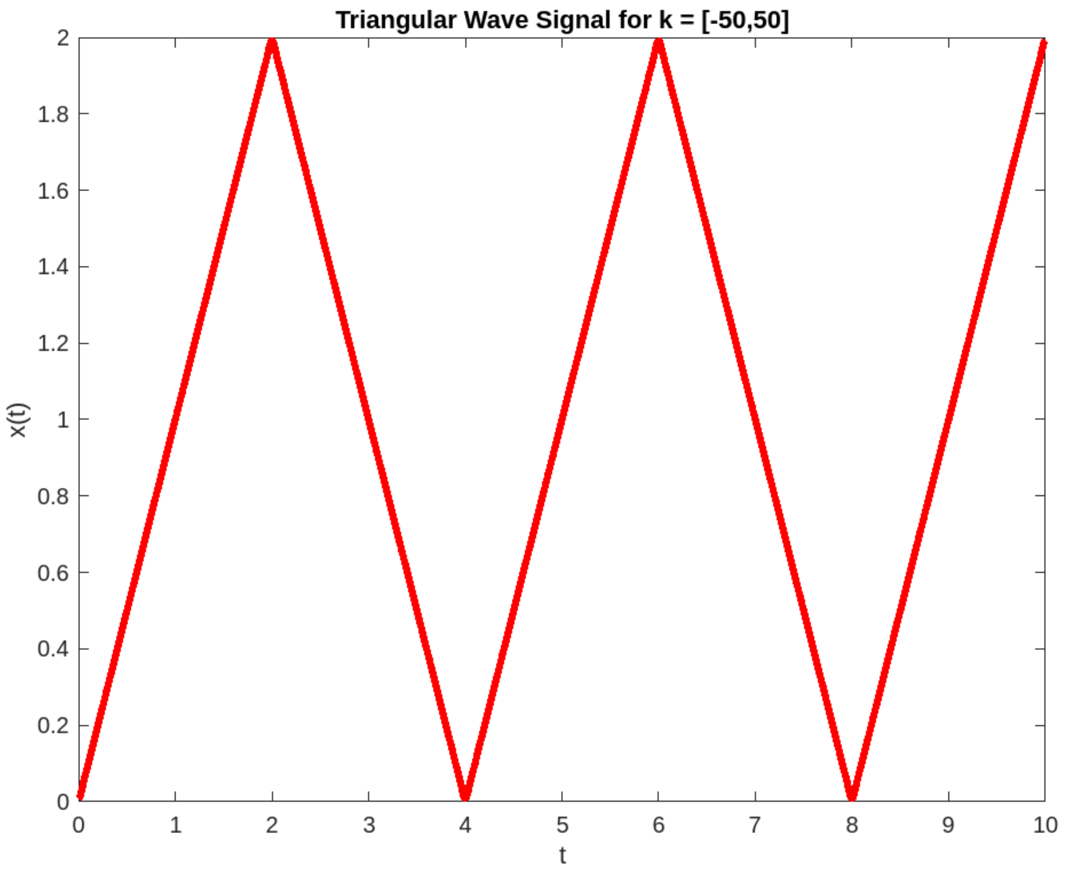
MATLAB Code of Triangular Wave Signal

```
period = 4;
frequency = 1 / period;
K = 2;
coefAmount = 2 * K + 1;
ak = zeros(1, coefAmount);

for k = -K: K
   fourierFunc = @(t) ((t) .* (t < 2 & t >= 0) + (4 - t) .* (t
< 4 & t >= 2)) .* exp(-1j * 2 * pi * frequency * k * t);
   ak(k + (K + 1)) = integral(fourierFunc, 0, period) /
period;
end

time_plot = linspace(0, 10, 1000);
reconstructedSignal = zeros(size(time_plot));

for k = -K: K
   reconstructedSignal = reconstructedSignal + ak(k + (K + 1))
.* exp(1j * 2 * pi * frequency * k * time_plot);
end

figure;
plot(time_plot, real(reconstructedSignal), 'r-', 'LineWidth',
3);
xlabel("t");
ylabel("x(t)")
title(['Triangular Wave Signal for k = [' num2str(-K) ','
num2str(K) ']']);
```

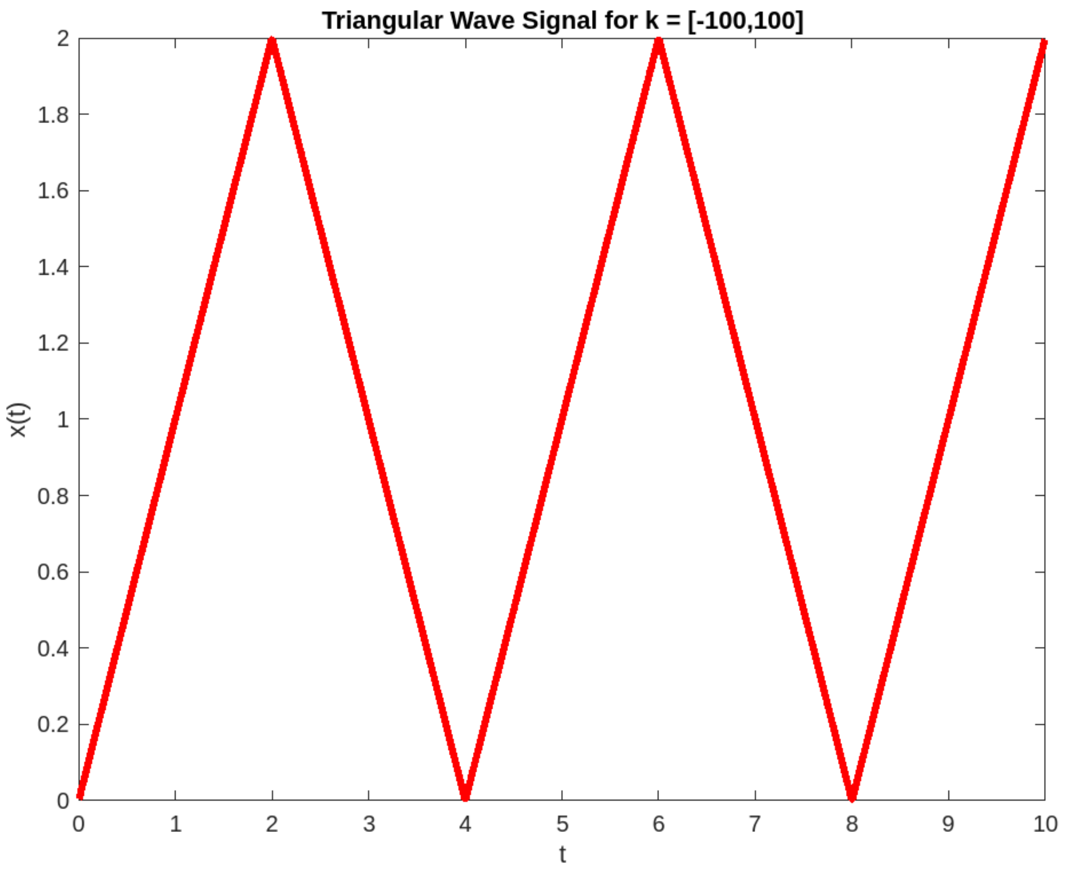
Figures of Triangular Wave Signal







Triangular Wave Signal for k = [-50,50]
x(t)
t
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
1
2
3
4
5
6
7
8
9
10



Triangular Wave Signal for k = [-100,100]
x(t)
t
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
1
2
3
4
5
6
7
8
9
10

Outcomes of Triangular Wave Signal Experiment

As the figures clearly depict, small values for k results in a signal wave that looks like a sinusoidal wave more than a triangular wave. When the value of k slightly increases the resulting signal is still a periodic function which slightly starts to depict the characteristics of the triangular function. Only when the value of k becomes significantly larger relative to the prior values, the signal starts to look like a triangular signal wave. However, this time a lower value of k (lower than that of the square wave example) is enough to obtain a signal looking like a triangular signal wave. Since the sinusoidal signals resemble triangular signals more than square signals, approximation error becomes lower earlier compared to square wave case.

Again, the outcome is consistent with the claims of the Fourier. Therefore, as k increases, the approximation error decreases; and when k goes to infinity, the resulting signal will be equivalent to the original triangular wave signal.

Part 2 (a)

$signal_1 = cos(2\pi * 220 * 2^{10/12} * t8)$

$signal_2 = cos(2\pi * 220 * 2^{6/12} * t2)$

$signal_3 = cos(2\pi * 220 * 2^{8/12} * t8)$

$signal_4 = cos(2\pi * 220 * 2^{5/12} * t2)$

Part 2 (b, c, d)

```
f0 = 220;
fs = 8000;
n1 = 2;
t8 = 1/fs: 1/fs: n1/8;
t2 = 1/fs: 1/fs: n1/2;
sd = zeros(1, round(length(t8) / 10));
rest = zeros(1, length(t8));

signal1 = cos(2 * pi * f0 * (2^(10 / 12)) * t8);
signal2 = cos(2 * pi * f0 * (2^(6 / 12)) * t2);
signal3 = cos(2 * pi * f0 * (2^(8 / 12)) * t8);
signal4 = cos(2 * pi * f0 * (2^(5 / 12)) * t2);

signalsArray = [signal1, sd, signal1, sd, signal1, sd,
signal2, sd, rest, sd, signal3, sd, signal3, sd, signal3, sd,
signal4];
sound(signalsArray, fs);
```

Part 2 (e)

These four signals correspond to the four notes comprising the beginning of Beethoven's 5th Symphony.