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");
stem(stem_sig);

%%reconstruct
recon = interp1(n/fs, stem_sig, t, 'spline');
figure("Name", "recon");
plot(recon);
```

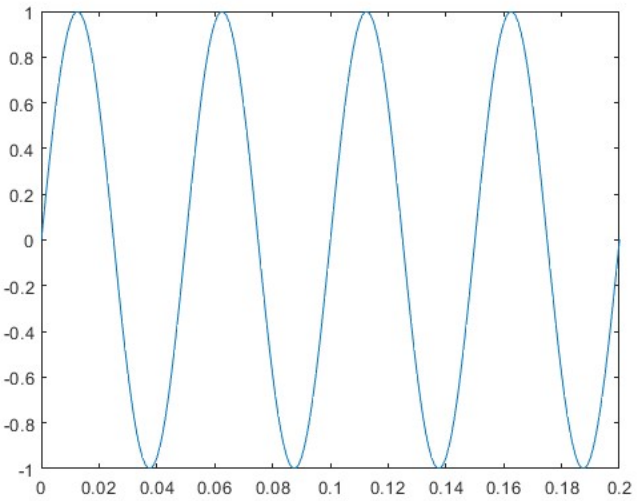


Figure 1: Original Signal

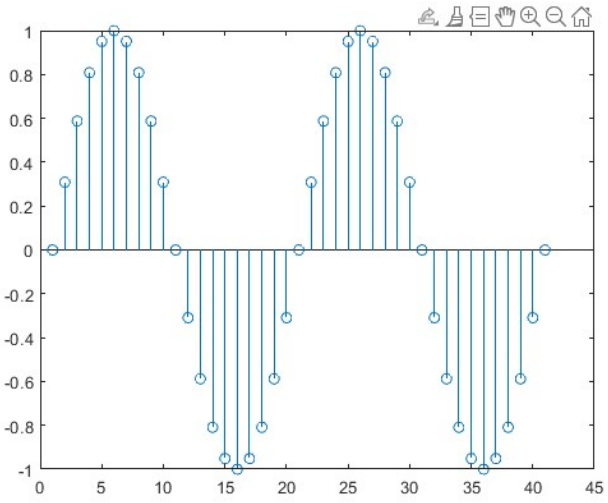


Figure 2: Sampled at 200 Hz

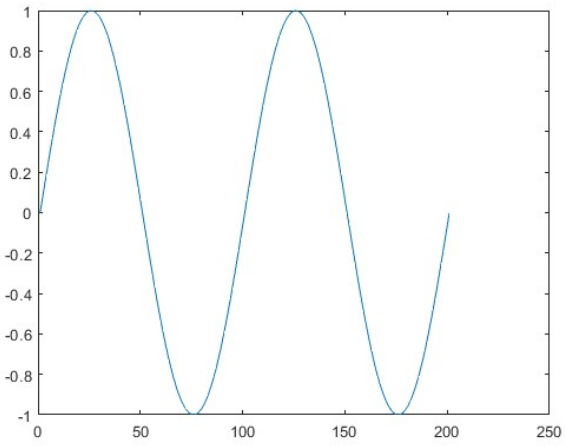


Figure 3: Reconstructed Signal after sampling

For 50 Hz

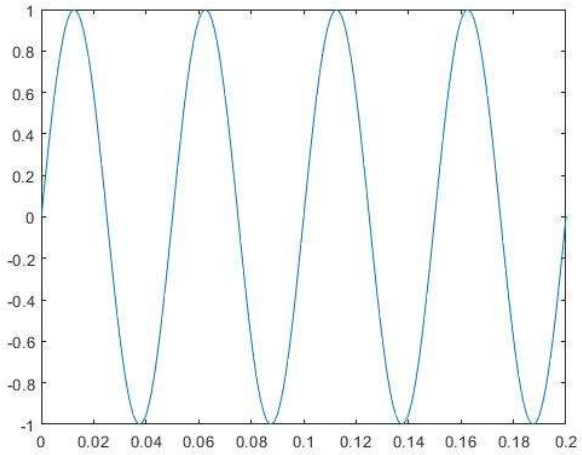


Figure 5: Original signal

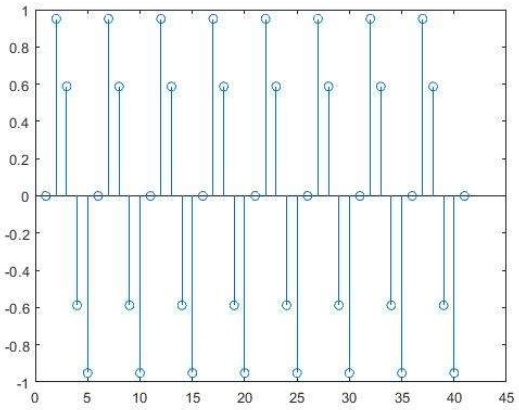


Figure 4: Sampled at 50 Hz

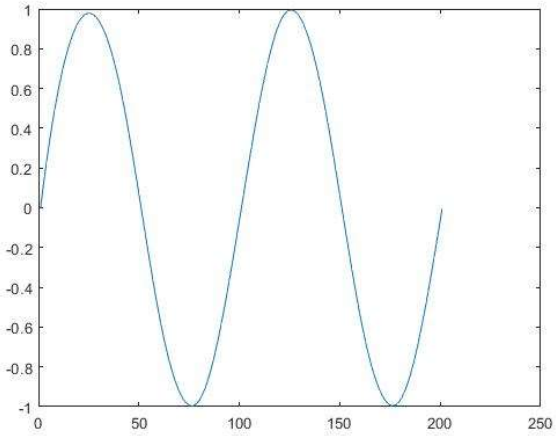


Figure 6: Reconstructed after sampling in 50 Hz

For 100 Hz

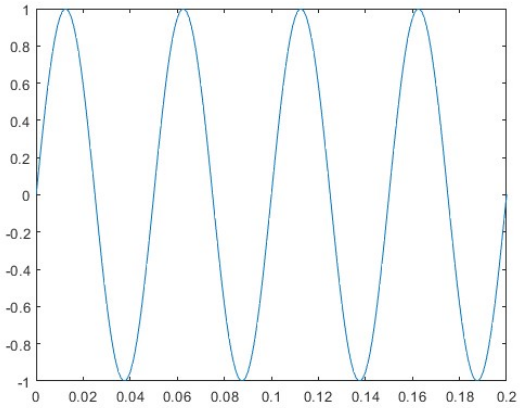


Figure 9: Original Signal

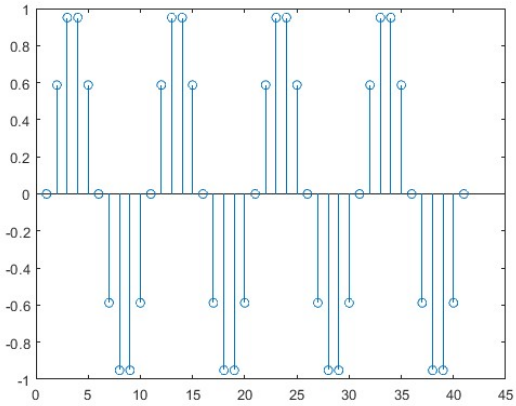


Figure 7: Sampled at 100 Hz

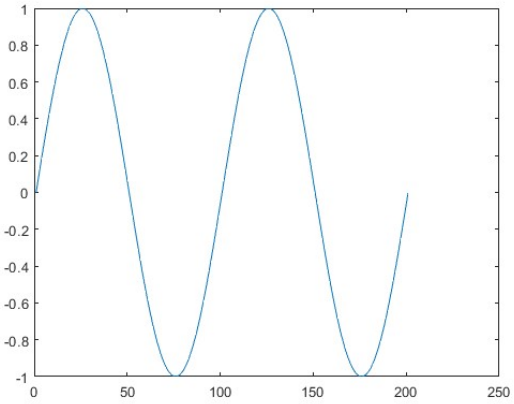


Figure 8: Reconstructed signal sampled at 100 Hz

For 10 Hz

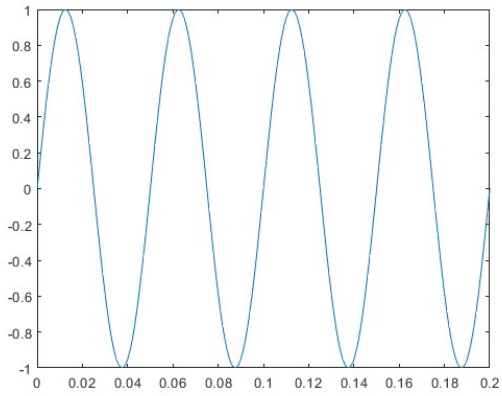


Figure 12: original Signal

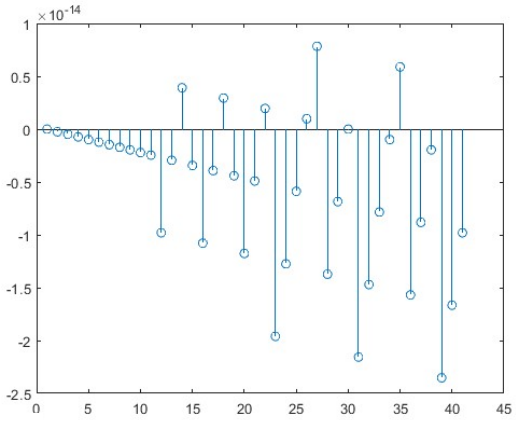


Figure 11: Sampled at 10 Hz

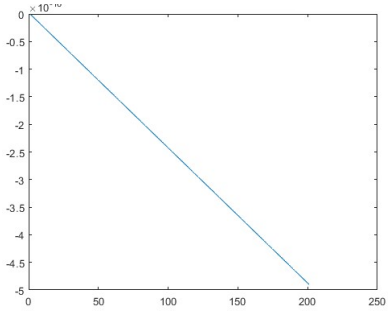


Figure 10: Reconstructed after sampling

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")
```

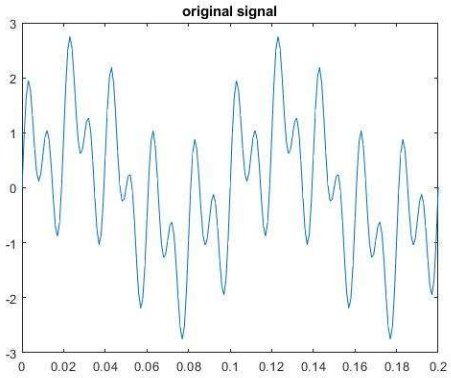


Figure 13: Plot of the original Signal

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
```

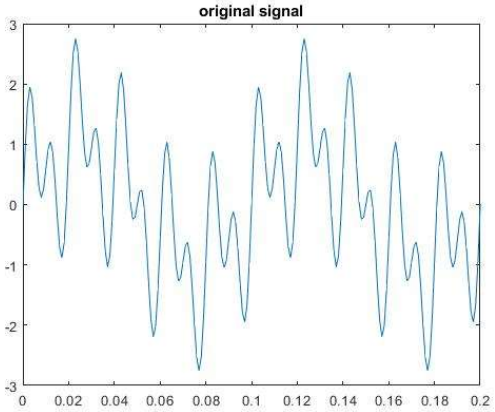
```
clc
clear all
close all
%%load the signal
t = 0:0.001:0.2;
signal = sin(2*pi*10*t)+sin(2*pi*50*t)+sin(2*pi*100*t);


%%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);
```

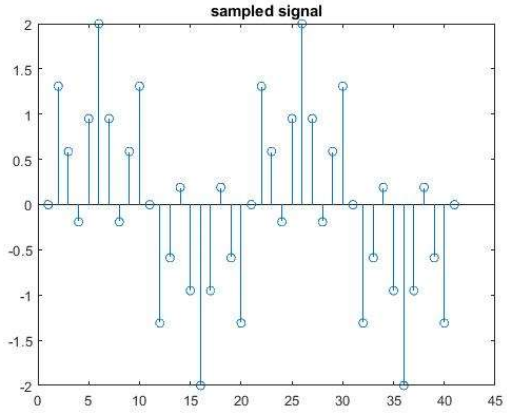
```
figure("Name", "quantized");
stem(y_quantized);
title("quantized")

%%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;
```

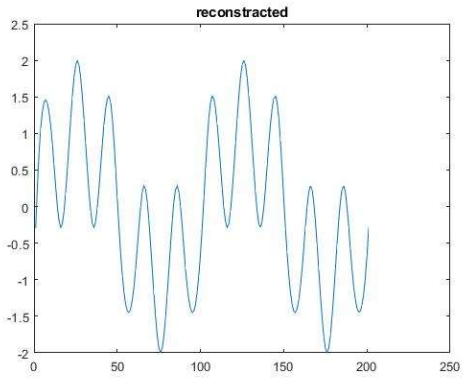
Outputs

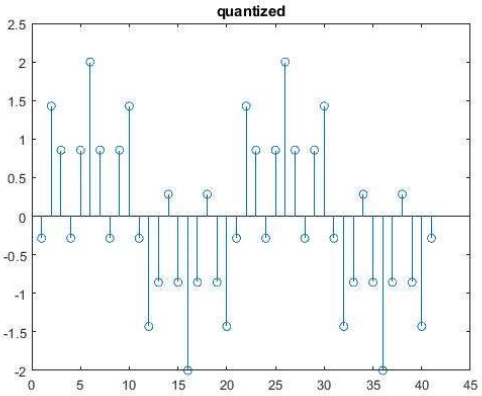
SQNR in theory: 19.760000000000000

SQNR in Experiment: 15.067902412367973





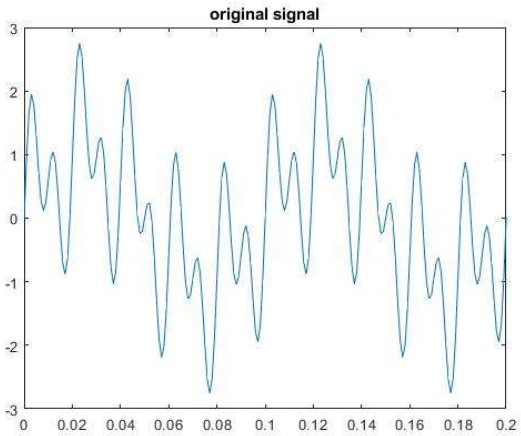


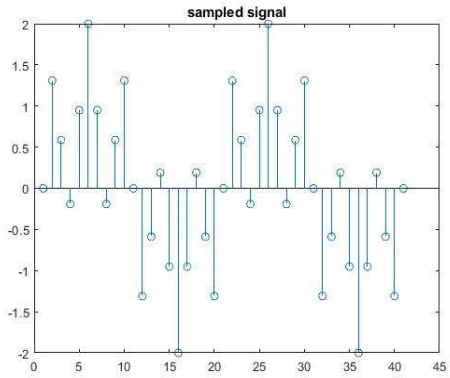


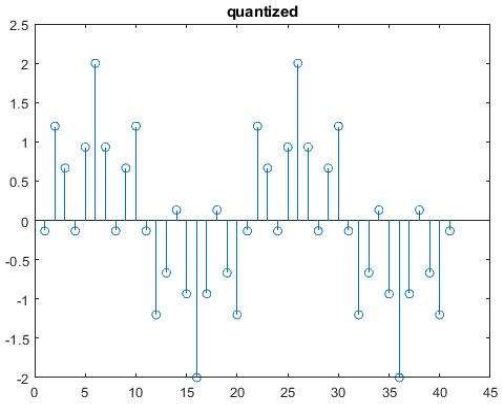
Uniform 4 bit Quantizer :

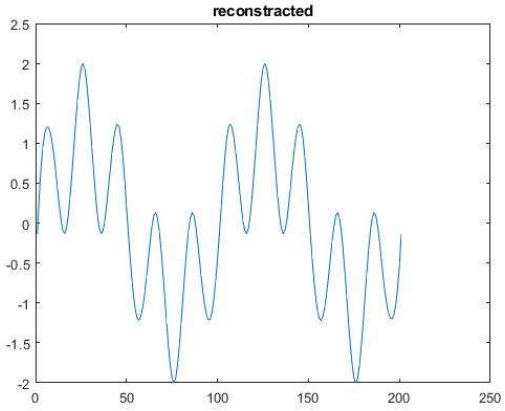
SQNR in theory: 25.7600000000000

SQNR in Experiment: 21.823594170954390





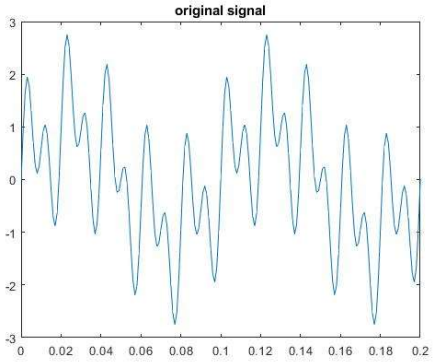


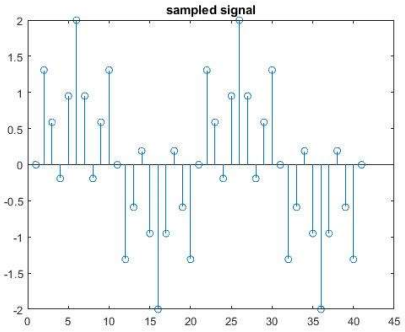


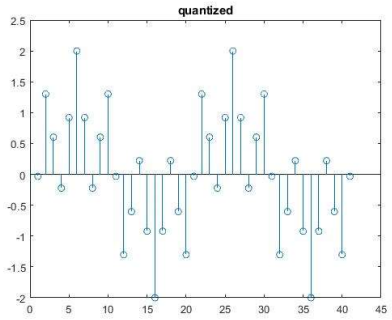
Uniform 6-bit Quantiezer

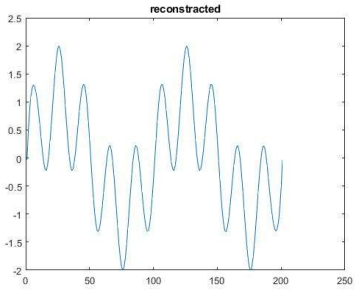
SQNR in theory: 37.760000000000000

SQNR in experiment: 32.482166417366216



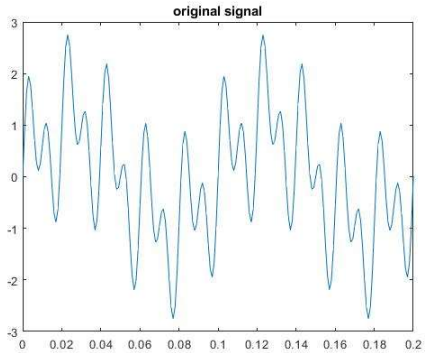


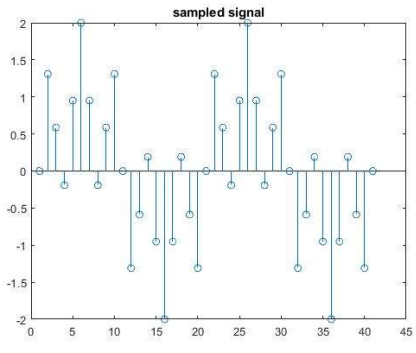


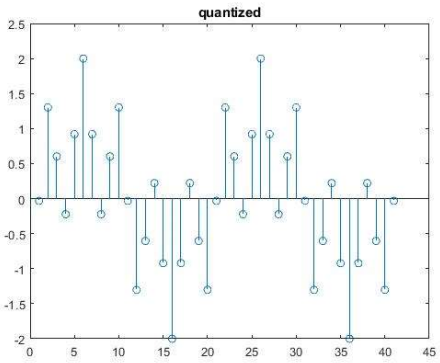


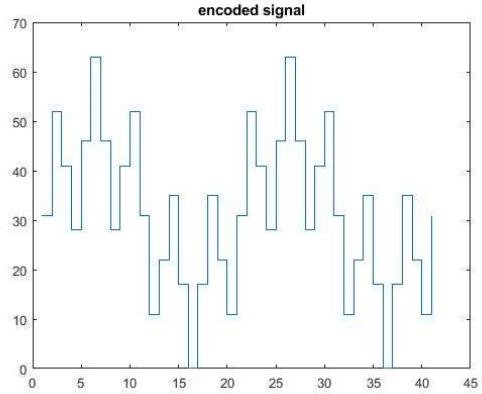
Comment: By increasing the number of bits, quantization noise power is decreasing due to increased number of quantization level and reconstructed signal will be more accurate.

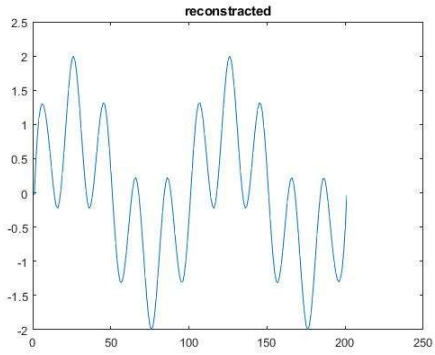
Part C:


original signal


sampled signal


quantized


encoded signal


reconstracted

Home task

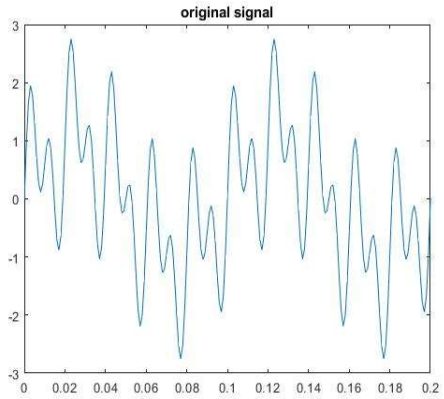
```
function sampled_com = u_law_compressor(signal, u)
signal = double(signal)
sampled_norm = signal/max(abs(signal))
figure("Name", "sampled_norm")
stem(sampled_norm)
sampled_com=log(1+u*abs(sampled_norm))/log(1+u).*sign(sampled_norm);


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);
```

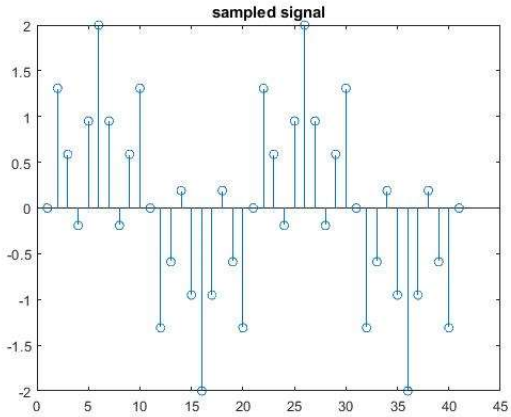
```
title("sampled signal")

%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;
```

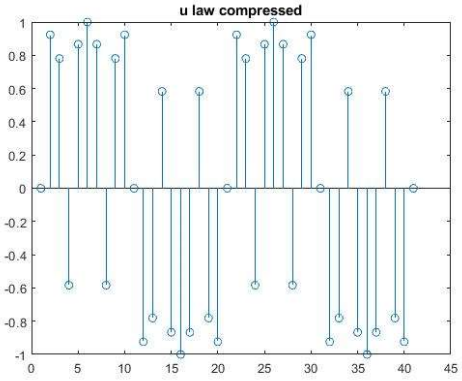
Output

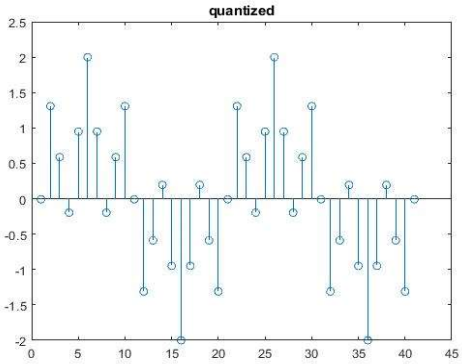
```
encod values =
     '01111111'
     '11010011'
     '10100101'
     '01110011'
     '10111100'
     '11111110'
     '10111100'
     '01110011'
     '10100101'
     '11010011'
     '01111111'
     '00101011'
     '01011010'
     '10001100'
     '01000011'
     '00000000'
     '01000011'
     '10001100'
```

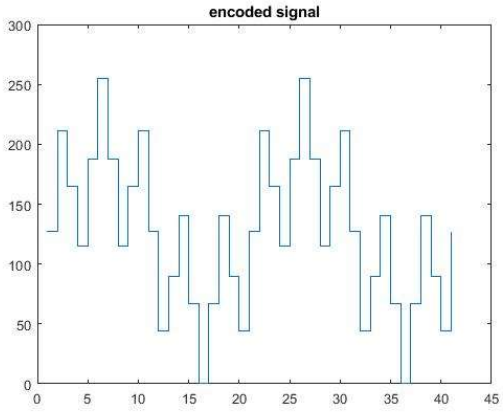
'01011010'
'00101011'
'01111111'
'11010011'
'10100101'
'01110011'
'10111100'
'11111110'
'10111100'
'01110011'
'10100101'
'11010011'
'01111111'
'00101011'
'01011010'
'10001100'
'01000011'
'00000000'
'01000011'
'10001100'
'01011010'
'00101011'
'01111111'

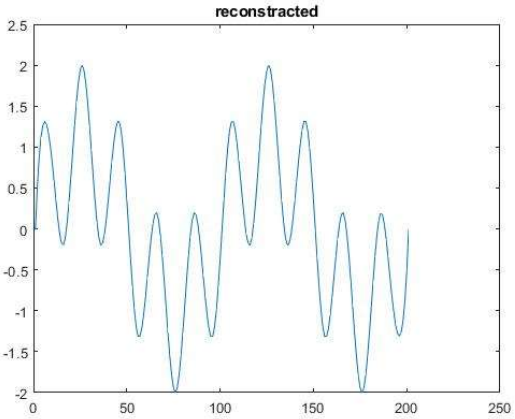












Codes : link to github repository