Digital Signal Processing Lab

Laboratory report submitted for the partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Mohit Akhouri - 19ucc023

Course Coordinator
Dr. Divyang Rawal

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

September 2021

Contents

Chapter *2*

Experiment - 2

2.1 Aim of the Experiment

- Quantization and Encoding
- Simulink based Quantization

2.2 Software Used

- MATLAB
- Simulink

2.3 Theory

2.3.1 About Quantization :

Quantization, in mathematics and digital signal processing, is the process of **mapping** input values from a large set (often a continuous set) to output values in a **countable smaller set**, often with a finite number of elements. **Rounding** and **truncation** are typical examples of quantization processes.Quantization is involved to some degree in nearly all digital signal processing, as the process of representing a signal in digital form ordinarily involves rounding. Quantization also forms the core of essentially all **lossy compression** algorithms. There are different types of Quantizer which are as follows :

- Analog-to-digital converter
- Rate–distortion optimization
- Mid-riser and mid-tread uniform quantizers
- Dead-zone quantizer

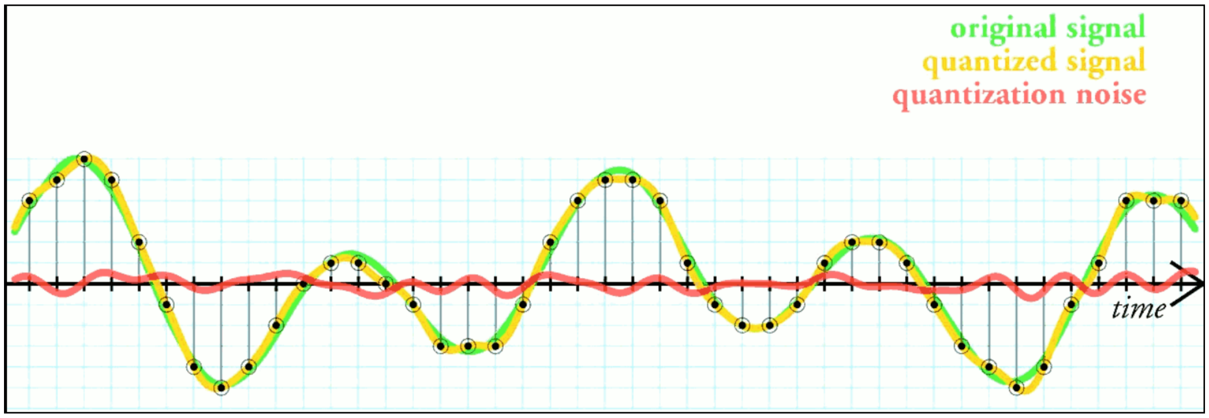


Figure 2.1 Original signal, Quantized signal and Quantization Noise

Quantized value can be calculated by first calculating the **index** value (i) which can be done as follows :

$$i = round(\frac{x - x_{min}}{\Delta}) \tag{2.1}$$

Now the **Quantization level** (x_q) of a particular sample can be calculated as:

$$x_q = x_{min} + i\Delta \tag{2.2}$$

$$\Delta = \frac{x_{max} - x_{min}}{L} \tag{2.3}$$

In the above equations , **x** is the original signal , x_{max} is the maximum value of the original signal, x_{min} is the minimum value of the original signal, Δ is the **step size** and **L** is the **number of levels** provided to the quantizer.

2.3.1.1 About Encoding :

The process of representing quantized values digitally is called **encoding**. The quantized value is coded into binary form. Typically, each binary digit is assigned an output line. At read-out time, these lines carry a 0 or 1. The number of digits used to represent each quantized value can be given as:

$$b >= log(L) \tag{2.4}$$

where **b** is the number of digits used to represent the quantized value and **L** is the number of levels.

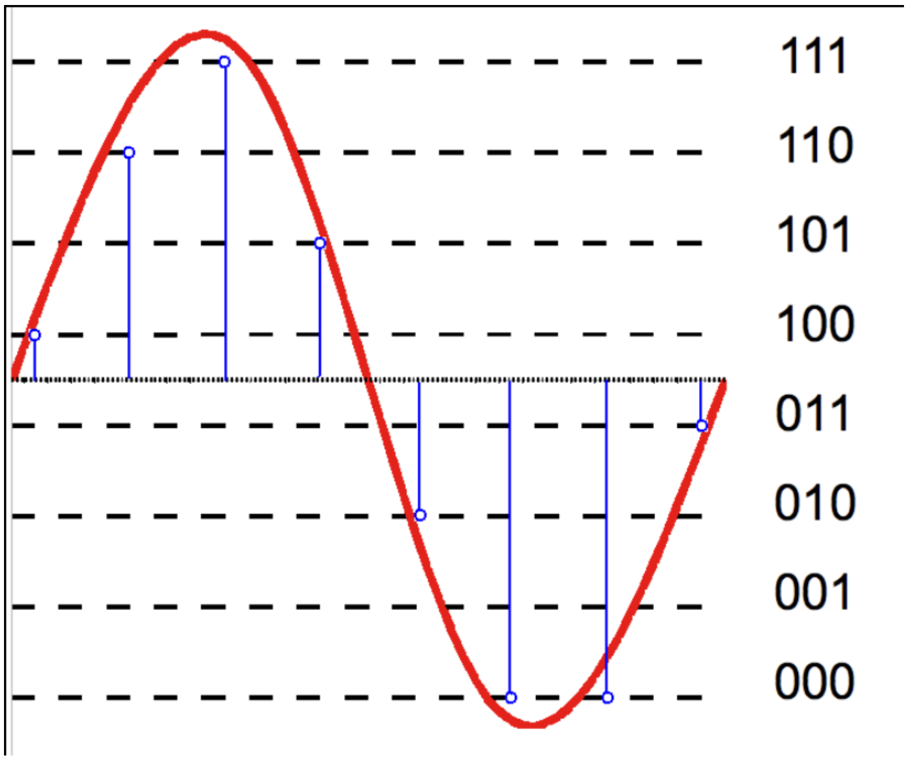


Figure 2.2 Quantized and Encoded signal for L=8

2.3.2 About Quantization Noise Error :

When an Analog-Digital Converter (ADC) converts a continuous signal into a discrete digital representation, there is a range of input values that produces the same output. That range is called quantum (Q) and is equivalent to the Least Significant Bit (LSB). The difference between input and output is called the quantization error. Therefore, the quantization error can be between $\pm\frac{Q}{2}$.

2.3.3 About Signal to Quantization Noise ratio (SQNR) :

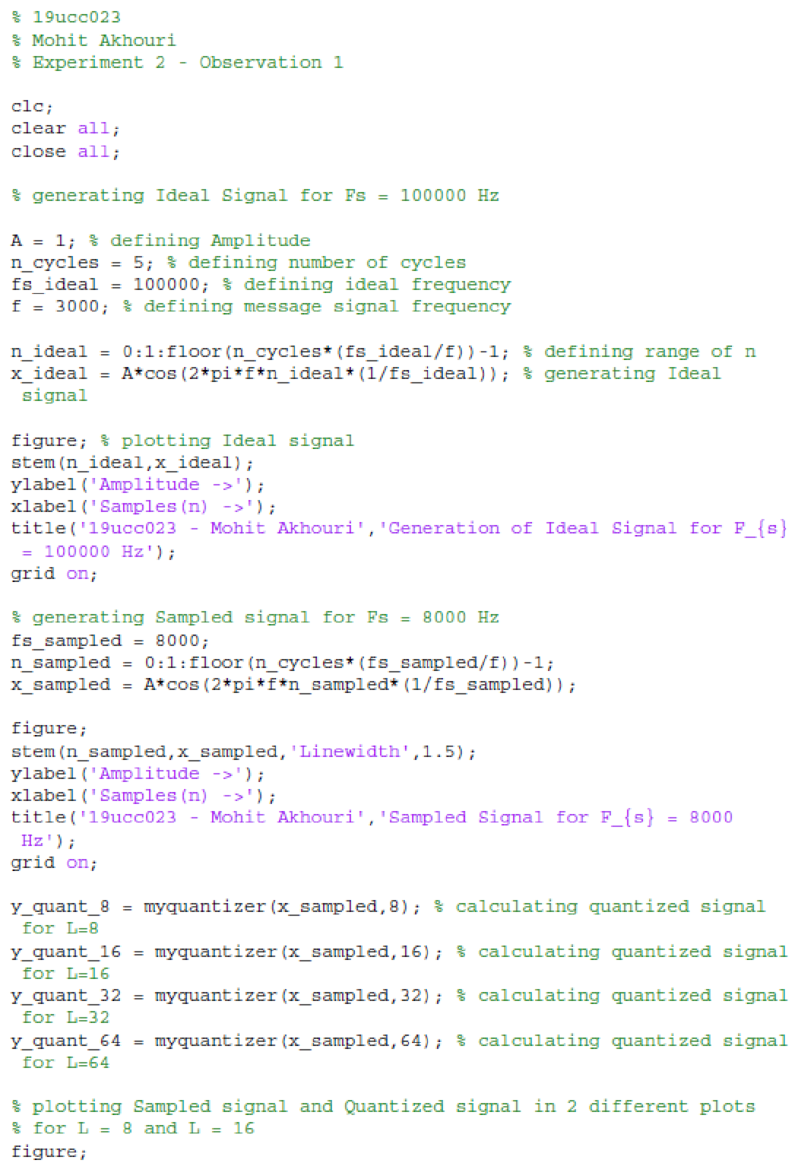
SQNR, short for signal to quantization noise ratio, is a measure of the quality of the quantization, or digital conversion of an analog signal. Defined as **normalized signal power** divided by **normalized quantization noise power**. The SQNR in dB is approximately equal to **6 times** the number of bits of the analog-to-digital converter (ADC). For example, the maximum SQNR for 16 bits is approximately 96dB. The Formula for calculating SQNR is given as :

$$SQNR = \frac{3}{2}(2^{2b}) = 1.76 + 6.02(b) \tag{2.5}$$

In the above equation , SQNR is in decibel (**dB**) and **b** represents the number of bits used to represent the Quantized signal samples. Relationship between number of levels(L) ans number of bits(b) is given as $L = 2^b$.

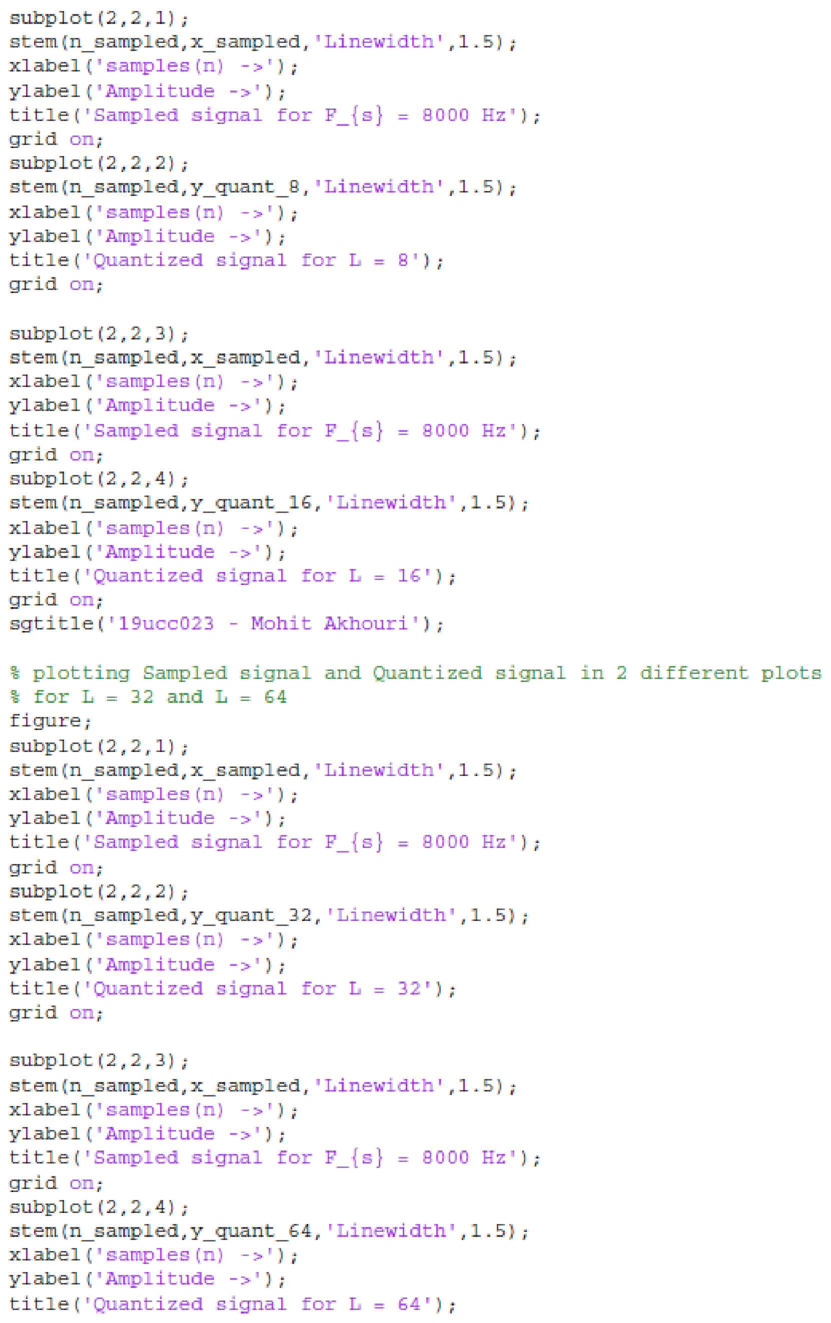
2.4 Code and results

2.4.1 Plot 5 cycles of sampled and quantized signal for various values of L=8,16,32,64 :



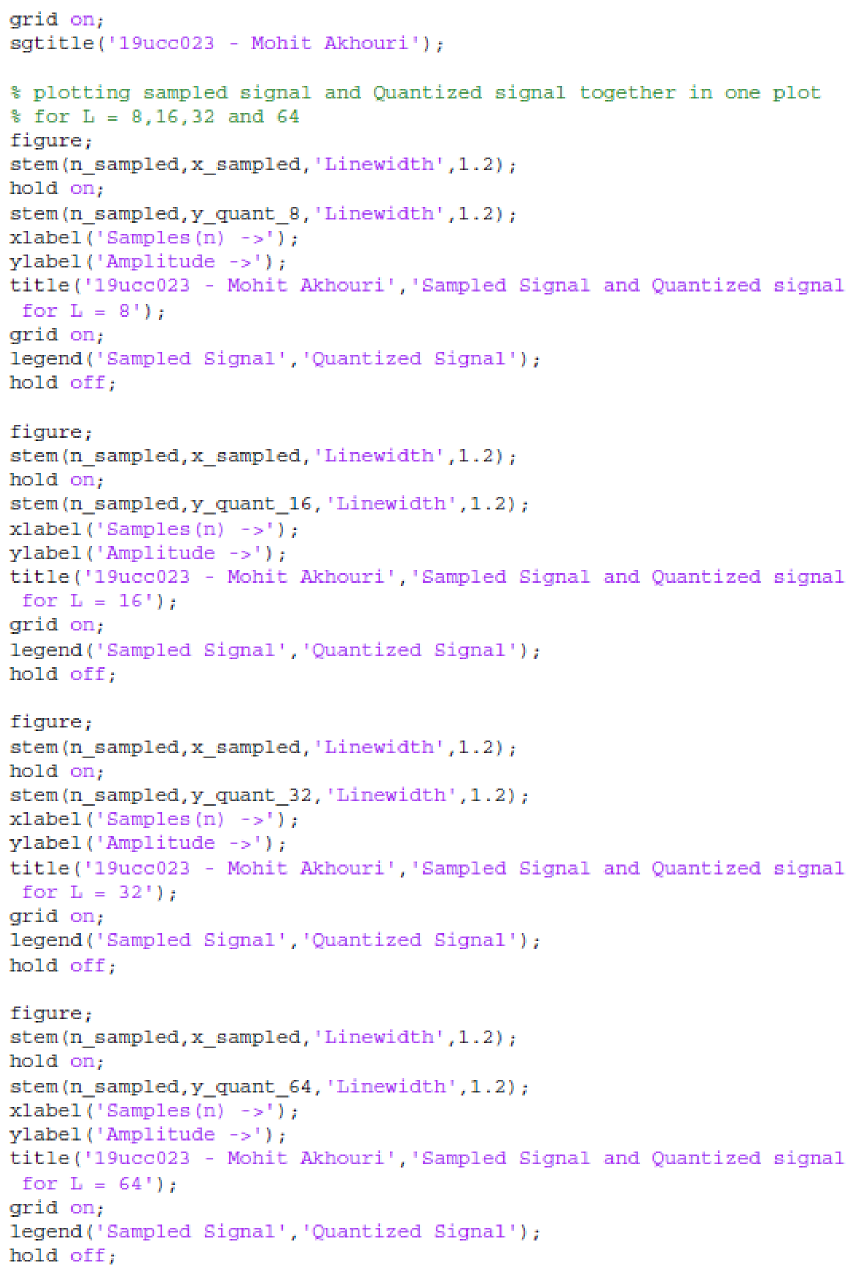
```
% 19ucc023
% Mohit Akhouri
% Experiment 2 - Observation 1

clc;
clear all;
close all;

% generating Ideal Signal for Fs = 100000 Hz

A = 1; % defining Amplitude
n_cycles = 5; % defining number of cycles
fs_ideal = 100000; % defining ideal frequency
f = 3000; % defining message signal frequency

n_ideal = 0:1:floor(n_cycles*(fs_ideal/f))-1; % defining range of n
x_ideal = A*cos(2*pi*f*n_ideal*(1/fs_ideal)); % generating Ideal
 signal

figure; % plotting Ideal signal
stem(n_ideal,x_ideal);
ylabel('Amplitude ->');
xlabel('Samples(n) ->');
title('19ucc023 - Mohit Akhouri','Generation of Ideal Signal for F_{s}
 = 100000 Hz');
grid on;

% generating Sampled signal for Fs = 8000 Hz
fs_sampled = 8000;
n_sampled = 0:1:floor(n_cycles*(fs_sampled/f))-1;
x_sampled = A*cos(2*pi*f*n_sampled*(1/fs_sampled));

figure;
stem(n_sampled,x_sampled,'Linewidth',1.5);
ylabel('Amplitude ->');
xlabel('Samples(n) ->');
title('19ucc023 - Mohit Akhouri','Sampled Signal for F_{s} = 8000
 Hz');
grid on;

y_quant_8 = myquantizer(x_sampled,8); % calculating quantized signal
 for L=8
y_quant_16 = myquantizer(x_sampled,16); % calculating quantized signal
 for L=16
y_quant_32 = myquantizer(x_sampled,32); % calculating quantized signal
 for L=32
y_quant_64 = myquantizer(x_sampled,64); % calculating quantized signal
 for L=64

% plotting Sampled signal and Quantized signal in 2 different plots
% for L = 8 and L = 16
figure;
```

Figure 2.3 Part 1 of the code for observation 1



```
subplot(2,2,1);
stem(n_sampled,x_sampled,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Sampled signal for F_{s} = 8000 Hz');
grid on;
subplot(2,2,2);
stem(n_sampled,y_quant_8,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Quantized signal for L = 8');
grid on;

subplot(2,2,3);
stem(n_sampled,x_sampled,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Sampled signal for F_{s} = 8000 Hz');
grid on;
subplot(2,2,4);
stem(n_sampled,y_quant_16,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Quantized signal for L = 16');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting Sampled signal and Quantized signal in 2 different plots
% for L = 32 and L = 64
figure;
subplot(2,2,1);
stem(n_sampled,x_sampled,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Sampled signal for F_{s} = 8000 Hz');
grid on;
subplot(2,2,2);
stem(n_sampled,y_quant_32,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Quantized signal for L = 32');
grid on;

subplot(2,2,3);
stem(n_sampled,x_sampled,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Sampled signal for F_{s} = 8000 Hz');
grid on;
subplot(2,2,4);
stem(n_sampled,y_quant_64,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Quantized signal for L = 64');
```

Figure 2.4 Part 2 of the code for observation 1



```
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting sampled signal and Quantized signal together in one plot
% for L = 8,16,32 and 64
figure;
stem(n_sampled,x_sampled,'Linewidth',1.2);
hold on;
stem(n_sampled,y_quant_8,'Linewidth',1.2);
xlabel('Samples(n) ->');
ylabel('Amplitude ->');
title('19ucc023 - Mohit Akhouri','Sampled Signal and Quantized signal
 for L = 8');
grid on;
legend('Sampled Signal','Quantized Signal');
hold off;

figure;
stem(n_sampled,x_sampled,'Linewidth',1.2);
hold on;
stem(n_sampled,y_quant_16,'Linewidth',1.2);
xlabel('Samples(n) ->');
ylabel('Amplitude ->');
title('19ucc023 - Mohit Akhouri','Sampled Signal and Quantized signal
 for L = 16');
grid on;
legend('Sampled Signal','Quantized Signal');
hold off;

figure;
stem(n_sampled,x_sampled,'Linewidth',1.2);
hold on;
stem(n_sampled,y_quant_32,'Linewidth',1.2);
xlabel('Samples(n) ->');
ylabel('Amplitude ->');
title('19ucc023 - Mohit Akhouri','Sampled Signal and Quantized signal
 for L = 32');
grid on;
legend('Sampled Signal','Quantized Signal');
hold off;

figure;
stem(n_sampled,x_sampled,'Linewidth',1.2);
hold on;
stem(n_sampled,y_quant_64,'Linewidth',1.2);
xlabel('Samples(n) ->');
ylabel('Amplitude ->');
title('19ucc023 - Mohit Akhouri','Sampled Signal and Quantized signal
 for L = 64');
grid on;
legend('Sampled Signal','Quantized Signal');
hold off;
```

Figure 2.5 Part 3 of the code for observation 1

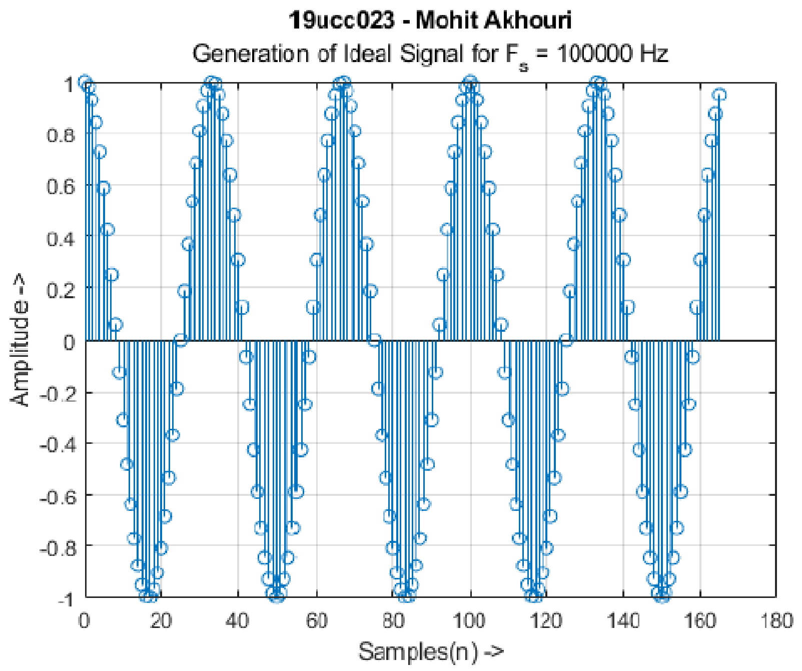


Figure 2.6 Ideal Signal For Sampling frequency = 100000 Hz

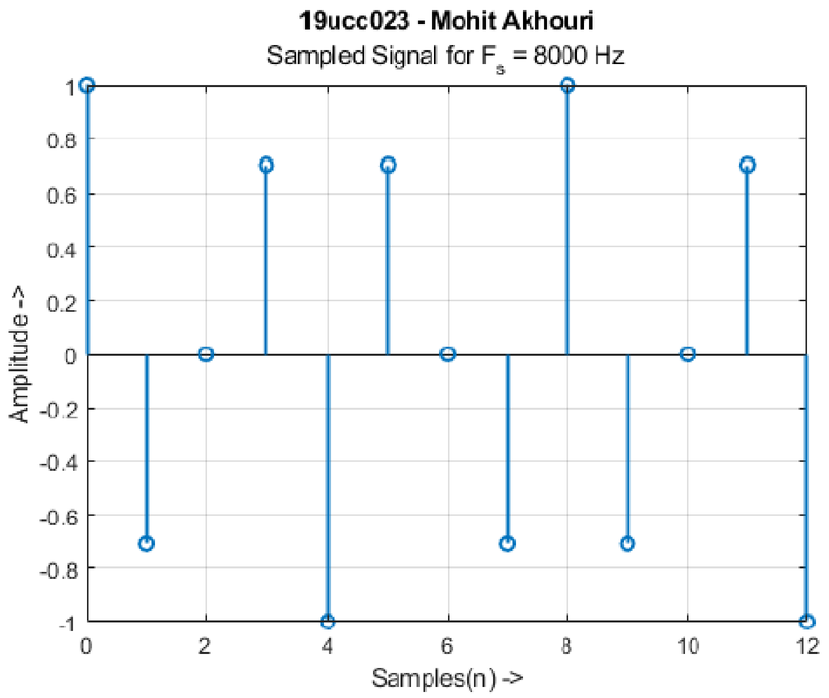


Figure 2.7 Sampled Signal For Sampling frequency = 8000 Hz

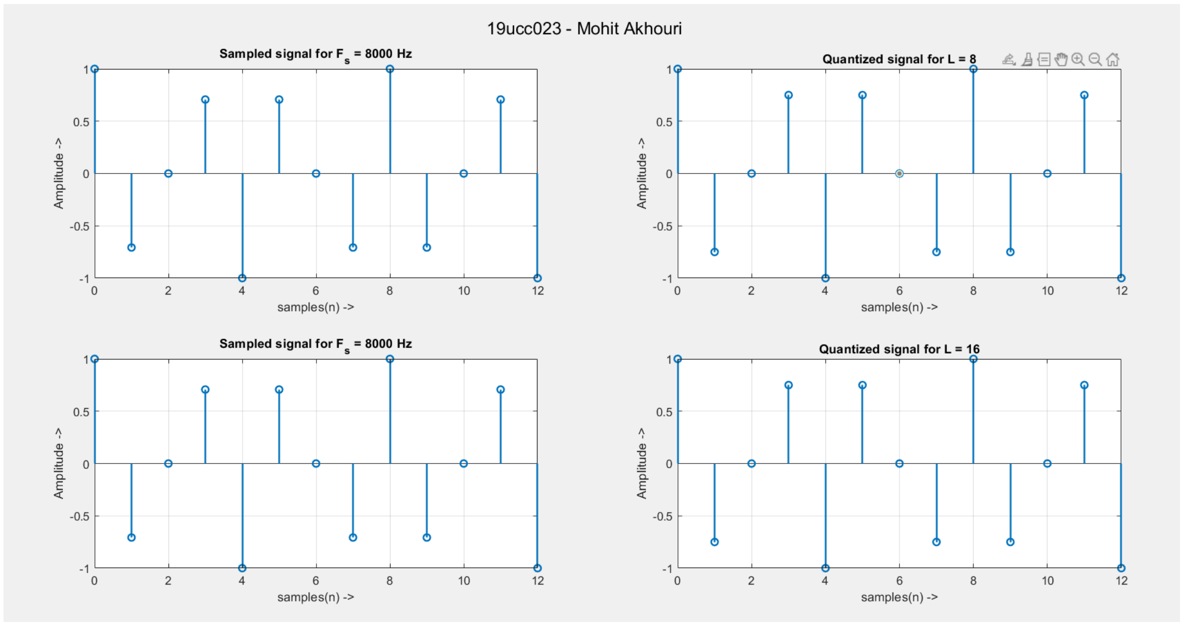


Figure 2.8 Sampled and Quantized signal (Different Plots) for number of levels(L) = 8 and 16

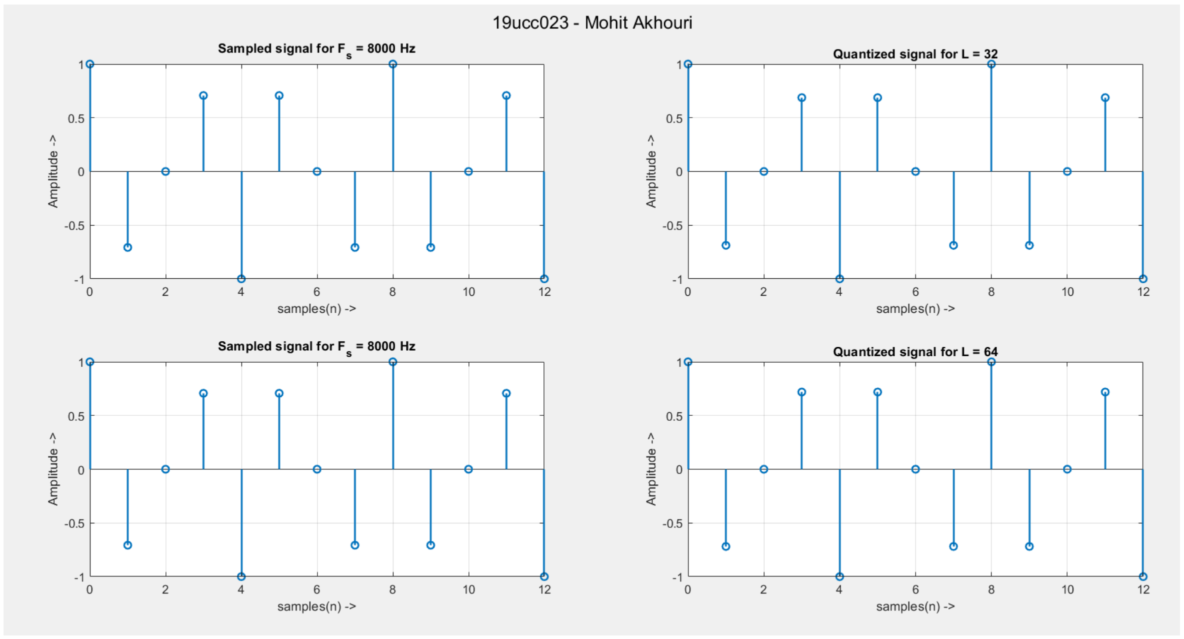


Figure 2.9 Sampled and Quantized signal (Different Plots) for number of levels(L) = 32 and 64

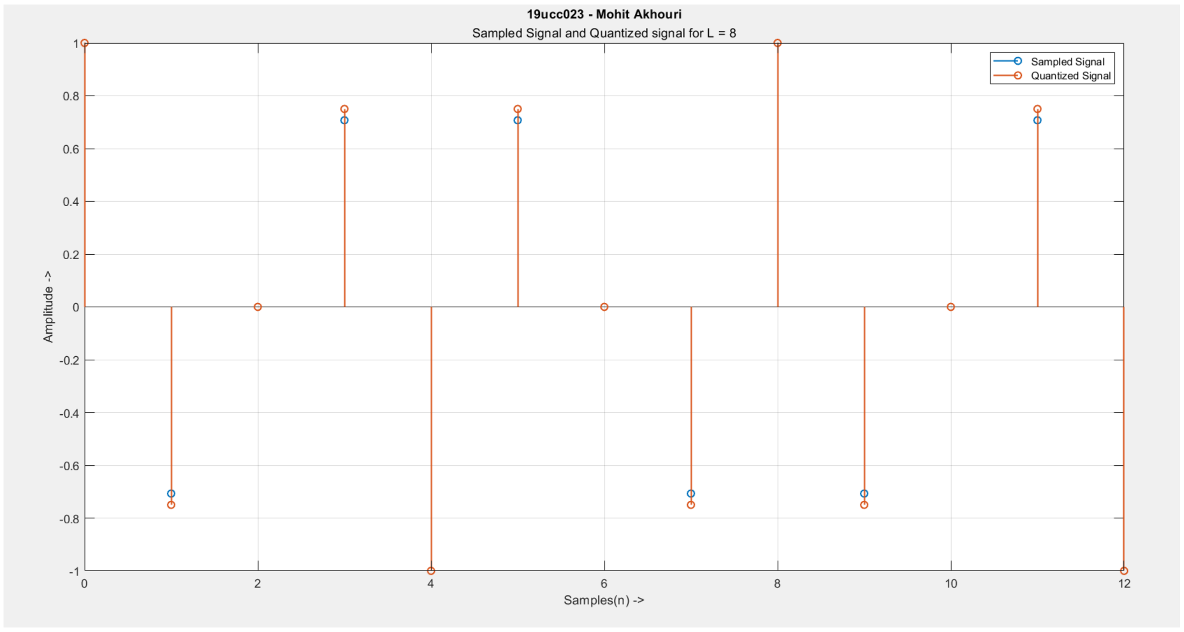


Figure 2.10 Sampled and Quantized signal (Same plot using **hold on**) for number of levels(L) = 8

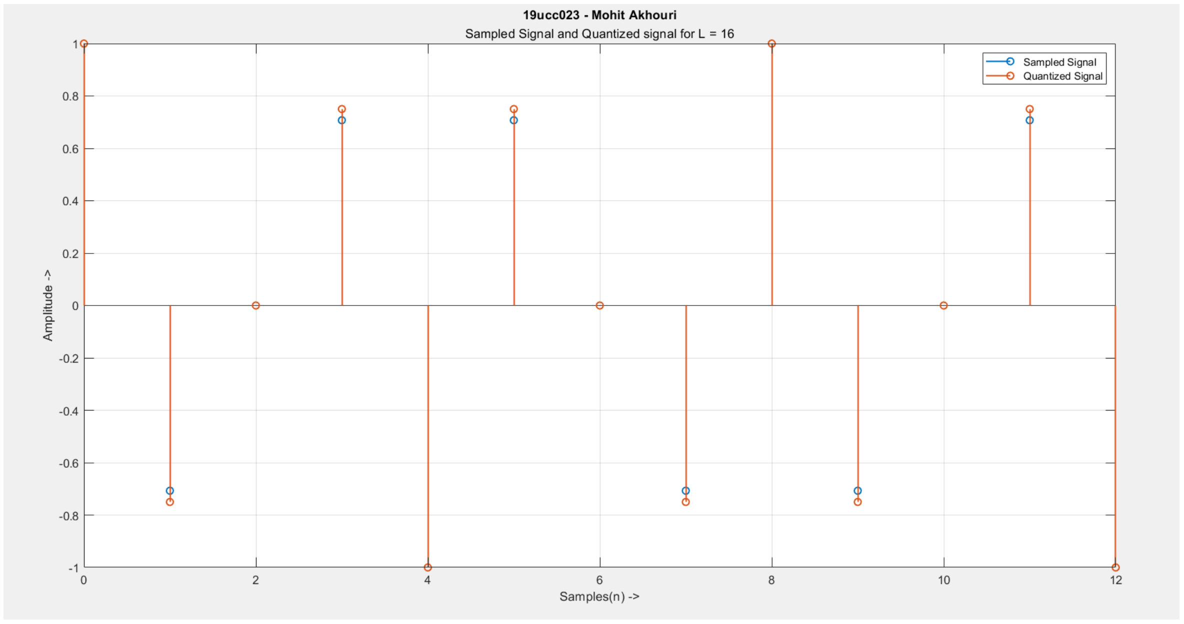


Figure 2.11 Sampled and Quantized signal (Same plot using **hold on**) for number of levels(L) = 16

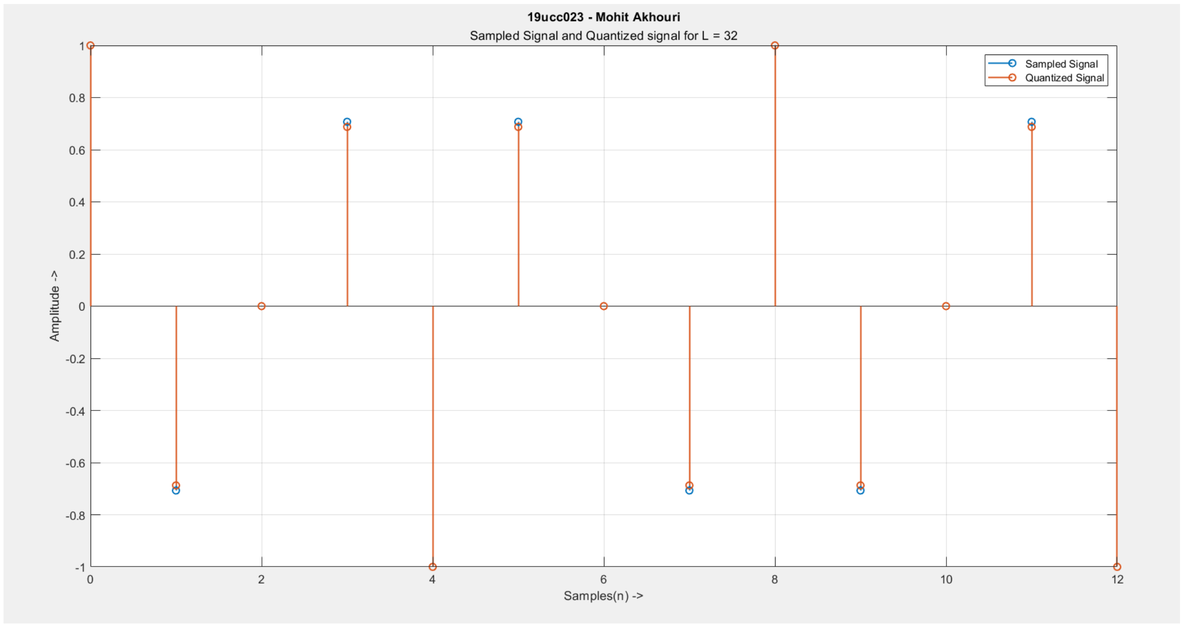


Figure 2.12 Sampled and Quantized signal (Same plot using **hold on**) for number of levels(L) = 32

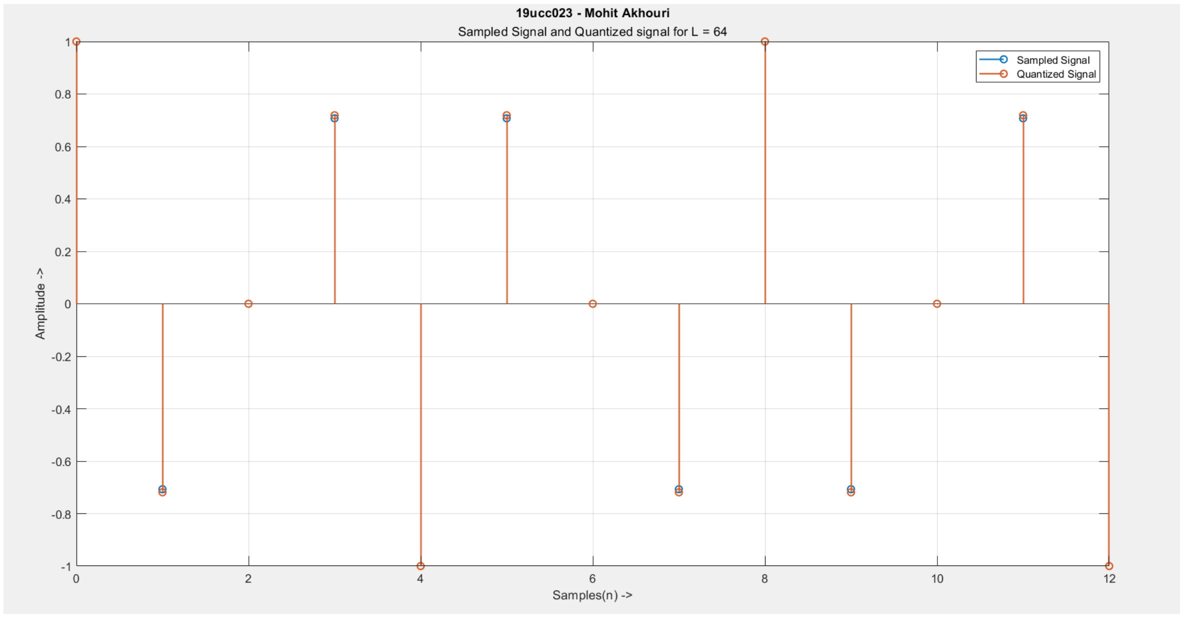
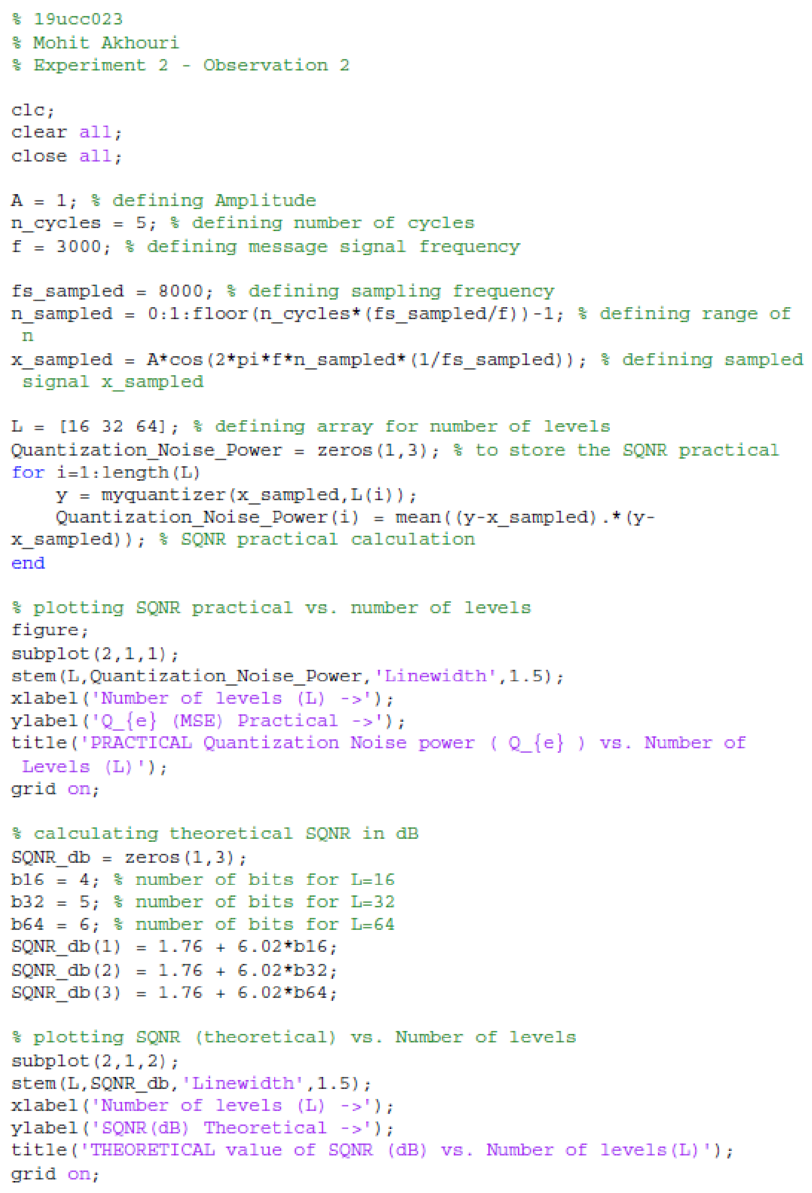


Figure 2.13 Sampled and Quantized signal (Same plot using **hold on**) for number of levels(L) = 64

2.4.2 Plot the graph b/w Quantization Noise Power (Q_e) and quantization levels L=16,32,64:



```
% 19ucc023
% Mohit Akhouri
% Experiment 2 - Observation 2

clc;
clear all;
close all;

A = 1; % defining Amplitude
n_cycles = 5; % defining number of cycles
f = 3000; % defining message signal frequency

fs_sampled = 8000; % defining sampling frequency
n_sampled = 0:1:floor(n_cycles*(fs_sampled/f))-1; % defining range of
 n
x_sampled = A*cos(2*pi*f*n_sampled*(1/fs_sampled)); % defining sampled
 signal x_sampled

L = [16 32 64]; % defining array for number of levels
Quantization_Noise_Power = zeros(1,3); % to store the SQNR practical
for i=1:length(L)
    y = myquantizer(x_sampled,L(i));
    Quantization_Noise_Power(i) = mean((y-x_sampled).*(y-
x_sampled)); % SQNR practical calculation
end

% plotting SQNR practical vs. number of levels
figure;
subplot(2,1,1);
stem(L,Quantization_Noise_Power,'Linewidth',1.5);
xlabel('Number of levels (L) ->');
ylabel('Q_{e} (MSE) Practical ->');
title('PRACTICAL Quantization Noise power ( Q_{e} ) vs. Number of
 Levels (L)');
grid on;

% calculating theoretical SQNR in dB
SQNR_db = zeros(1,3);
b16 = 4; % number of bits for L=16
b32 = 5; % number of bits for L=32
b64 = 6; % number of bits for L=64
SQNR_db(1) = 1.76 + 6.02*b16;
SQNR_db(2) = 1.76 + 6.02*b32;
SQNR_db(3) = 1.76 + 6.02*b64;

% plotting SQNR (theoretical) vs. Number of levels
subplot(2,1,2);
stem(L,SQNR_db,'Linewidth',1.5);
xlabel('Number of levels (L) ->');
ylabel('SQNR(dB) Theoretical ->');
title('THEORETICAL value of SQNR (dB) vs. Number of levels(L)');
grid on;
```

Figure 2.14 Code for observation 2

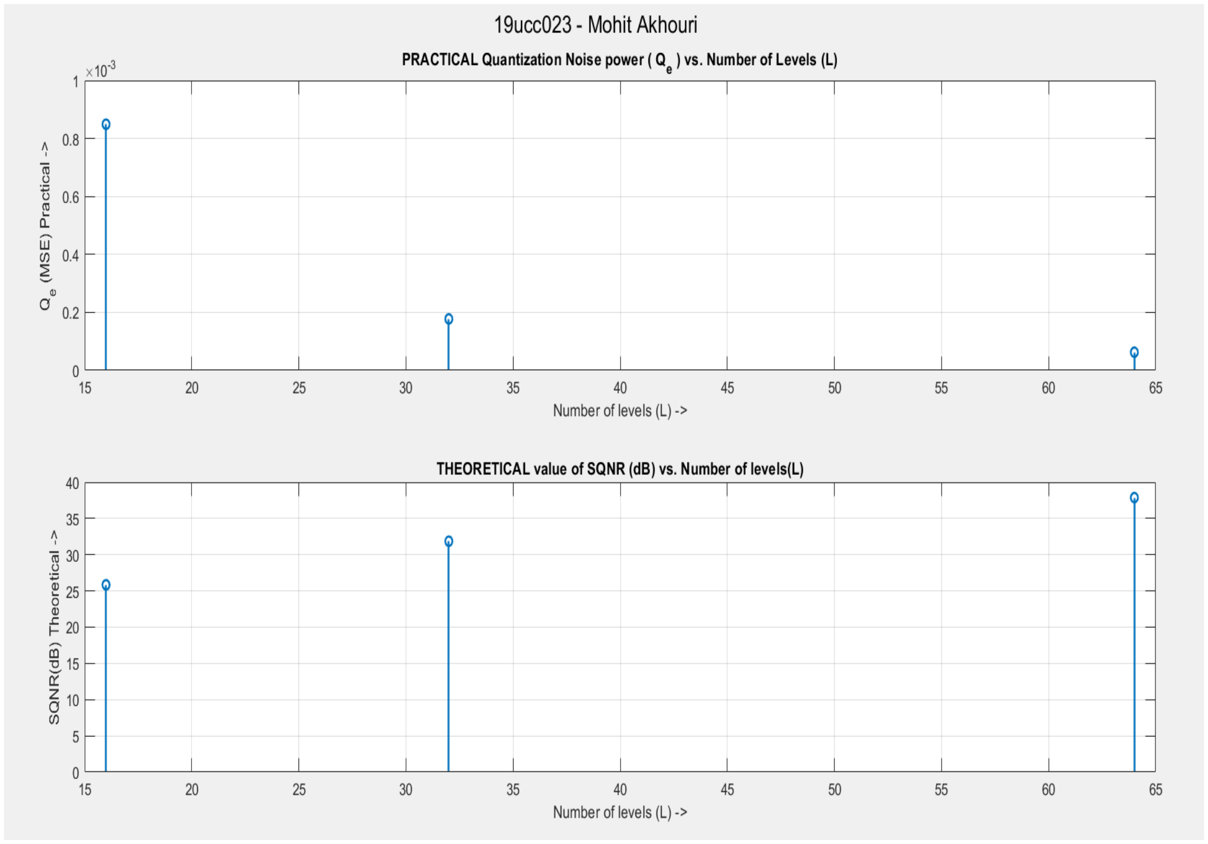
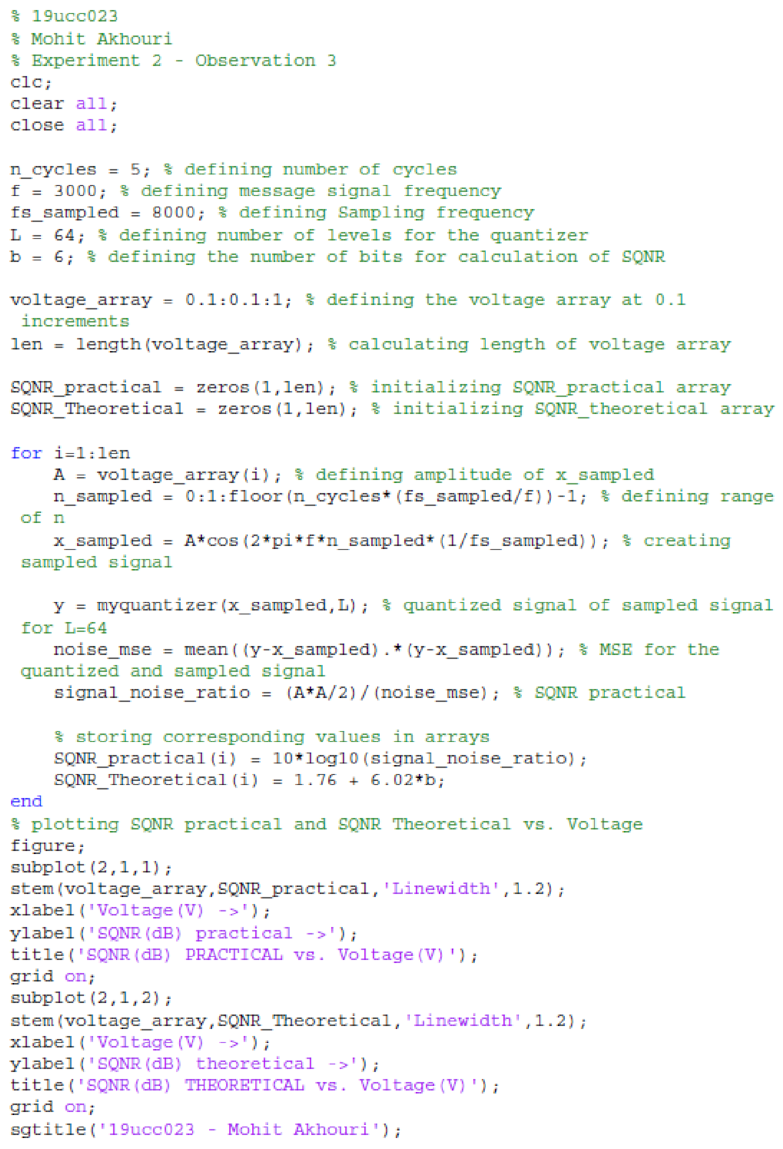


Figure 2.15 Plots of Practical and Theoretical (Q_e) vs. Number of levels

2.4.3 Plot SQNR vs. Input voltage for voltage levels in step size 0.1V :



```
% 19ucc023
% Mohit Akhouri
% Experiment 2 - Observation 3
clc;
clear all;
close all;

n_cycles = 5; % defining number of cycles
f = 3000; % defining message signal frequency
fs_sampled = 8000; % defining Sampling frequency
L = 64; % defining number of levels for the quantizer
b = 6; % defining the number of bits for calculation of SQNR

voltage_array = 0.1:0.1:1; % defining the voltage array at 0.1
 increments
len = length(voltage_array); % calculating length of voltage array

SQNR_practical = zeros(1,len); % initializing SQNR_practical array
SQNR_Theoretical = zeros(1,len); % initializing SQNR_theoretical array

for i=1:len
    A = voltage_array(i); % defining amplitude of x_sampled
    n_sampled = 0:1:floor(n_cycles*(fs_sampled/f))-1; % defining range
 of n
    x_sampled = A*cos(2*pi*f*n_sampled*(1/fs_sampled)); % creating
 sampled signal

    y = myquantizer(x_sampled,L); % quantized signal of sampled signal
 for L=64
    noise_mse = mean((y-x_sampled).*(y-x_sampled)); % MSE for the
 quantized and sampled signal
    signal_noise_ratio = (A*A/2)/(noise_mse); % SQNR practical

    % storing corresponding values in arrays
    SQNR_practical(i) = 10*log10(signal_noise_ratio);
    SQNR_Theoretical(i) = 1.76 + 6.02*b;
end
% plotting SQNR practical and SQNR Theoretical vs. Voltage
figure;
subplot(2,1,1);
stem(voltage_array,SQNR_practical,'Linewidth',1.2);
xlabel('Voltage(V) ->');
ylabel('SQNR(dB) practical ->');
title('SQNR(dB) PRACTICAL vs. Voltage(V)');
grid on;
subplot(2,1,2);
stem(voltage_array,SQNR_Theoretical,'Linewidth',1.2);
xlabel('Voltage(V) ->');
ylabel('SQNR(dB) theoretical ->');
title('SQNR(dB) THEORETICAL vs. Voltage(V)');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 2.16 Code for observation 3

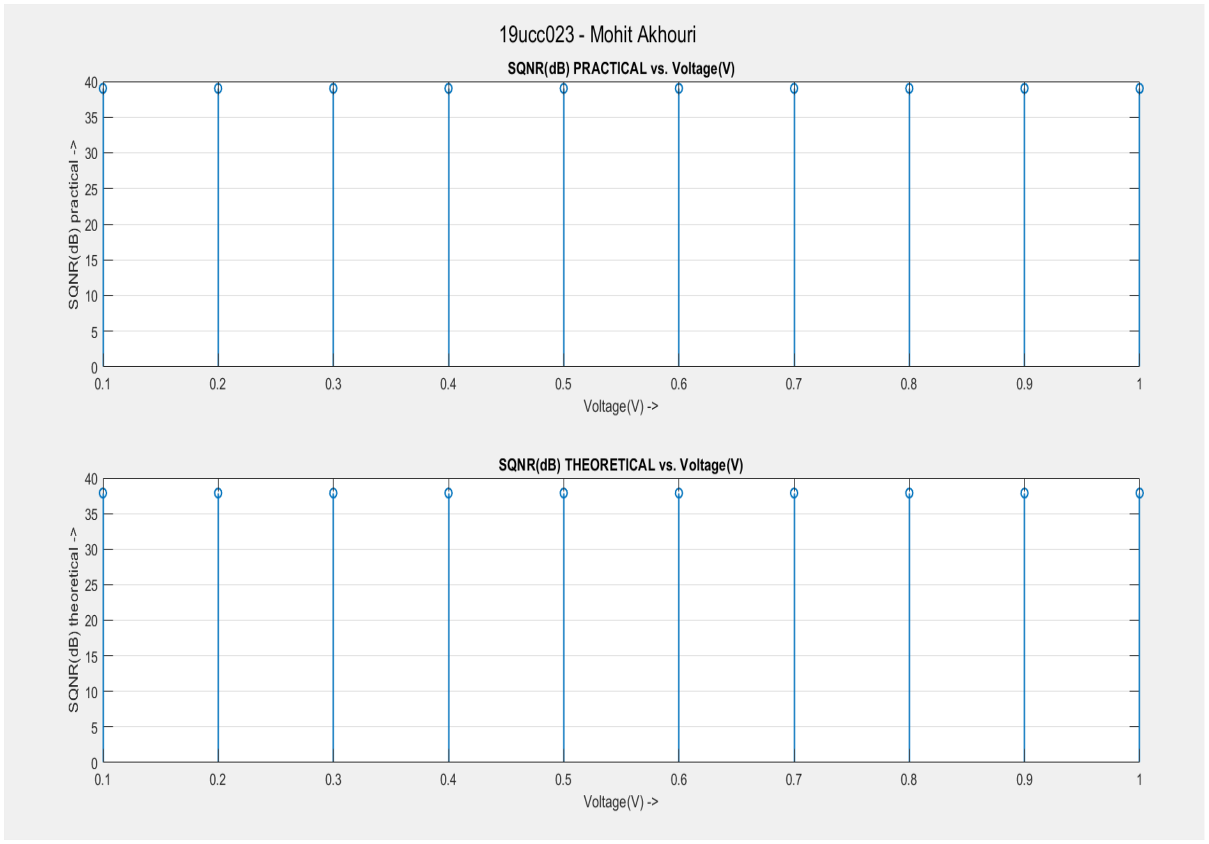
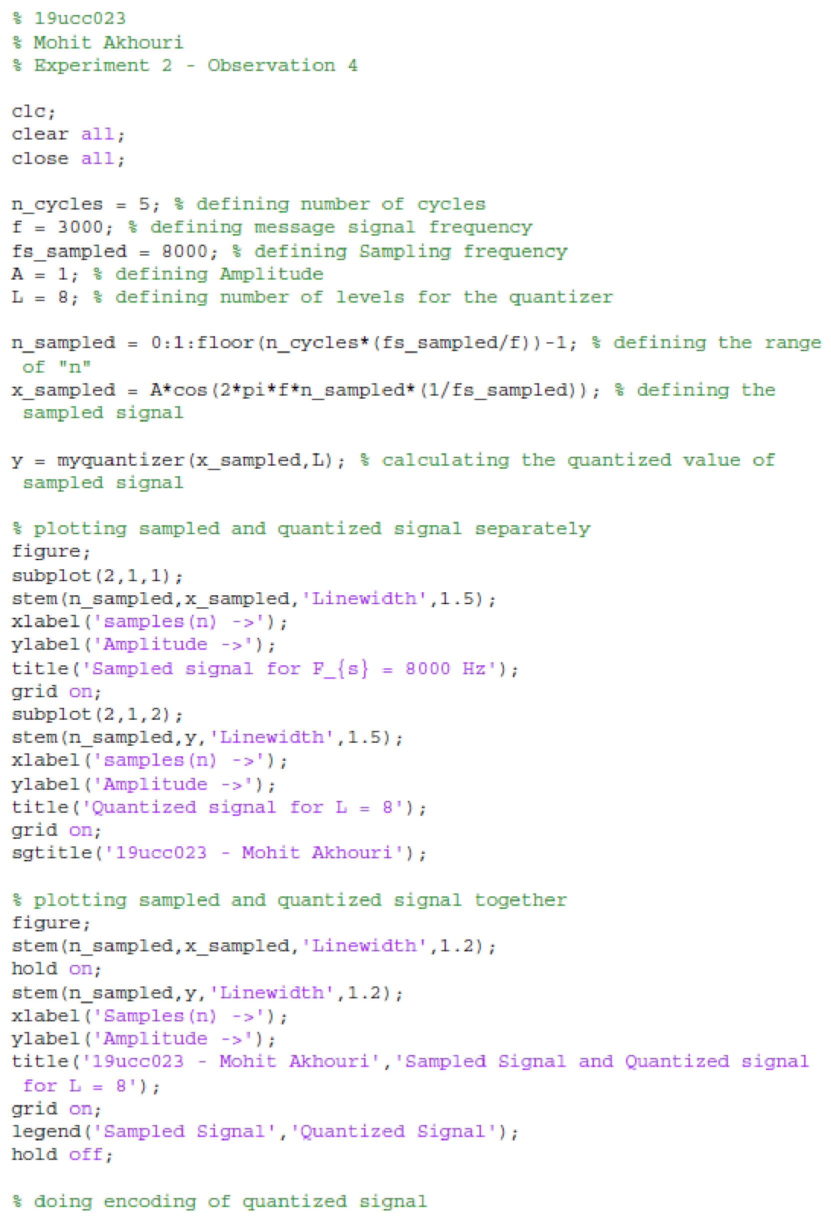


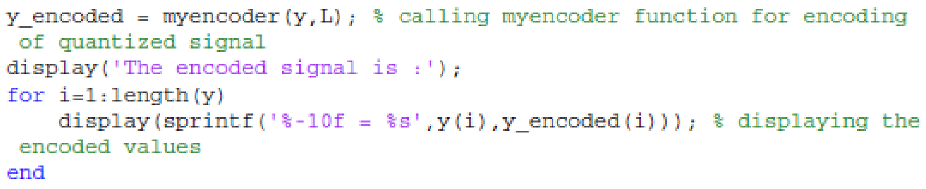
Figure 2.17 Plots of SQNR(dB) vs. Voltage levels in 0.1V step size

2.4.4 Plot the sampled, quantized and encoded signal for L=8 :



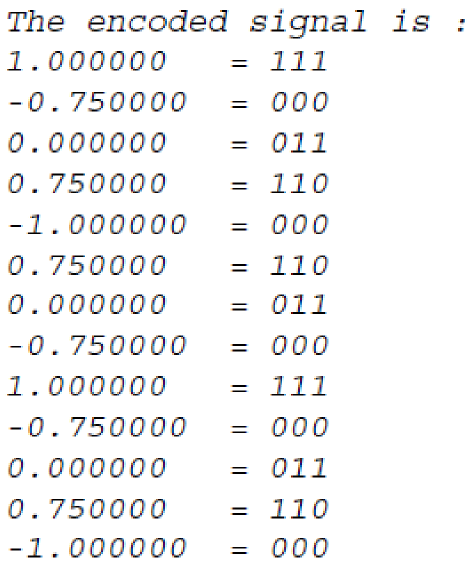
```
% 19ucc023
% Mohit Akhouri
% Experiment 2 - Observation 4

clc;
clear all;
close all;

n_cycles = 5; % defining number of cycles
f = 3000; % defining message signal frequency
fs_sampled = 8000; % defining Sampling frequency
A = 1; % defining Amplitude
L = 8; % defining number of levels for the quantizer

n_sampled = 0:1:floor(n_cycles*(fs_sampled/f))-1; % defining the range
 of "n"
x_sampled = A*cos(2*pi*f*n_sampled*(1/fs_sampled)); % defining the
 sampled signal

y = myquantizer(x_sampled,L); % calculating the quantized value of
 sampled signal

% plotting sampled and quantized signal separately
figure;
subplot(2,1,1);
stem(n_sampled,x_sampled,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Sampled signal for F_{s} = 8000 Hz');
grid on;
subplot(2,1,2);
stem(n_sampled,y,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Quantized signal for L = 8');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting sampled and quantized signal together
figure;
stem(n_sampled,x_sampled,'Linewidth',1.2);
hold on;
stem(n_sampled,y,'Linewidth',1.2);
xlabel('Samples(n) ->');
ylabel('Amplitude ->');
title('19ucc023 - Mohit Akhouri','Sampled Signal and Quantized signal
 for L = 8');
grid on;
legend('Sampled Signal','Quantized Signal');
hold off;

% doing encoding of quantized signal
```

Figure 2.18 Part 1 of the Code for observation 4



```
y_encoded = myencoder(y,L); % calling myencoder function for encoding
 of quantized signal
display('The encoded signal is :');
for i=1:length(y)
    display(sprintf('%-10f = %s',y(i),y_encoded(i))); % displaying the
 encoded values
end
```

Figure 2.19 Part 2 of the Code for observation 4



```
The encoded signal is :
1.000000    = 111
-0.750000   = 000
0.000000    = 011
0.750000    = 110
-1.000000   = 000
0.750000    = 110
0.000000    = 011
-0.750000   = 000
1.000000    = 111
-0.750000   = 000
0.000000    = 011
0.750000    = 110
-1.000000   = 000
```

Figure 2.20 Displaying **encoded values** for different quantized values

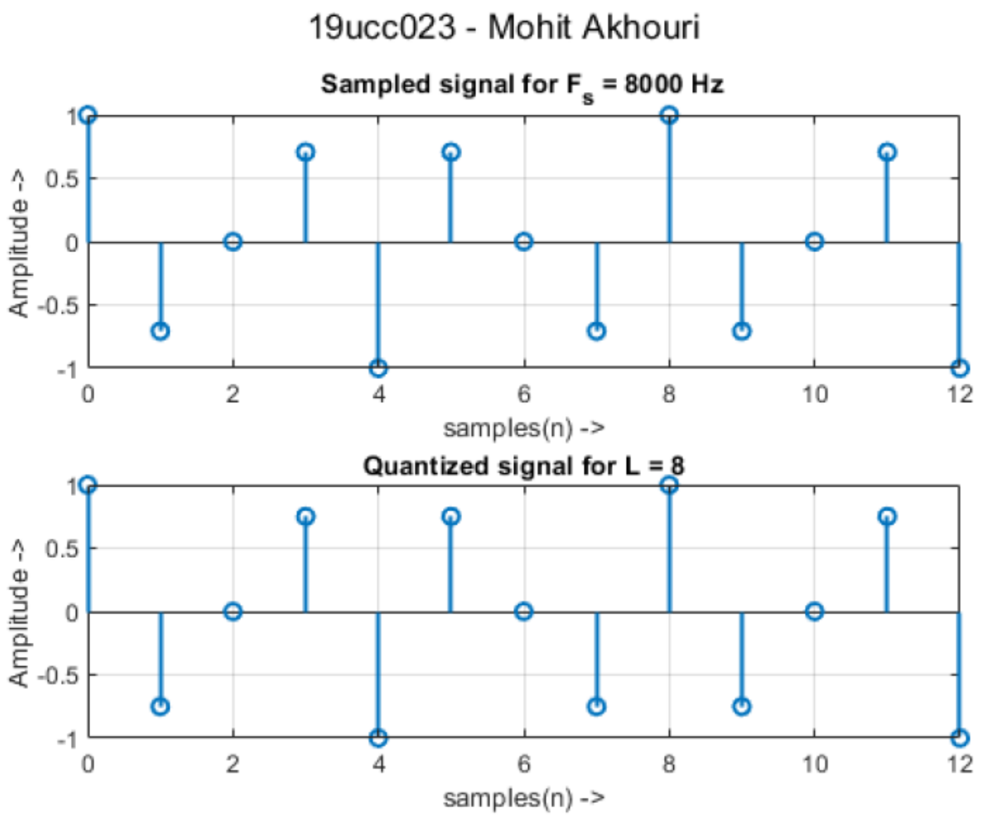


Figure 2.21 Sampled and Quantized signal (Different Plots) for number of levels(L) = 8

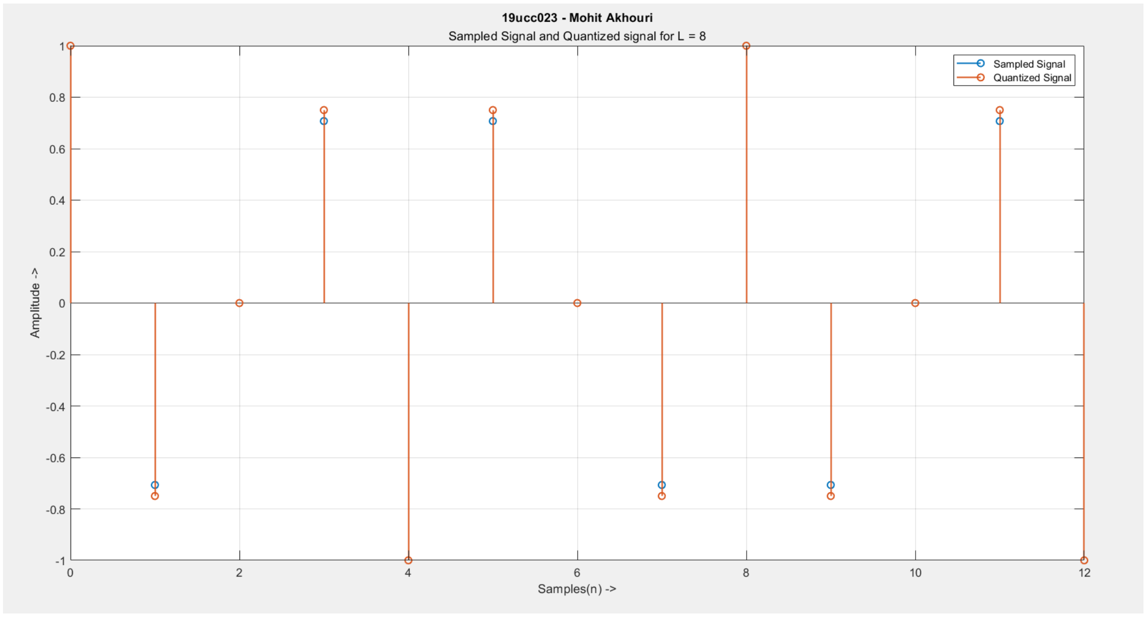


Figure 2.22 Sampled and Quantized signal (Same plots using **hold on**) for number of levels(L) = 8

2.4.5 Quantization and encoding in Simulink :

```
% 19ucc023
% Mohit Akhouri
% Experiment 2 - Observation 5

% doing quantization and encoding for L=8 via Simulink model

sim('Simulink_Observation_5'); % calling the Simulink model

n_cycles = 5; % defining number of cycles
f = 3000; % defining message signal frequency
fs_sampled = 8000; % defining Sampling frequency
A = 1; % defining Amplitude
L = 8; % defining number of levels for the quantizer

n_sampled = 0:1:floor(n_cycles*(fs_sampled/f))-1; % defining the range
 of "n"
x_sampled = A*cos(2*pi*f*n_sampled*(1/fs_sampled)); % defining the
 sampled signal

% plotting sampled and quantized signal separately
figure;
subplot(2,1,1);
stem(n_sampled,x_sampled,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Sampled signal for F_{s} = 8000 Hz');
grid on;
subplot(2,1,2);
stem(n_sampled,out.y_sampled.data,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('Amplitude ->');
title('Quantized signal for L = 8');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting sampled and quantized signal together
figure;
stem(n_sampled,x_sampled,'Linewidth',1.2);
hold on;
stem(n_sampled,out.y_sampled.data,'Linewidth',1.2);
xlabel('Samples(n) ->');
ylabel('Amplitude ->');
title('19ucc023 - Mohit Akhouri','Sampled Signal and Quantized signal
 for L = 8');
grid on;
legend('Sampled Signal','Quantized Signal');
hold off;

% doing encoding of quantized signal via Simulink
y = out.y_sampled.data;
y_encoded = out.y_encoded.data;
% doing encoding of quantized signal
```

Figure 2.23 Part 1 of the Code for observation 5

```
display('The encoded signal is :');
for i=1:length(y)
    display(sprintf('%-10f = %s',y(i),dec2bin(y_encoded(i),3))); %
 displaying the encoded values
end
```

Figure 2.24 Part 2 of the Code for observation 5

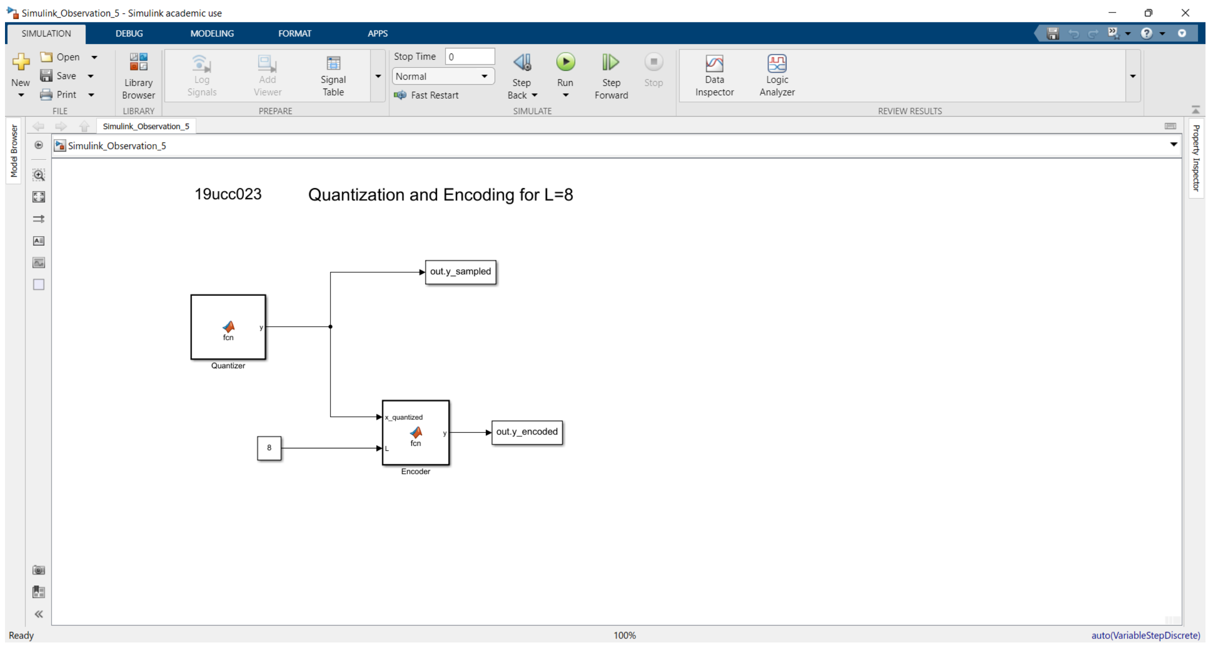
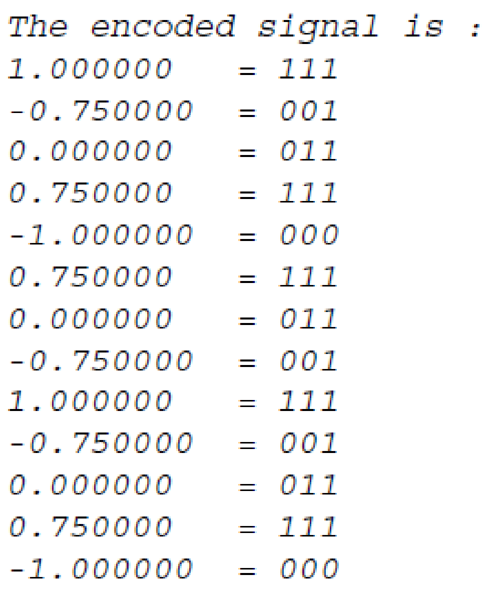


Figure 2.25 Simulink model for Quantization and Encoding



```
The encoded signal is :
1.000000    = 111
-0.750000   = 001
0.000000    = 011
0.750000    = 111
-1.000000   = 000
0.750000    = 111
0.000000    = 011
-0.750000   = 001
1.000000    = 111
-0.750000   = 001
0.000000    = 011
0.750000    = 111
-1.000000   = 000
```

Figure 2.26 Encoded values for the quantized signal for L=8

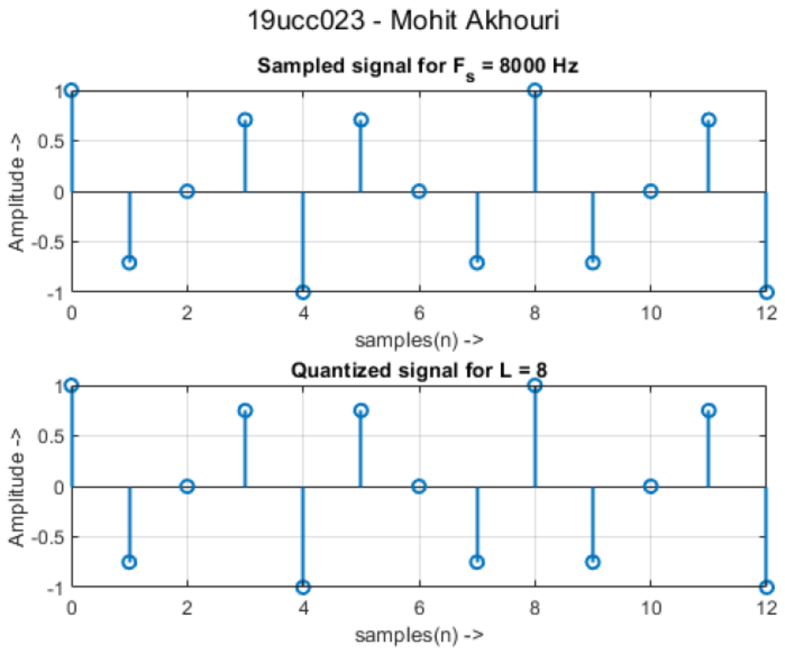


Figure 2.27 Sampled and Quantized signal (Different Plots) for L = 8 using Simulink model

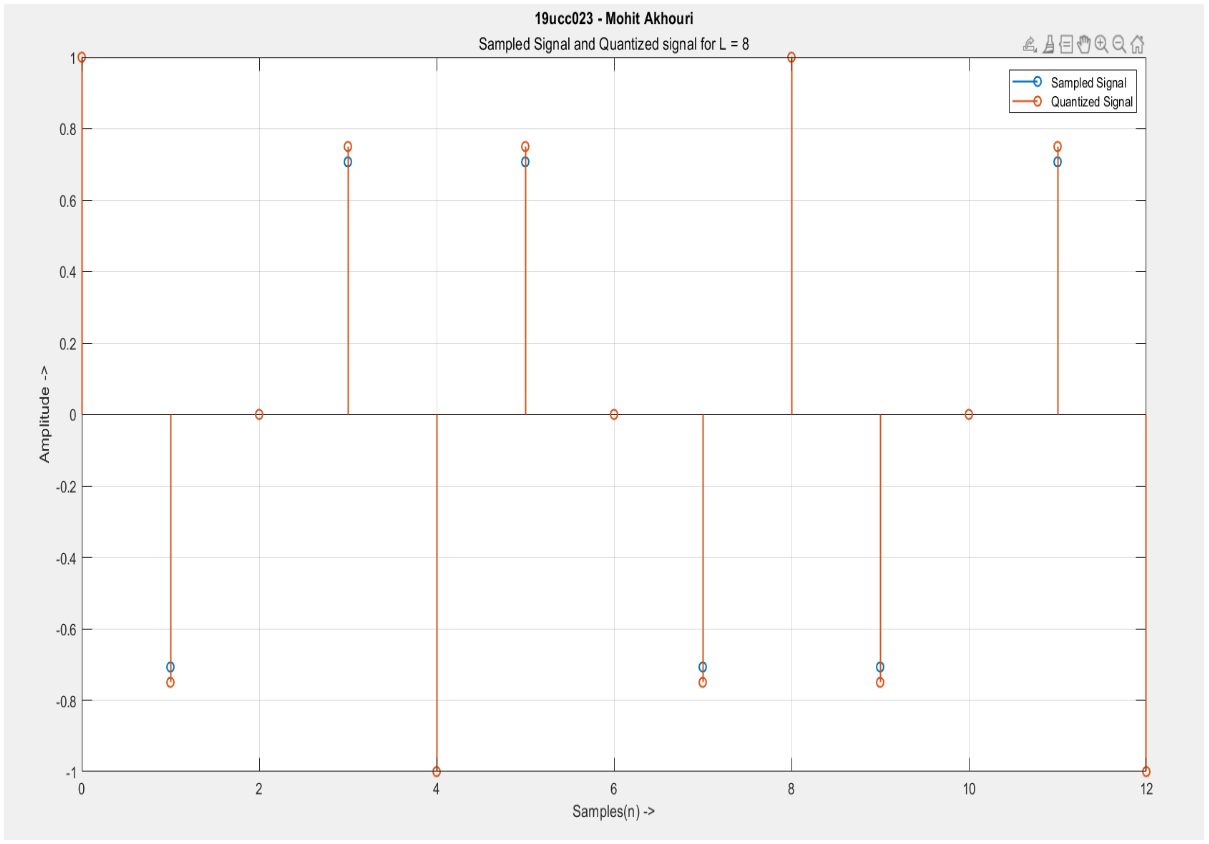
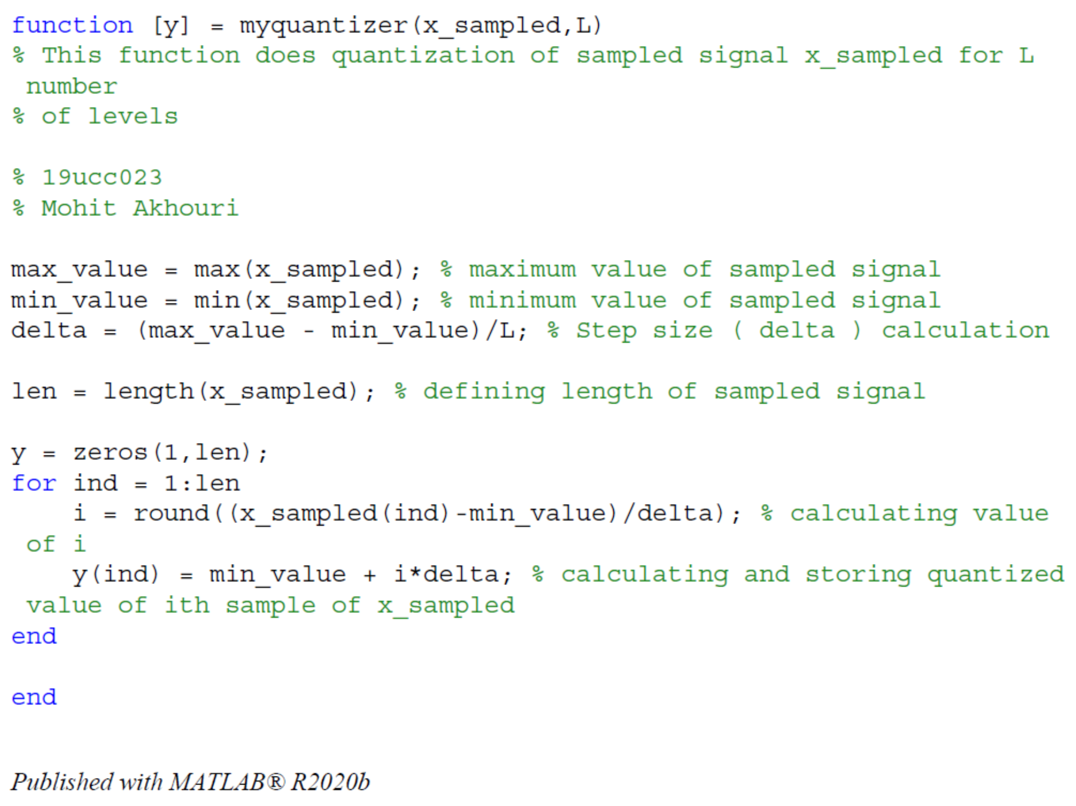


Figure 2.28 Sampled and Quantized signal (Same Plots) for L = 8 using Simulink model

2.4.6 Functions used in main codes for Quantization and encoding :

2.4.6.1 myquantizer.m function code :

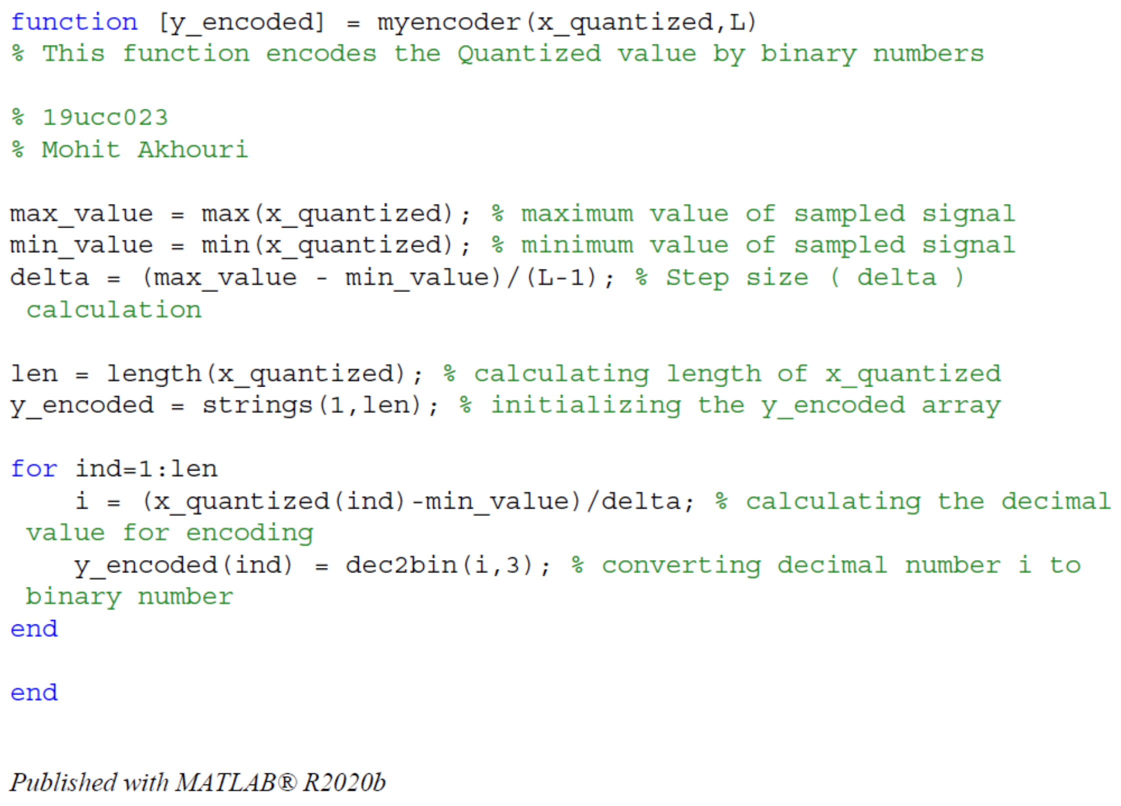


```
function [y] = myquantizer(x_sampled,L)
% This function does quantization of sampled signal x_sampled for L
 number
% of levels

% 19ucc023
% Mohit Akhouri

max_value = max(x_sampled); % maximum value of sampled signal
min_value = min(x_sampled); % minimum value of sampled signal
delta = (max_value - min_value)/L; % Step size ( delta ) calculation

len = length(x_sampled); % defining length of sampled signal

y = zeros(1,len);
for ind = 1:len
    i = round((x_sampled(ind)-min_value)/delta); % calculating value
 of i
    y(ind) = min_value + i*delta; % calculating and storing quantized
 value of ith sample of x_sampled
end

end
```

Published with MATLAB® R2020b

Figure 2.29 myquantizer function for Quantization computation

2.4.6.2 myencoder.m function code :



```
function [y_encoded] = myencoder(x_quantized,L)
% This function encodes the Quantized value by binary numbers

% 19ucc023
% Mohit Akhouri

max_value = max(x_quantized); % maximum value of sampled signal
min_value = min(x_quantized); % minimum value of sampled signal
delta = (max_value - min_value)/(L-1); % Step size ( delta )
 calculation

len = length(x_quantized); % calculating length of x_quantized
y_encoded = strings(1,len); % initializing the y_encoded array

for ind=1:len
    i = (x_quantized(ind)-min_value)/delta; % calculating the decimal
 value for encoding
    y_encoded(ind) = dec2bin(i,3); % converting decimal number i to
 binary number
end

end
```

Published with MATLAB® R2020b

Figure 2.30 myencoder function for Encoding of Quantized signal

2.5 Conclusion

In this experiment , we learnt the concepts of **Quantization** and **encoding** of Digital Signal Processing. We learnt how do we do quantization by dividing the voltage range into number of levels and then rounding each of sample values. We also learnt the process of encoding different quantized values to different binary numbers. We also learnt the concepts of **Quantization Noise error** and **Signal to Noise ratio (SQNR)** and how to compute them both practically and theoretically. We concluded that **Quantization noise** is **lower** for **more number of levels**. We also concluded the effect of increasing voltage levels on SQNR. We performed the same experiment in Simulink and cross-checked the results obtained with the MATLAB code.