Lab Report 2

Fourier Series and Fourier Transform

Performed by:

Aidyn Ardabek
a.ardabek@jacobs-university.de

Aya El Mai
a.elmai@jacobs-university.de

Author of the report:

Aidyn Ardabek
Mailbox Number: NA-527

Instructor:

Uwe Pagel
u.pagel@jacobs-university.de

Date of experiment: 13 October 2021
Date of submission: 17 October 2021

Contents

1 Introduction

In this experiment, Fourier series and Fourier transform of different signals will be studied. The main goal of this experiment is to get more detailed understanding about Fourier transform. During the Prelab and the experiment, practical implementation of the FFT (Fast Fourier Transform) will be described. Theoretical values will be calculated and plotted using Matlab to compare with experimental values.

2 Prelab Fourier Series and Fourier Transform

2.1 Problem 1: Decibels

1. Given $x(t) = 2cos(2\pi 1000t)$,

   (a) Since $cos(x)$ goes from -1 to 1, $V_{pp} = 4V$

   (b)

$$\begin{aligned}
V_{rms} =& \sqrt{\frac{1}{2\pi/(2\pi 1000)}\int_0^{\frac{1}{1000}}[2cos(2\pi 1000t)]^2 dt} \\
=& \sqrt{1000\int_0^{\frac{1}{1000}} 4\frac{1+cos(4\pi 1000t)}{2}dt} = 2000\left[t + \frac{sin(4\pi 1000t)}{4\pi 1000}\right]_0^{\frac{1}{1000}} \\
=& \sqrt{2000\left[\frac{1}{1000} + \frac{sin(4\pi)}{4\pi 1000} - 0 - \frac{sin(0)}{4\pi 1000}\right]} = \sqrt{2}V
\end{aligned}$$

   (c)

$$20\,log\frac{V_{rms}}{V_{ref}} = 20\,log\frac{\sqrt{2}}{1} = 3.01\,dBV_{rms}$$

2. (a)

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

$$= \sqrt{\frac{1}{T}\left[4\frac{T}{2} - 0 + 4T - 4\frac{T}{2}\right]} = 2V$$

   (b)

$$20\,log\frac{V_{rms}}{V_{ref}} = 20\,log\frac{2}{1} = 6.02\,dBV_{rms}$$

2.2 Problem 2: Determination of Fourier series coefficients

1. Note that $f(t)$ is an odd function. It means that the Fourier Series contains only sine terms.

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

$$b_k = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \frac{2}{T} t\, sin(\omega k t) = \frac{4}{T^2}\int_{-\frac{T}{2}}^{\frac{T}{2}} t\, sin(\omega k t) = \frac{4}{T^2}\Big[-\frac{t\,cos(\omega k t)}{\omega k} + \frac{sin(\omega k t)}{\omega^2 k^2}\Big]_{-\frac{T}{2}}^{\frac{T}{2}}$$

Also note that expression ωt when $t = T/2$ equals to π. Thus,

$$b_k = \frac{4}{T^2}\Big[-\frac{T/2\,cos(\pi k)}{2k\pi/T} + \frac{sin(\pi k)}{4\pi^2 k^2/T^2} + \frac{-T/2\,cos(-\pi k)}{2k\pi/T} - \frac{sin(-\pi k)}{4\pi^2 k^2/T^2}\Big] =$$

$$= \frac{4}{T^2}\Big[-\frac{T^2\,cos(\pi k)}{4k\pi} - \frac{T^2\,cos(-\pi k)}{4k\pi}\Big] = -2\frac{(-1)^k}{k\pi}$$

Therefore, Fourier Series coefficients up to the 5th harmonics are:

$$b_1 = \frac{2}{k\pi} \qquad b_2 = \frac{-2}{k\pi} \qquad b_3 = \frac{2}{k\pi} \qquad b_4 = \frac{-2}{k\pi} \qquad b_5 = \frac{2}{k\pi}$$

2. Figure 1 shows the function plotted using the calculated coefficients by Matlab and Figure 3 shows the code used to build this graph.

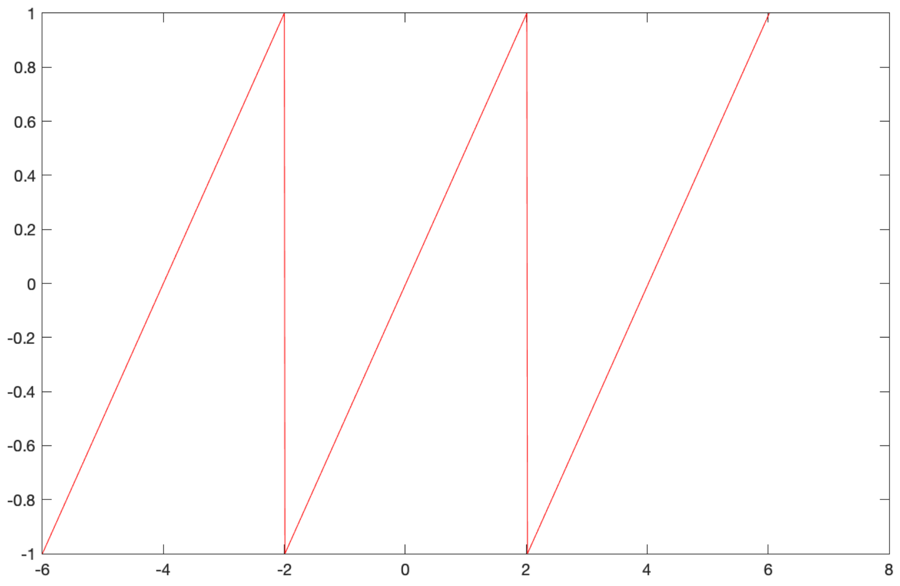


Figure 1: Original function

3. Figure 2 shows the original function plotted by using Matlab. Note that the red line is the graph of the original function and the green line is the graph plotted usinbg Fourier Series. Also, Figure 3 shows the code used to build this graph.

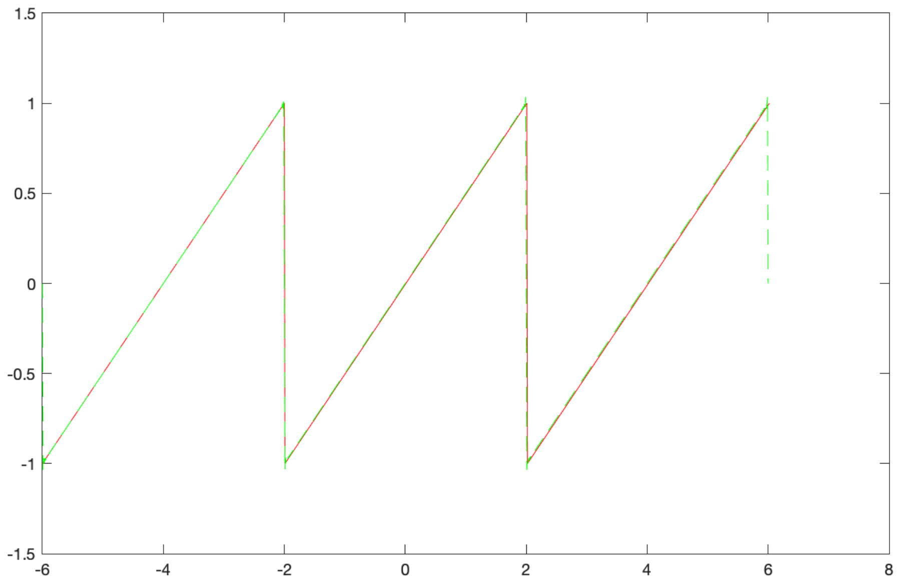
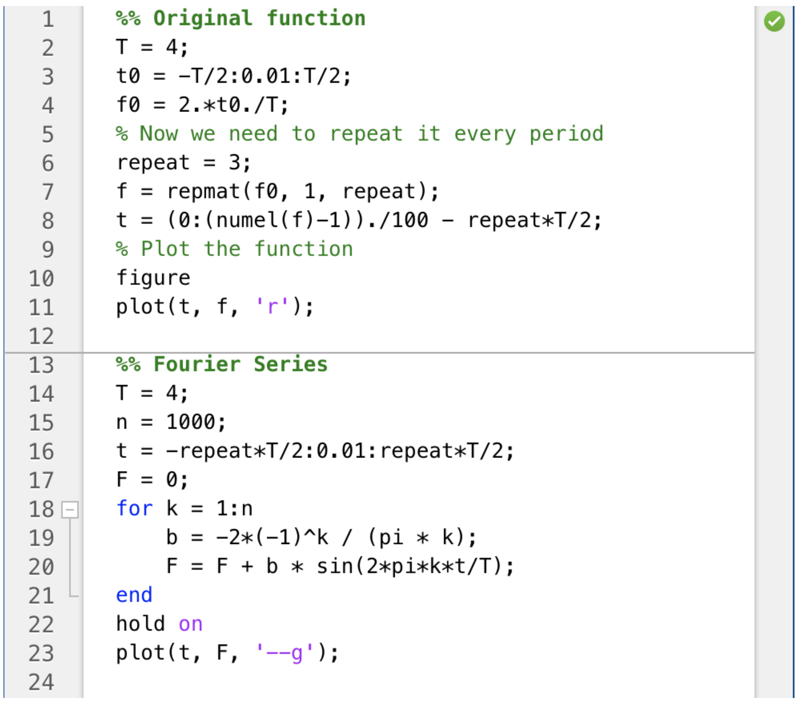


Figure 2: Using Fourier series

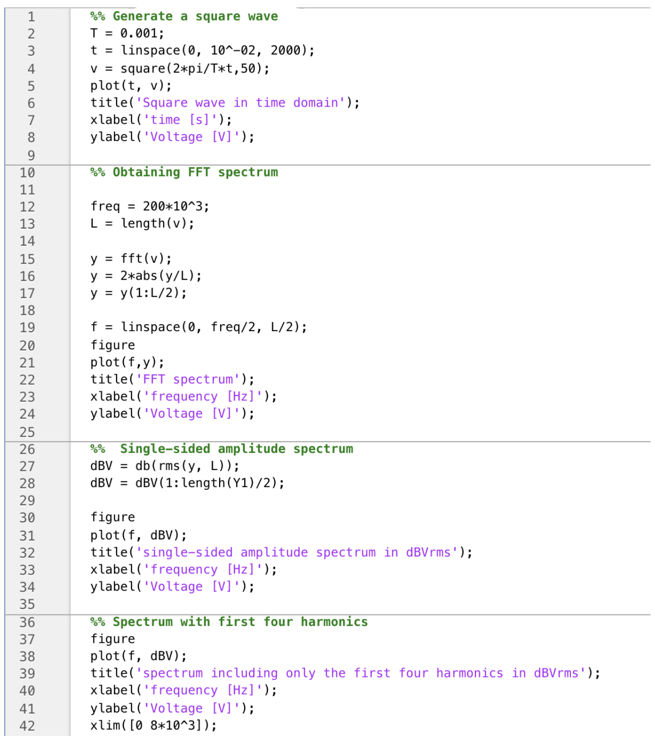


```
%% Original function
T = 4;
t0 = -T/2:0.01:T/2;
f0 = 2.*t0./T;
% Now we need to repeat it every period
repeat = 3;
f = repmat(f0, 1, repeat);
t = (0:(numel(f)-1))./100 - repeat*T/2;
% Plot the function
figure
plot(t, f, 'r');

%% Fourier Series
T = 4;
n = 1000;
t = -repeat*T/2:0.01:repeat*T/2;
F = 0;
for k = 1:n
    b = -2*(-1)^k / (pi * k);
    F = F + b * sin(2*pi*k*t/T);
end
hold on
plot(t, F, '--g');

```

Figure 3: Matlab code

2.3 Problem 3: FFT of a Square/Rectangular Wave

1. To generate a square wave of $1ms$ period, $2V_{pp}$, no offset, and duty cycle 50%, Matlab code shown in Figure 4 was used.



```
%% Generate a square wave
T = 0.001;
t = linspace(0, 10^-02, 2000);
v = square(2*pi/T*t,50);
plot(t, v);
title('Square wave in time domain');
xlabel('time [s]');
ylabel('Voltage [V]');

%% Obtaining FFT spectrum

freq = 200*10^3;
L = length(v);

y = fft(v);
y = 2*abs(y/L);
y = y(1:L/2);

f = linspace(0, freq/2, L/2);
figure
plot(f,y);
title('FFT spectrum');
xlabel('frequency [Hz]');
ylabel('Voltage [V]');

%%  Single-sided amplitude spectrum
dBV = db(rms(y, L));
dBV = dBV(1:length(Y1)/2);

figure
plot(f, dBV);
title('single-sided amplitude spectrum in dBVrms');
xlabel('frequency [Hz]');
ylabel('Voltage [V]');

%% Spectrum with first four harmonics
figure
plot(f, dBV);
title('spectrum including only the first four harmonics in dBVrms');
xlabel('frequency [Hz]');
ylabel('Voltage [V]');
xlim([0 8*10^3]);
```

Figure 4: Matlab code

2. The square wave in the time domain was plotted by using Matlab and it is represented in Figure 5. The Matlab code used to plot is shown in Figure 4.

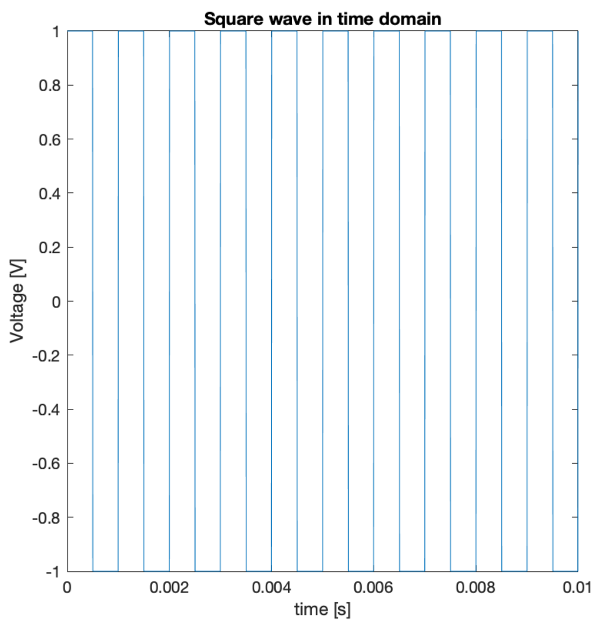


Figure 5: Square wave in time domain

3. The FFT spectrum was obtained using Matlab as shown in Figure 4. The result is shown in Figure 6

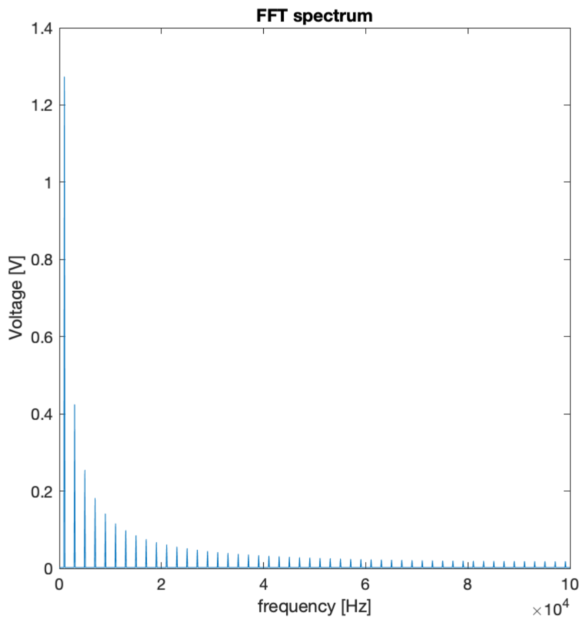


Figure 6: FFT spectrum

4. Then, the single-sided amplitude spectrum in dBV_{rms} is plotted (Figure 7) by using the code shown in Figure 4.

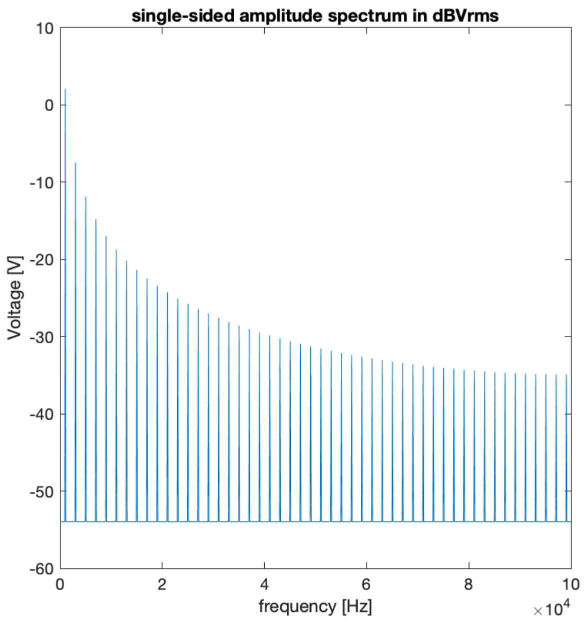


Figure 7: Single-sided amplitude spectrum in dBV_{rms}

5. Finally, in Matlab 'xlim' command was used to display the first four harmonics in dBV_{rms} Figure 4 shows the code and Figure 8 shows the graph.

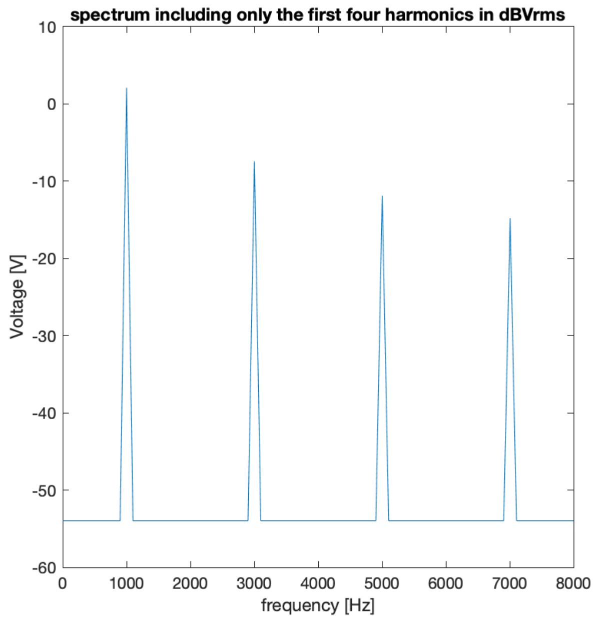


Figure 8: Spectrum including only the first four harmonics in dBV_{rms}

6. The same process was repeated twice for 20% and 33% duty cycles. The obtained graphs are shown in Figure 9 and Figure 10 for 20% and 33% respectively.

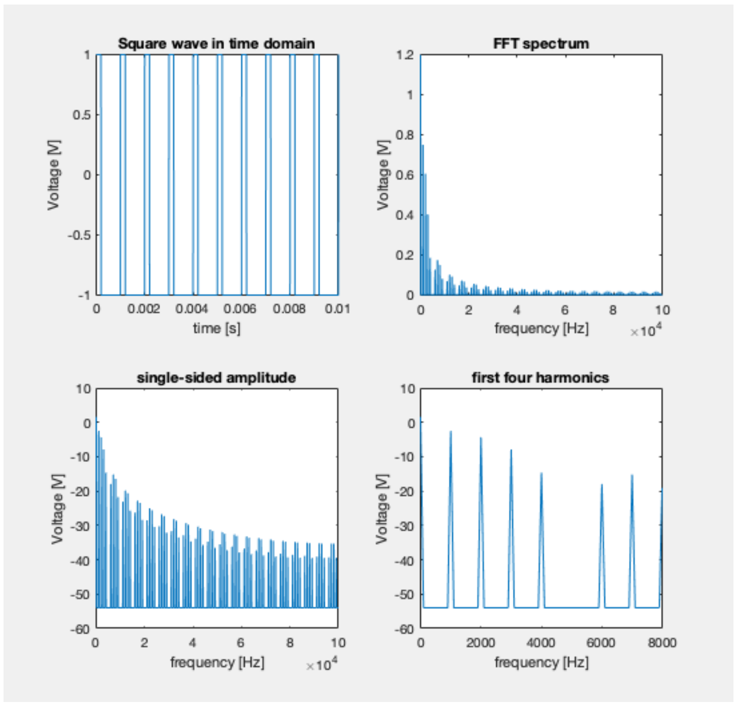


Figure 9: All graphs with 20% duty cycle

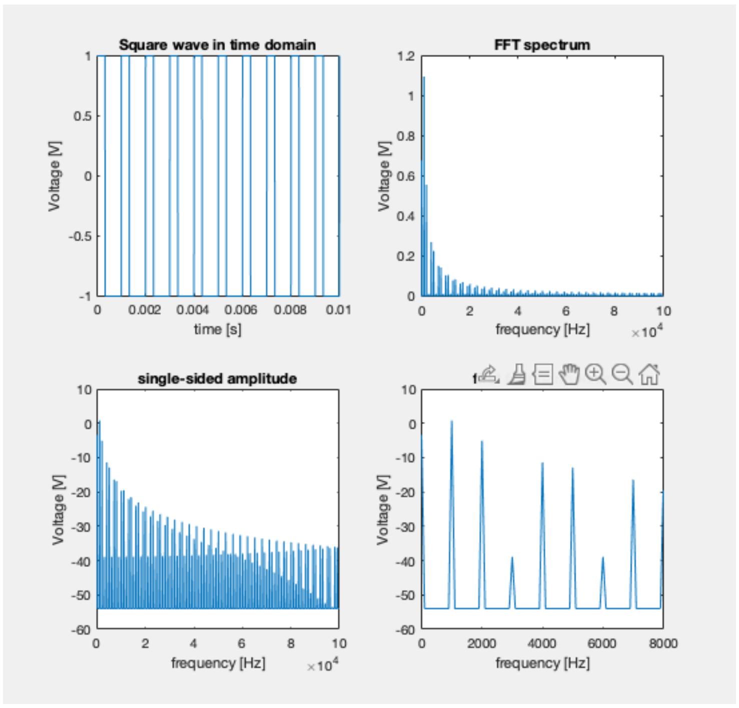


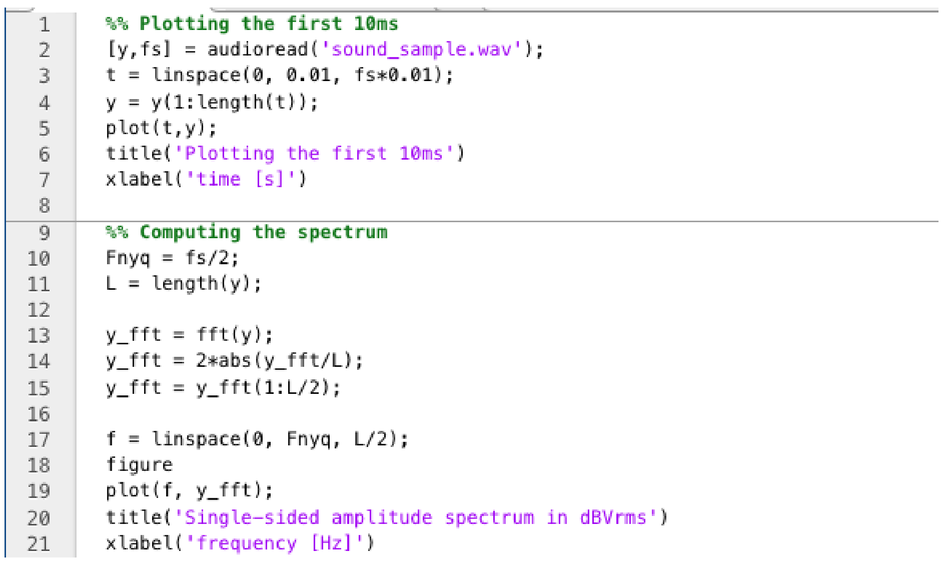
Figure 10: All graphs with 33% duty cycle

7.
$$c_k = \frac{2}{k\omega_0 T} sin(k\omega_0 T_1) = \frac{1}{k\pi} sin(k\omega_0 T_1) \tag{1}$$
where T is the period of the signal and T_1 is the width of the periodic pulses.

Note from the plots of different duty cycles that the magnitude spectrum differs for each duty cycle. When the duty cycle is smaller, the power of the signal is less spread out, and is compressed over a smaller period of time. Thus, the amplitude of the spectrum is higher and more varying, meaning it has larger variations over values. Also, when the duty cycle cecreases, the magnitude spectrum reaches zero for different multiples of ω_0. For duty cycles of 50%, 33%, and 20%, the magnitude reaches zero for frequencies $\omega = 2n\omega_0$, $\omega = 3n\omega_0$, and $\omega = 5n\omega_0$ respectively.

2.4 Problem 4: FFT of a sound sample

1. To plot the sound, Matlab with the code shown in Figure 11 was used. The result is shown in Figure 12.



```matlab
%% Plotting the first 10ms
[y,fs] = audioread('sound_sample.wav');
t = linspace(0, 0.01, fs*0.01);
y = y(1:length(t));
plot(t,y);
title('Plotting the first 10ms')
xlabel('time [s]')

%% Computing the spectrum
Fnyq = fs/2;
L = length(y);

y_fft = fft(y);
y_fft = 2*abs(y_fft/L);
y_fft = y_fft(1:L/2);

f = linspace(0, Fnyq, L/2);
figure
plot(f, y_fft);
title('Single-sided amplitude spectrum in dBVrms')
xlabel('frequency [Hz]')
```

Figure 11: Matlab code

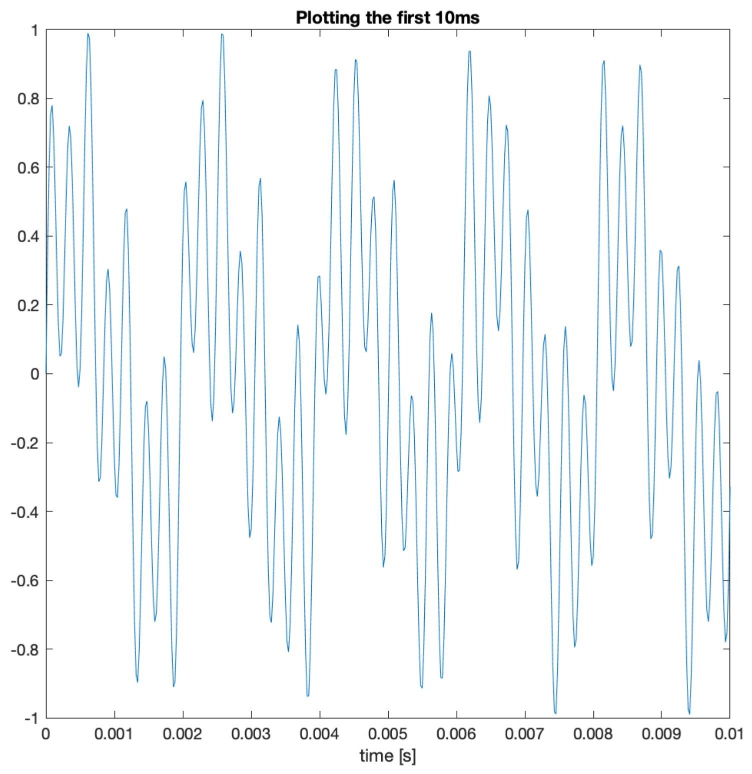


Figure 12: Sound wave

2. Then, Matlab FFT function was used to compute the spectrum as shown in Figure 11. From obtained numbers, the graph represented in Figure 13.

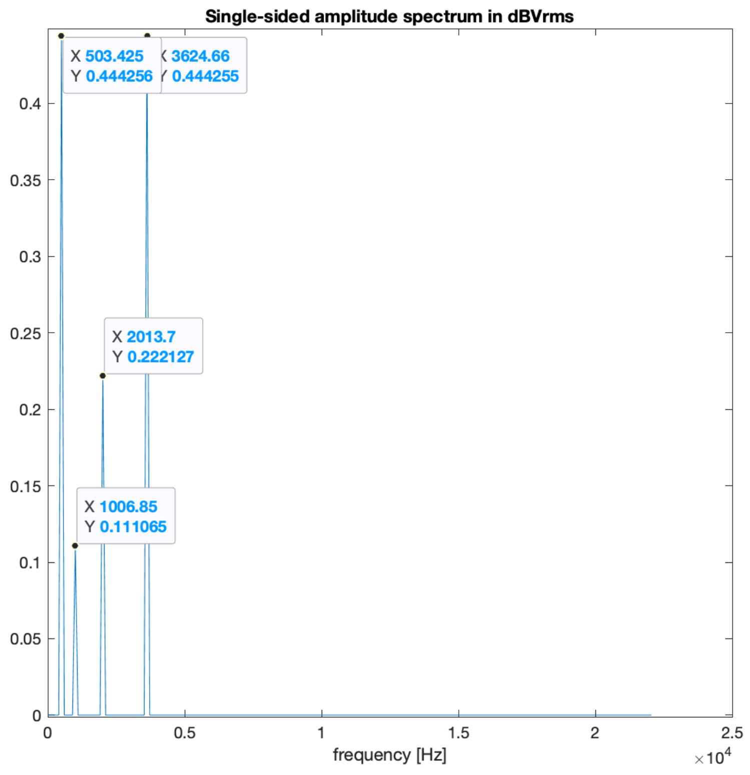


Figure 13: Amplitude spectrum

3. The tones forming this signal are $503Hz$, $1006Hz$, $2013Hz$, and $3624Hz$.

3 Execution Fourier Series and Fourier Transform

3.1 Problem 1: FFT of Single Tone sinusoidal wave

Firstly, the function generator was used to generate a sinusoidal wave with $500Hz$ frequency, $2V_{pp}$ amplitude, and no offset. Figure 15 shows the wave in time domain. Then, the built-in oscilloscope FFT function was used to obtain the FFT spectrum. Figure 16 shows the complete spectra and zoomed spectra of the sinusoidal wave.
To obtain $0dB$ spectrum peak, the function generator was set to sinusoidal wave with $2KHz$ frequency and zero offset. Peak-to-peak voltage was calculated as:

$$20\, log\frac{V_{rms}}{1} = 0 \quad \Rightarrow \quad \frac{V_{rms}}{1} = 1 \quad \Rightarrow \quad V_{rms} = 1 = \frac{V_{pp}}{2\sqrt{2}} \quad \Rightarrow \quad V_{pp} = 2\sqrt{2} \approx 2.828V$$

Hard copies from the oscilloscope in time (Figure 17a) and frequency domain (Figure 17b) were taken.

3.2 Problem 2: FFT of square wave

Firstly, the function generator was used to generate a square wave with $1ms$ period, $2V_{pp}$ amplitude, and no offset. Figure 18 shows the wave in time domain. Then, the built-in oscilloscope FFT function was used to obtain the FFT spectrum. Figure 19 shows the complete spectra of the square wave.
Using the FFT zoom control in oscilloscope, amplitudes and frequencies of the fundamental and the first four harmonics are determined. Figure 20 and Figure 21 show the obtained images from the oscilloscope.
The same process was repeated with 20% duty cycle. Hardcopues of the signal in time and frequency domains are shown in Figure 22.

3.3 Problem 3: FFT of Multiple-Tone Sinusoidal wave

In this part of the experiment, the auxiliary signal generator and agilent signal generator were combined using the circuit shown in Figure 14 and auxiliary signal generator was set to sinusoidal wave with $2V_{pp}$ amplitude, $10KHz$ frequency, and zero offset. Figure 25 shows the hardcopis of the signal in time domain and of the FFT spectrum.

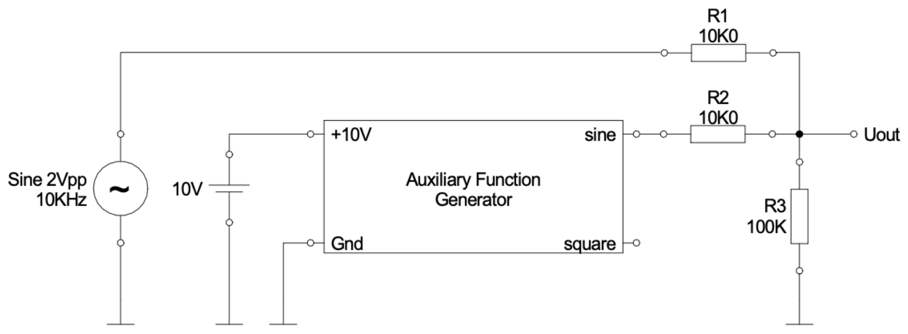


Figure 14: Circuit with two signal generators

3.4 Results

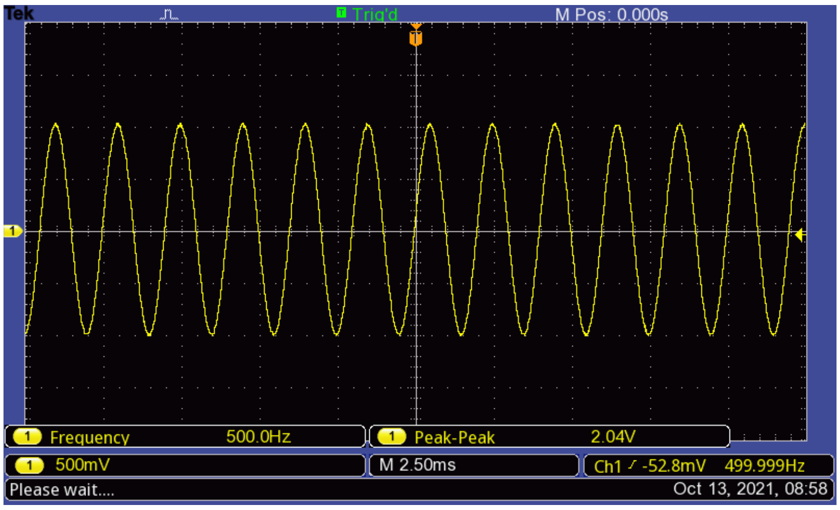
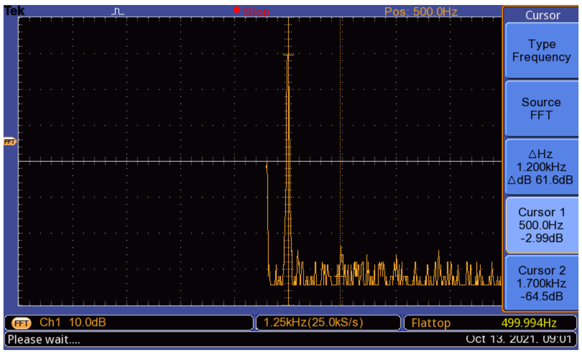
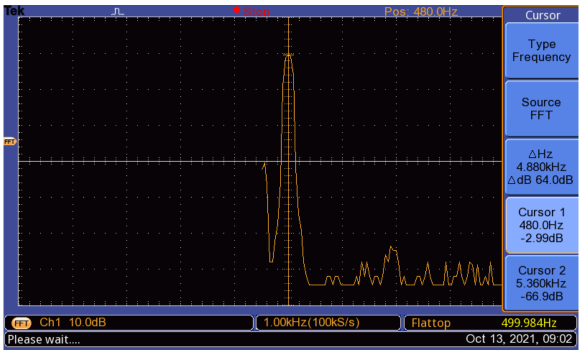


Figure 15: Sinusoidal wave in time domain

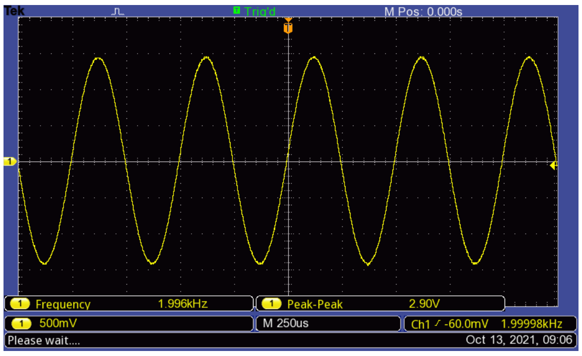


(a) Complete spectra

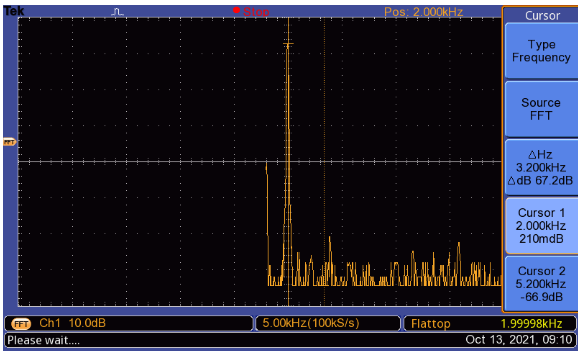


(b) Zoomed spectra

Figure 16: Spectra of sinusoidal wave

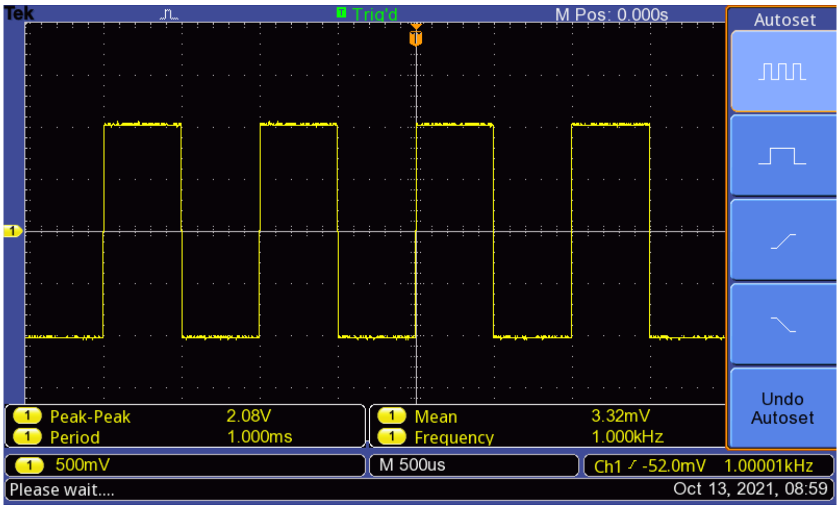


(a) in time domain



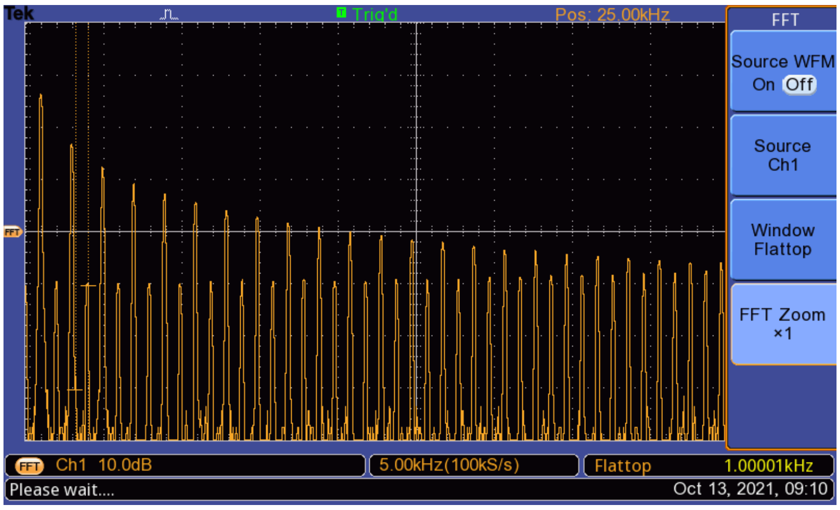
(b) in frequency domain

Figure 17: Sinusoidal wave with 0dB



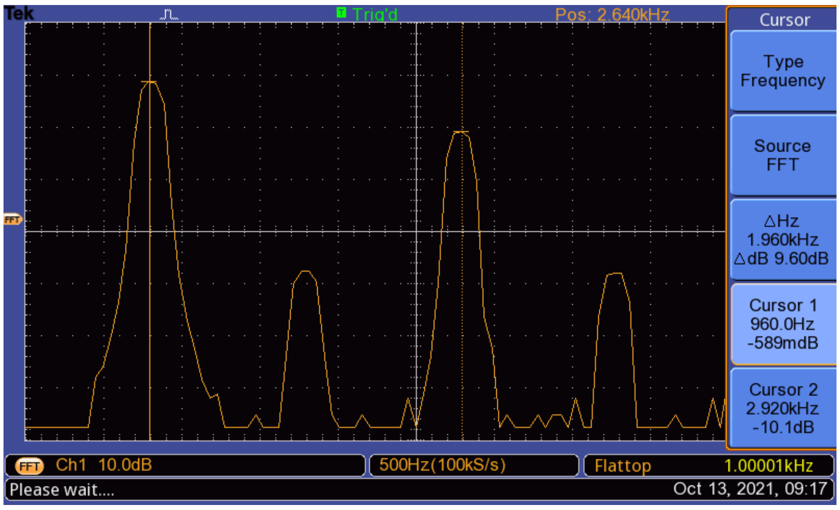
Peak-Peak 2.08V
Period 1.000ms
Mean 3.32mV
Frequency 1.000kHz
500mV
M 500us
Ch1 ⁄ -52.0mV 1.00001kHz
Please wait....
Oct 13, 2021, 08:59

Figure 18: Square wave in time domain



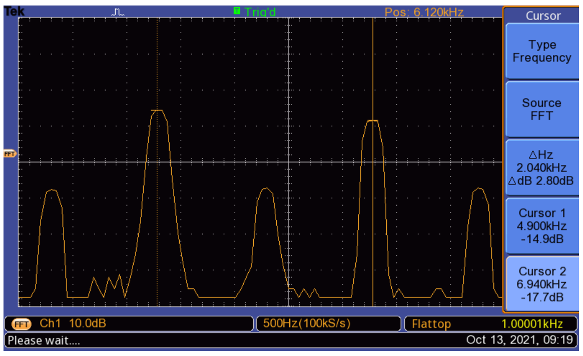
Source WFM On [✓ Off]
Source Ch1
Window Flattop
FFT Zoom ×1
Ch1 10.0dB
5.00kHz(100kS/s)
Flattop 1.00001kHz
Please wait....
Oct 13, 2021, 09:10

Figure 19: Complete spectra of Square wave



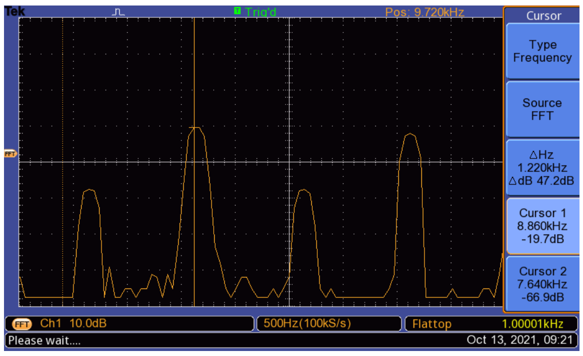
Type Frequency
Source FFT
ΔHz 1.960kHz
ΔdB 9.60dB
Cursor 1 960.0Hz -589mdB
Cursor 2 2.920kHz -10.1dB
Ch1 10.0dB
500Hz(100kS/s)
Flattop 1.00001kHz
Please wait....
Oct 13, 2021, 09:17

Figure 20: Amplitudes and frequencies of the fundamental and the first harmonic for square wave



Pos: 6.120kHz
Type Frequency
Source FFT
ΔHz 2.040kHz ΔdB 2.80dB
Cursor 1 4.900kHz -14.9dB
Cursor 2 6.940kHz -17.7dB
FFT Ch1 10.0dB
500Hz(100kS/s)
Flattop 1.00001kHz
Please wait....
Oct 13, 2021, 09:19

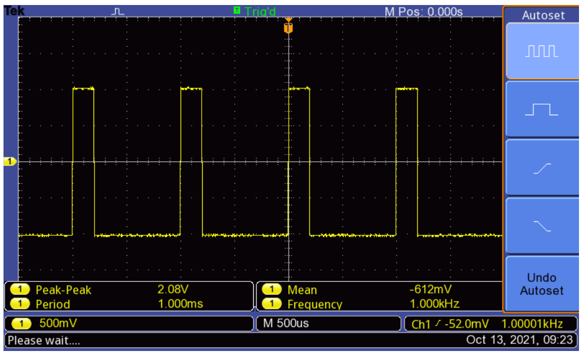
(a) of the second and third harmonics



Pos: 9.720kHz
Type Frequency
Source FFT
ΔHz 1.220kHz ΔdB 47.2dB
Cursor 1 8.860kHz -19.7dB
Cursor 2 7.640kHz -66.9dB
FFT Ch1 10.0dB
500Hz(100kS/s)
Flattop 1.00001kHz
Please wait....
Oct 13, 2021, 09:21

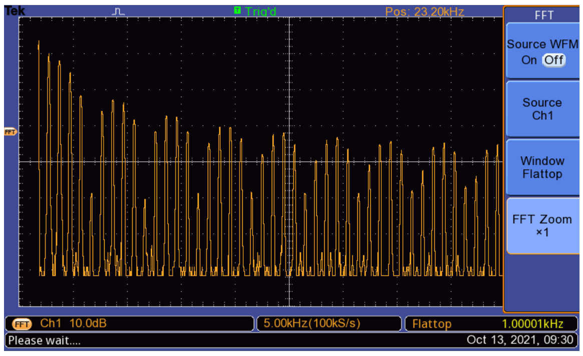
(b) of the fourth harmonic

Figure 21: Amplitudes and frequencies for square wave



M Pos: 0.000s
Peak-Peak 2.08V
Period 1.000ms
Mean -612mV
Frequency 1.000kHz
1 500mV
M 500us
Ch1 ⟋ -52.0mV 1.00001kHz
Please wait....
Oct 13, 2021, 09:23

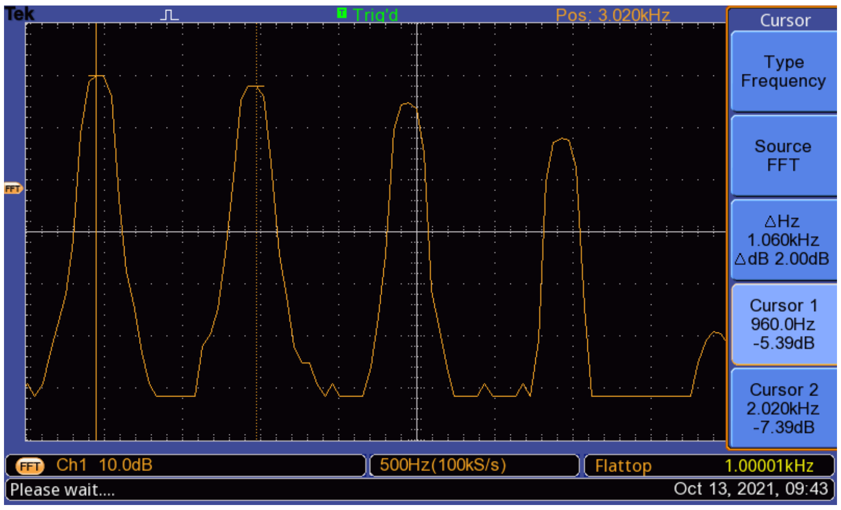
(a) in time domain



Pos: 23.20kHz
Source WFM On Off
Source Ch1
Window Flattop
FFT Zoom ×1
FFT Ch1 10.0dB
5.00kHz(100kS/s)
Flattop 1.00001kHz
Please wait....
Oct 13, 2021, 09:30

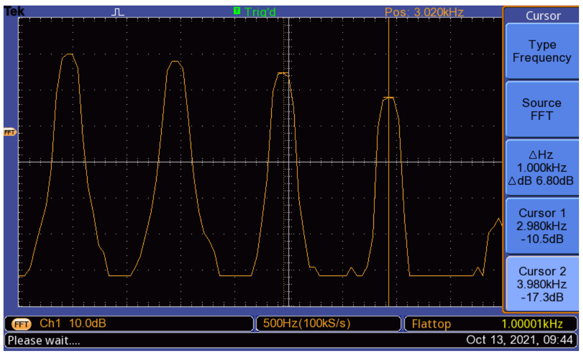
(b) in frequency domain

Figure 22: Square wave with 20% duty cycle

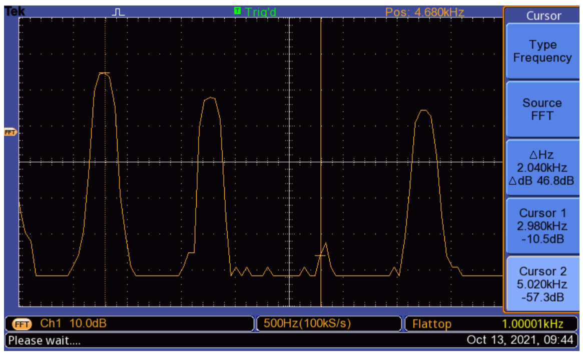


Pos: 3.020kHz
Type Frequency
Source FFT
ΔHz 1.060kHz ΔdB 2.00dB
Cursor 1 960.0Hz -5.39dB
Cursor 2 2.020kHz -7.39dB
FFT Ch1 10.0dB
500Hz(100kS/s)
Flattop 1.00001kHz
Please wait....
Oct 13, 2021, 09:43

Figure 23: Amplitudes and frequencies of the fundamental and the first harmonic for square wave with 20% duty cycle

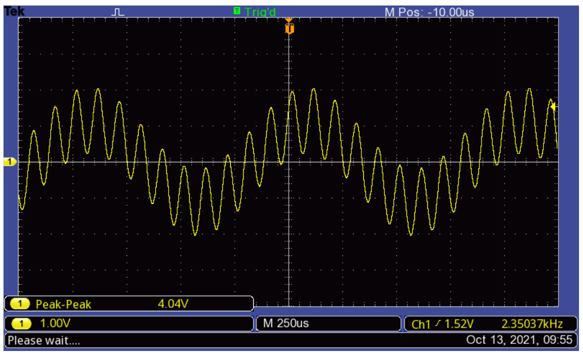


(a) of the second and third harmonics

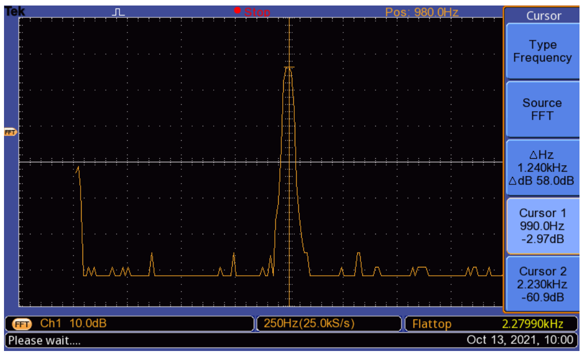


(b) of the fourth harmonic

Figure 24: Amplitudes and frequencies for square wave with 20% duty cycle



(a) in time domain



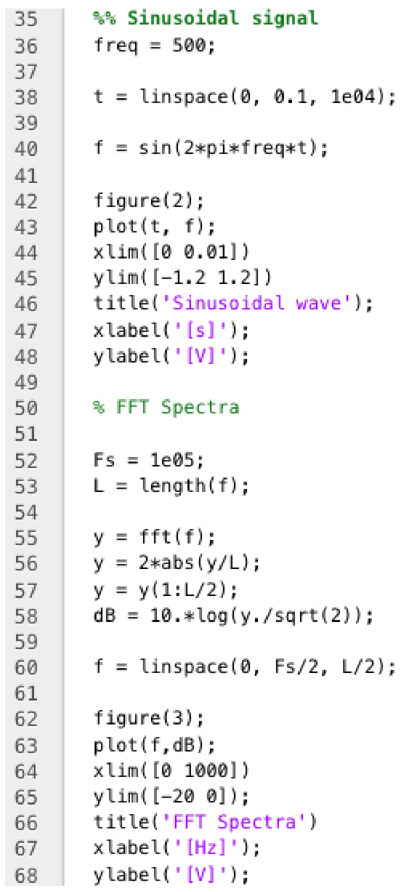
(b) in frequency domain

Figure 25: Combination of two signal generators

4 Evaluation

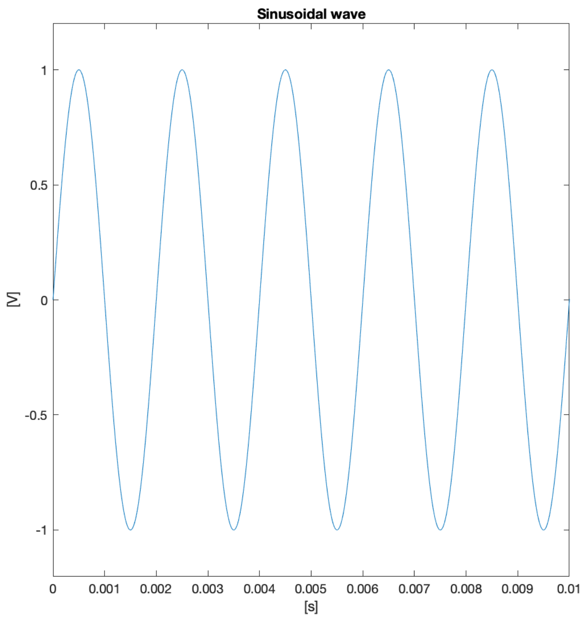
4.1 Problem 1: FFT of Single Tone sinusoidal wave

1. Reference value of the oscilloscope for $0dB$ equals $1V_{rms}$.
2. To calculate the expected FFT spectra, Matlab with the code shown in Figure 29a was used. The signal generator produced the sinusoidal signal as expected, shown in Figure 29b. Also, spectra peak can be observed around $480 - 500Hz$ from Figure 15. Note that at spectra peak, it shows -2.99dB, which is very close to -3dB. It is very close approximation considering the fact that both oscilloscope and generator have some instrumental errors.

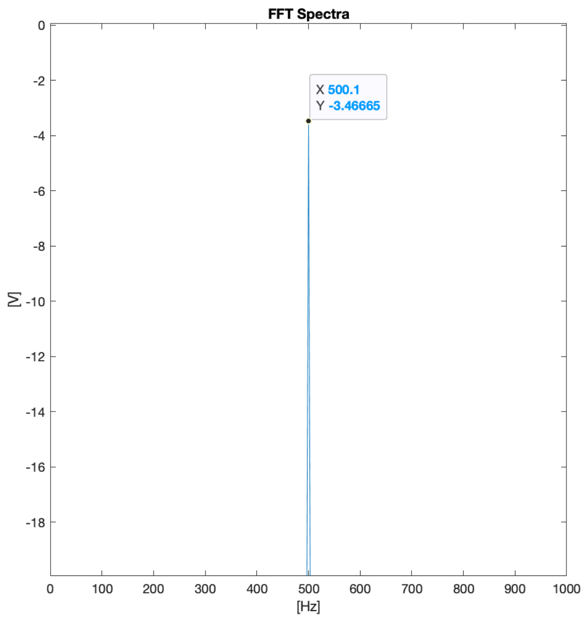


```
%% Sinusoidal signal
freq = 500;

t = linspace(0, 0.1, 1e04);

f = sin(2*pi*freq*t);

figure(2);
plot(t, f);
xlim([0 0.01])
ylim([-1.2 1.2])
title('Sinusoidal wave');
xlabel('[s]');
ylabel('[V]');

% FFT Spectra

Fs = 1e05;
L = length(f);

y = fft(f);
y = 2*abs(y/L);
y = y(1:L/2);
dB = 10.*log(y./sqrt(2));

f = linspace(0, Fs/2, L/2);

figure(3);
plot(f,dB);
xlim([0 1000])
ylim([-20 0]);
title('FFT Spectra')
xlabel('[Hz]');
ylabel('[V]');
```

Figure 26: Matlab code



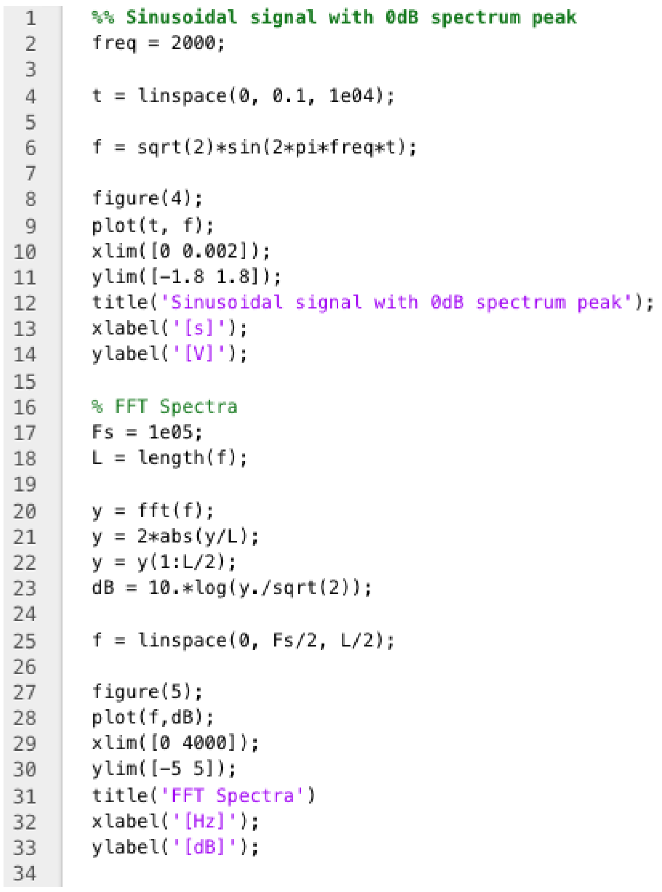
(a) Signal in time domain



(b) FFT spectra

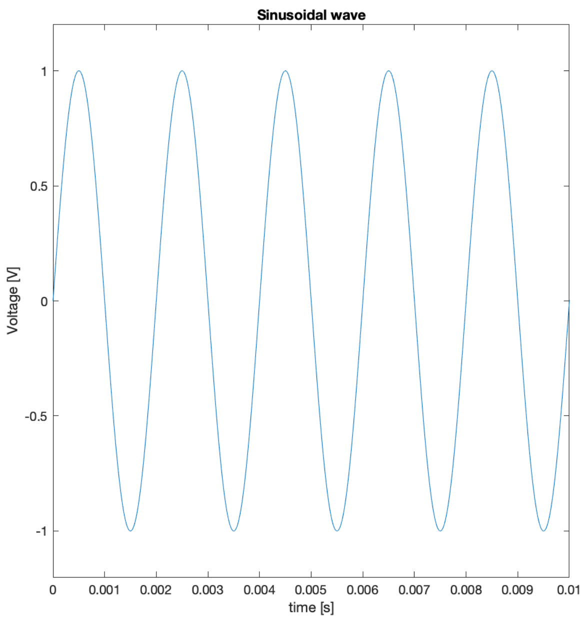
Figure 27: Expected graphs of sinusoidal signal

3. To calculate the expected FFT spectra, Matlab with the code shown in Figure 27a was used. The signal generator produced the sinusoidal signal as expected, shown in Figure 27b. Note that the cursor shows that at spectra peak we have 210mdB, which is very close to 0dB. Whereas, the Matlab shows -0.007dB. It is very close approximation considering the fact that both oscilloscope and generator have some instrumental errors. Also, spectra peak can be observed around 2000Hz from Figure 17a and at 2000Hz from Figure 26.

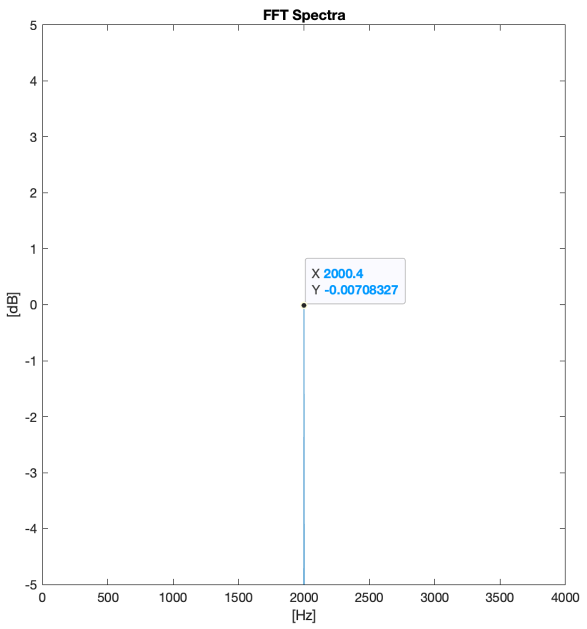


```matlab
%% Sinusoidal signal with 0dB spectrum peak
freq = 2000;

t = linspace(0, 0.1, 1e04);

f = sqrt(2)*sin(2*pi*freq*t);

figure(4);
plot(t, f);
xlim([0 0.002]);
ylim([-1.8 1.8]);
title('Sinusoidal signal with 0dB spectrum peak');
xlabel('[s]');
ylabel('[V]');

% FFT Spectra
Fs = 1e05;
L = length(f);

y = fft(f);
y = 2*abs(y/L);
y = y(1:L/2);
dB = 10.*log(y./sqrt(2));

f = linspace(0, Fs/2, L/2);

figure(5);
plot(f,dB);
xlim([0 4000]);
ylim([-5 5]);
title('FFT Spectra')
xlabel('[Hz]');
ylabel('[dB]');

```

Figure 28: Matlab code



(a) Signal in time domain



(b) FFT spectra

Figure 29: Expected graphs of sinusoidal signal with 0dB

4. Overall, the obtained values and expected values are very close to each other. It means that our experimental values were confirmed with expected values. However, the small difference could be the result of instrumental errors of signal generator and oscilloscope. Nevertheless, the error is very small.

4.2 Problem 2: FFT of square wave

1. This setting is a scale factor. If the setting is $1ms$, each horizontal division represents $1ms$ and the total screen width represents $10ms$, or ten divisions. Changing the sec/div setting enables you to look at longer and shorter time intervals of the input signal. Increasing the time-base increases the frequency axis, meaning that the Nyquist frequency will increase too. Also, the distorted frequency parts will go back to their frequencies. However, if the time-base is decreased, images of the waves can be clearer. Also, some information, such as high frequency, can be lost. It means that obtained values would not be so accurate.

2. The duty cycle with specific percentage means only that percentage of the wave is represented. For example, comparing Figure 18 and Figure 22a, it is clearly seen that with 20% of duty cycle, the squares are narrower than in the original wave. This is due to the fact that only 20% of the wave is in active state.

4.3 Problem 3

In this part of the experiment, two signal generators were used with two different frequencies. Since the Fourier Transform is linear, the resulting output from two signal generators should give the sum of two sinusoidal signals with $2KHz$ frequency and $10KHz$ frequency. From Figure 25a, it can be seen that one division is 2.35KHz. The small sinusoidal signal has a period in an almost one division. It means that the signal has frequency of around 1.2KHz, which is true since our Auxiliary Function Generator has frequency of 1KHz. Also, the big sinusoidal signal has a period over almost 4 divisions. It means that the signal should have frequency of around 11.2KHz, which is also true since our Agilent signal generator has frequency of 10Khz.

5 Conclusion

In this experiment, expected FFT spectra were compared with experimentally obtained ones. The Matlab was used to build theoretical graphs. Generally, all values are convinced. Three different types of signal were described: simple sinusoidal wave, square wave, and combined sinusoidal signal. For all cases, FFT spectra was calculated. However, there were some small errors, which is mainly due to the instrumental errors. For example, signal generators can not generate exactly the desired output.

6 Prelab Sampling

6.1 Problem 1: The Sampling Theorem

1. Usually, analog signals are passed through a low-pass filter before sampling to avoid the aliasing phenomenon. Since frequencies, that are higher than our message signal maximum frequency, are not needed, the low-pass filter weakens these frequency components.

2. The sampling theoren states that a real signal, which is band-limited to fHz can be reconstructed without error from samples taken uniformly at a rate $r > 2f$ samples per second. This minimum sampling frequency, $f_s = 2fHz$, is called the Nyquist rate or the Nyquist frequency.
$$2 \times 3KHz = 6KHz$$

3. The Nyquist frequency, also called the Nyquist limit, is the highest frequency that can be coded at a given sampling rate in order to be able to fully reconstruct the signal.

4. If a sampling frequency of $5KHz$ samples the following input frequencies, the resulting frequency would be:

5. Frequencies of signal that alias to a $f_a = 7Hz$ signal can be found using the following expression:
$$f = |n \times f_s + f_a|$$
where sampling frequency is $f_s = 30Hz$ and n is an integer number. Thus, it can be easily determined that $37Hz$, $67Hz$, and $97Hz$ frequencies can alias to a $7Hz$ signal.

Input frequency	Resulting frequency
$500Hz$	$500Hz$
$2.5KHz$	$0Hz$ or $2.5KHz$
$5KHz$	$0Hz$
$5.5KHz$	$500Hz$

Table 1: Resulting frequency after sampling

6.2 Problem 2: Impulse Train Sampling and Real Sampling

1. To plot under sampling, Nyquist sampling, and over sampling cases, the Matlab code shown in Figure 31 was used. The result is shown in Figure 30

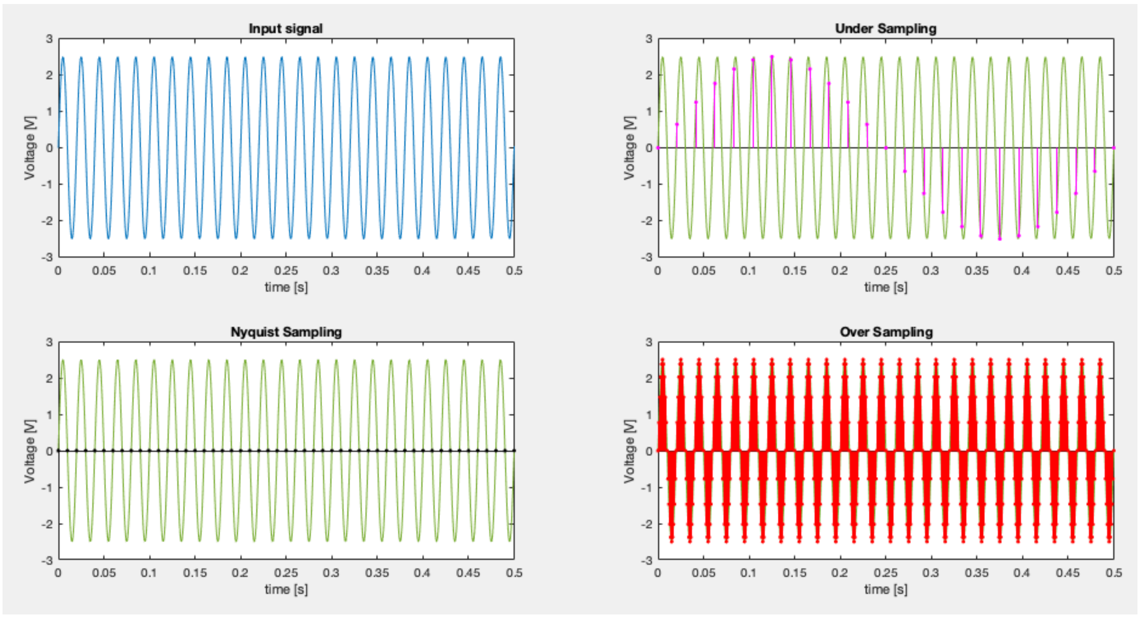
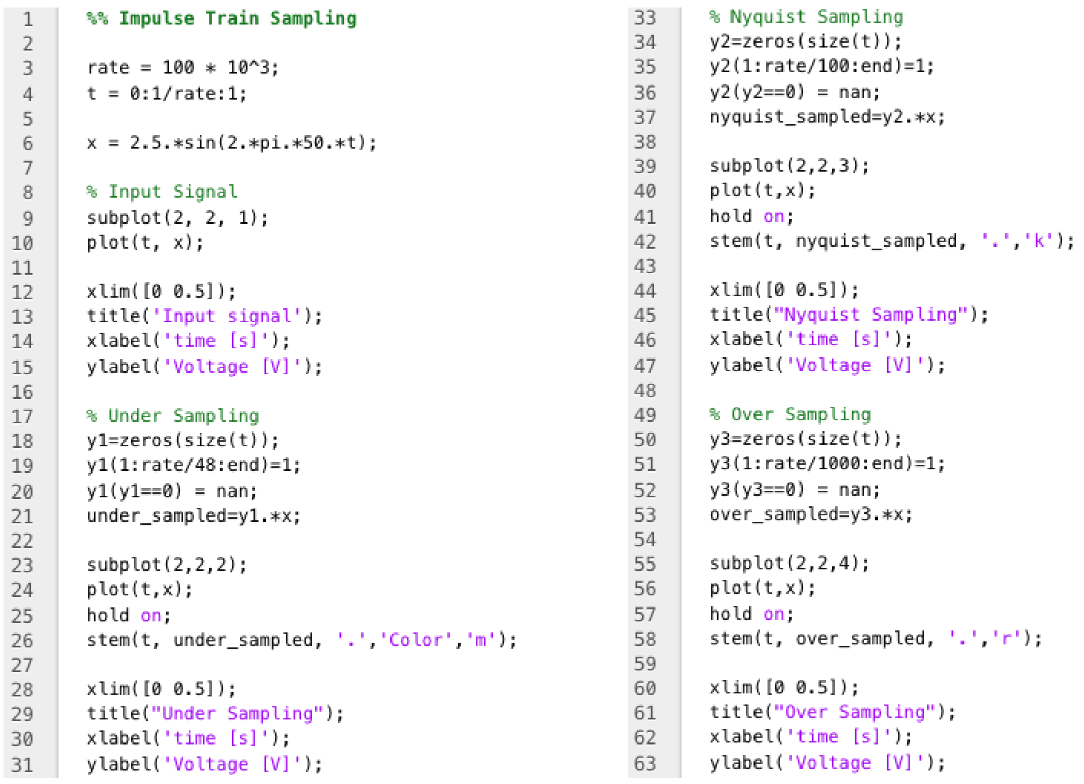


Figure 30: Simulations of all 3 cases



```matlab
%% Impulse Train Sampling

rate = 100 * 10^3;
t = 0:1/rate:1;

x = 2.5.*sin(2.*pi.*50.*t);

% Input Signal
subplot(2, 2, 1);
plot(t, x);

xlim([0 0.5]);
title('Input signal');
xlabel('time [s]');
ylabel('Voltage [V]');

% Under Sampling
y1=zeros(size(t));
y1(1:rate/48:end)=1;
y1(y1==0) = nan;
under_sampled=y1.*x;

subplot(2,2,2);
plot(t,x);
hold on;
stem(t, under_sampled, '.','Color','m');

xlim([0 0.5]);
title("Under Sampling");
xlabel('time [s]');
ylabel('Voltage [V]');
```

```matlab
% Nyquist Sampling
y2=zeros(size(t));
y2(1:rate/100:end)=1;
y2(y2==0) = nan;
nyquist_sampled=y2.*x;

subplot(2,2,3);
plot(t,x);
hold on;
stem(t, nyquist_sampled, '.','k');

xlim([0 0.5]);
title("Nyquist Sampling");
xlabel('time [s]');
ylabel('Voltage [V]');

% Over Sampling
y3=zeros(size(t));
y3(1:rate/1000:end)=1;
y3(y3==0) = nan;
over_sampled=y3.*x;

subplot(2,2,4);
plot(t,x);
hold on;
stem(t, over_sampled, '.','r');

xlim([0 0.5]);
title("Over Sampling");
xlabel('time [s]');
ylabel('Voltage [V]');
```

Figure 31: Matlab code

2. To plot under sampling, Nyquist sampling, and over sampling cases, the Matlab code shown in Figure 33 was used. The result is shown in Figure 32

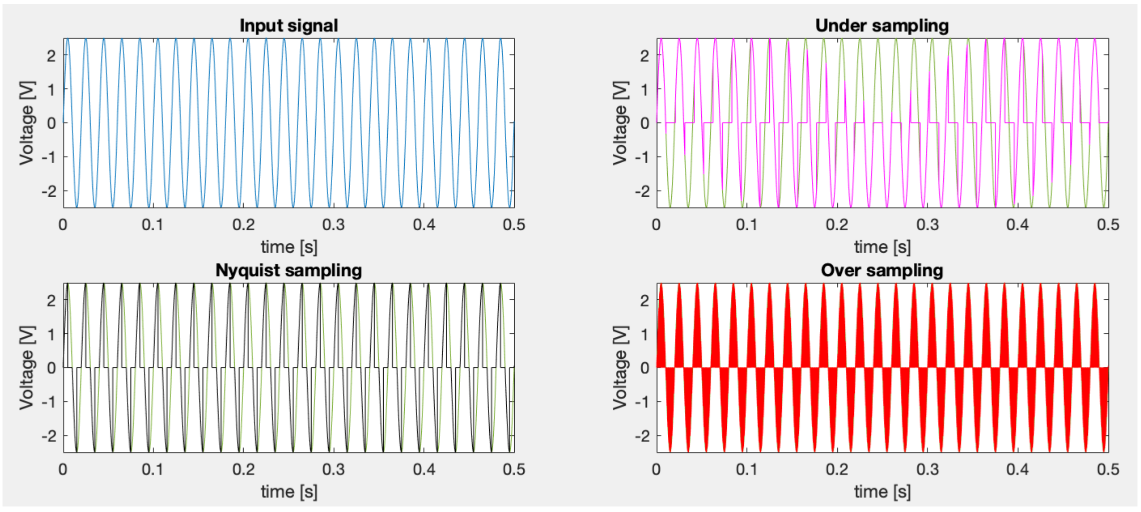
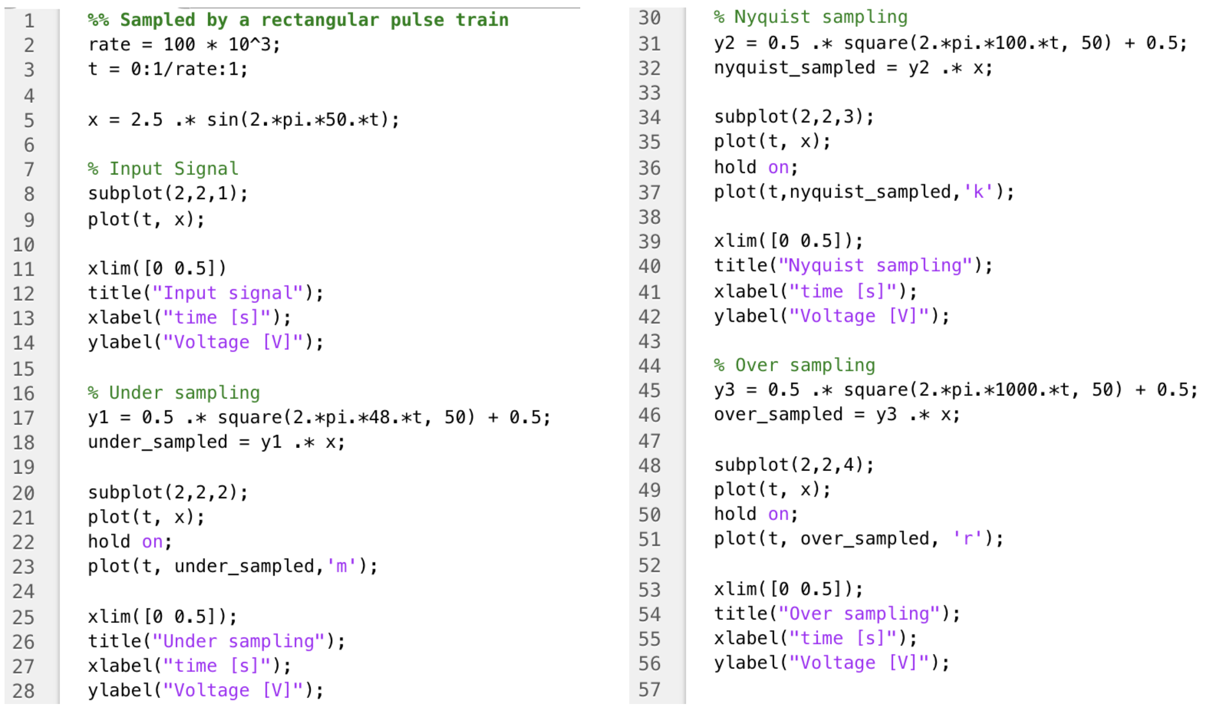


Figure 32: Simulations of all 3 cases



```
%% Sampled by a rectangular pulse train
rate = 100 * 10^3;
t = 0:1/rate:1;

x = 2.5 .* sin(2.*pi.*50.*t);

% Input Signal
subplot(2,2,1);
plot(t, x);

xlim([0 0.5])
title("Input signal");
xlabel("time [s]");
ylabel("Voltage [V]");

% Under sampling
y1 = 0.5 .* square(2.*pi.*48.*t, 50) + 0.5;
under_sampled = y1 .* x;

subplot(2,2,2);
plot(t, x);
hold on;
plot(t, under_sampled,'m');

xlim([0 0.5]);
title("Under sampling");
xlabel("time [s]");
ylabel("Voltage [V]");

% Nyquist sampling
y2 = 0.5 .* square(2.*pi.*100.*t, 50) + 0.5;
nyquist_sampled = y2 .* x;

subplot(2,2,3);
plot(t, x);
hold on;
plot(t,nyquist_sampled,'k');

xlim([0 0.5]);
title("Nyquist sampling");
xlabel("time [s]");
ylabel("Voltage [V]");

% Over sampling
y3 = 0.5 .* square(2.*pi.*1000.*t, 50) + 0.5;
over_sampled = y3 .* x;

subplot(2,2,4);
plot(t, x);
hold on;
plot(t, over_sampled, 'r');

xlim([0 0.5]);
title("Over sampling");
xlabel("time [s]");
ylabel("Voltage [V]");

```

Figure 33: Matlab code

6.3 Problem 3: Sampling using a Sampling bridge

The circuit should have only one sampling source to sample the input. This can be done by removing the negative V_s. This results to one sampling signal provided by positive supply.

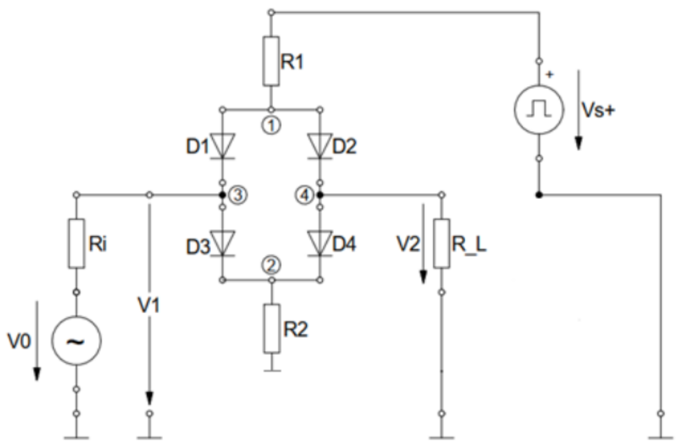


Figure 34: Modified circuit