Jacobs University Bremen

CO-520-B: Signals and Systems Lab

Fall 2021

Lab Experiment 3 - Fourier Series and Fourier Transform

Priontu Chowdhury

Introduction

Theory

In this lab, we studied the different signals in terms of their Fourier coefficients and got a deeper understanding of the Fourier Transform. A signal can be represented in the time domain or the frequency domain. The frequency domain representation of the signal is called the spectrum of the signal, and Fourier analysis is the technique that is used to decompose the signal into sinusoids.

Any time varying signal can be constructed by superimposing sinusoidal waves. The Fourier Transform is used to transform a signal from the time domain to the frequency domain. Continuous can be performed on a class of signals, but for arbitrary signals, the signal must be digitized, which requires a discrete Fourier Transform.

We can also use the inverse Fourier Transform to convert a signal from the frequency domain to the time domain.

Prelab: Fourier Series and Fourier Transform

Problem 1: Decibels

Question 1

a)

$$x(t) = 2\cos(2000\pi t)$$

$$-1 \leq \cos(\theta) \leq 2$$

$$Vpp = 2 - (-2) = 4$$

b)

$$V_{rms} = \sqrt{\frac{1}{T}\int_{t0}^{t0+T} V^2(t)\,dt}$$

$$V(t) = x(t) = 2\cos(2000\pi t)$$

$$\omega_0 = 2000\pi$$

$$T = \frac{2\pi}{2000\pi} = 10^{-3}$$

$$V_{rms} = \sqrt{\frac{1}{10^{-3}}\int_0^{10^{-3}} x^2(t)\,dt} = \sqrt{1000\int_0^{10^{-3}} 4\cos^2(2000\pi t)\,dt}$$

$$V_{rms} = \sqrt{\frac{4000}{2}\int_0^{10^{-3}} (1 + \cos(4000\pi t)dt} = \sqrt{2000\left[t + \frac{\sin(4000\pi t)}{4000\pi}\right]_0^{10^{-3}}}$$

$$V_{rms} = \sqrt{2000\left[\left\{10^{-3} + \frac{sin 4\pi}{4000\pi}\right\} - \left\{0 + \frac{\sin(0)}{4000\pi}\right\}\right]}$$

$$V_{rms} = \sqrt{2000\ [10^{-3}]} = \sqrt{2}$$

$$V_{rms} = \sqrt{2}$$

c)

$$dBV_{rms} = 20\log(V_{rms}) = 20\log\left(2^{\frac{1}{2}}\right) = 3.010\ dB$$

<u>Question 2</u>

a)

$$V_{rms} = \sqrt{\frac{1}{T}\int_{t0}^{t0+T} V^2(t)\ dt}$$

$$V_{rms} = \sqrt{\left(\frac{1}{T}\right)\left\{\int_0^{\frac{T}{2}} 2^2 dt\ +\ \int_{\frac{T}{2}}^{T} (-2)^2 dt\right\}}$$

$$V_{rms} = \sqrt{\left(\frac{1}{T}\right)\left\{4\left(\frac{T}{2}\right) + 4\left\{T - \frac{T}{2}\right\}\right\}} = \sqrt{\left(\frac{1}{T}\right)\{2T + 2T\}} = \sqrt{\frac{4T}{T}}$$

$$V_{rms} = \sqrt{4} = 2$$

b)

$$dBV_{rms} = 20log(V_{rms}) = 20\log(2) = 6.021$$

Problem 2: Determination of Fourier Series coefficients

Question 1

$$f(t) = \frac{2t}{T}\ , -\frac{T}{2} < 0 < \frac{T}{2}$$

$$\omega_0 = \frac{2\pi}{T}$$

$$\Delta T = \frac{T}{2} - \left(-\frac{T}{2}\right) = T$$

$$f(t) = -f(-t)$$

Therefore, $f(t)$ is an odd function.

$$a_0 = 0, a_n = 0$$

$$b_n = \left(\frac{4}{T}\right)\int_0^{\frac{T}{2}} f(t)\sin(n\omega_0 t)\,dt$$

$$b_n = \left(\frac{4}{T}\right)\int_0^{\frac{T}{2}} \frac{2t}{T}\sin(n\omega_0 t)\,dt$$

$$b_n = \frac{8}{T^2}\int_0^{\frac{T}{2}} t\sin(n\omega_0 t)\,dt$$

$$b_n = \frac{8}{T^2}\left[-\frac{tcos(n\omega_0 t)}{n\omega_0} + \int\frac{\cos(n\omega_0 t)}{n\omega_0}dt\right]_0^{\frac{T}{2}}$$

$$b_n = \frac{8}{T^2}\left[-\frac{tcos(n\omega_0 t)}{\omega_0 n} + \frac{\sin(n\omega_0 t)}{\omega_0^2 n^2}\right]_0^{\frac{T}{2}}$$

$$b_n = \frac{8}{T^2}\left\{-\frac{T}{2}\frac{\cos\left(\omega_0 n\frac{T}{2}\right)}{n\omega_0} + \left(\frac{1}{\omega_0^2 n^2}\sin\left(\frac{\omega_0 nT}{2}\right)\right)\right\}$$

$$b_n = \frac{2}{T^2}\left[\frac{-T\cos\left(\frac{2\pi}{T}n\frac{T}{2}\right)}{\frac{2\pi}{T}n} + \frac{T^2}{4\pi^2 n^2}\sin(n\pi)\right]$$

$$b_n = \frac{2}{T^2}\left[-\frac{T^2\cos(n\pi)}{n\pi} + \frac{T^2}{4\pi^2 n^2}(0)\right]$$

$$b_n = 2\left[-\frac{(-1)^n}{n\pi}\right] = \frac{2(-1)^{n+1}}{n\pi}$$

$$f(t) = \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{n}\sin\,(n\omega_0 t)$$

$n = 1: f(t) = \frac{2}{\pi}\sin(\omega_0 t)$

$n = 2: f(t) = \frac{2}{\pi}\left(\sin(\omega_0 t) - \frac{1}{2}\sin(2\omega_0 t)\right)$

$n = 3: f(t) = \frac{2}{\pi}\left(\sin(\omega_0 t) - \frac{1}{2}\sin(2\omega_0 t) + \frac{1}{3}\sin(3\omega_0 t)\right)$

$n = 4: f(t) = \frac{2}{\pi}\left(\sin(\omega_0 t) - \frac{1}{2}\sin(2\omega_0 t) + \frac{1}{3}\sin(3\omega_0 t) - \frac{1}{4}\sin(4\omega_0 t)\right)$

$n = 5: f(t) = \frac{2}{\pi}\left(\sin(\omega_0 t) - \frac{1}{2}\sin(2\omega_0 t) + \frac{1}{3}\sin(3\omega_0 t) - \frac{1}{4}\sin(4\omega_0 t) + \frac{1}{5}\sin\,(5\omega_0 t)\right)$

Question 2

Matlab plot of the function:

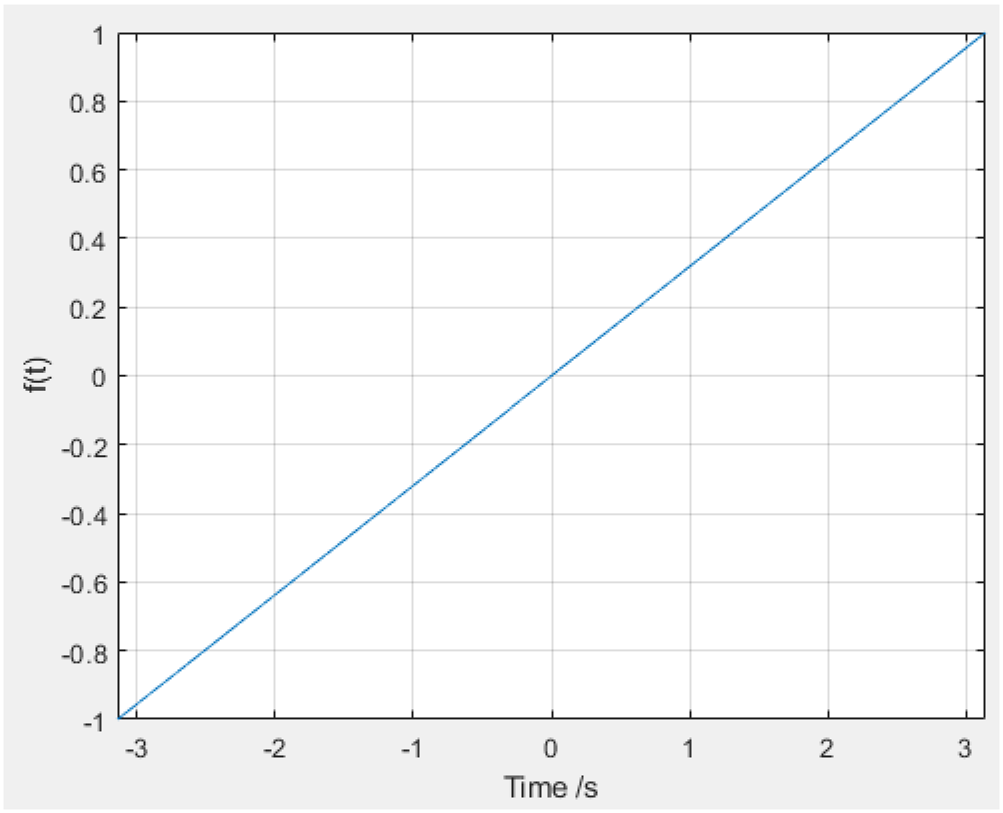


Figure 1: Matlab plot of original function f(t) = 2t/T

Matlab code for the figure:

```
T = 2*pi;                       %period of function
t = (-T/2:0.0001:T/2);          %setting up time domain
y = 2*t/T;                      %setting up function y
figure(1);                      %setting up figure
plot(t,y);                      %plotting the function
xlim([-pi,pi]);                 %setting up plot limits
grid on;
xlabel("Time /s");              %labeling the axis
ylabel("f(t)");
```

Question 3

Using the calculated coefficients for the first five harmonics, we can see that the fourier series for f(t) looks more similar to the original function as we add more harmonics. The figure is provided below:

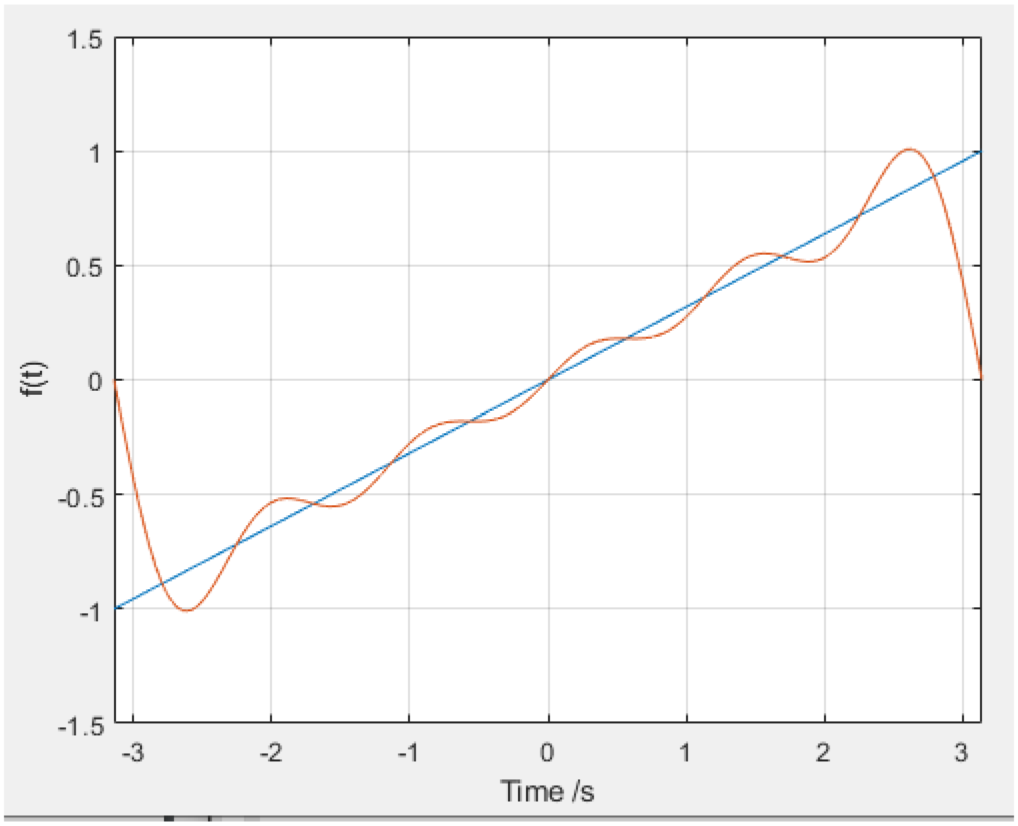


Figure 2: Fourier Series of f(t)

The matlab code is provided below:

```
T = 2*pi;                     %period of function
t = (-T/2:0.0001:T/2);        %setting up time domain
y = 2*t/T;
f = 0;
for n=1:5                     %developing the fourier series
 f = f + ((-2*(-1)^n/(n*pi)))*sin(n*2*pi/T*t);
end
figure(2);
plot(t,y);                     %plotting the line
grid on;
hold on;
plot(t,f);                     %plotting the fourier series
xlim([-pi,pi]);                %boundary of plot
xlabel("Time /s");             %labeling the axis
ylabel("f(t)");
grid on;
hold on;
```

Problem 3 : FFT of a Square/Rectangular Wave

<u>Question 1</u>

The matlab script for square wave plot generation is provided below:

```
Fs = 200000;              % sampling frequency
t = 0.00001:1/Fs:0.05;    %time domain
T = 10^(-3);              %period
f = 1/T;                  % frequency of signal
w = 2*pi*f;               %calculating radian frequency
y = square(w*t, 50);      %developing square wave
plot (t, y);
xlim([0, 0.002]);         %x-axis boundary
ylim([-2, 2]);            %y-axis boundary
xlabel("Time/s");
ylabel("f(t)");
title("Square wave");
grid on;
hold on;
```

Question 2

The plot for the square wave is provided below:

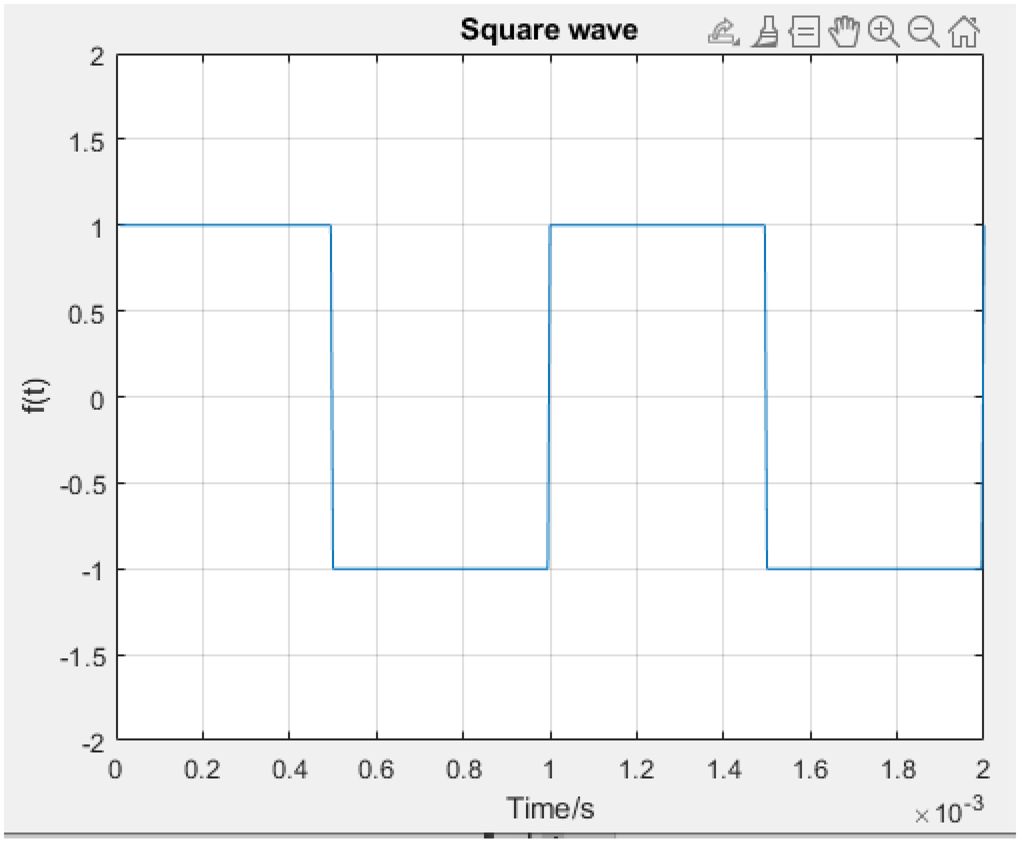


Figure 3: Square wave plot on Matlab

The script is provided as question 1 solution.

Question 3

On this task, we develop the FFT Spectrum of the square wave. The script for this is provided below:

```
Fs = 200000;                          % sampling frequency
t = 0:1/Fs:0.01;                      %domain
sw = square(2*pi*1000*t, 50);          %square wave

%FFT:
L = length(sw);
F_nyq = Fs/2;                         %Nyquist frequency
sw_fft = fft(sw);                       %Applying FFT
sw_fft = 2*abs(sw_fft)/L;
sw_fft = sw_fft(1:length(sw)/2);
f = linspace(0, F_nyq, L/2);
%plot FFT
```

```
plot(f, sw_fft);
xlabel('Frequency/Hz');          %Labeling the axes
ylabel('f(t)');
title("FFT Spectrum of Square Wave");
grid on;
hold on;
```

The plot is provided below:

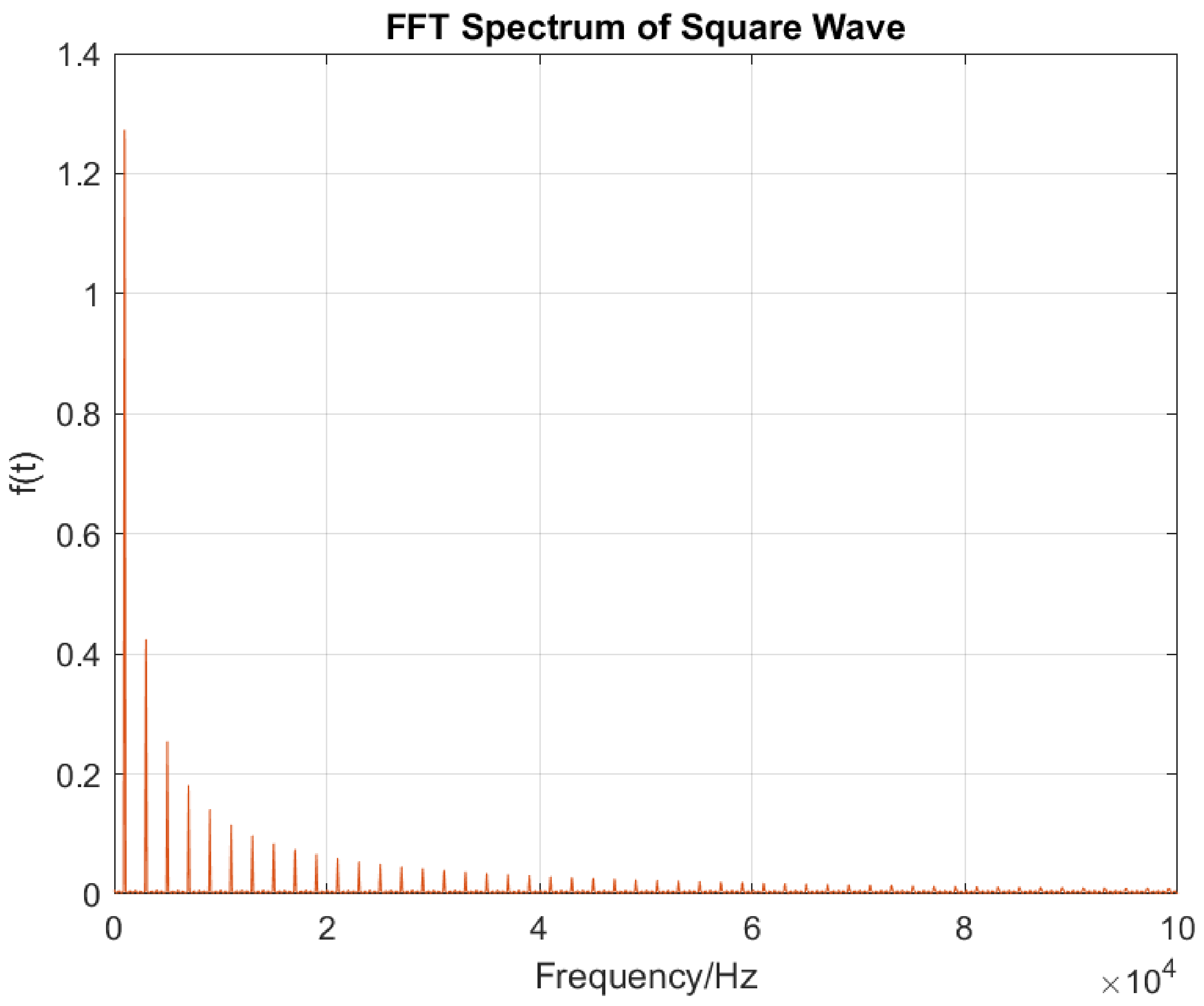


Figure 4: FFT Spectrum of square wave

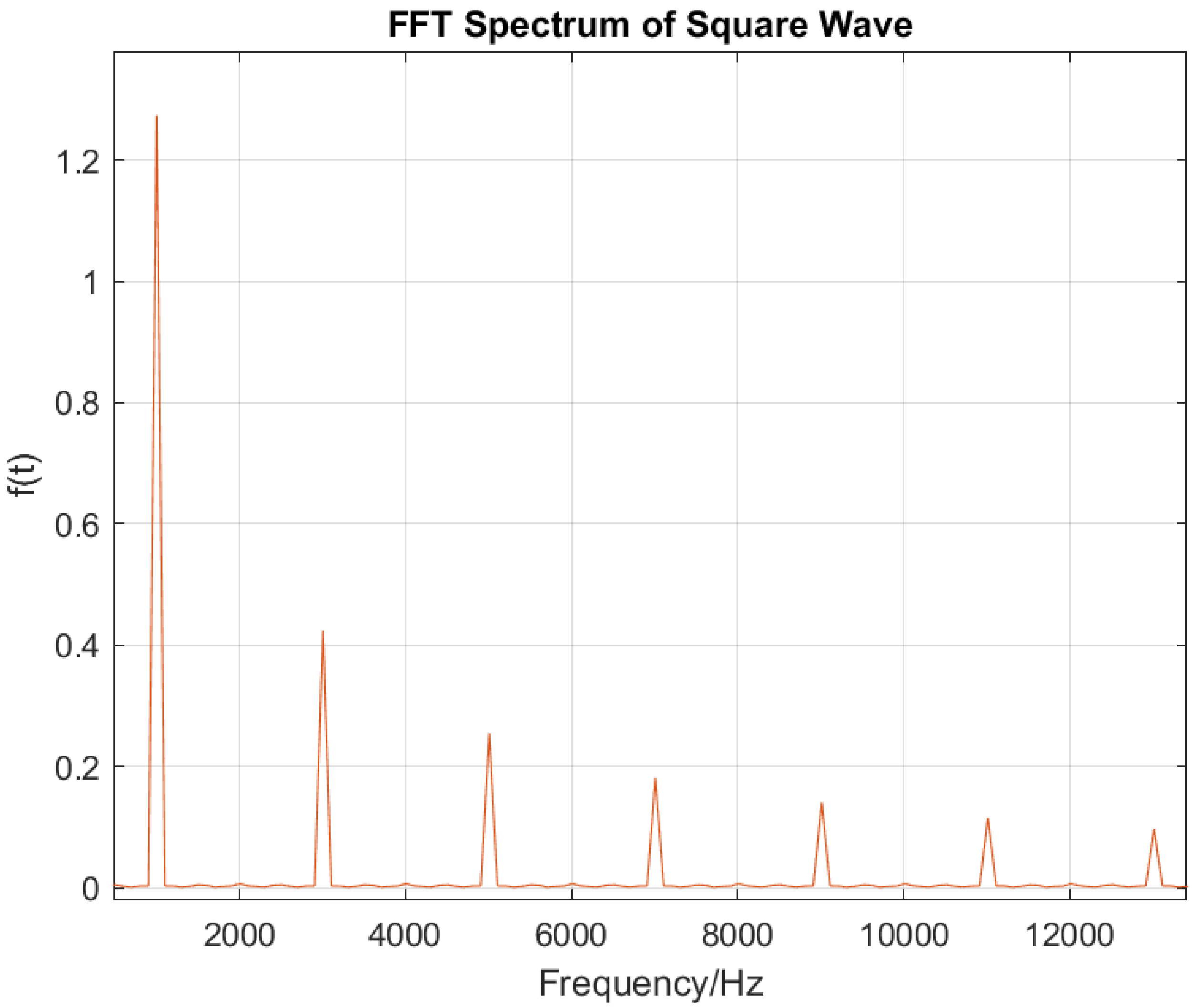


Figure 5: Close up of fundamental frequency and first six harmonics

<u>Question 4</u>

For this problem we are required to produce a single sided spectrum. The matlab code is provided below:

```
Fs = 200000; %sampling frequency
t = 0:1/Fs:0.01;
f = 1000;
w = 2*pi*f;
sw = square(w*t, 50); %Generating Square Wave
%Implementing FFT:
F_nyq = Fs/2;
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft)/length(sw);
sw_fft = sw_fft(1:length(y)/2);
f = Fs*(0:(length(y)/2))/length(sw);
```

```
%Calculating Single-Sided Spectrum in dB:
SSSpec = abs(2*fft(sw)/length(sw));
SSSpec = db(SSSpec/sqrt(2));
SSSpec = SSSpec(1:length(sw)/2+1);
SSSpec(2:end-1) = 2*SSSpec(2:end-1);
%Plotting the Single Sided Spectrum:
plot(f, SSSpec, '-r');
xlabel('Frequency/Hz'); %labeling Axes
ylabel('Voltage/dB');
title("Single-sided Spectrum");
grid on;
```

The resulting curve is provided below:

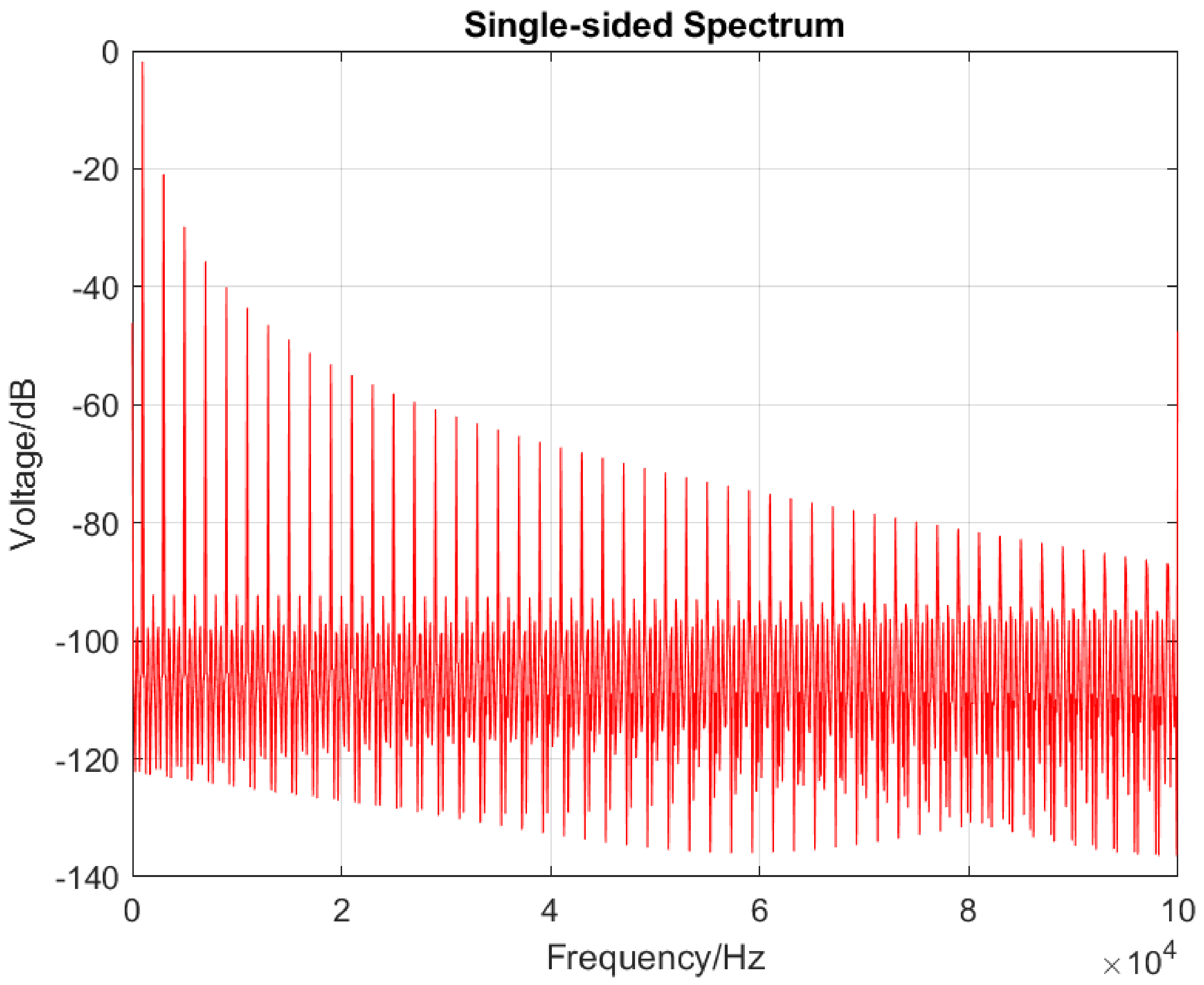


Figure 6: Single-sided Spectrum

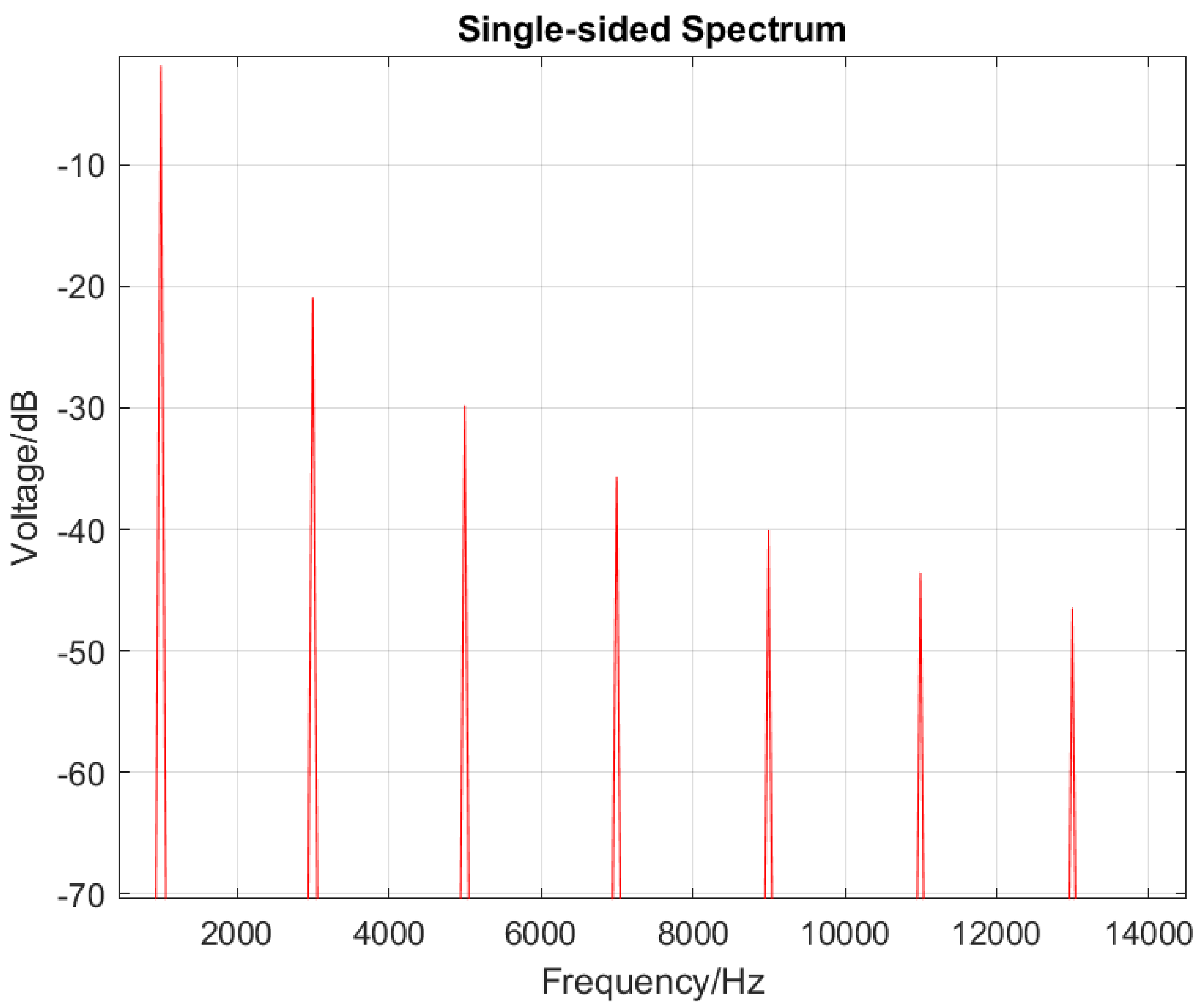


Figure 7: Close-up of Singe-sided Spectrum (Showing fundamental frequency and first 6 harmonics)

Question 5

Matlab code is provided below:

```
Fs = 200000; %sampling frequency
t = 0:1/Fs:0.01;
f = 1000;
w = 2*pi*f;
sw = square(w*t, 50); %Generating Square Wave
%Implementing FFT:
F_nyq = Fs/2;
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft)/length(sw);
sw_fft = sw_fft(1:length(y)/2);
f = Fs*(0:(length(y)/2))/length(sw);
%Calculating Single-Sided Spectrum in dB:
SSSpec = abs(2*fft(sw)/length(sw));
SSSpec = db(SSSpec/sqrt(2));
SSSpec = SSSpec(1:length(sw)/2+1);
SSSpec(2:end-1) = 2*SSSpec(2:end-1);
%Plotting the Single Sided Spectrum:
plot(f, SSSpec, '-r');
xlabel('Frequency/Hz'); %labeling Axes
ylabel('Voltage/dB');
title("First Four Harmonics of Spectrum");
xlim([0,10000]);
grid on;
```

The plot is provided below:

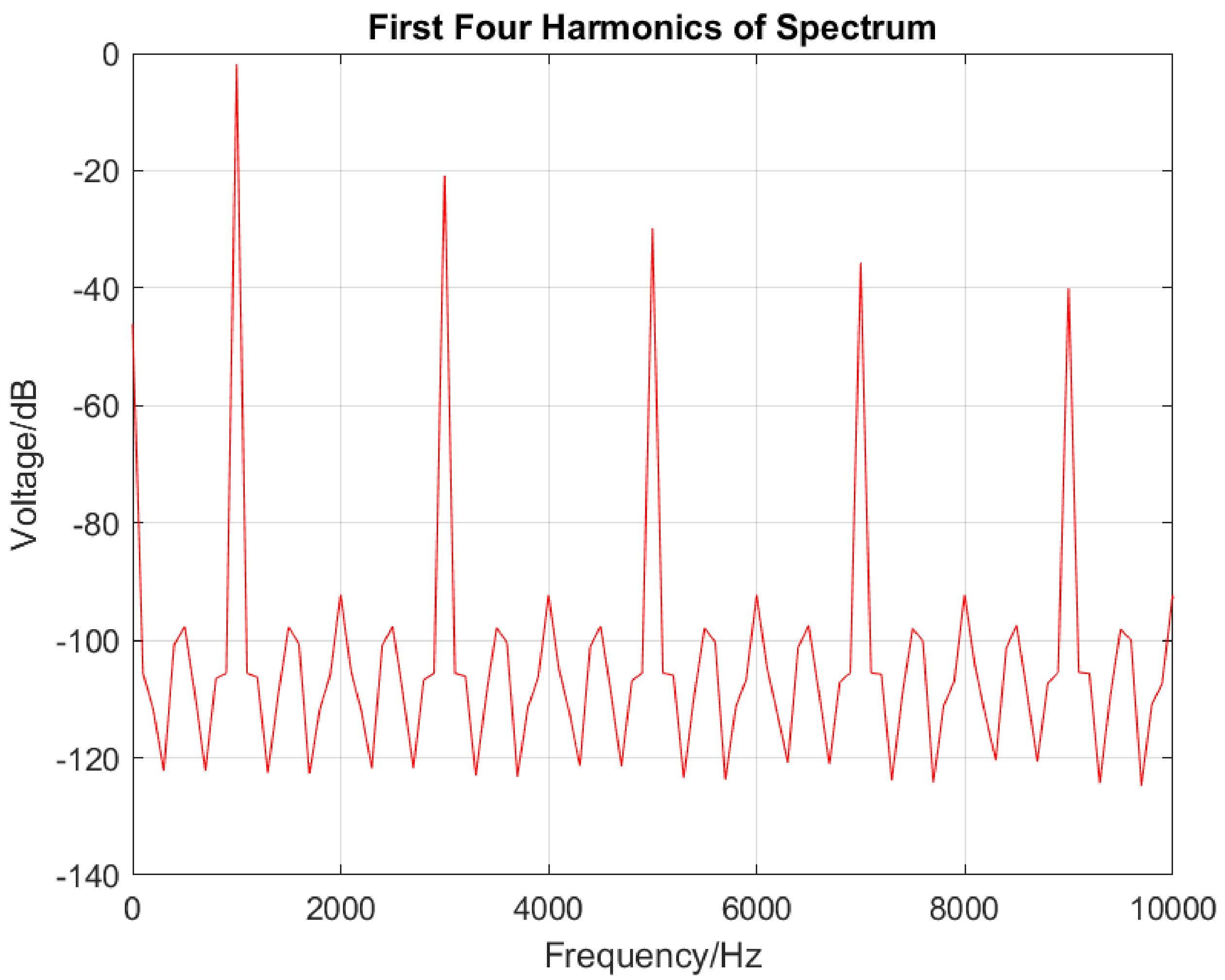


Figure 8: First four harmonics of spectrum

Question 6

Part 1 (20% Duty Cycle):

Now, we are required to generate a square wave with 20% duty cycle, implement FFT to it, and then plot the square wave, its spectrum, the single sided spectrum and the first four harmonics.

The matlab code for the implementation is provided below:

```
%%Square Wave(20% Duty Cycle) FFT Generation

%Square Wave Generation(20% Duty Cycle):
t = 0:0.00001:0.01;     %Setting up domain
f = 1000;
w = 2*pi*f;
sw = square(w*t, 20);
%Implementing FFT:
Fs = 200000;
F_nyq = Fs/2;
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft)/length(sw);
sw_fft = sw_fft(1:length(sw)/2);
freq = Fs*(0:(length(sw)/2))/length(sw);
%Generating Single-sided Spectrum in dB:
SSSpec = abs(2*fft(sw)/length(sw));
SSSpec = db(SSSpec);
SSSpec = SSSpec(1:length(sw)/2+1);
SSSpec(2:end-1) = 2*SSSpec(2:end-1);

%Plotting the square wave:
figure(1);
plot(t, sw);
title("Square Wave(20% Duty Cycle)");
xlabel('Time/s');
ylabel('Voltage/V');
ylim([-1.5, 1.5]);
xlim([0, 0.005]);
grid on;

%Plotting FFT Spectrum of Square wave:
figure(2);
freq_dom = linspace(0, F_nyq, length(sw)/2);
plot(freq_dom, sw_fft);
title("FFT Spectrum of Square Wave(20% Duty Cycle)");
xlabel('Frequency/Hz');
ylabel('Voltage/V');
grid on;
```

```
%Plotting Single-sided spectrum:
figure(3);
plot(freq, SSSpec);
title("Single-sided Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
grid on;

%Plotting first 4 harmomics:
figure(4);
plot(freq, SSSpec);
title("First 4 Harmonics of Square Wave Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
xlim([0 11000]);
grid on;
%%
```

The plots are provided below:

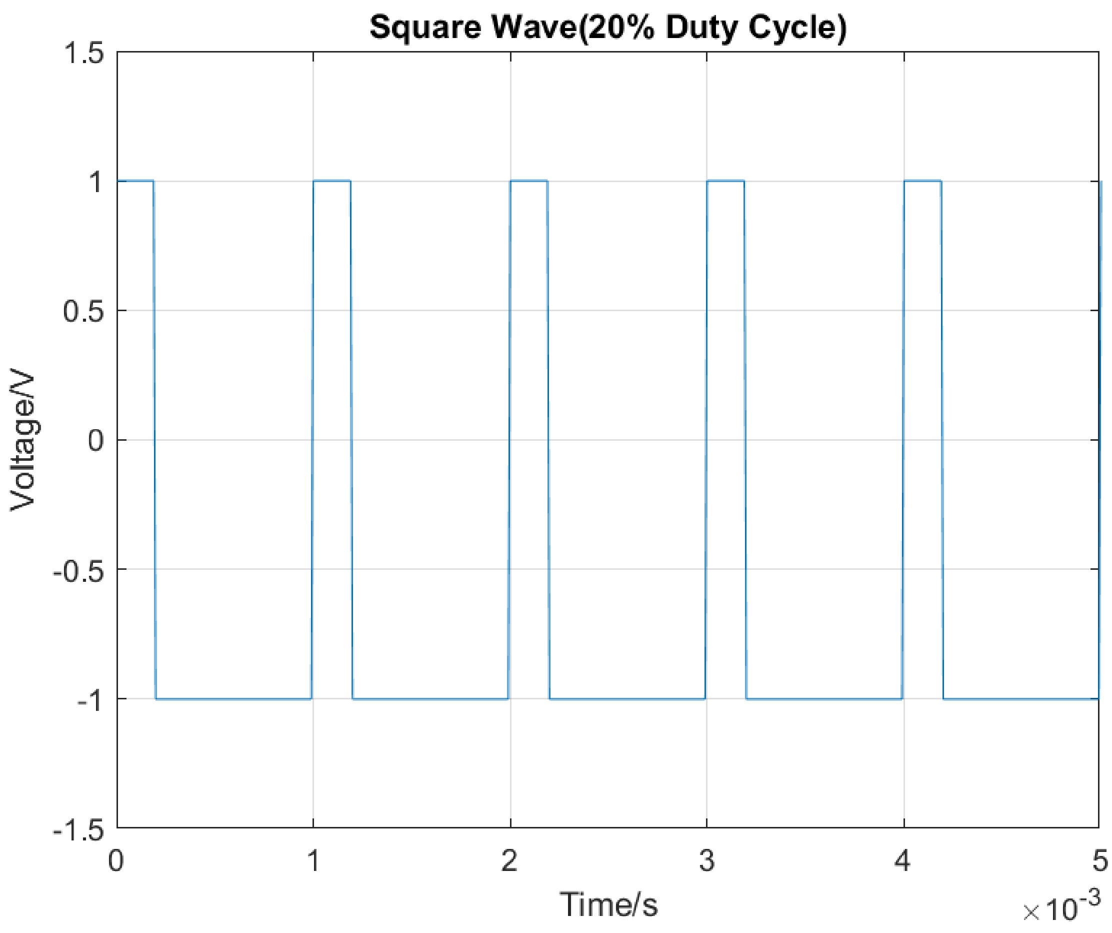


Figure 9: Square wave (20% duty cycle)

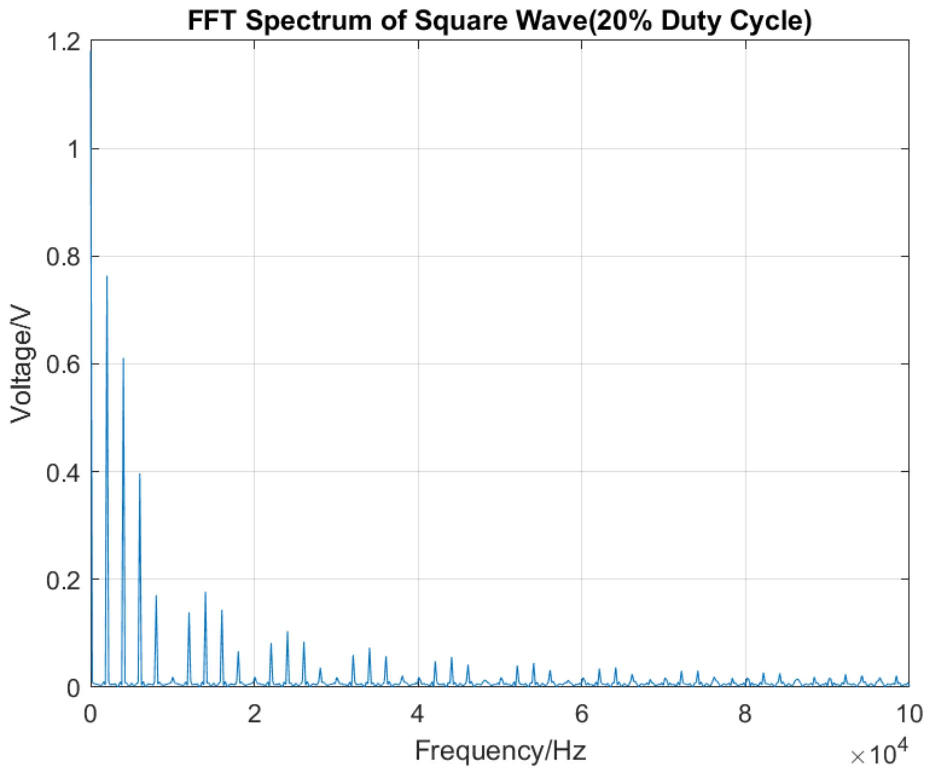


Figure 10: FFT Spectrum of the Square wave above

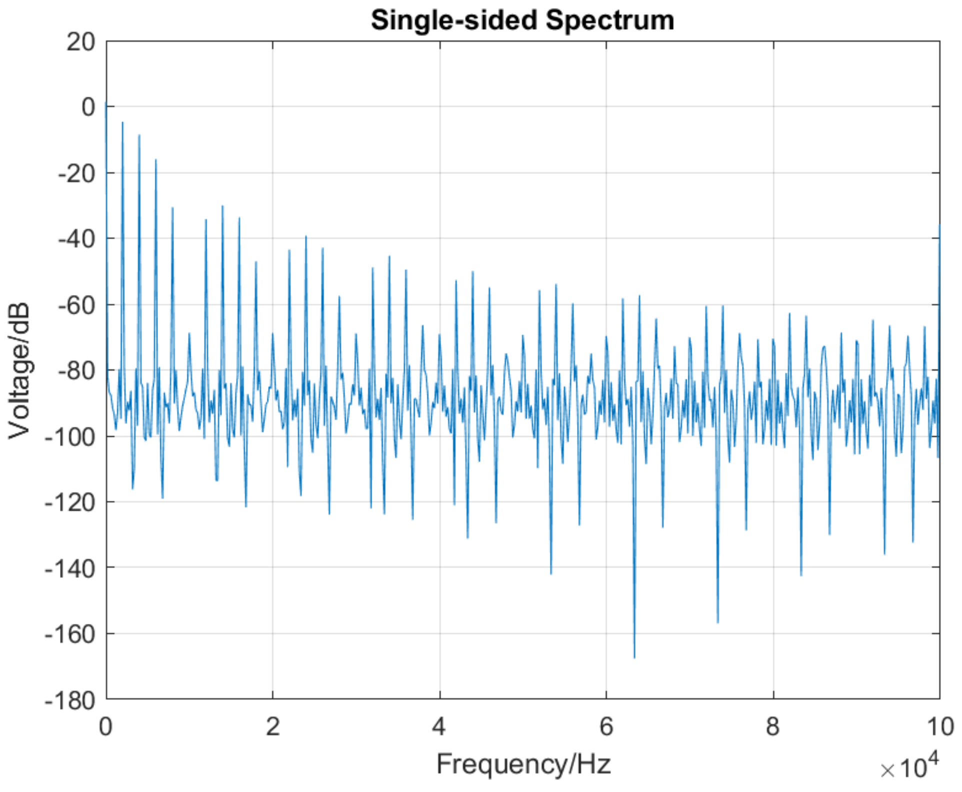


Figure 11 : Single-sided spectrum of above square wave

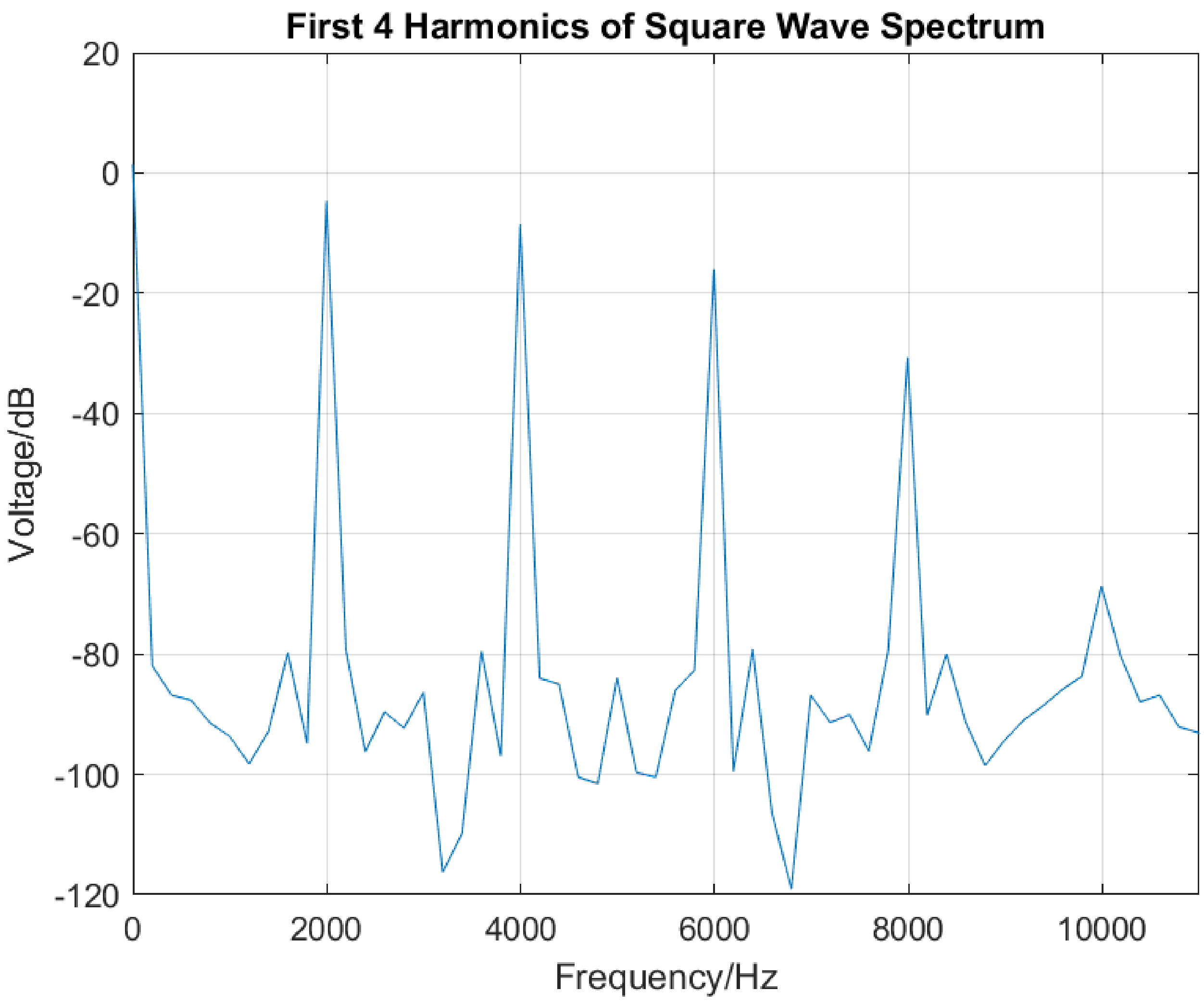


Figure 12: First four harmonics of above single-sided

Part 2 (33% Duty Cycle):

Now, we are required to generate a square wave with 20% duty cycle, implement FFT to it, and then plot the square wave, its spectrum, the single sided spectrum and the first four harmonics.

The matlab code is provided below:

```
%%Square Wave(33% Duty Cycle) FFT Generation

%Square Wave Generation(33% Duty Cycle):
t = 0:0.00001:0.01;      %Setting up domain
f = 1000;
w = 2*pi*f;
sw = square(w*t, 33);
%Implementing FFT:
Fs = 200000;
F_nyq = Fs/2;
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft)/length(sw);
sw_fft = sw_fft(1:length(sw)/2);
freq = Fs*(0:(length(sw)/2))/length(sw);
%Generating Single-sided Spectrum in dB:
SSSpec = abs(2*fft(sw)/length(sw));
SSSpec = db(SSSpec);
SSSpec = SSSpec(1:length(sw)/2+1);
SSSpec(2:end-1) = 2*SSSpec(2:end-1);

%Plotting the square wave:
figure(1);
plot(t, sw);
title("Square Wave(33% Duty Cycle)");
xlabel('Time/s');
ylabel('Voltage/V');
ylim([-1.5, 1.5]);
xlim([0, 0.005]);
grid on;

%Plotting FFT Spectrum of Square wave:
figure(2);
freq_dom = linspace(0, F_nyq, length(sw)/2);
plot(freq_dom, sw_fft);
title("FFT Spectrum of Square Wave(33% Duty Cycle)");
xlabel('Frequency/Hz');
ylabel('Voltage/V');
grid on;

%Plotting Single-sided spectrum:
```

```
figure(3);
plot(freq, SSSpec);
title("Single-sided Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
grid on;

%Plotting first 4 harmonics:
figure(4);
plot(freq, SSSpec);
title("First 4 Harmonics of Square Wave Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
xlim([0 11000]);
grid on;
%%
```

The plots are provided below:

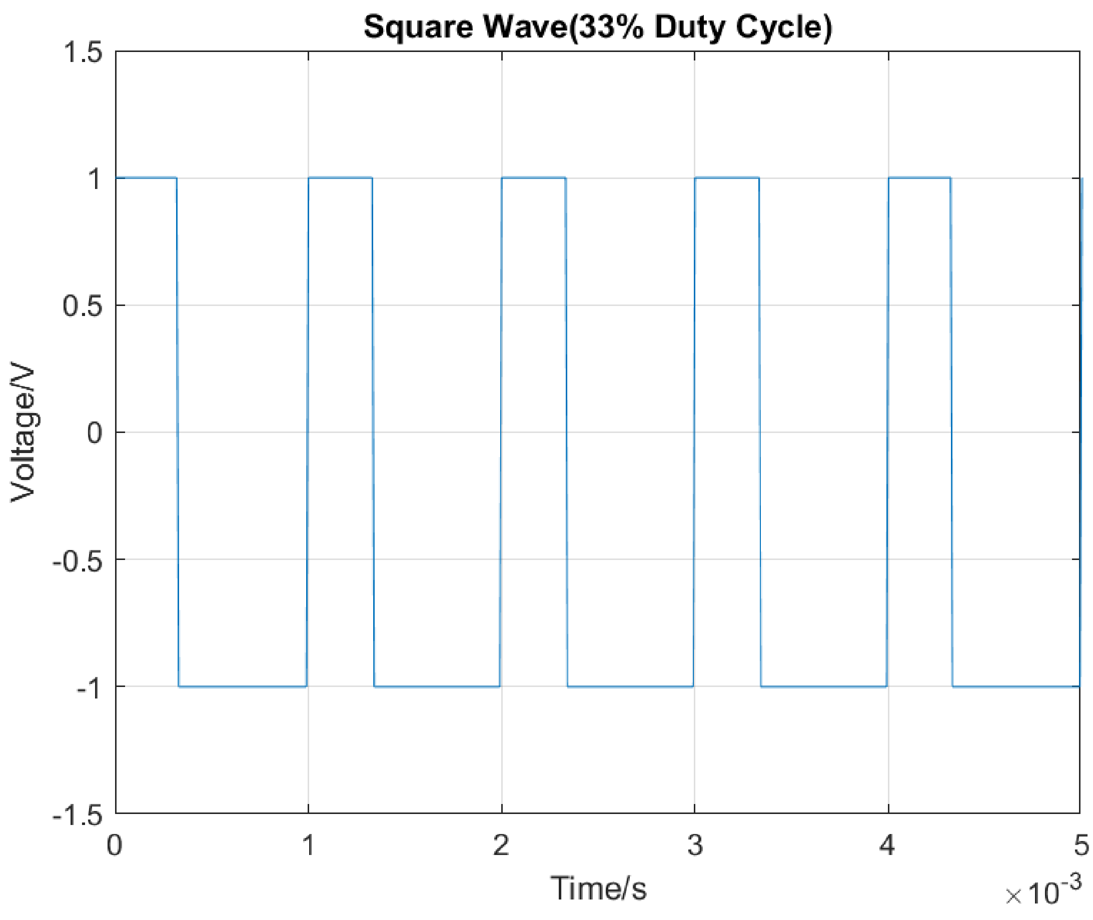


Figure 13: Square wave (33% duty cycle)

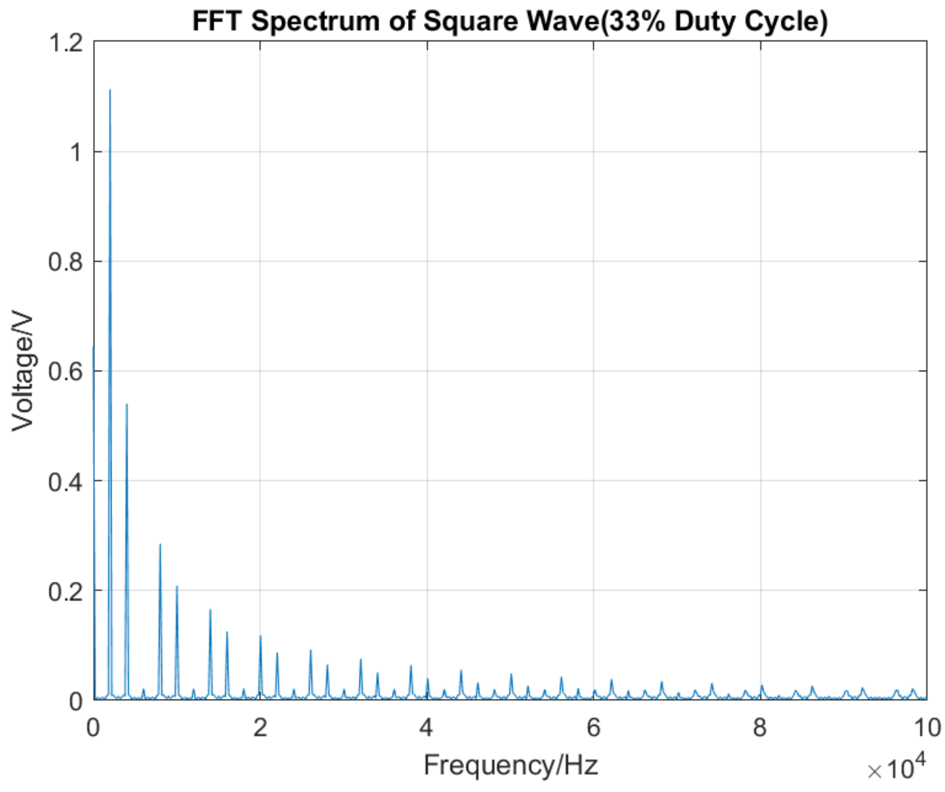


Figure 14: FFT Spectrum of Square wave (33% duty cycle)

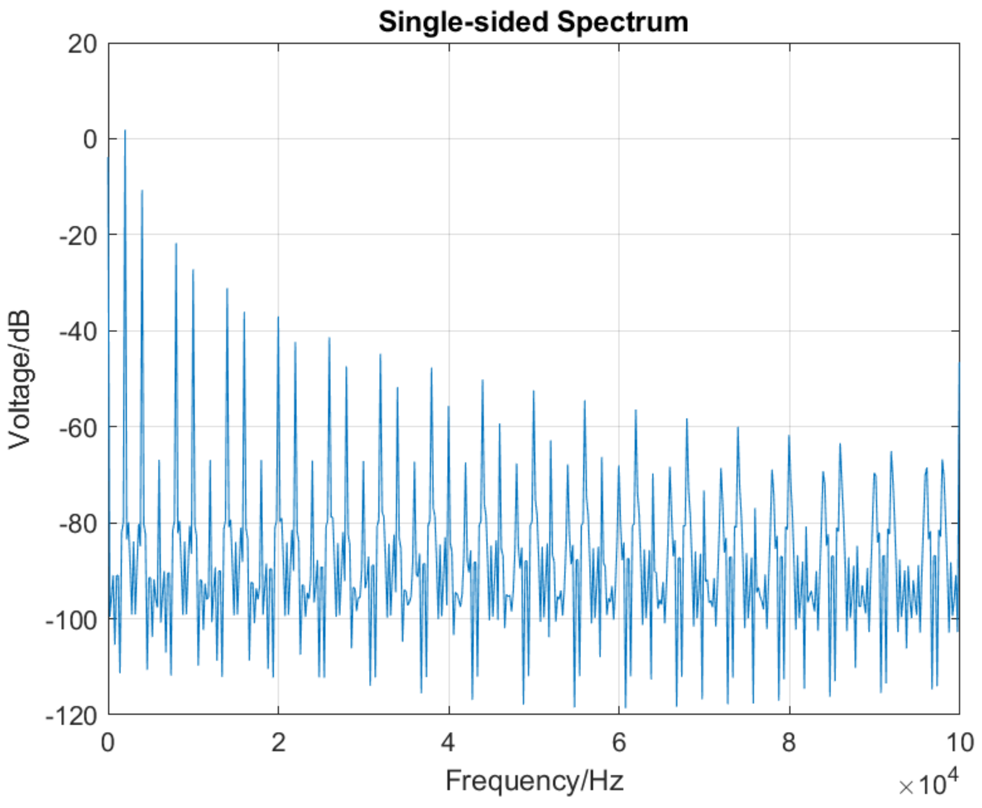


Figure 15: Single-sided Spectrum

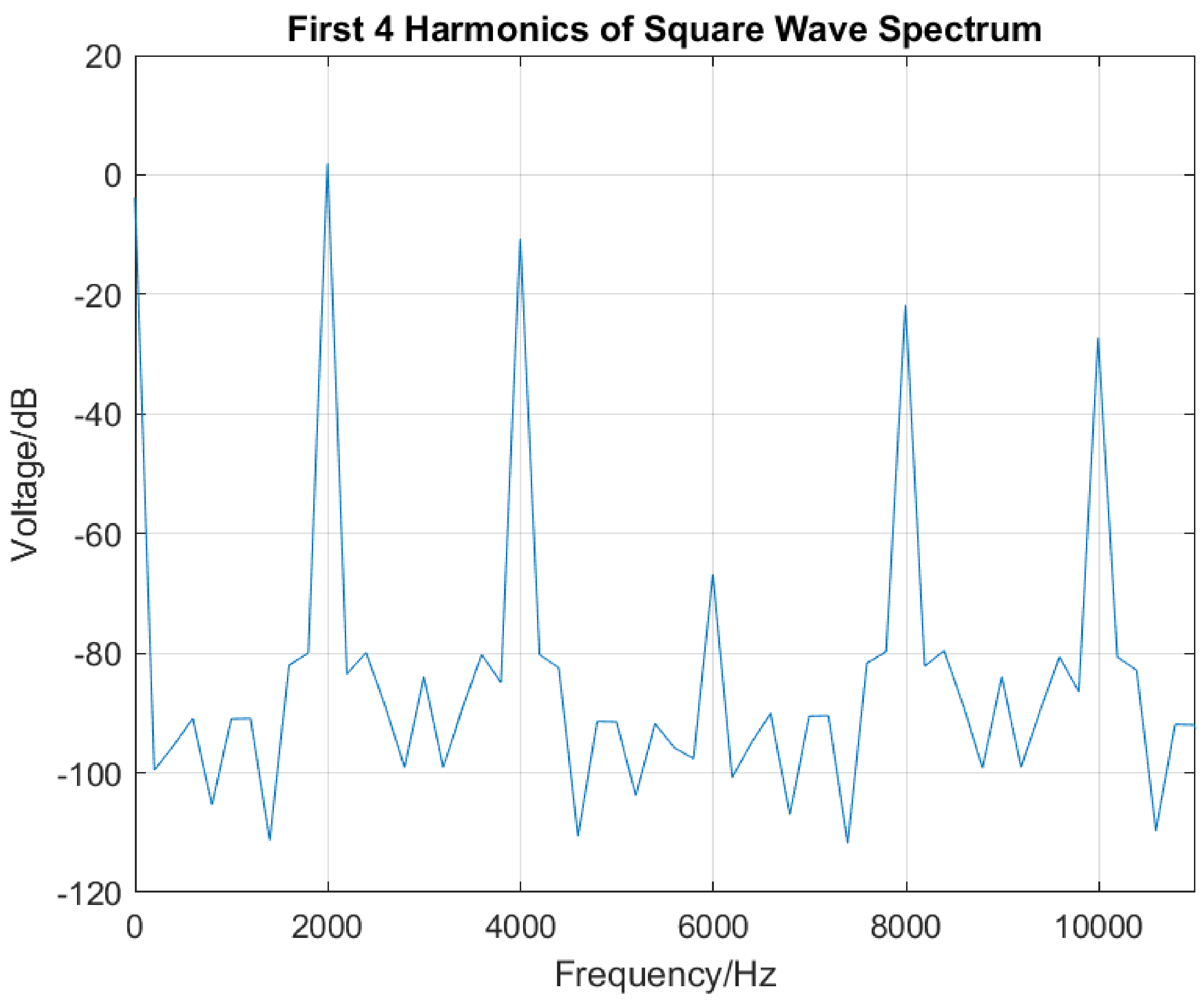


Figure 16: First four harmonics of Square Wave Spectrum

Question 7

We will now put everything in subplots. The matlab code is provided below:

```
%%Square Wave(20% Duty Cycle) FFT Generation

%Square Wave Generation(20% Duty Cycle):
t = 0:0.00001:0.01;      %Setting up domain
f = 1000;
w = 2*pi*f;
sw = square(w*t, 20);
%Implementing FFT:
Fs = 200000;
F_nyq = Fs/2;
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft)/length(sw);
sw_fft = sw_fft(1:length(sw)/2);
freq = Fs*(0:(length(sw)/2))/length(sw);
%Generating Single-sided Spectrum in dB:
```

```
SSSpec = abs(2*fft(sw)/length(sw));
SSSpec = db(SSSpec);
SSSpec = SSSpec(1:length(sw)/2+1);
SSSpec(2:end-1) = 2*SSSpec(2:end-1);

%Plotting the square wave:
figure(1);
subplot(2, 2, 1);
plot(t, sw);
title("Square Wave(20% Duty Cycle)");
xlabel('Time/s');
ylabel('Voltage/V');
ylim([-1.5, 1.5]);
xlim([0, 0.005]);
grid on;

%Plotting FFT Spectrum of Square wave:
subplot(2, 2, 2);
freq_dom = linspace(0, F_nyq, length(sw)/2);
plot(freq_dom, sw_fft);
title("FFT Spectrum of Square Wave(20% Duty Cycle)");
xlabel('Frequency/Hz');
ylabel('Voltage/V');
grid on;

%Plotting Single-sided spectrum:
subplot(2, 2, 3);
plot(freq, SSSpec);
title("Single-sided Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
grid on;

%Plotting first 4 harmomics:
subplot(2, 2, 4);
plot(freq, SSSpec);
title("First 4 Harmonics of Square Wave Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
xlim([0 11000]);
grid on;

%%
%%Square Wave(33% Duty Cycle) FFT Generation

%Square Wave Generation(33% Duty Cycle):
t = 0:0.00001:0.01;     %Setting up domain
f = 1000;
w = 2*pi*f;
sw = square(w*t, 33);
%Implementing FFT:
Fs = 200000;
F_nyq = Fs/2;
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft)/length(sw);
sw_fft = sw_fft(1:length(sw)/2);
```

```
freq = Fs*(0:(length(sw)/2))/length(sw);
%Generating Single-sided Spectrum in dB:
SSSpec = abs(2*fft(sw)/length(sw));
SSSpec = db(SSSpec);
SSSpec = SSSpec(1:length(sw)/2+1);
SSSpec(2:end-1) = 2*SSSpec(2:end-1);

%Plotting the square wave:
figure(2);
subplot(2, 2, 1);
plot(t, sw);
title("Square Wave(33% Duty Cycle)");
xlabel('Time/s');
ylabel('Voltage/V');
ylim([-1.5, 1.5]);
xlim([0, 0.005]);
grid on;

%Plotting FFT Spectrum of Square wave:
subplot(2, 2, 2);
freq_dom = linspace(0, F_nyq, length(sw)/2);
plot(freq_dom, sw_fft);
title("FFT Spectrum of Square Wave(33% Duty Cycle)");
xlabel('Frequency/Hz');
ylabel('Voltage/V');
grid on;

%Plotting Single-sided spectrum:
subplot(2, 2, 3);
plot(freq, SSSpec);
title("Single-sided Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
grid on;

%Plotting first 4 harmomics:
subplot(2, 2, 4);
plot(freq, SSSpec);
title("First 4 Harmonics of Square Wave Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
xlim([0 11000]);
grid on;
%%
%%
%%Square Wave(50% Duty Cycle) FFT Generation

%Square Wave Generation(50% Duty Cycle):
t = 0:0.00001:0.01;     %Setting up domain
f = 1000;
w = 2*pi*f;
sw = square(w*t, 50);
%Implementing FFT:
Fs = 200000;
F_nyq = Fs/2;
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft)/length(sw);
sw_fft = sw_fft(1:length(sw)/2);
freq = Fs*(0:(length(sw)/2))/length(sw);
%Generating Single-sided Spectrum in dB:
SSSpec = abs(2*fft(sw)/length(sw));
SSSpec = db(SSSpec);
SSSpec = SSSpec(1:length(sw)/2+1);
SSSpec(2:end-1) = 2*SSSpec(2:end-1);

%Plotting the square wave:
figure(3);
subplot(2, 2, 1);
plot(t, sw);
title("Square Wave(50% Duty Cycle)");
xlabel('Time/s');
ylabel('Voltage/V');
ylim([-1.5, 1.5]);
xlim([0, 0.005]);
grid on;

%Plotting FFT Spectrum of Square wave:
subplot(2, 2, 2);
freq_dom = linspace(0, F_nyq, length(sw)/2);
plot(freq_dom, sw_fft);
title("FFT Spectrum of Square Wave(33% Duty Cycle)");
xlabel('Frequency/Hz');
ylabel('Voltage/V');
grid on;

%Plotting Single-sided spectrum:
subplot(2, 2, 3);
plot(freq, SSSpec);
title("Single-sided Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
grid on;

%Plotting first 4 harmonics:
subplot(2, 2, 4);
plot(freq, SSSpec);
title("First 4 Harmonics of Square Wave Spectrum");
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
xlim([0 11000]);
grid on;
%%
```

The subplots are provided below:

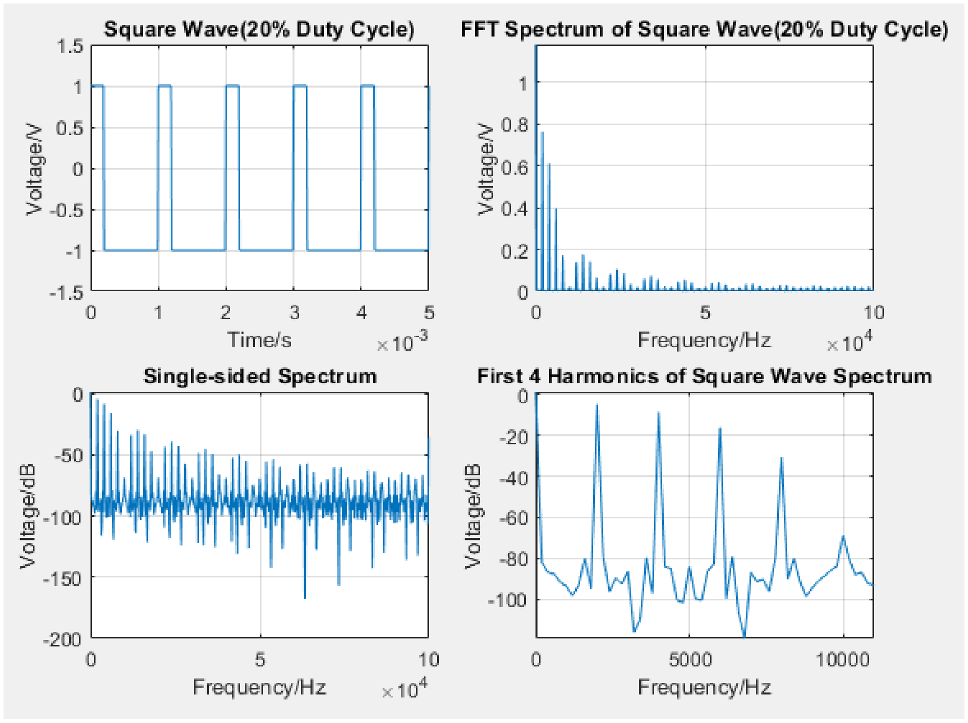


Figure 17: Subplot for Square Wave (20% duty cycle)

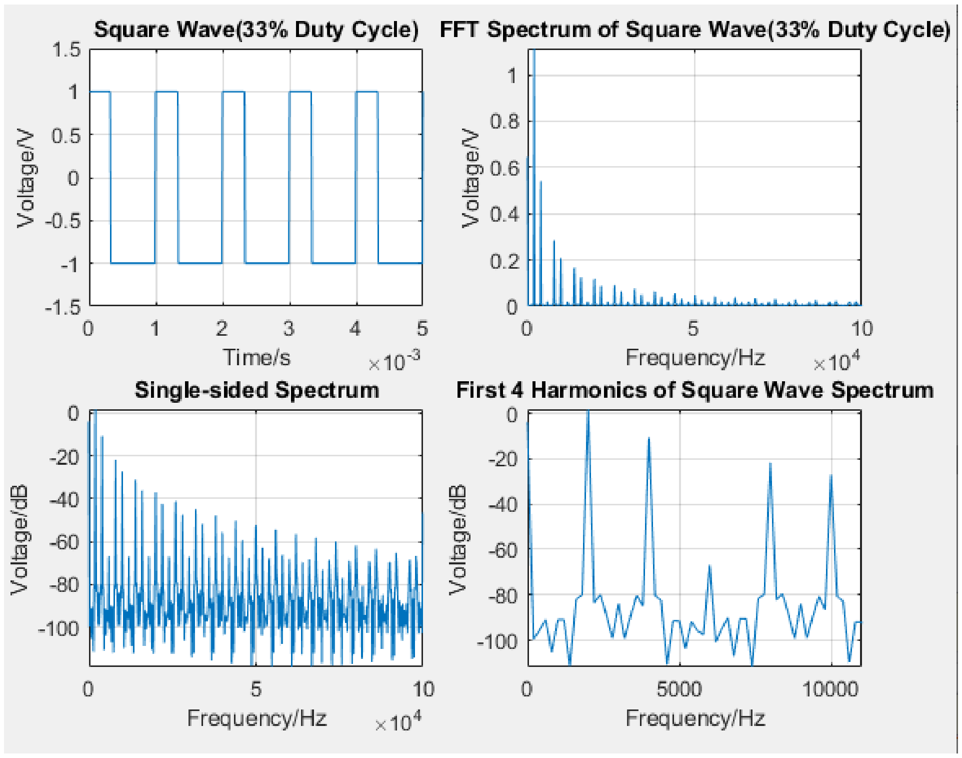


Figure 18: Subplot for Square Wave (20% duty cycle)

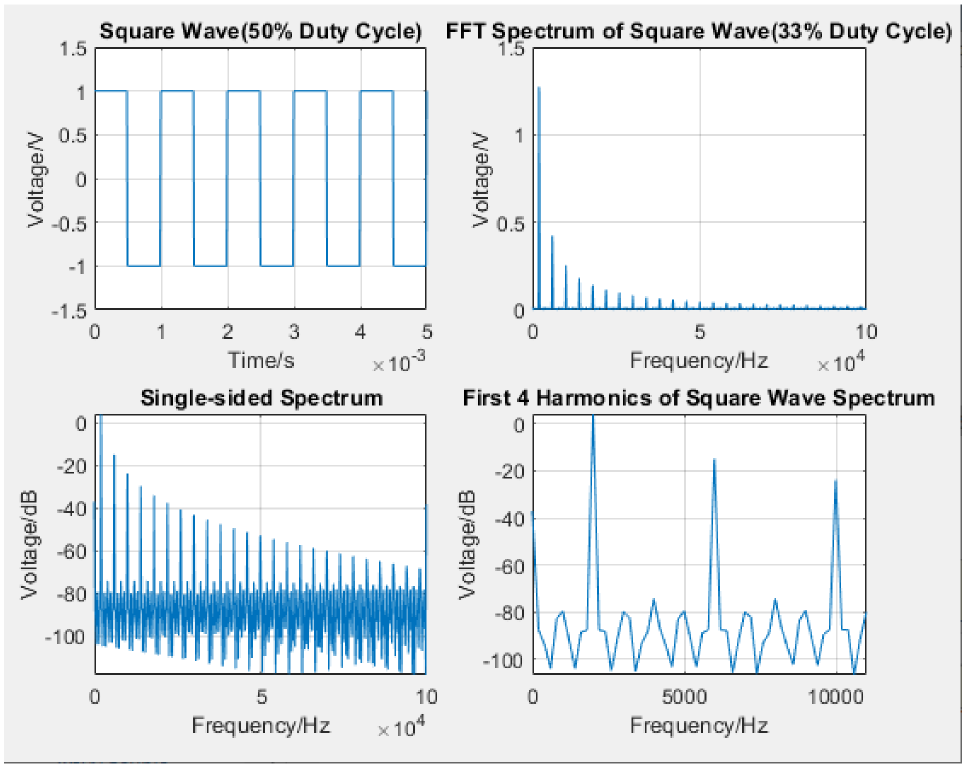


Figure 19: Subplot for Square Wave (50% duty cycle)

We require the following equation to describe this phenomenon:

$$c_k = \frac{2}{k\omega_0 T}\sin(k\omega_0 T_1) = \frac{1}{k\pi}\sin\,(k\omega_0 T_1)$$

From the equation, we see that when a function has lower pulse width T_1, the magnitude of its spectrum is higher. Again, when the function has higher T_1, the magnitude of the spectrum is lower. We see this phenomenon in our subplots. For 20% duty cycle, the magnitude of its spectrum is highest. However, for 50% duty cycle, the magnitude of the spectrum is lowest.

Problem 4: FFT of a sound sample

Question 1 & 2

```
[AudioInput,Fs] = audioread("sound_sample.wav");
t = 0:1/Fs:(numel(AudioInput)-1)/Fs;
L = 450;
t_dom = t(1:450);
Data = AudioInput(1:L);
plot(t_dom, Data);
xlabel("Time/s");
ylabel("Amplitude");
ylim([-1.2, 1.2]);
xlim([0, 0.01]);
title("Sound Sample in Time Domain");
```

```
grid on;
```

The plot is provided below:

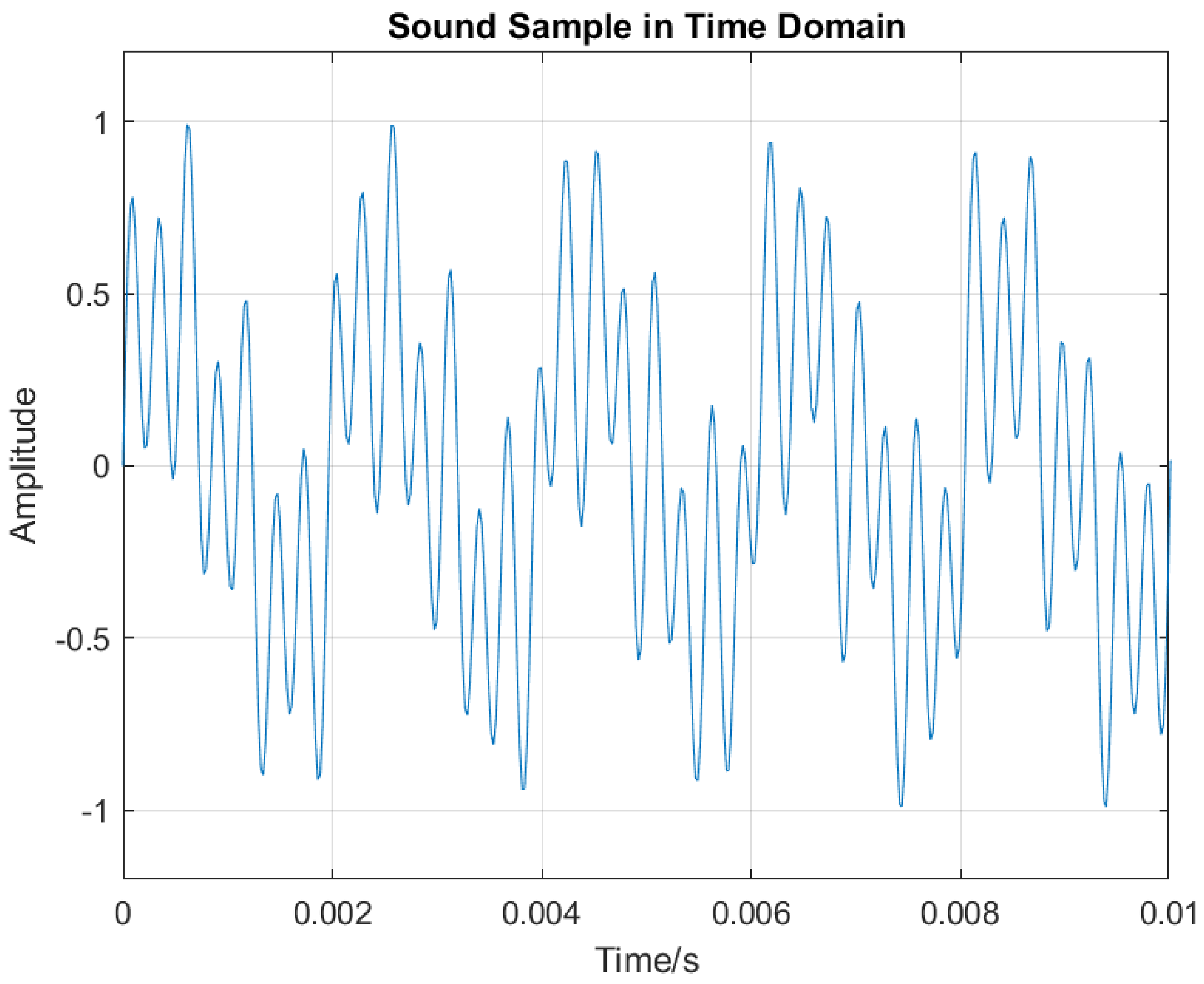


Figure 20: Sound Sample in Time Domain

<u>Question 3</u>

Matlab script is provided below:

```
[AudioInput,Fs] = audioread("sound_sample.wav");
L = numel(AudioInput);
t = 0:1/Fs:(L-1)/Fs;
Spec = fft(AudioInput)/L;
Spec2 = abs(Spec(1:floor((L+1)/2)));
Dom = (0:(L-1)/2);
fDom = Fs*Dom/L;
plot(fDom, Spec2);
xlim([0,4000]);
```

```
xlabel("Frequency");
ylabel("Amplitude");
grid on;
```

Matlab Plot is provided below:

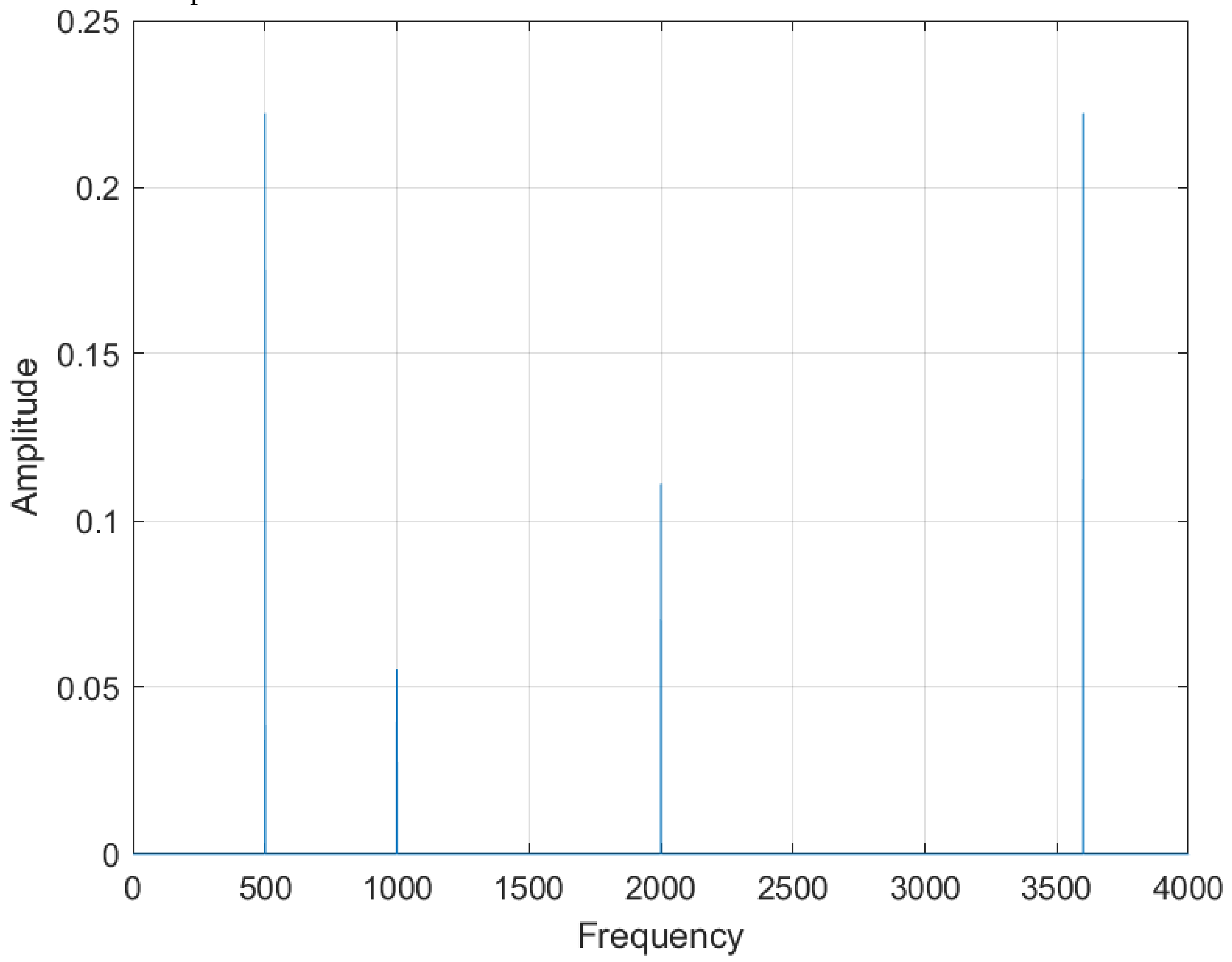


Figure 21: Matlab Plot of Spectrum

Question 4

The tones forming the signal are 500Hz, 1000Hz, 2000Hz and 3600Hz.

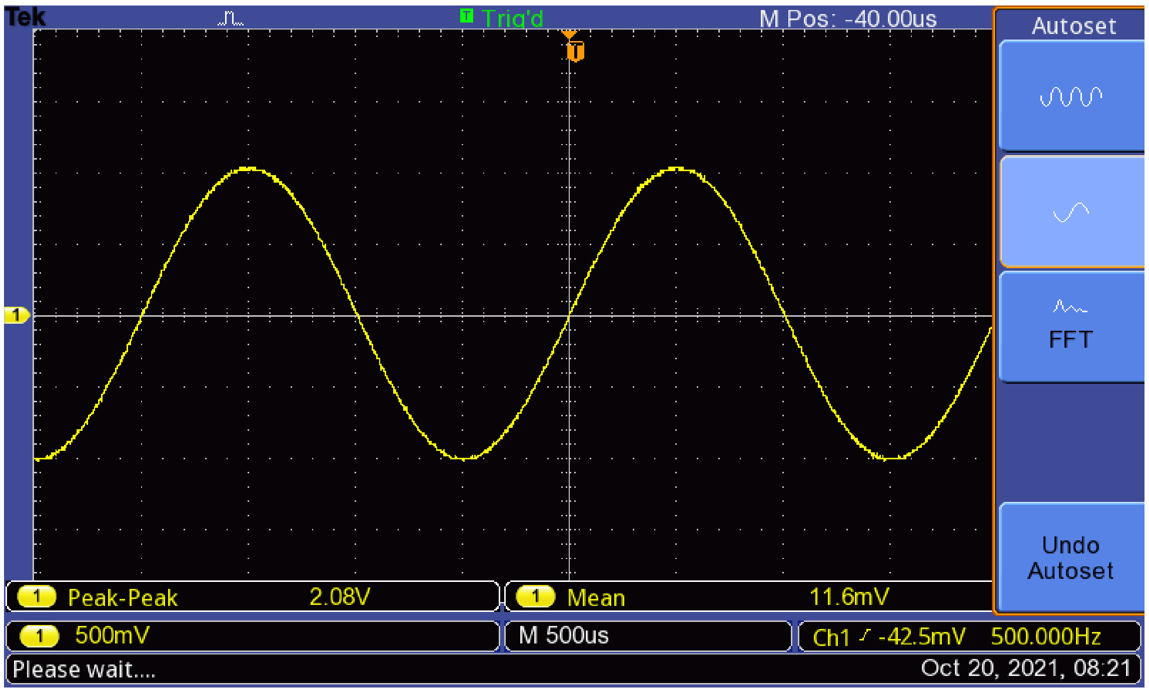
Experimental Setup and Results

Experimental Set-up:

- Breadboard
- Tektronix Oscilloscope
- BNC Cable
- Agilent Signal Generator
- Auxiliary Signal Generator
- Resistors

To begin with, we used a function generator to generate a sinusoidal wave having 500Hz frequency, 2Vpp amplitude, and no offset. Then we used the measure function to verify all the properties of the function.

On the screen, the time domain function looked as follows:



1 Peak-Peak 2.08V
1 Mean 11.6mV
1 500mV
M 500us
Ch1 -42.5mV 500.000Hz
Oct 20, 2021, 08:21

Figure 22: Sine function, 500 Hz, 2Vpp, No Offset

Then, we obtained the FFT spectrum using the oscilloscope FFT function. We used the cursor to measure the properties. Using this we obtained the complete spectra.

A zoomed picture of the spectra is provided below:

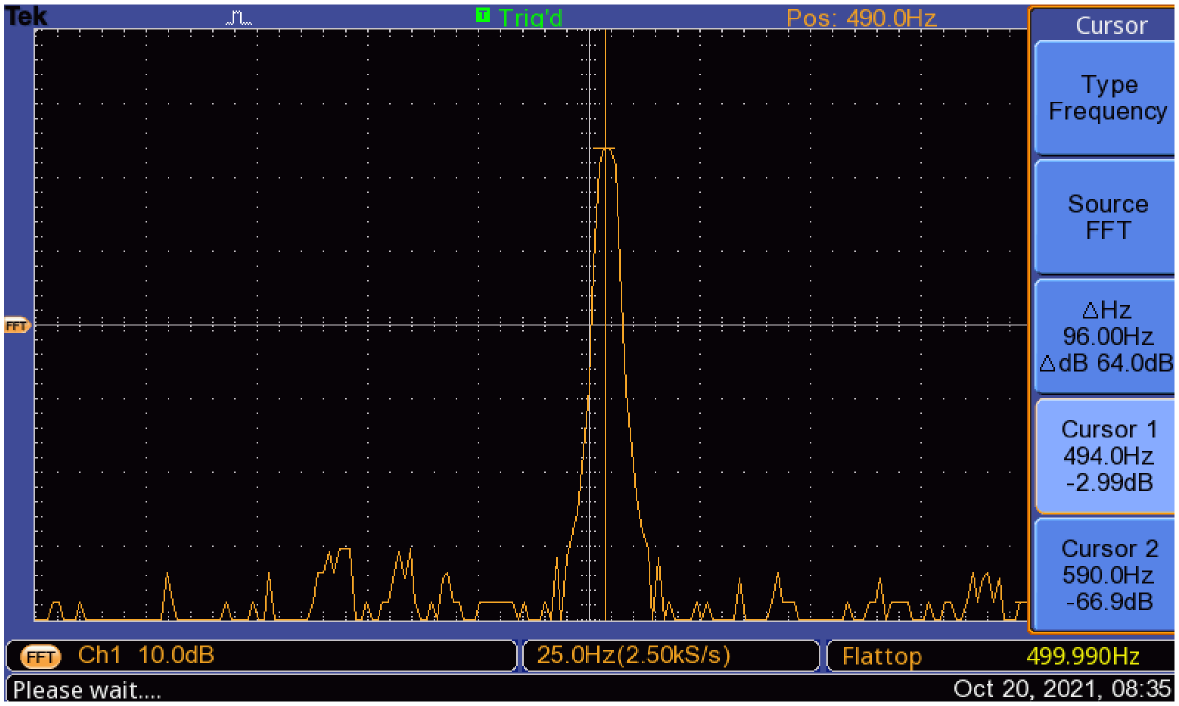


Figure 23: Zoomed picture of Spectrum

We then generated a sinusoidal wave with 0dB spectrum peak, 2kHz frequency, and no dc offset. The result is provided below:

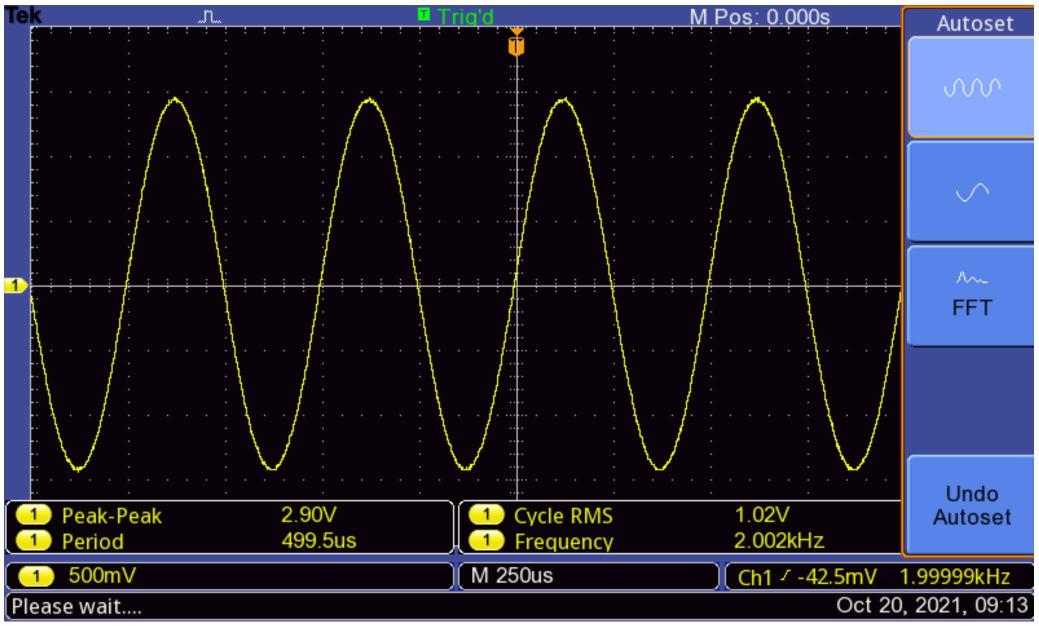


Figure 24: Sine wave for 0dB in time domain

A picture of the spectrum of the sine wave for 0dB is provided below:

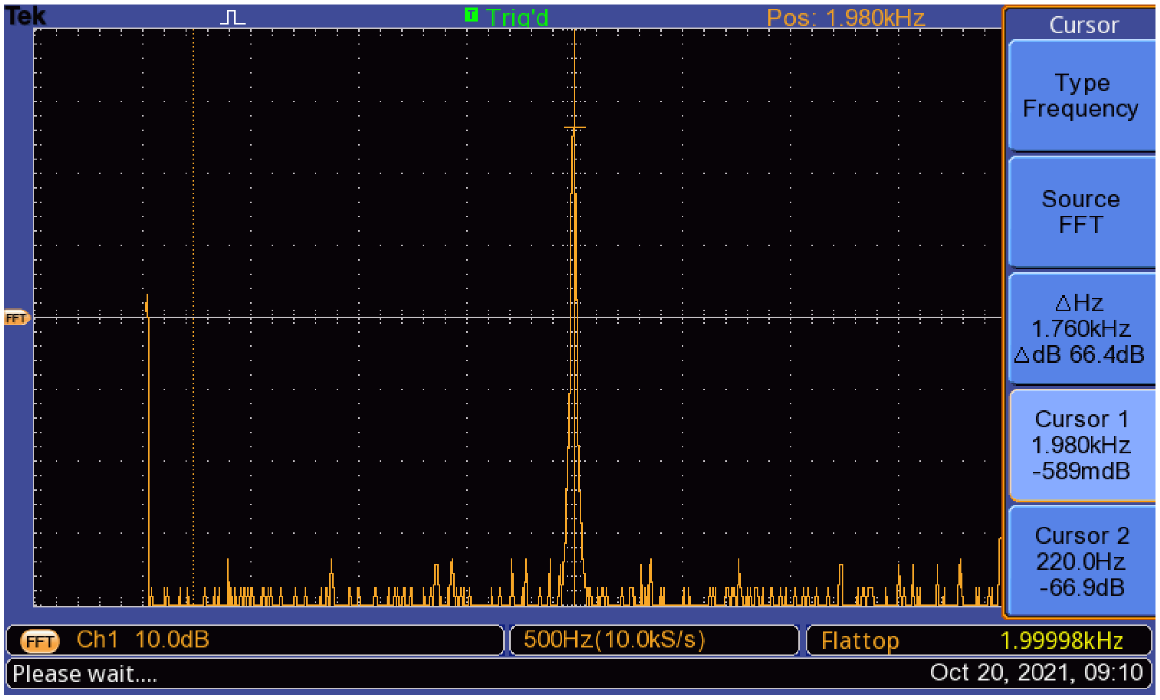


Figure 25: Spectrum peak of sine wave at 0dB

Using the measure function, we can draw the following conclusions from the first picture (sine wave in time domain):

$Vpp = 2.90V, Amplitude = 1.45\ V,\ \ Frequency = 2002\ Hz,\ \ Period = 499.5\mu s$

$Cycle\ RMS = 1.02V,$

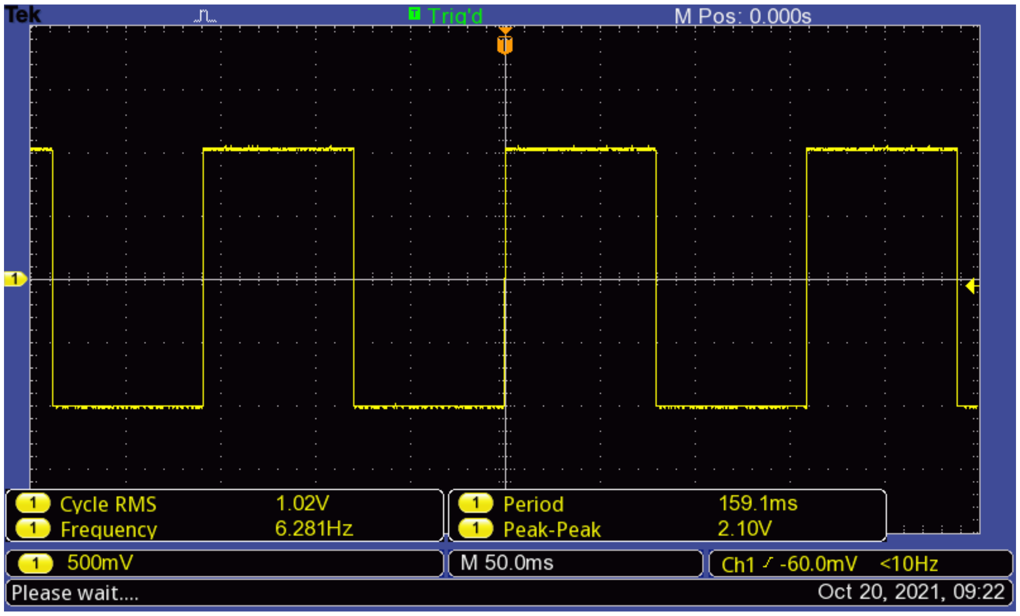
Using the cursor, we can draw the following conclusions from the second picture, which is the spectrum of the sine wave:

$$Spectrum\ Peak\ Amplitude = \ -589\ mdB$$

$$Peak\ appears\ at\ Frequency = 1980\ Hz$$

Problem 2: FFT of square wave

In this section, we used the function generator to generate a square wave with a 1ms period, 2 Vpp amplitude and no offset. The output is provided below:



```
Cycle RMS    1.02V        Period      159.1ms
Frequency    6.281Hz      Peak-Peak   2.10V
500mV                     M 50.0ms    Ch1 ⌐ -60.0mV   <10Hz
Please wait....                       Oct 20, 2021, 09:22
```

Figure 26: Square wave signal (Period = 1ms, Amplitude = 2Vpp)

Using the measured function, we saw that the wave has the following properties:

$$Cycle\ RMS = 1.02V$$

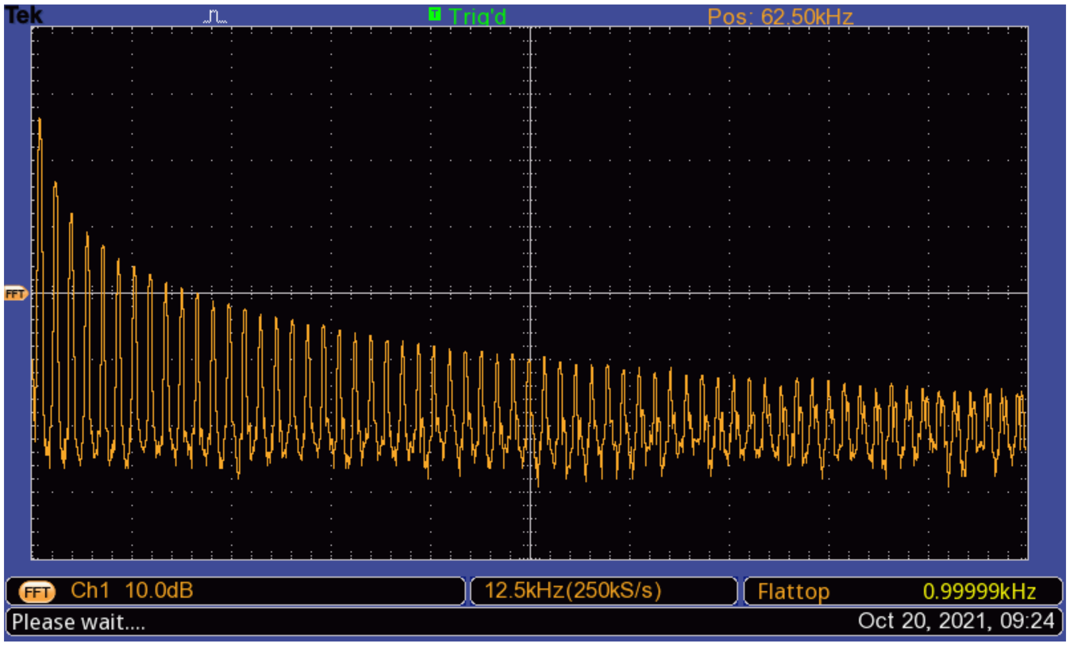
$$Frequency = 6.281\ Hz$$

$$Period = 159.1\ ms$$

$$Vpp = 2.10V$$

We then obtained the FFT spectrum of the square wave.

The spectrum is shown below:

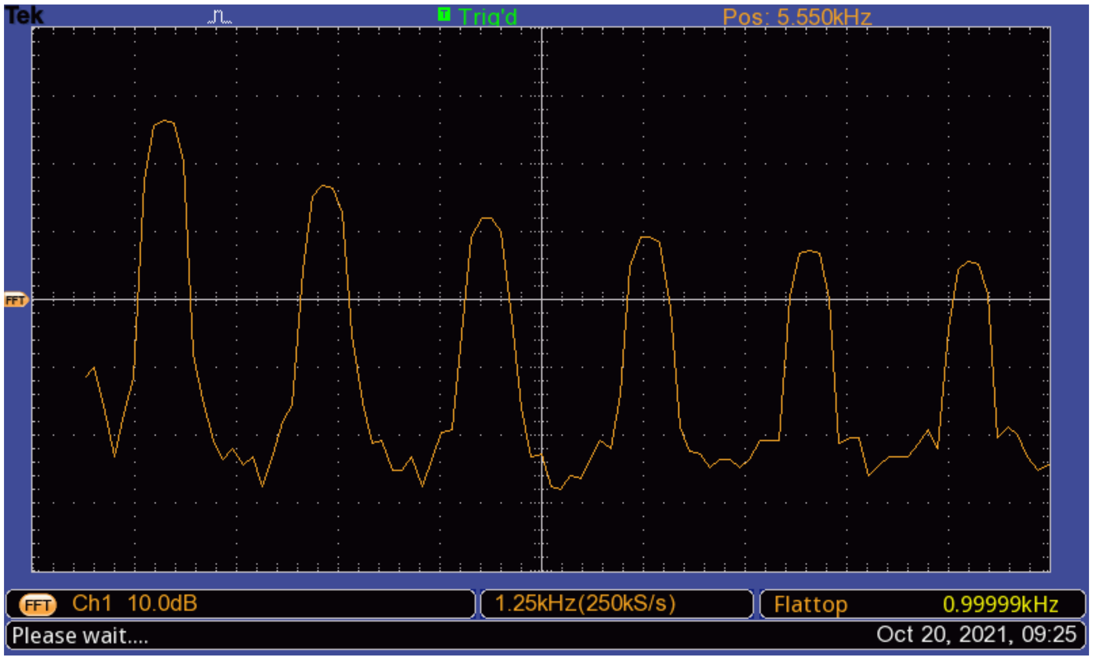


Ch1 10.0dB | 12.5kHz(250kS/s) | Flattop 0.99999kHz

Please wait.... Oct 20, 2021, 09:24

Figure 27: Spectrum of square wave (Period = 1ms, Amplitude = 2Vpp)

We used the FFT zoom control on this part rather than the time base (sec/div). This provided us a zoom factor of up to 10. The result is provided below:

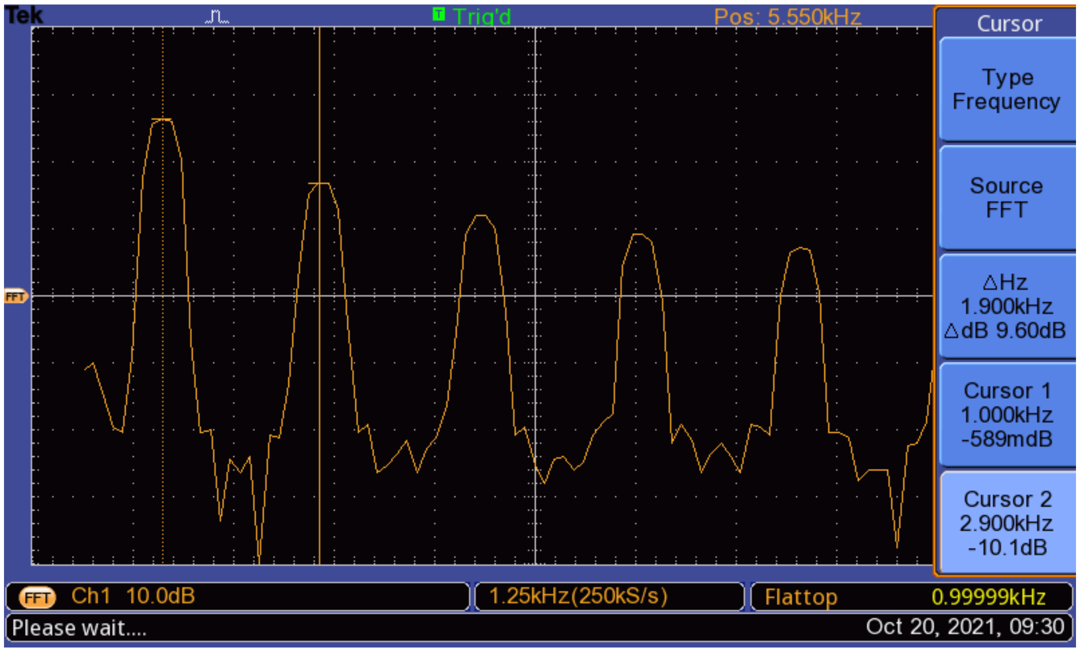


Ch1 10.0dB | 1.25kHz(250kS/s) | Flattop 0.99999kHz

Please wait.... Oct 20, 2021, 09:25

Figure 28: Spectrum after applying zoom control

We used the cursors to determine the amplitudes and the frequency of the fundamental and the first four harmonics. The output of the oscilloscope is provided below:

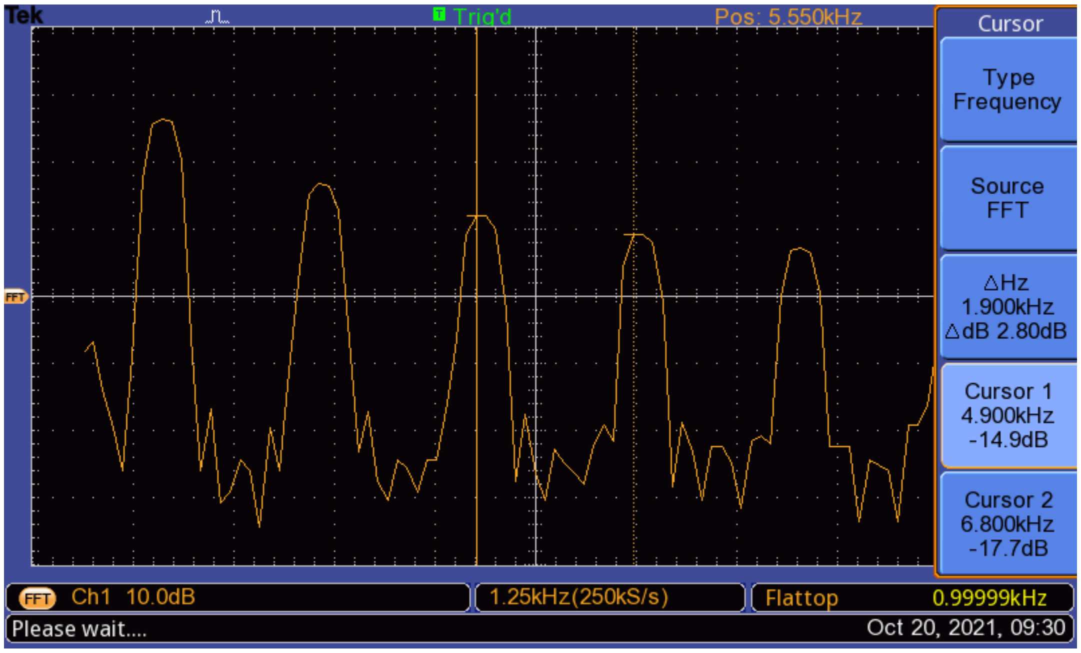


Pos: 5.550kHz

Cursor
Type Frequency
Source FFT
ΔHz 1.900kHz
ΔdB 9.60dB
Cursor 1 1.000kHz -589mdB
Cursor 2 2.900kHz -10.1dB

Ch1 10.0dB | 1.25kHz(250kS/s) | Flattop 0.99999kHz
Please wait.... Oct 20, 2021, 09:30

Figure : Figure 29: Measurement of amplitude of fundamental and first harmonic



Pos: 5.550kHz

Cursor
Type Frequency
Source FFT
ΔHz 1.900kHz
ΔdB 2.80dB
Cursor 1 4.900kHz -14.9dB
Cursor 2 6.800kHz -17.7dB

Ch1 10.0dB | 1.25kHz(250kS/s) | Flattop 0.99999kHz
Please wait.... Oct 20, 2021, 09:30

Figure 30: Measurement of amplitude of the second and third harmonics

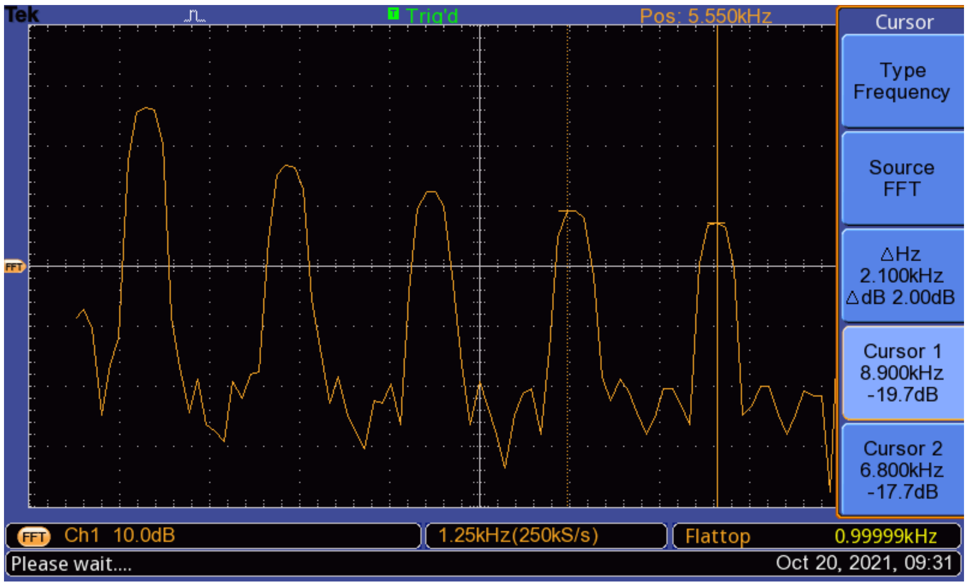


Figure 31: Measurement of amplitude of the fourth harmonic

The results are provided below:

Harmonics	Frequency	Amplitude
Fundamental	1000 Hz	-589 mdB
First	2900 Hz	-10.1 dB
Second	4900 Hz	-14.9 dB
Third	6800 Hz	-17.7 dB
Fourth	8900 Hz	-19.7 dB

Then, we generated another square wave at 20% duty cycles. The output is provided below:

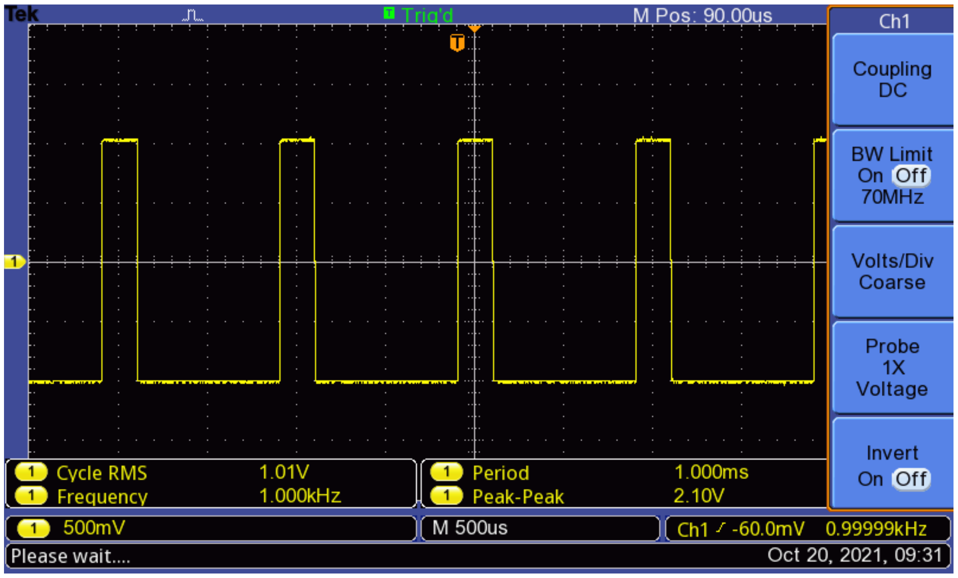


Figure 32: Square wave (20% duty cycle)

We then generated the spectrum of this square wave. The display is provided below:

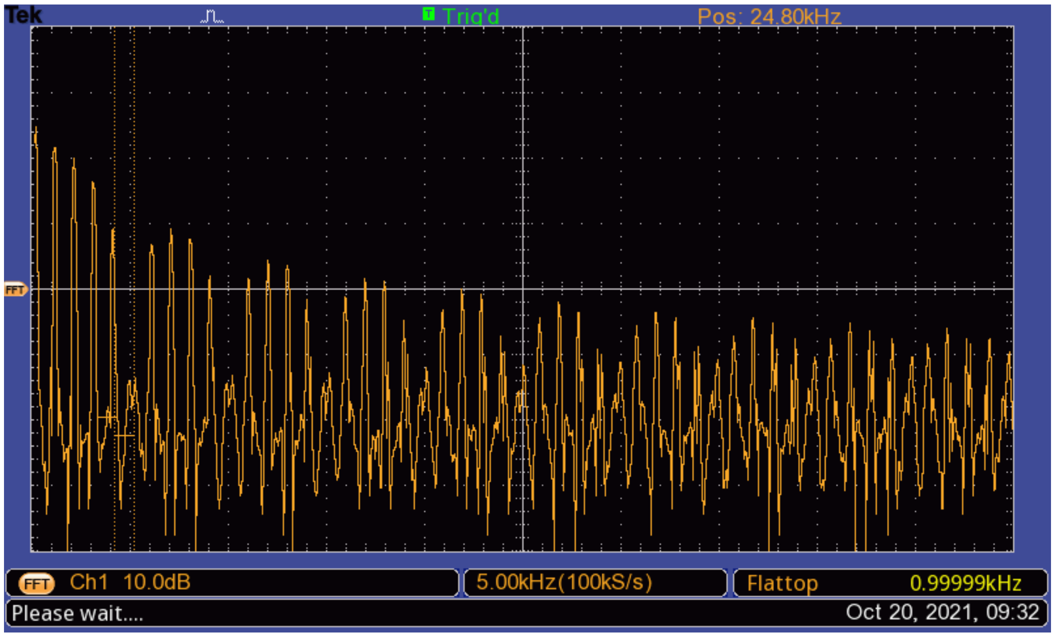


Figure 33: Spectrum of square wave (20% duty cycle)

Then, we measured the amplitudes of the fundamental frequency and the first four harmonics. The results are provided below:

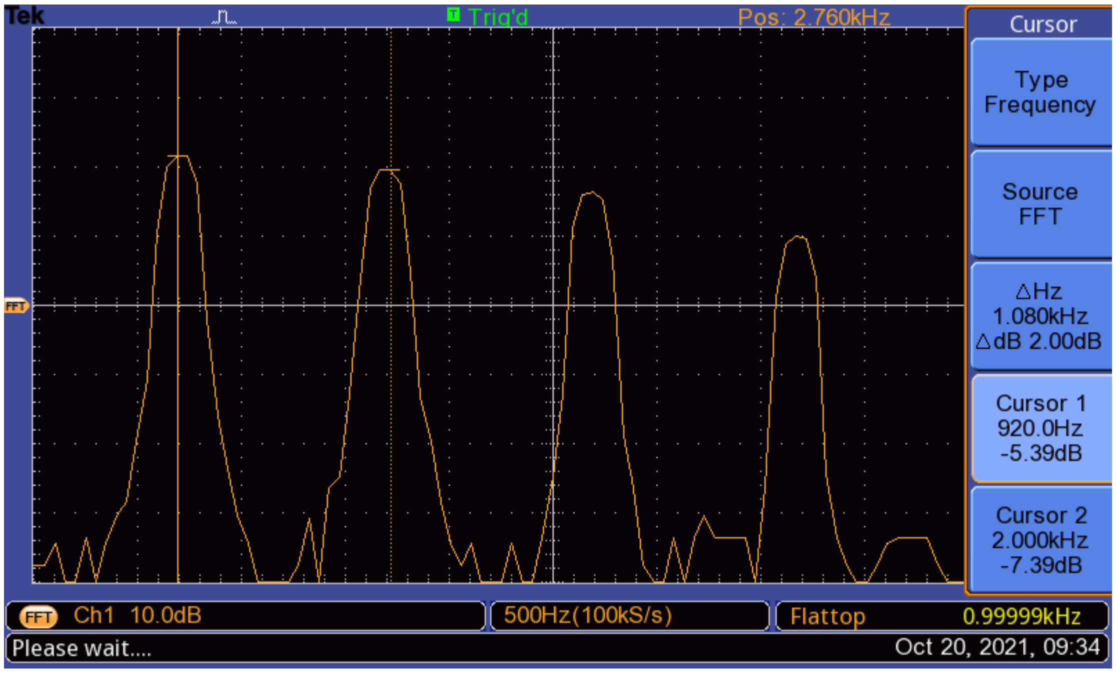


Figure 34: Measurement of amplitude of fundamental and first harmonic

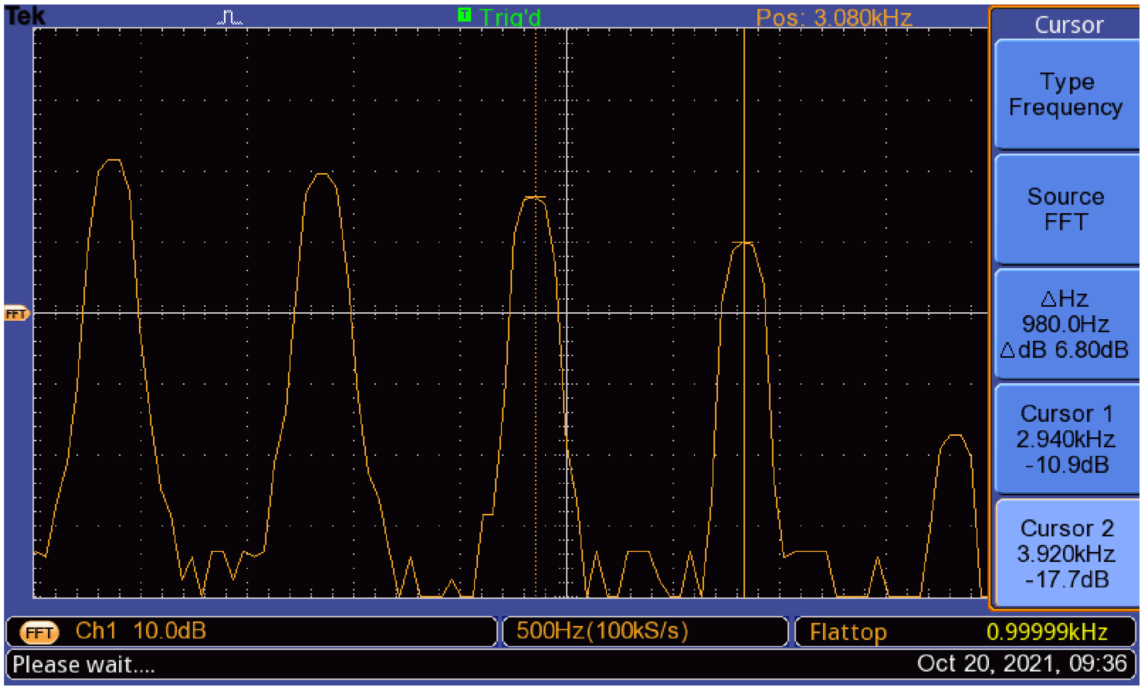


Figure 35: Measurement of amplitude of second and third harmonic

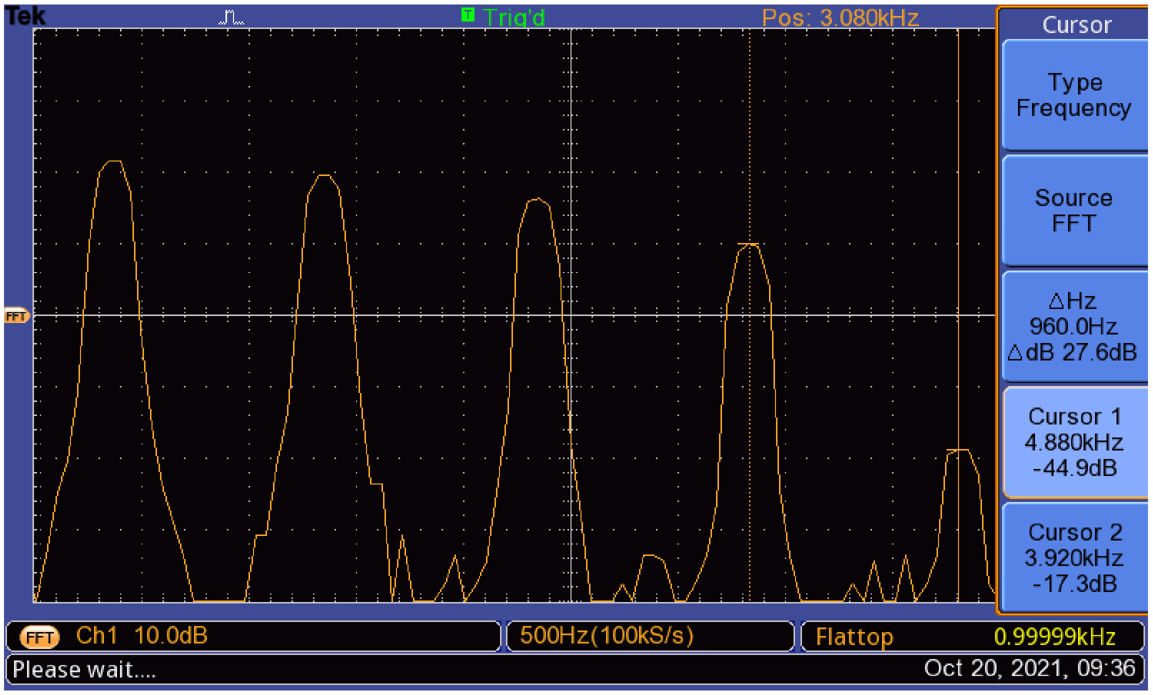


Figure 36: Measurement of amplitude of fourth harmonic

After making the measurements, we tabulated the results.

The table with the measurements is provided below:

Harmonics	Frequency	Amplitude
Fundamental	920 Hz	-5.39 dB
First	2000 Hz	-7.39 dB
Second	2940 Hz	-10.5 dB
Third	3920 Hz	-17.3 dB
Fourth	4880 Hz	-44.9 dB

Problem 2: FFT of Multiple-Tone sinusoidal wave

To execute the task on this section, we generated a 2 Vpp, 10 KHz sinusoidal wave on the Agilent signal generator, and combined the signal from the sine output of the auxiliary signal generator with it. The circuit has the following setup:

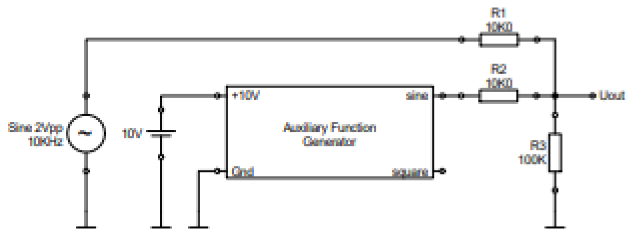


Figure : Circuit setup for multi-tone sinusoidal wave generation

The output on the oscilloscope was as follows:

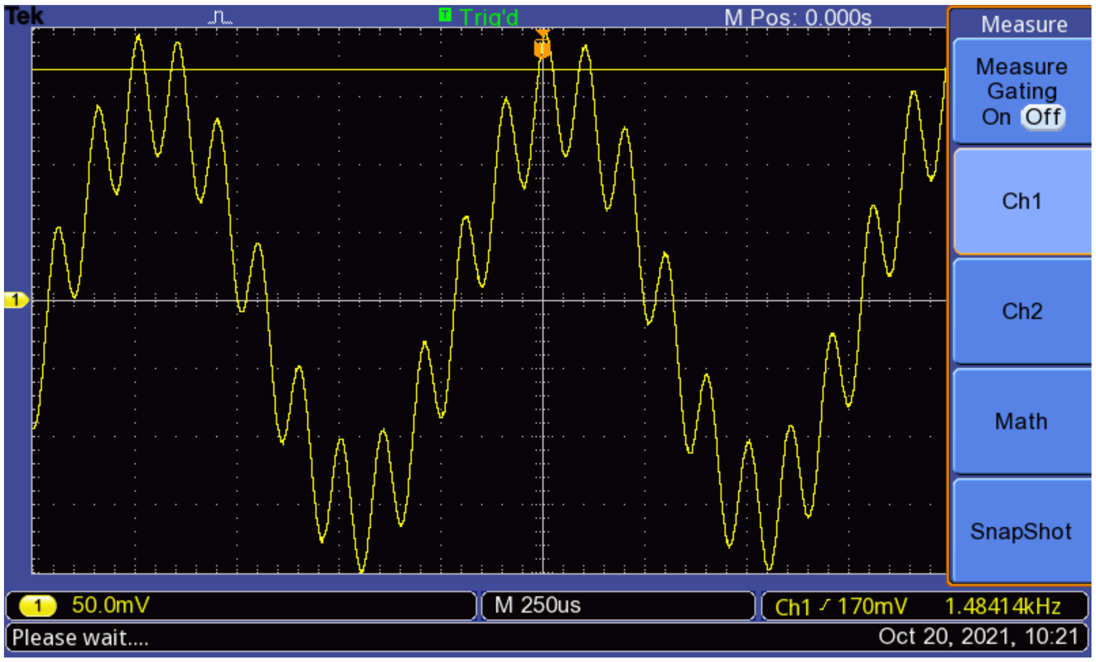


Figure 37: Oscilloscope output for multiple-tone sinusoidal wave

We then generated the FFT spectrum of the signal on the oscilloscope. The output is provided below:

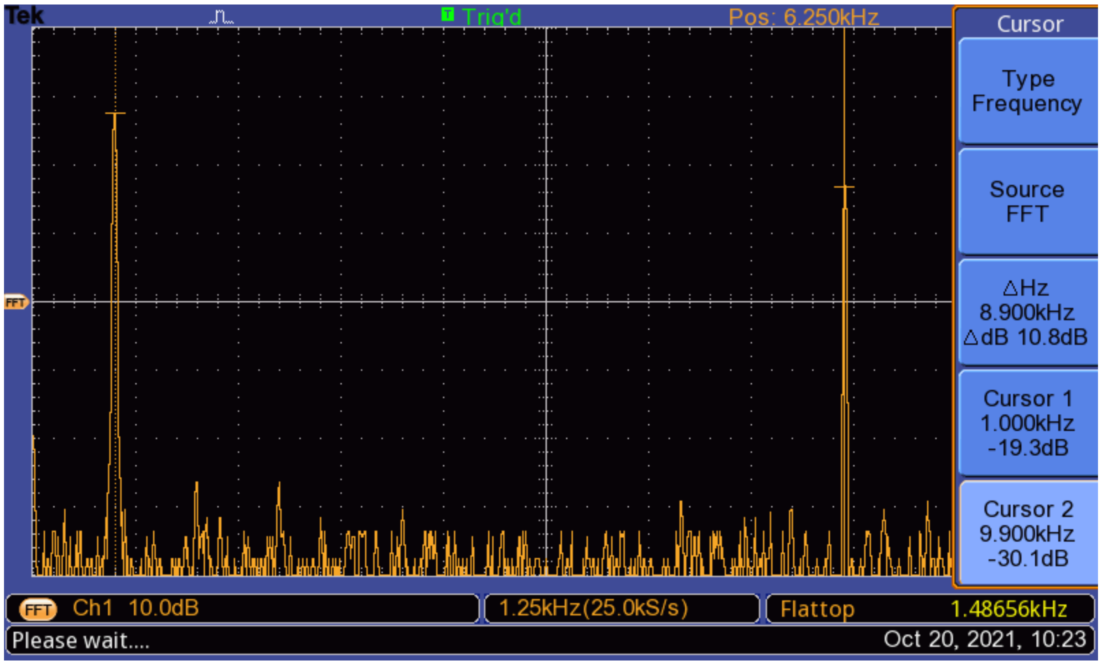


Figure 38: Spectrum of multi-tone sine wave

Evaluation

Problem 1: FFT of Single Tone sinusoidal wave

Question 1

The reference value for the oscilloscope at 0dB is 1 V_{RMS}.

Question 2

We have a sine wave with 500 Hz frequency and 2Vpp amplitude. We can generate a Matlab plot for the sinusoidal wave and its spectrum with the following script:

```
%Generating Sine wave:
t = 0:0.00001:0.01;
f = 500;
w0 = 2*pi*f;
sw = sin(w0*t);

%Plotting sine wave:
figure(1);
plot(t, sw, '-r');
xlabel('Time/s');
ylabel('Voltage/V');
title('Sine Wave Plot');
ylim([-1.2, 1.2]);
grid on;

%Implementing FFT:
Fs = 100000;
F_nyq = Fs/2;
L = length(sw);
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft/L);
db_sw_fft = db(sw_fft/sqrt(2));
db_sw_fft = db_sw_fft(1:L/2+1);


%plotting the spectrum:
figure(2);
freq_dom = linspace(0, F_nyq, L/2+1);
plot(freq_dom, db_sw_fft, '-r');
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
title('FFT Spectrum');
xlim([0, 1000]);
ylim([-40, 0]);
```

```
grid on;
```

The results are provided below:

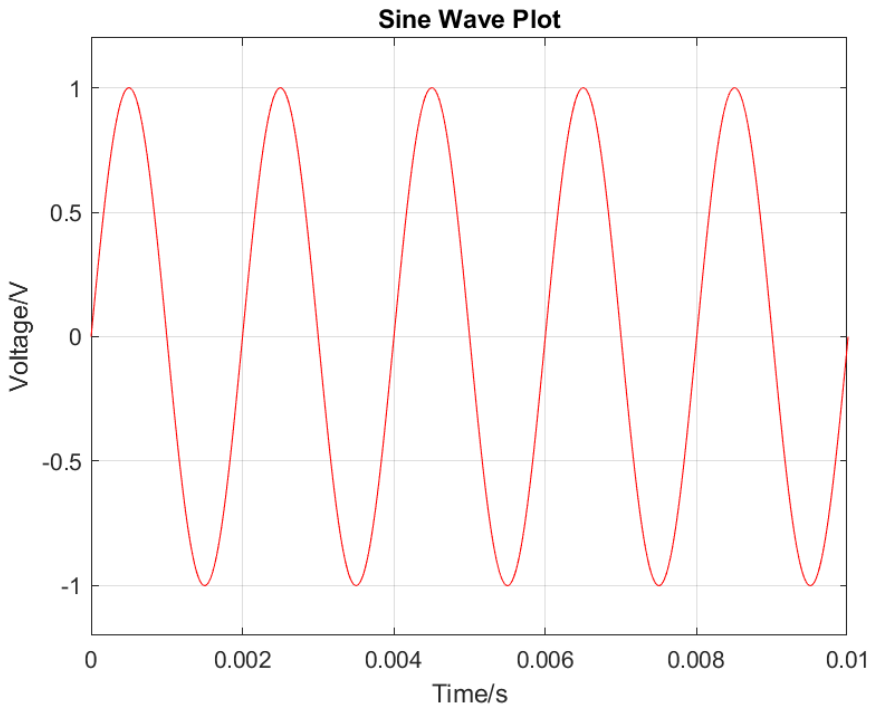


Figure 39: Matlab plot of Sine wave in time domain

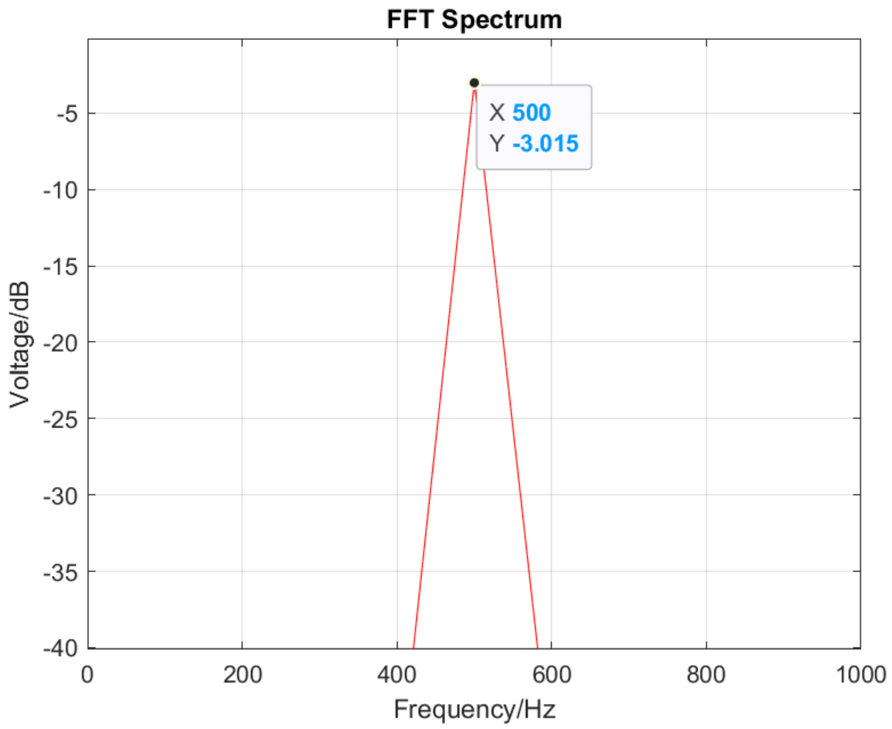


Figure 40: Matlab plot of Sine wave in frequency domain

The results of our measurements and calculations can be tabulated as follows:

	Frequency	Amplitude
Measured (Oscilloscope)	494 Hz	-2.99 dB
Calculated (Matlab)	500 Hz	-3.015 dB

We see on the Matlab plot that the spectrum is represented by a peak at 500 Hz, where it has an amplitude of -3.015 dB. The measured values are very close to the calculated values, with measured amplitude being -2.99 dB at 494 Hz. The error is very small (E(dBVrms) < 1%), which means the calculated spectrum is consistent with the measured spectrum. It's possible that the small error is due to error in measurement of the oscilloscope or small deviation in the precision of cursor as it's very difficult to align the cursor with the peak. There might also be some fault in the accuracy with which the signal generator operates.

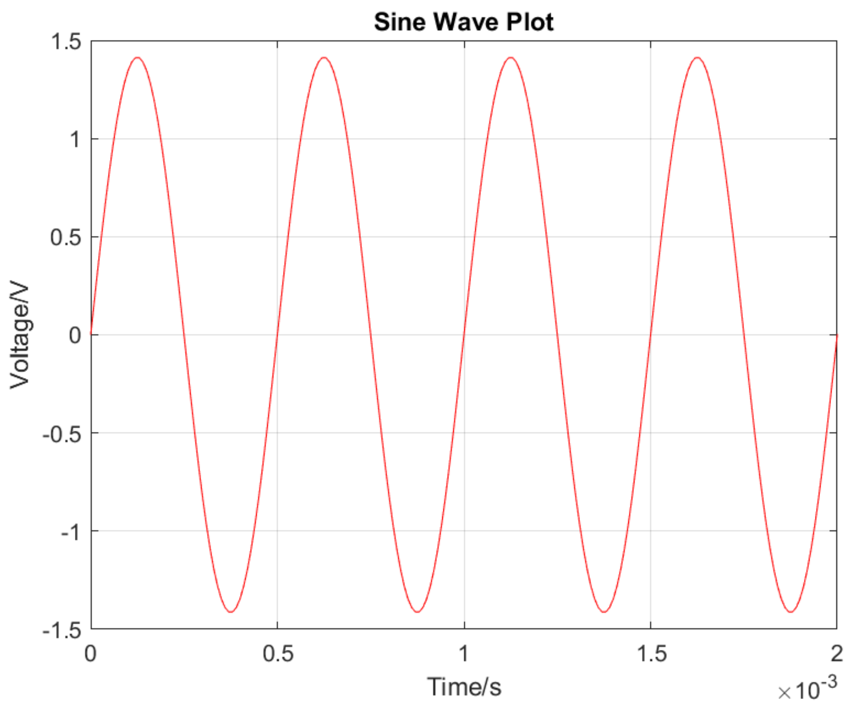
Question 3

The next FFT spectra can be calculated using the following script:

```
Fs = 100000;
%Generating Sine wave:
t = 0:1/Fs:0.01;
f = 2000;
w0 = 2*pi*f;
sw = sqrt(2)*sin(w0*t);

%Plotting sine wave:
figure(1);
plot(t, sw, '-r');
xlabel('Time/s');
ylabel('Voltage/V');
title('Sine Wave Plot');
ylim([-1.5, 1.5]);
xlim([0,0.002]);
grid on;

%Implementing FFT:
F_nyq = Fs/2;
L = length(sw);
sw_fft = fft(sw);
sw_fft = 2*abs(sw_fft/L);
db_sw_fft = db(sw_fft/sqrt(2)); %calculating dB of Vrms
db_sw_fft = db_sw_fft(1:L/2+1);


%plotting the spectrum:
figure(2);
```

```
freq_dom = linspace(0, F_nyq, L/2+1);
plot(freq_dom, db_sw_fft, '-r');
xlabel('Frequency/Hz');
ylabel('Voltage/dB');
title('FFT Spectrum');
xlim([0, 4000]);
ylim([-30, 1]);
grid on;
```

The plot of the sine function is provided below:



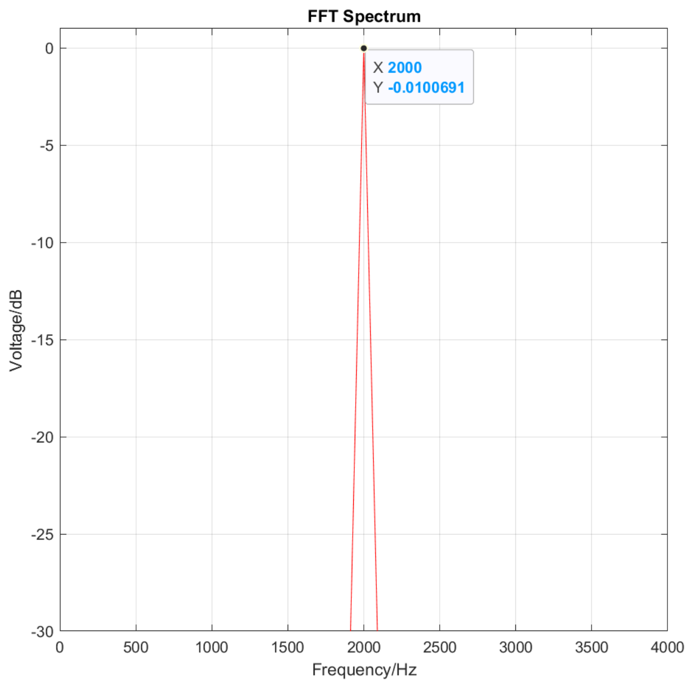


Figure 41: Sine wave constructed for 0dB

Figure 42: Spectrum at 0dB

The results can be tabulated as follows:

	Frequency	Amplitude
Measured (Oscilloscope)	1980 Hz	-0.589 dB
Calculated (Matlab)	2000 Hz	-0.0100691 dB

We see on the Matlab plot that the spectrum is represented by a peak at 2000 Hz, where it has an amplitude of approximately -0.01dB. In contrast, the measured amplitude at 1980 Hz is -0.589 dB. Even though both values are considered sufficiently close to the intended 0dB peak, the difference in dB scale in this case is very large, which means the calculated spectrum is not consistent with the measured spectrum in dB. It's possible that the large error is due to limitations of the equipment. Some faults are, again, the error in measurement of the oscilloscope or the small deviation in the precision of cursor as it's very difficult to align the cursor with the peak. There might also be some fault in the accuracy with which the signal generator operates.

Question 4

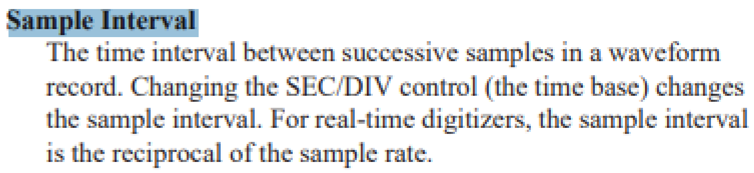
The spectra obtained through calculation have more defined peaks compared to those obtained through the oscilloscope. The calculated spectra have pointed peaks while the peaks on the oscilloscope are slightly curved. The values of the calculated data also differ somewhat from the measured data, but the difference is very small and within a certain range, the values confirm each other.

Differences in values can be attributed to some limitations of the equipment. For example, there might be error in measurements made by the oscilloscope, or the signal generator might not be generating a signal that is exactly accurate. It is also not possible to position the cursor in the oscilloscope exactly on the peak due to limitations in precision.

Problem 2: FFT of square wave

Question 1

In the TDS200 Manual, we find the following:

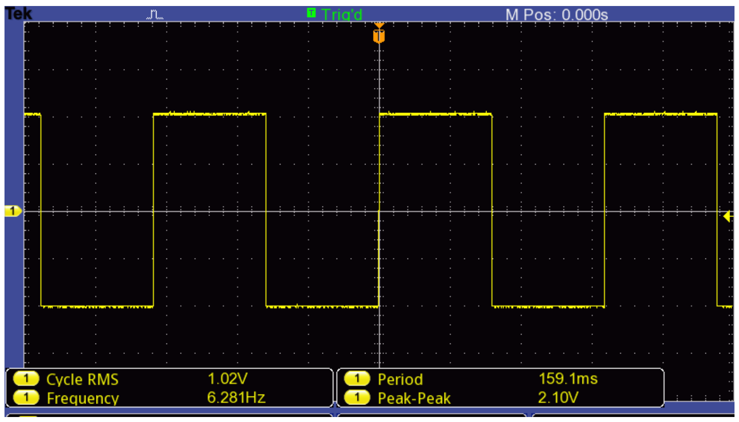


Sample Interval

The time interval between successive samples in a waveform record. Changing the SEC/DIV control (the time base) changes the sample interval. For real-time digitizers, the sample interval is the reciprocal of the sample rate.

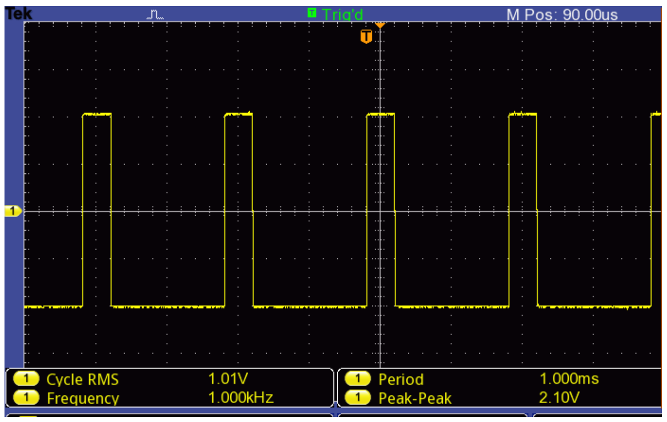
Therefore, as we increase SEC/DIV and consequently the sample interval, we are also decreasing the sampling rate. Hence, our data is sampled at larger intervals, and we are taking less samples in a fixed period of time. Therefore, this may reduce the bandwidth of the signal.

Question 2



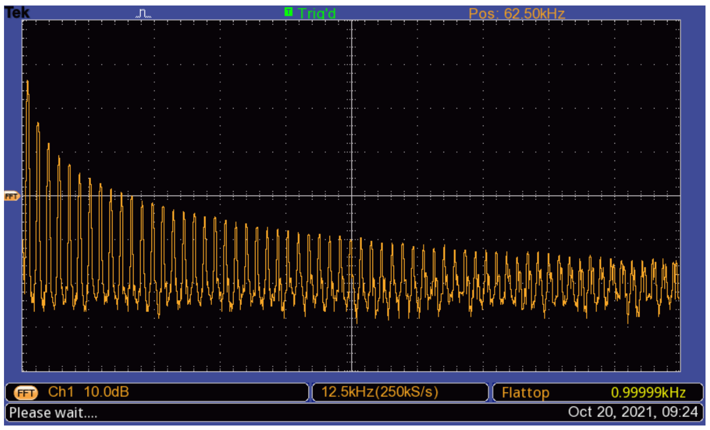
Cycle RMS 1.02V
Frequency 6.281Hz
Period 159.1ms
Peak-Peak 2.10V

Figure 43: Square Wave (50% duty cycle)



Cycle RMS 1.01V
Frequency 1.000kHz
Period 1.000ms
Peak-Peak 2.10V

Figure 44: Square Wave (20% duty cycle)



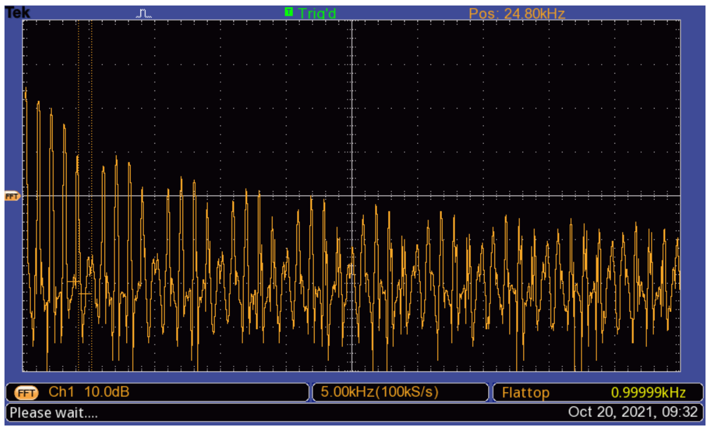


Figure 45: Square Wave Spectrum (50% duty cycle)

Figure 46: Square Wave Spectrum (20% duty cycle)

The duty cycle is the ratio of pulse width to period. A higher pulse width results in a wider pulse. In the figures above, we see that when the duty cycle is 50%, the square wave is wider compared to the square wave created with 20% duty cycle. However, the spectrum generated for the square wave with 50% duty cycle has lower width in between peaks compared to the spectrum of the square wave made with 20% duty cycle. Therefore, a larger duty cycle results in narrower FFT spectrum width and vice versa.

Problem 3: FFT of Multiple-Tone sinusoidal waves

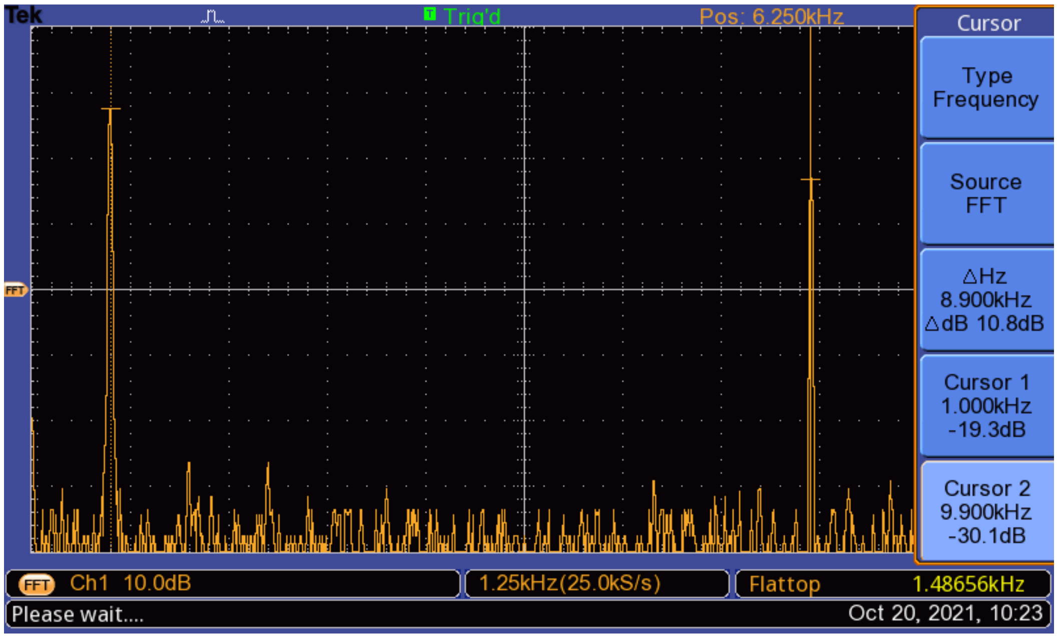


Figure 47: FFT of multi-tone sinusoidal waves

The signal produced in the time domain is constructed by combining two sinusoidal signals with 1kHz frequency and 10 kHz frequency. Mathematically, we can think of this as summing two sine functions with frequencies 1kHz and 10 kHz.

From linearity, we know that if we add two functions and take Fourier Transform of the sum, then the result is the same as what we will get from summing the individual Fourier Transforms of the individual functions.

Mathematically,

$$F\{f(x) + g(x)\} = F\{f(x)\} + F\{g(x)\}$$

Therefore, when we took FFT of the sum of the functions, we found peaks at frequencies of the sinusoids that were combined to construct the signal that we see on the oscilloscope, namely, at approximately 1kHz and 10kHz. The small difference is due to the error of the equipment.

This means FFT maintains the linearity property. We would get a peak at 1kHz if we took FFT of the sinusoid with 1kHz frequency, and a peak at 10kHz if we took an FFT of the sinusoid with 10kHz frequency. If we added these, we would get two peaks, one at 1kHz and 10kHz. However, since we summed the sinusoids first and then took FFT, we are seeing peaks at 1kHz and 10kHz.

Conclusion

In this lab we obtained a deep understanding of the Fourier Transform. We learned how to calculate Fourier transform of different functions conceptually and using Matlab, and how to use the fft() function provided by Matlab to apply the Fast Fourier Transform on our signals to obtain the Spectrum of the function. We learned how to construct sinusoids and square waves in Matlab and how to analyze them by developing their spectrum and relating to features of the Fourier transform.

In the lab, we generated sinusoids and square waves using the signal generator and studied their spectra using the oscilloscope. The FFT feature, the measure function and the cursors were very helpful in this regard. They allowed us to take measurements of very specific values that we required for our experimentation.

We were also able to combine two sinusoids using the auxiliary signal generator and the source signal generator, and develop its FFT, which showed us two peaks. This allowed us to obtain an understanding of the linearity property of the Fourier Transform.

Comparing the information we obtained in the prelab with our experimental results, we were able to understand how experimental data relates to theoretical findings in case of the Fourier transform. We were able to understand the limitations of the equipment with regards to different situations, and how to make the best use of the equipment at hand.

In evaluation, we reconstructed theoretical evidence for our experimental data and made comparisons. We were able to notice the small nuances that arise that make the experimental outcomes different or similar when compared to the theoretical data. We also understood conditions that should hold when we try to validate our experimental data using theoretical means.

References

- Signals and Systems Lab Manual (Uwe Pagel)
- http://www.faculty.jacobs-university.de/upagel/
- https://download.tek.com/manual/071039803.pdf

Appendix

Prelab: Sampling

Problem 1: The Sampling Theorem

Question 1

A low pass filter is a filter that passes signals with frequency lower than the selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. When sampling, we do not require frequencies that are higher than the Nyquist frequency. Therefore, the low pass filter is used to remove those frequencies to avoid aliasing, which could avoid any distortion or artifact.

Question 2

The original signal can be reconstructed if the sampling frequency is greater than two times the maximum frequency of the band-pass limited signal.

Therefore, the signal can be reconstructed if:

$$f_{sampling} \geq 2 \times f_0 = 2 \times 3kHz = 6kHz$$

f_0 is the fundamental frequency of the sampled signal.

Question 3

The Nyquist Frequency is the minimum rate at which a signal can be sampled so that the no errors are introduced and the original signal can still be fully reconstructed.

Question 4

The resulting frequencies are calculated using the formula:

$$f_{alias} = \left| f_0 - f_s \cdot nint\left(\frac{f}{f_s}\right) \right|$$

$$f_0 = input\ signal\ frequency$$

$$f_s = sampling\ frequency$$

$$nint(n)\ provides\ the\ nearest\ integer\ of\ n$$

Based on this, the resulting frequencies are: 500Hz, 2500Hz, 0Hz and 500Hz

Question 5

We know,

$$f_r = |nf_s + f_a|$$

Here,

$$f_r = resulting\ frequency$$

$$n = an\ integer$$

$$f_s = sampling\ frequency$$

$$f_a = aliasing\ frequency$$

Based on the formula, we have:

$$f_s = 30\ Hz$$

$$n = 1, 2, 3\ (for\ example)$$

$$f_a = 7\ Hz$$

Therefore, the resulting alias to 7Hz are 37Hz, 67Hz and 97Hz.

Problem 2: Impulse Train Sampling and Real Sampling

Question 1

(a) Under Sampling

The results of under-sampling can be achieved using the following Matlab script:

```
Fs=100*10^3; %sampling rate
fs=48;       %sampling rate
t=0:1/Fs:1;
fx = 50;
w0 = 2*pi*fx;
x= 5*sin(w0*t); %Input signal

subplot(3,1,1);
plot(t,x);      %generating input signal plot
title('Input Signal');
ylabel('X(t)/V');
grid on;
hold on;

wp = 2*pi*t*fs;
sw=(1+square(wp, 0.1))/2; %sampling signal
subplot(3,1,2);
plot(t,sw);                %plotting sampling signal
title('Sampling Unity Impulse Train');
ylabel('p(t)/V]');
grid on
hold on;

subplot(3,1,3);
```

```
xp=x.*sw;              %output signal
plot(t,xp);            %plotting output signal
title('Output Signal');
ylabel('Xp(t)/V');
grid on;
hold on;
xlabel('Time/s');

xlim([0, 1]);
ylim([-5, 5]);
```

The plot is provided below:

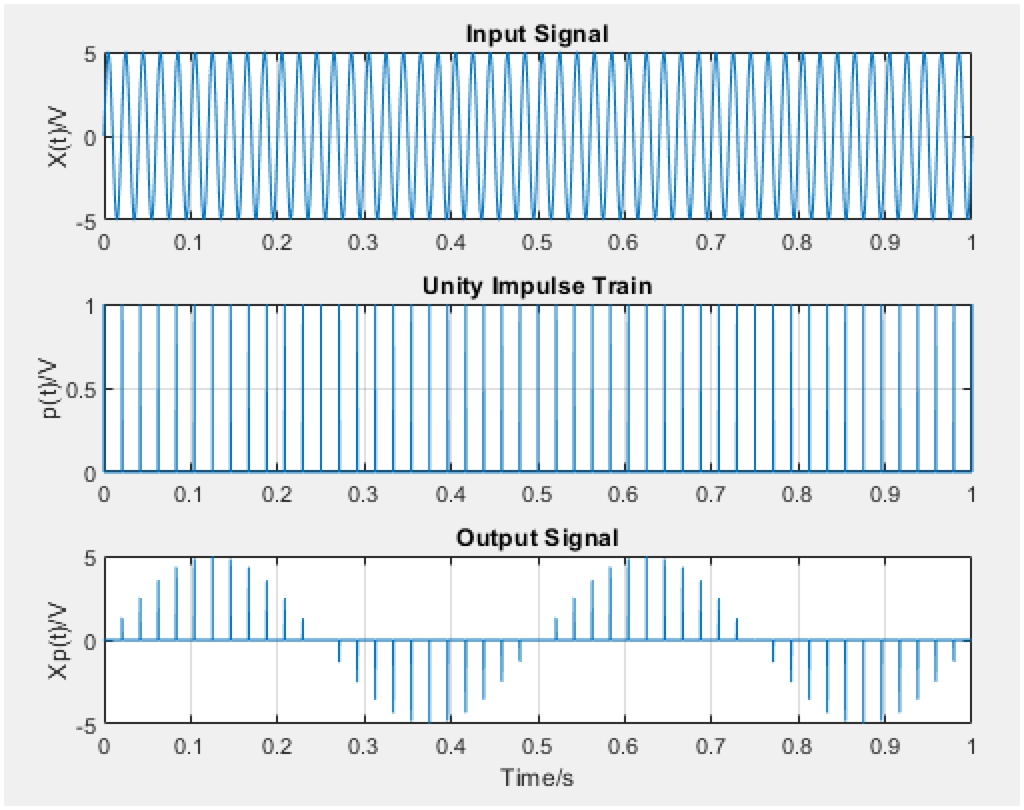


Figure 48: Results of under-sampling

(b) Nyquist Sampling

Nyquist sampling can be achieved using the following Matlab script:

```
Fs=100*10^3; %sampling rate
fs=100;        %sampling rate
t=0:1/Fs:1;
fx = 50;
w0 = 2*pi*fx;
x= 5*sin(w0*t); %Input signal
```

```
subplot(3,1,1);
plot(t,x);       %generating input signal plot
title('Input Signal');
ylabel('X(t)/V');
grid on;
hold on;
wp = 2*pi*t*fs;
sw=(1+square(wp, 0.1))/2; %sampling signal
subplot(3,1,2);
plot(t,sw);                %plotting sampling signal
title('Unity Impulse Train');
ylabel('p(t)/V');
grid on
hold on;
subplot(3,1,3);
xp=x.*sw;             %output signal
plot(t,xp);           %plotting output signal
title('Output Signal');
ylabel('Xp(t)/V');
grid on;
hold on;
xlabel('Time/s');
xlim([0, 1]);
ylim([-5, 5]);
```

The results are provided below:

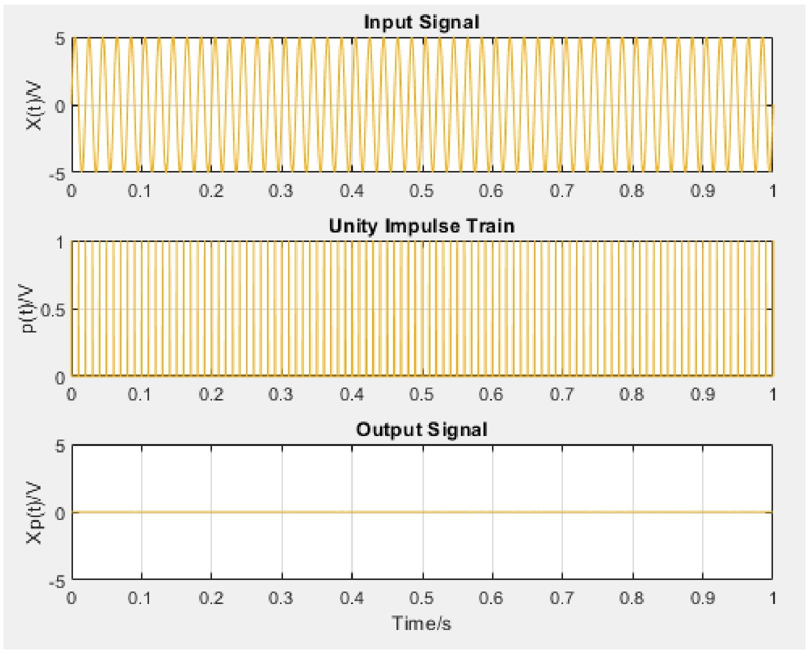


Figure 49: Results of Nyquist sampling

(c) Over Sampling

Over sampling can be achieved using the following Matlab script:

```
Fs=100*10^3; %sampling rate
fs=1000;       %sampling rate
t=0:1/Fs:0.1;
fx = 50;
w0 = 2*pi*fx;
x= 5*sin(w0*t); %Input signal

subplot(3,1,1);
plot(t,x);      %generating input signal plot
title('Input Signal');
ylabel('X(t)/V');
grid on;
hold on;

wp = 2*pi*fs;
sw=(1+square(wp*t, 0.1))/2; %sampling signal
subplot(3,1,2);
plot(t,sw);                %plotting sampling signal
title('Unity Impulse Train');
ylabel('p(t)/V');
grid on
hold on;

subplot(3,1,3);
xp=x.*sw;           %output signal
plot(t,xp);         %plotting output signal
title('Output Signal');
ylabel('Xp(t)/V');
grid on;
hold on;
xlabel('Time/s');

xlim([0, 0.1]);
ylim([-5, 5]);
```

The results are provided below:

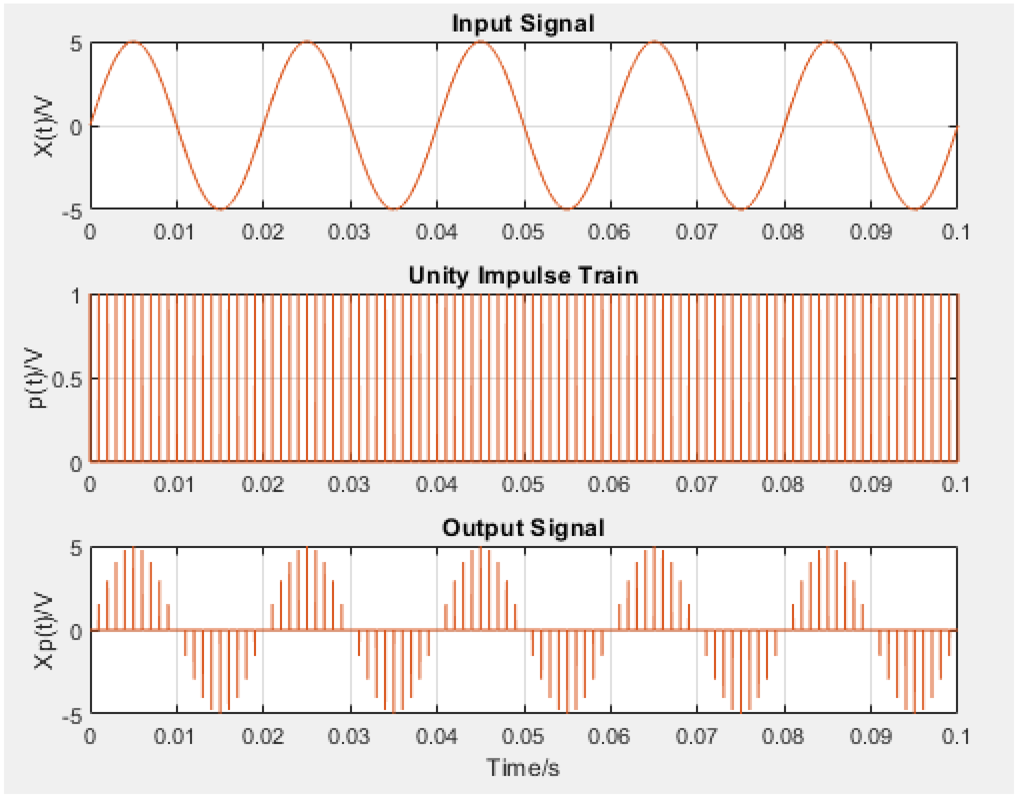


Figure 50: Results of Over sampling

<u>Question 2</u>

(a) Under Sampling

The results of under-sampling can be achieved using the following Matlab script:

```
Fs=100*10^3;
fp=48;  %sampling pulse frequency
t=0:1/Fs:0.5;   %time domain
w0 = 2*pi*50;   %freqency of x(t)
x= 5*sin(w0*t); %generating input signal
subplot(3,1,1);
plot(t,x,'b');  %plotting input signal
title('Input Signal');
ylabel('x(t)/V');
grid on;
hold on;
rec=[0.1, 50];
for train = 1: length(rec)
  for   rate = 1: length(fp)
```

```
    p = max(square (2*pi*fp(rate)*t,rec(train)),0); %generating
sampling pulse
    subplot(3,1,2);
    plot(t,p,'b');  %plotting sampling pulse
    title('Rectangular Pulse');
    ylabel('p(t)/V');
    grid on
    hold on;

    subplot(3,1,3);
    xp=x.*p;
    plot(t,xp,'b');
    title('Output Signal');
    ylabel('Xp(t)/V');
    grid on;
    hold on;
    xlabel('Time/s');

    xlim([0, 0.5]);
    ylim([-5, 5]);
  end
end
```
The results are provided below:

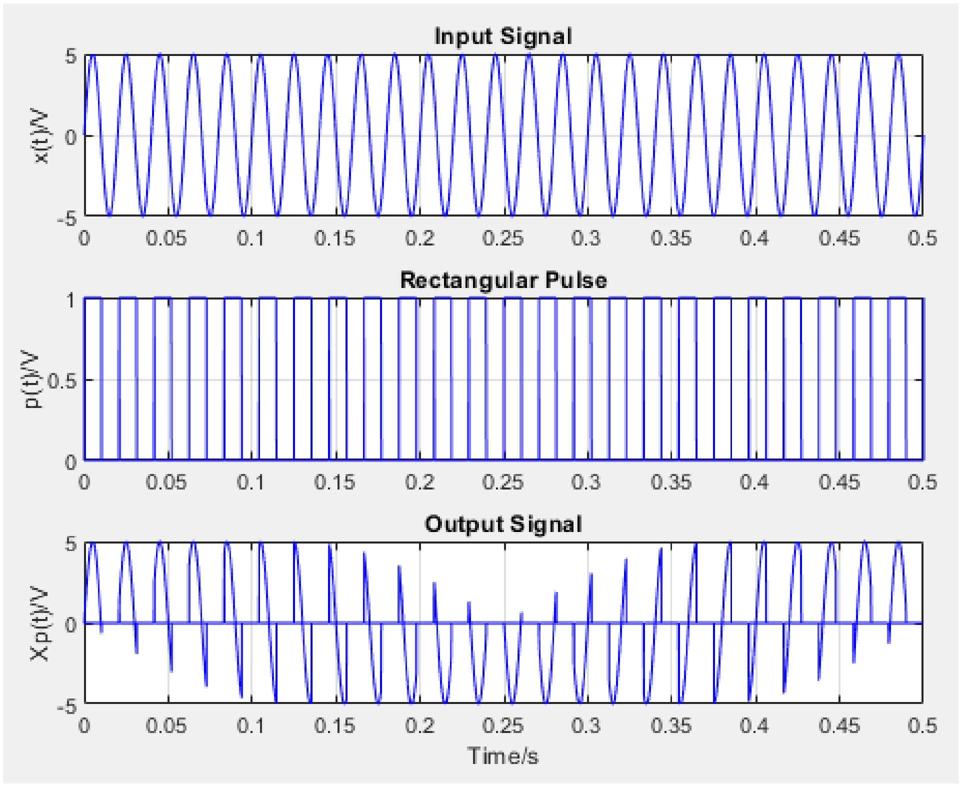


Figure 51: Results of under-sampling

(b) Nyquist Sampling

Nyquist sampling can be achieved using the following Matlab script:

```
Fs=100*10^3;
fp=100;  %sampling pulse frequency
t=0:1/Fs:0.1;   %time domain
w0 = 2*pi*50;   %freqency of x(t)
x= 5*sin(w0*t); %generating input signal
subplot(3,1,1);
plot(t,x,'b');  %plotting input signal
title('Input Signal'); ylabel('x(t)/V');
grid on;
hold on;
rec=[0.1, 50];
for train = 1: length(rec)
  for   rate = 1: length(fp)
    p = max(square(2*pi*fp(rate)*t,rec(train)),0); %generating
sampling pulse
    subplot(3,1,2);
    plot(t,p,'b');  %plotting sampling pulse
    title('Rectangular Pulse');
    ylabel('p(t)/V');
    grid on
    hold on;

    subplot(3,1,3);
    xp=x.*p;     %generating output signal
    plot(t,xp,'b'); %plotting output signal
    title('Output Signal');
    ylabel('Xp(t)/V');
    grid on;
    hold on;
    xlabel('Time/s');

    xlim([0, 0.1]);
    ylim([-5, 5]);
  end
end
```

The plot is provided below:

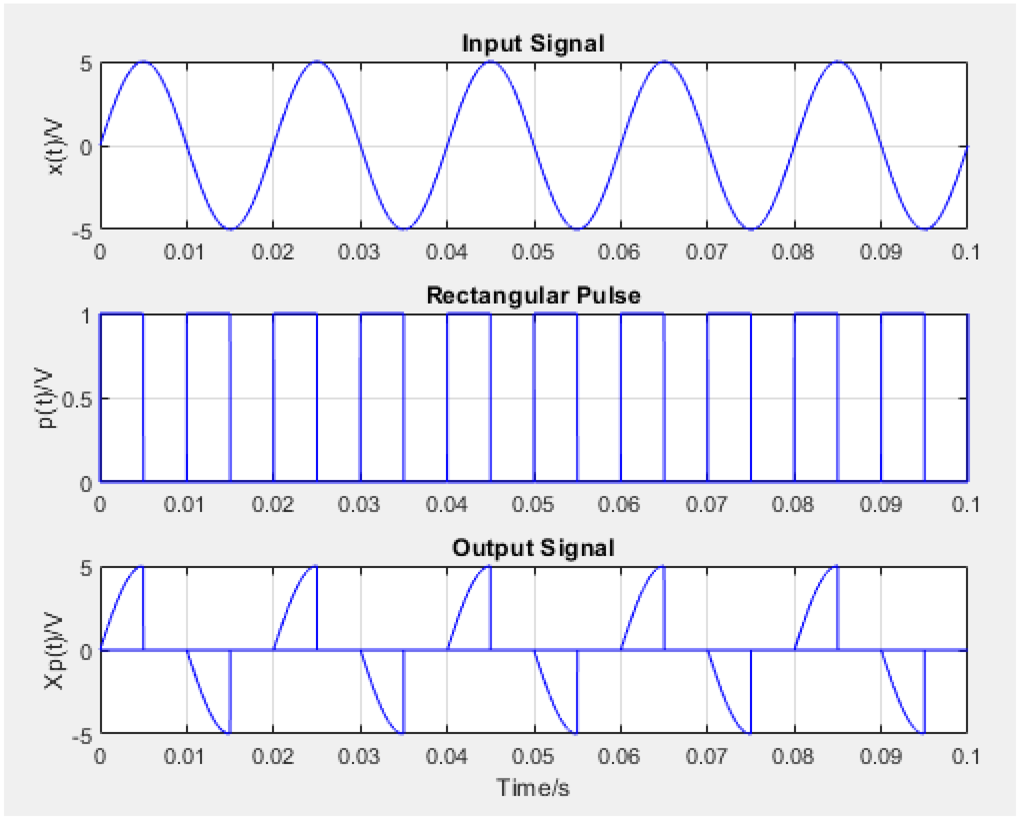


Figure 52: Results of Nyquist sampling

(c) Over Sampling

Over sampling can be achieved using the following Matlab script:

```
Fs=100*10^3;
fp=1000;  %sampling pulse frequency
t=0:1/Fs:0.1;   %time domain
w0 = 2*pi*50;   %freqency of x(t)
x= 5*sin(w0*t); %generating input signal
subplot(3,1,1);
plot(t,x,'b');  %plotting input signal
title('Input Signal');
ylabel('x(t)/V');
grid on;
hold on;
rec=[0.1, 50];
for train = 1: length(rec)
  for   rate = 1: length(fp)
    p = max(square(2*pi*fp(rate)*t,rec(train)),0); %generating
sampling pulse
    subplot(3,1,2);
    plot(t,p,'b');  %plotting sampling pulse
    title('Rectangular Pulse');
    ylabel('p(t)/V');
    grid on
    hold on;

    subplot(3,1,3);
    xp=x.*p;    %generating output signal
    plot(t,xp,'b'); %plotting output signal
    title('Output Signal');
    ylabel('Xp(t)/V');
    grid on;
    hold on;
    xlabel('Time/s');

    xlim([0, 0.1]);
    ylim([-5, 5]);
  end
end
```

The results are provided below:

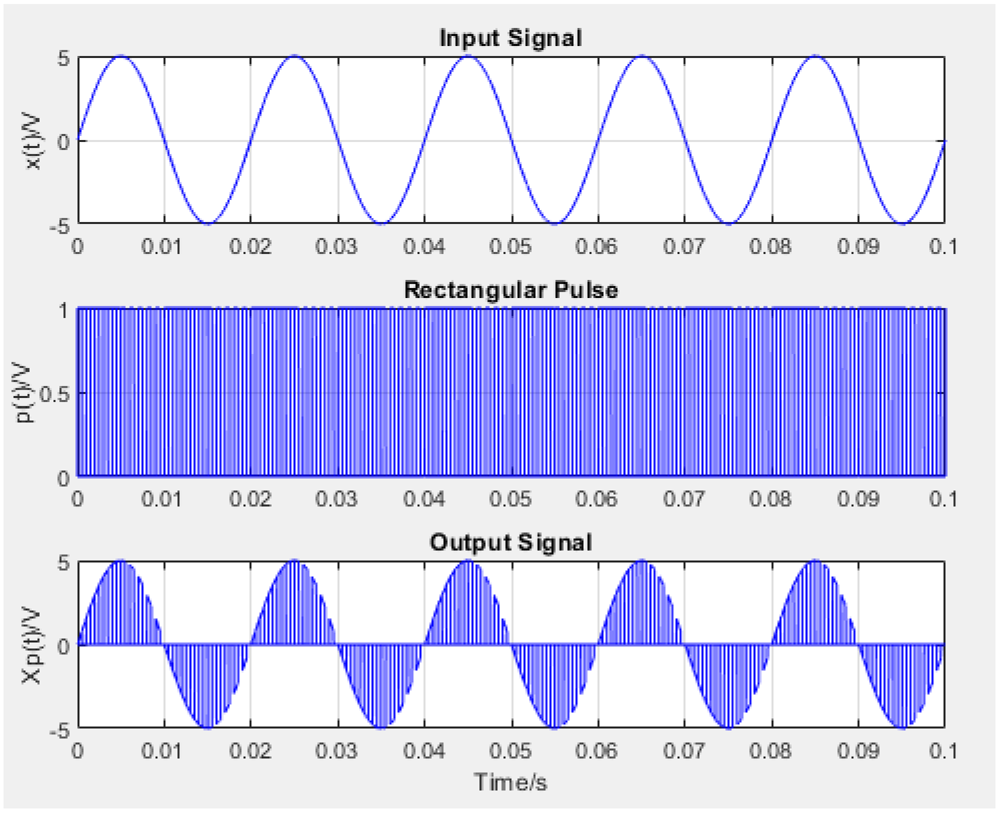


Figure 53: Results of Nyquist sampling

Problem 3: Sampling using a Sampling bridge

Question 1

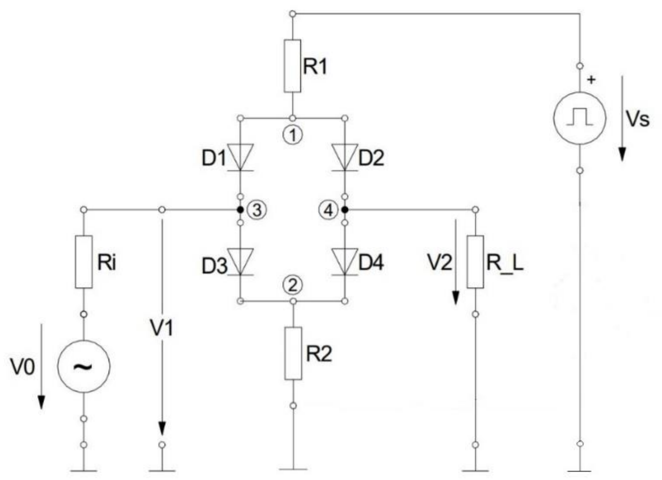


Figure 54: Modified Circuit

Question 2

One difference between the original circuit and the circuit shown above is the ground. In the original circuit, the signal changes around 0V. However, the modified circuit has an offset, which means the mathematically the resulting signal will be shifted by the offset amount. This is the result of removing the Vs-. Since we removed the negative sampling source, we will have only one sampling source to sample the signal.