# Bangladesh University of Engineering & Technology

## Department of Electrical & Electronic Engineering

Course No: 312

Course Title: Digital Signal Processing Lab

Date of Performance: 09.03.2021

Date of Submission: 16.03.2021

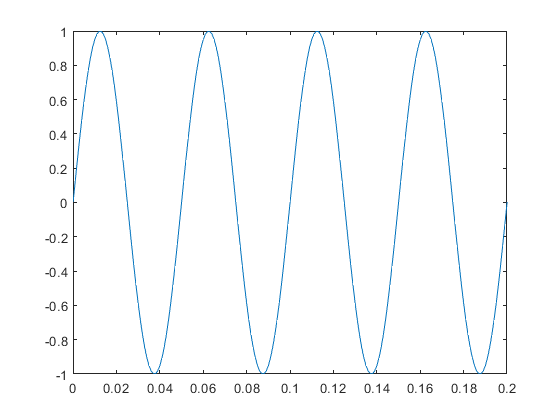
Tasnim Nishat Islam

Student Id: 1706092

Section: B1

Level 3 Term 1

Part A:

Codes:

clc

clear all

close all

%%load the signal

t = 0:0.001:0.2;

signal = sin(2\*pi\*20\*t);

%%sample the signal

n = 0:1:40

fs = 200

stem\_sig = sin(2\*pi\*(10/fs)\*n)

%%plot the original signal

figure("Name", "original");

plot(t, signal);

%plot the sampled version

figure("Name", "sampled\_signal");

Figure : Original Signal

stem(stem\_sig);

%%reconstruct

recon = interp1(n/fs, stem\_sig, t, 'spline');

figure("Name", "recon");

plot(recon);

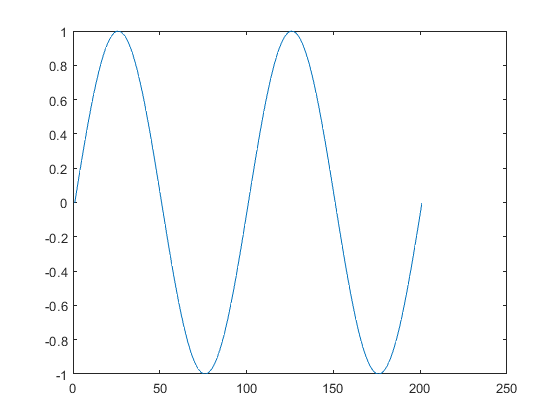
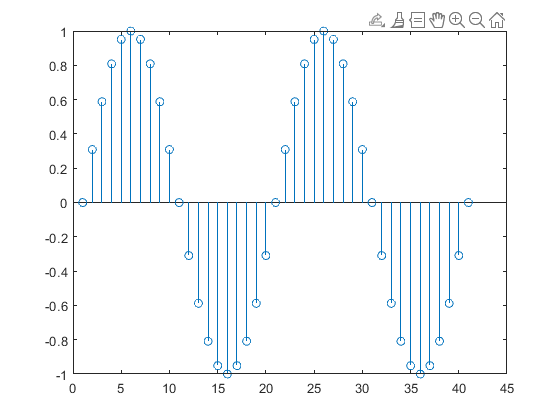


Figure : Sampled at 200 Hz

Figure : Reconstructed Signal after sampling

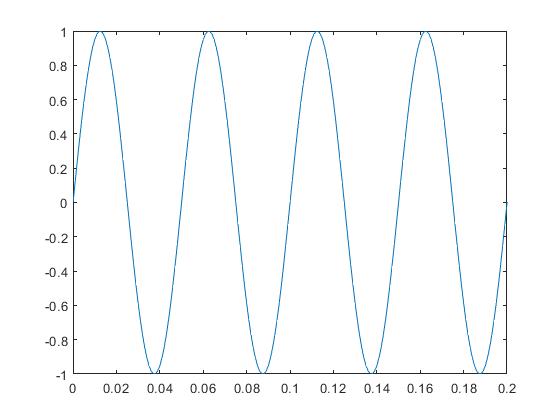
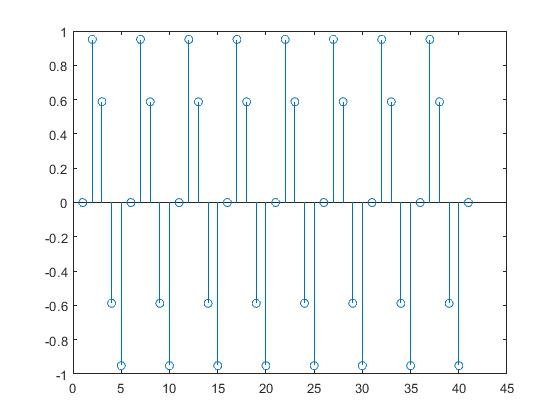
For 50 Hz

Figure : Sampled at 50 Hz

Figure : Original signal

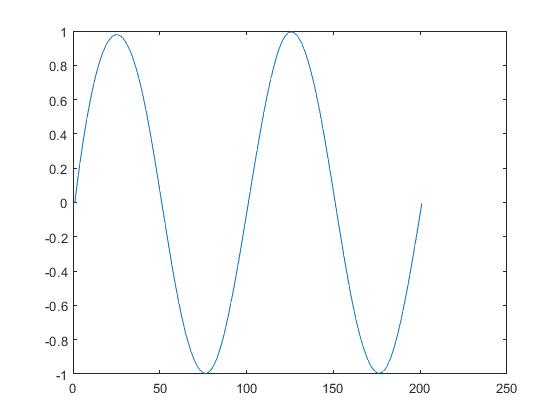


Figure : Reconstructed after sampling in 50 Hz

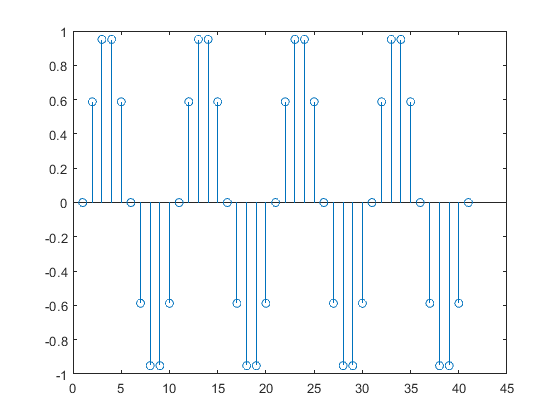
For 100 Hz

Figure : Sampled at 100 Hz

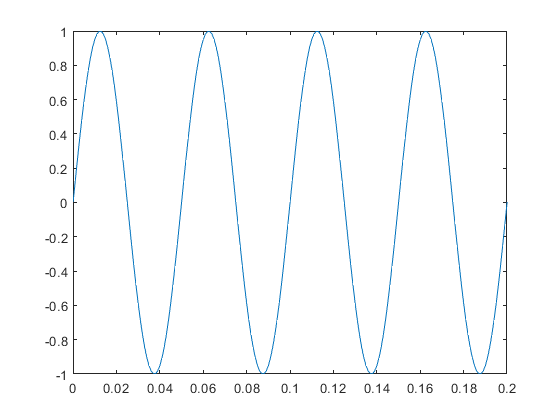
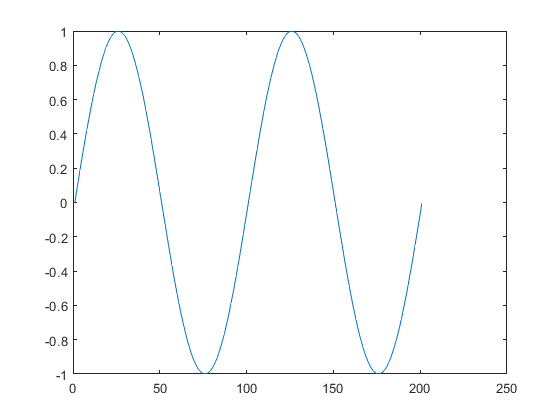


Figure : Reconstructed signal sampled at 100 Hz

Figure : Original Signal

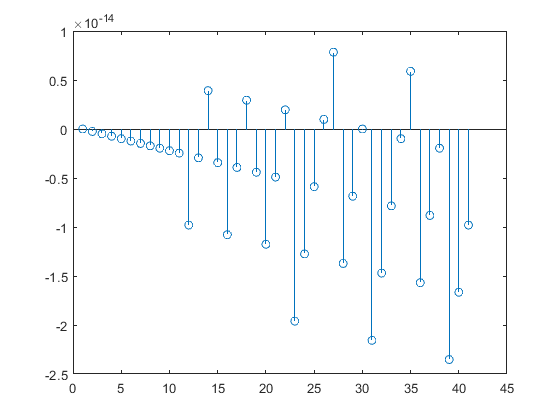
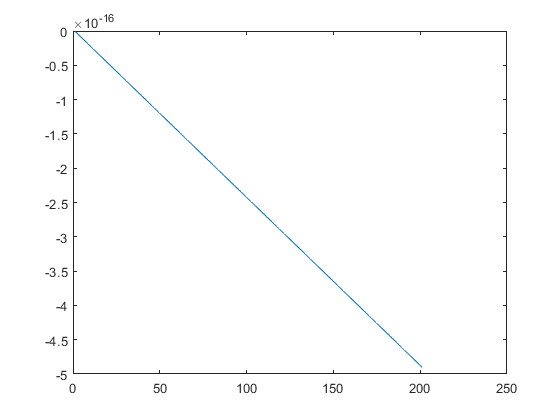
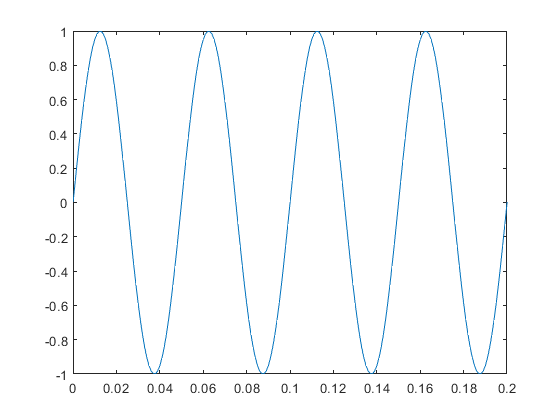
For 10 Hz

Figure : Reconstructed after sampling

Figure : Sampled at 10 Hz

Figure : original Signal

Step 5

signal = sin(2\*pi\*10\*t)+sin(2\*pi\*50\*t)+sin(2\*pi\*100\*t);

%%plot the original signal

figure("Name", "original");

plot(t, signal);

title("original signal")

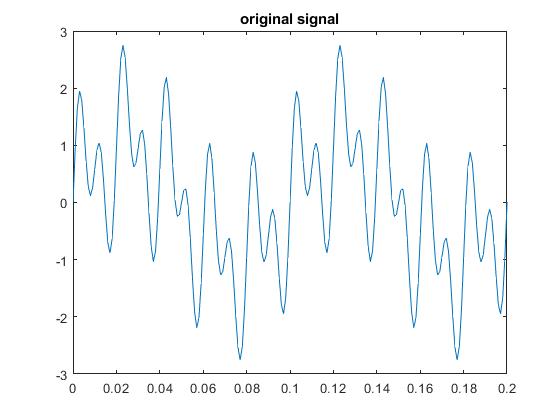


Figure : Plot of the original Signal

Part B:

Uniform 3-bit Quantizer:

function y\_quantized= quantize(x\_sampled,bit)

N=length(x\_sampled);

L=2^bit;

del=((max(x\_sampled)+0.0000000001)-min(x\_sampled))/(L-1);

for j=1:L

level(j)=min(x\_sampled)+del\*(j-1);

end

for i=1:N

for j=1:L-1

if(x\_sampled(i)>=level(j)&& x\_sampled(i)<level(j+1))

if(x\_sampled(i)>=level(j)+.5\*del)

y\_quantized(i)=level(j+1);

else

y\_quantized(i)=level(j);

end

end

end

end

y\_quantized

clc

clear all

close all

%%load the signal

t = 0:0.001:0.2;

signal = sin(2\*pi\*10\*t)+sin(2\*pi\*50\*t)+sin(2\*pi\*100\*t);

%%sample the signal

n = 0:1:40

fs = 200

stem\_sig = sin(2\*pi\*(10/fs)\*n)+sin(2\*pi\*(50/fs)\*n)+sin(2\*pi\*(100/fs)\*n);

%%plot the original signal

figure("Name", "original");

plot(t, signal);

title("original signal")

%plot the sampled version

figure("Name", "sampled\_signal");

stem(stem\_sig);

title("sampled signal")

%%quantized

bit = 3;

y\_quantized = quantize(stem\_sig,bit);

figure("Name", "quantized");

stem(y\_quantized);

title("quantized")

%%reconstruct

recon = interp1(n/fs,y\_quantized,t,'spline');

figure("Name", "recon");

plot(recon);

title("reconstracted")

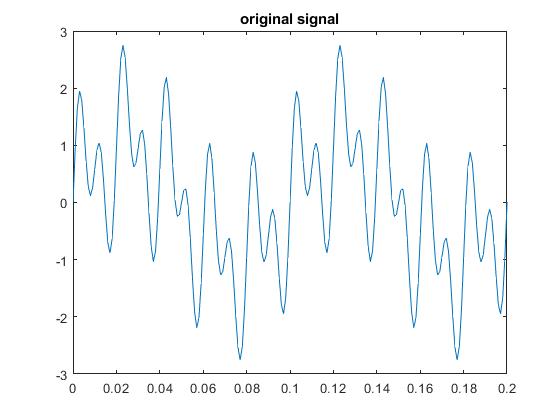
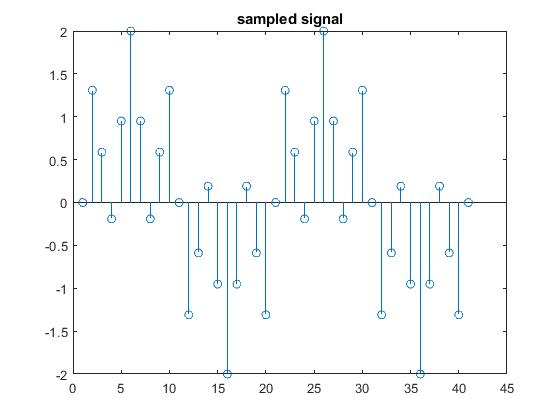
%%SQNR

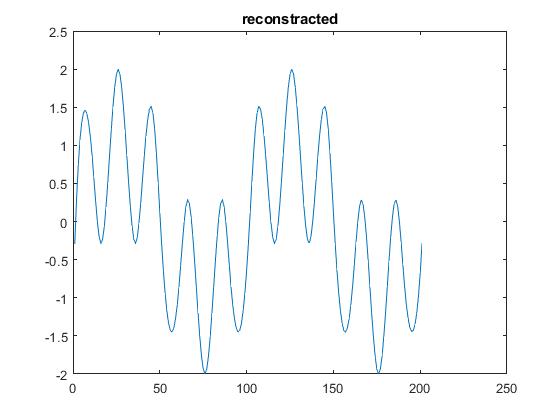
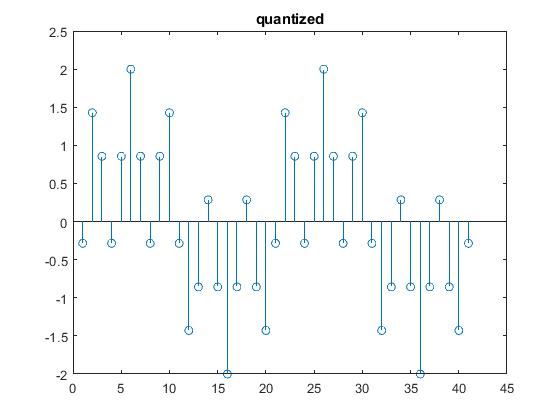
SQNR\_hard\_coded = signal\_noise\_ratio(stem\_sig, y\_quantized);

SQNR\_eq=1.76+6\*bit;

**Outputs**

SQNR in theory: 19.760000000000000

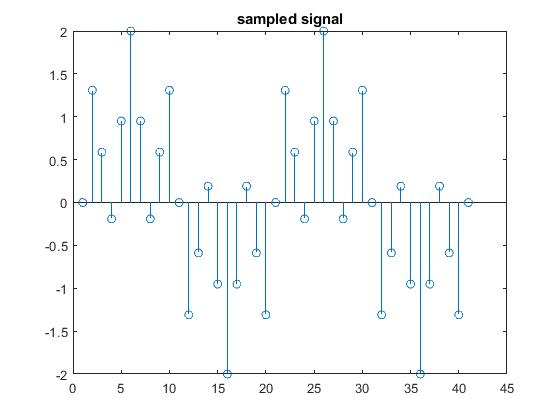
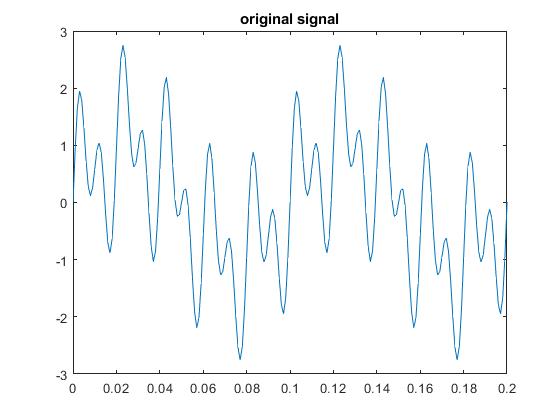
SQNR in Experiment: 15.067902412367973

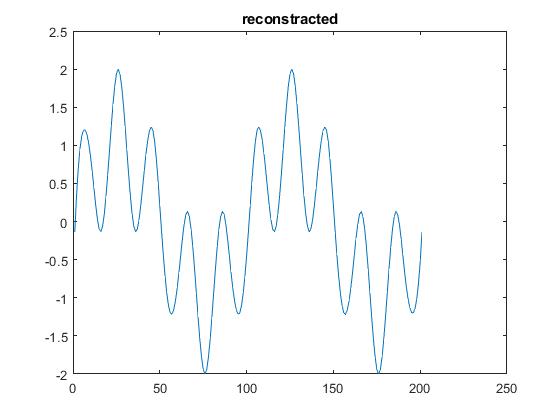
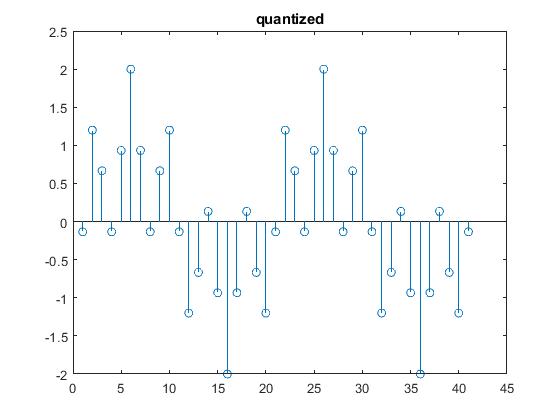


**Uniform 4 bit Quantizer :**

SQNR in theory: 25.7600000000000

SQNR in Experiment: 21.823594170954390

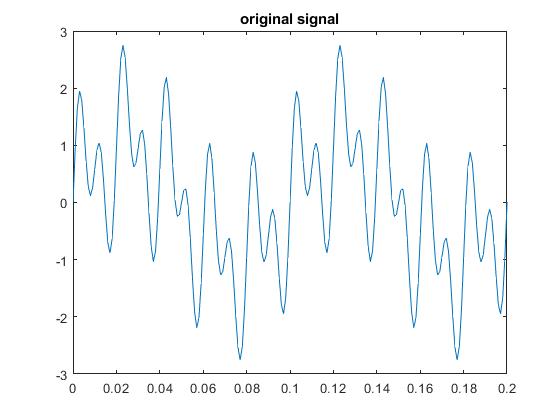
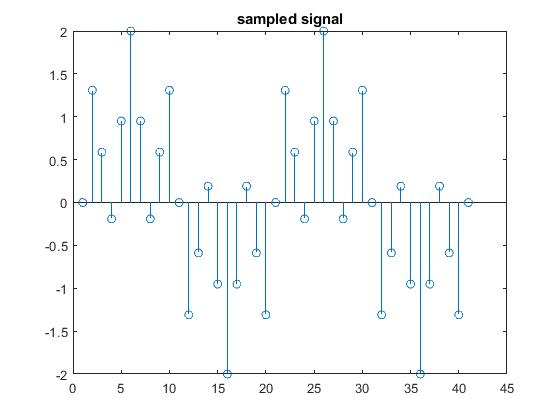
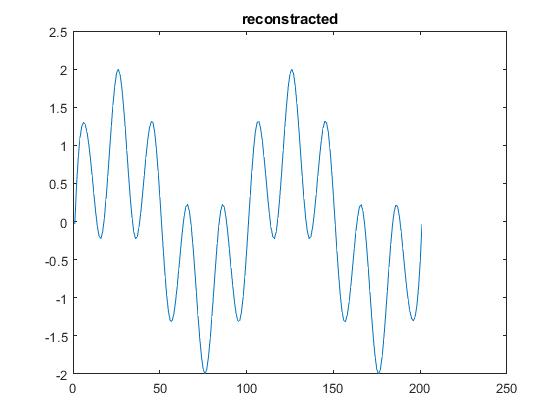
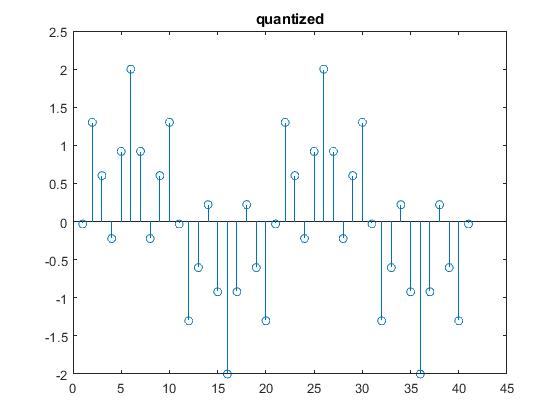
****

****

**Uniform 6-bit Quantiezer**

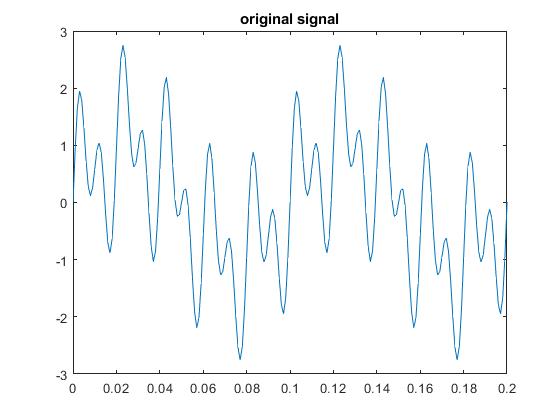
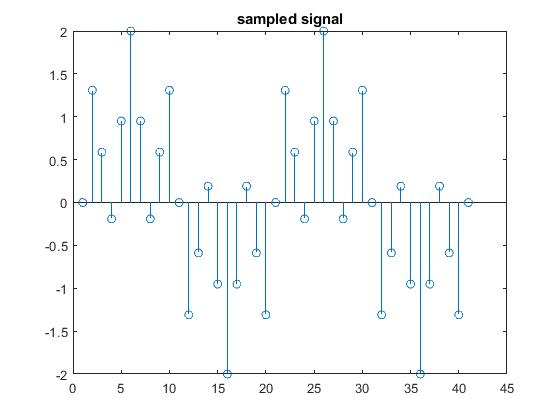
SQNR in theory: 37.760000000000000

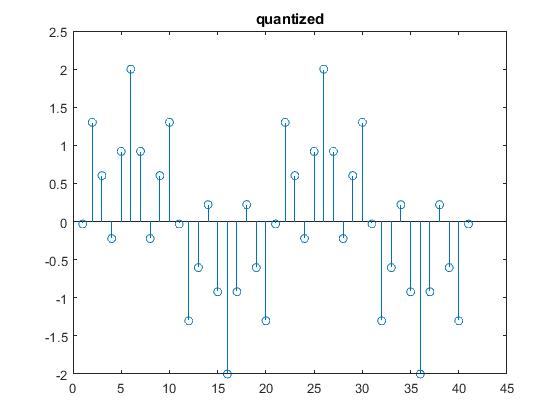
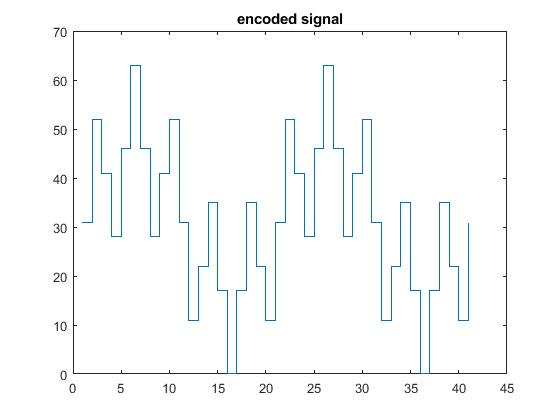
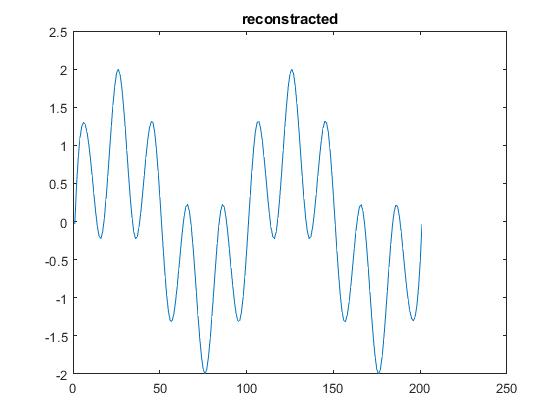
SQNR in experiment: 32.482166417366216



**Comment**: By increasing the number of bits, quantization noise power is decreasing due to increased number of quantization level and reconstructed signal will be more accurate.

**Part C:**





**Home task**

function sampled\_com = u\_law\_compressor(signal, u)

signal = double(signal)

sampled\_norm = signal/max(abs(signal))

figure("Name", "sampled\_norm")

stem(sampled\_norm)

sampled\_com=log(1+u\*abs(sampled\_norm))/log(1+u).\*sign(sampled\_norm);

function y\_quantized= quantize(x\_sampled,bit)

N=length(x\_sampled);

L=2^bit;

del=((max(x\_sampled)+0.0000000001)-min(x\_sampled))/(L-1);

for j=1:L

level(j)=min(x\_sampled)+del\*(j-1);

end

for i=1:N

for j=1:L-1

if(x\_sampled(i)>=level(j)&& x\_sampled(i)<level(j+1))

if(x\_sampled(i)>=level(j)+.5\*del)

y\_quantized(i)=level(j+1);

else

y\_quantized(i)=level(j);

end

end

end

end

y\_quantized

clc

clear all

close all

%%load the signal

t = 0:0.001:0.2;

signal = sin(2\*pi\*10\*t)+sin(2\*pi\*50\*t)+sin(2\*pi\*100\*t);

%%sample the signal

n = 0:1:40

fs = 200

stem\_sig = sin(2\*pi\*(10/fs)\*n)+sin(2\*pi\*(50/fs)\*n)+sin(2\*pi\*(100/fs)\*n);

%%plot the original signal

figure("Name", "original");

plot(t, signal);

title("original signal")

%plot the sampled version

figure("Name", "sampled\_signal");

stem(stem\_sig);

title("sampled signal")

%u\_law\_compressor

u = 255

sampled\_com = u\_law\_compressor(stem\_sig, u)

figure("Name", "u\_law\_compressed");

stem(sampled\_com)

title("u law compressed")

%quantized

bit = 8;

y\_quantized = quantize(stem\_sig,bit);

figure("Name", "quantized");

stem(y\_quantized);

title("quantized")

%reconstruct

recon = interp1(n/fs,y\_quantized,t,'spline');

figure("Name", "recon");

plot(recon);

title("reconstracted")

%SQNR

SQNR\_hard\_coded = signal\_noise\_ratio(stem\_sig, y\_quantized);

SQNR\_eq=1.76+6\*bit;

%Encoding

N = 8;

encoded\_n = encoder(N, stem\_sig, bit, y\_quantized);

encod = encoded\_n;

**Output**

encod values =

'01111111'

'11010011'

'10100101'

'01110011'

'10111100'

'11111110'

'10111100'

'01110011'

'10100101'

'11010011'

'01111111'

'00101011'

'01011010'

'10001100'

'01000011'

'00000000'

'01000011'

'10001100'

'01011010'

'00101011'

'01111111'

'11010011'

'10100101'

'01110011'

'10111100'

'11111110'

'10111100'

'01110011'

'10100101'

'11010011'

'01111111'

'00101011'

'01011010'

'10001100'

'01000011'

'00000000'

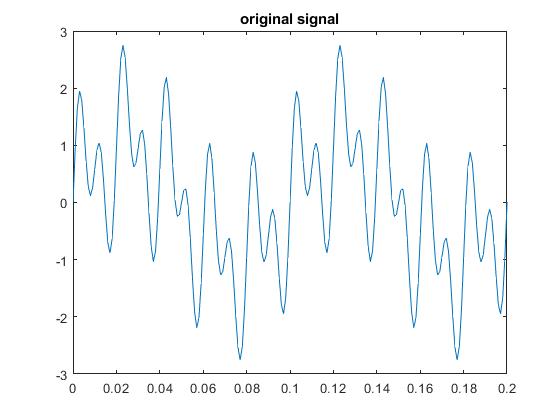
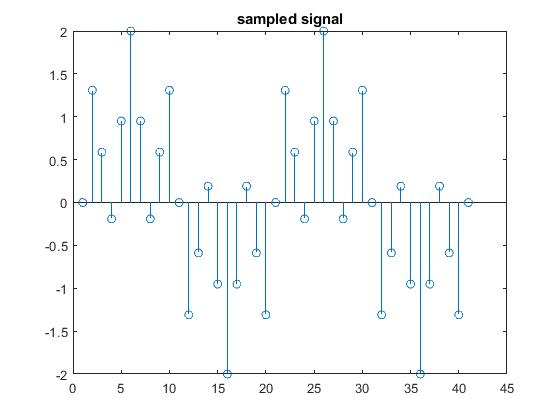
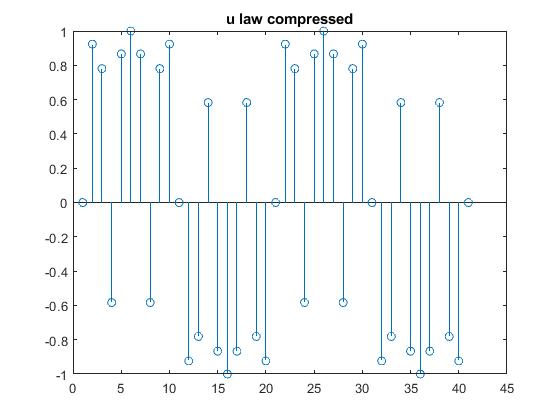
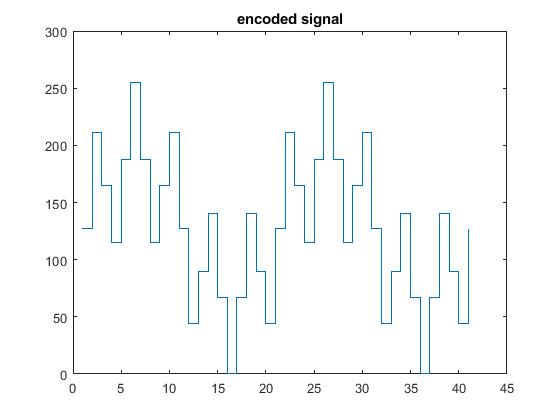
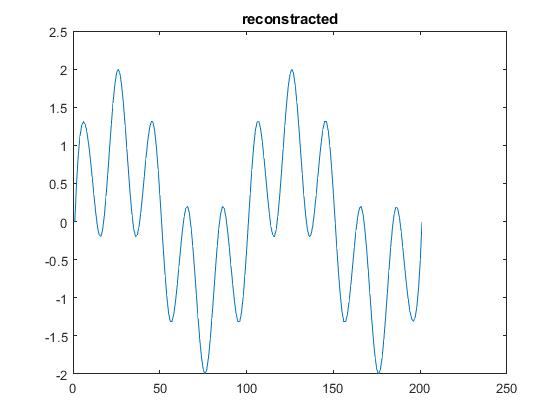
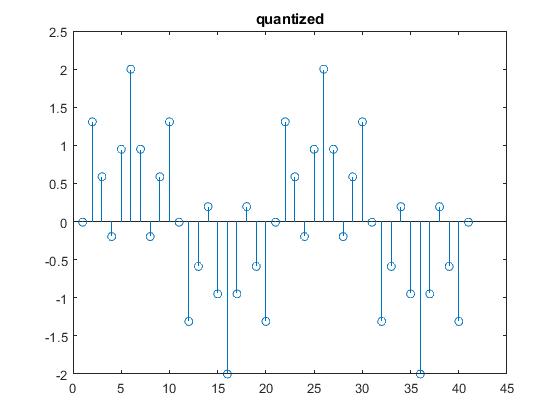
'01000011'

'10001100'

'01011010'

'00101011'

'01111111'

****

Codes : [link to github repository](https://github.com/tasnimislam/1706092_dsp_projects.git)