Cairo University
Faculty of Engineering

Department of Computer Engineering

ELC 325B – Spring 2023

Digital Communications

Assignment #1

Quantization

Submitted to

Dr. mai

Dr. hala

Eng. Mohamed Khaled

Submitted by

Name	Sec	BN
Norhan Reda Abdelwahed Ahmed	2	32
Hoda Gamal Hamouda Ismail	2	34

Contents

Part 1: Q1 4
Comment: 4
Part 2:Q2 6
Comment: 6
Part 3:Q3 8
Comment: 8
Part 4:Q4 9
Comment: 9
Part 5:Q5 10
Comment: 10
Part 6:Q6 11
Comment: 11
Index: Code 13

Figures

Figure 1 Fig 4
Figure 2 fig 6
Figure 3 Fig 8
Figure 4 Fig 9
Figure 5 Fig 10
Figure 6 Fig 11

Part 1:Q1

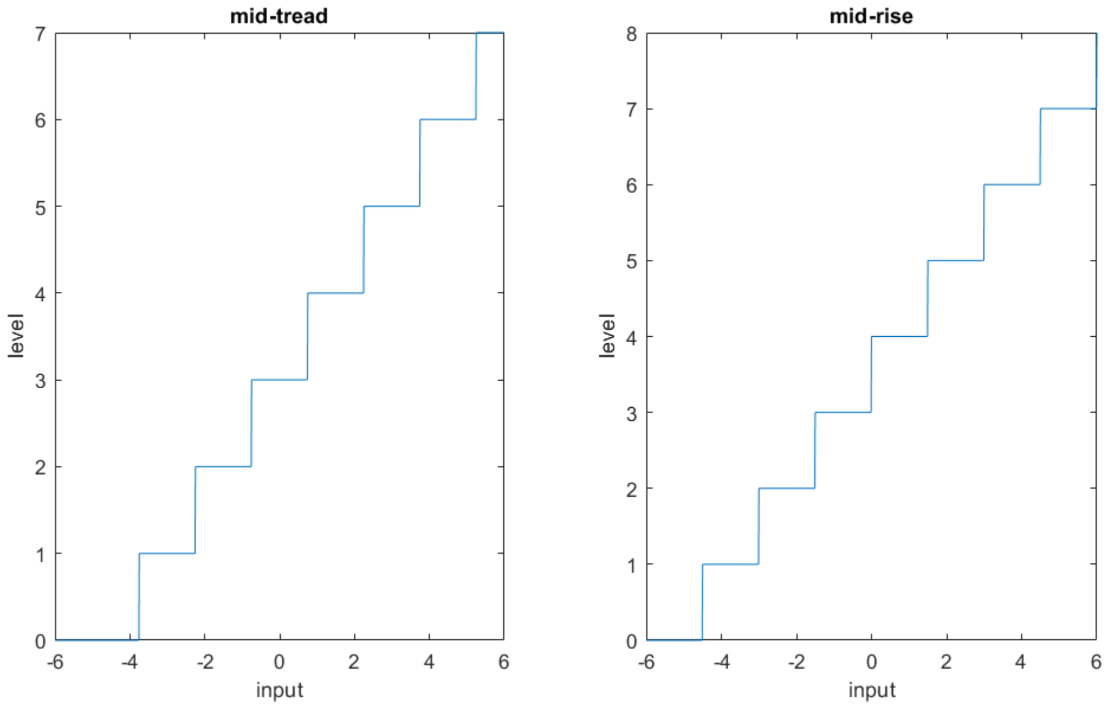

Figure 1 Fig

Comment:

The equation is used to map our data input to the correct level depending on m

if m=0 defines a “midrise” quantizer and if m=1 gives a “midtread” quantizer

```
% ----------------------Requirement 1---------------
function q_ind = UniformQuantizer(in_val, n_bits, xmax, m)
   levels = 2 ^ n_bits;
   delta = 2 * xmax / levels;
   q_ind = floor((in_val - ((m) * (delta / 2) - xmax)) / delta);
   q_ind(q_ind<0) = 0;
end
```

We get the distance between the input value and the absolute max value and divide by delta

for example: in mid rise

input	-4d:-3d	-3d:-2d	-2d:-d	-d:0	0:d	d:2d	2d:3d	3d:4d
level	0	1	2	3	4	5	6	7

level=(input value + absolute(max value))/delta

will get how the point far from the max

and get the floor to get the lower integer level

in mid tread:

same but with shifting by delta/2

level=(input value + absolute(max value)- (delta/2))/delta

input	-7d/2:-5d/2	-5d/2:-3d/2	-3d/2:-d/2	-d/2:d/2	d/2:3d/2	3d/2:5d/2	5d/2:7d/2	7d/2:9d/2
level	0	1	2	3	4	5	6	7

Part 2:Q2

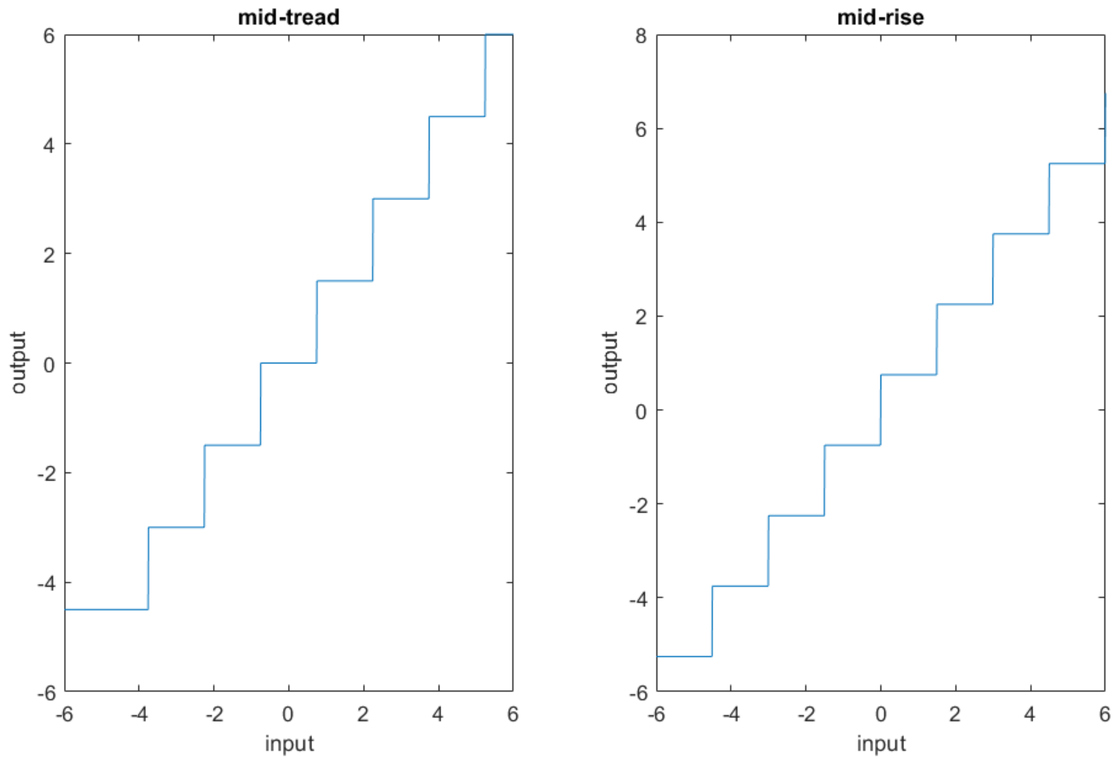

Figure 2 fig

Comment:

in this requirement we are performing the de-quantizer to map the quantized indices to the corresponding value level

```
% ---------------------Requirement 2----------------
function deq_val = UniformDequantizer(q_ind, n_bits, xmax, m)
   levels = 2 ^ n_bits;
   delta = 2 * xmax / levels;
   deq_val = ((q_ind) * delta) + ((m+1) * (delta / 2) - xmax);
end
```

since in quantization:

level=(value+max)/delta

so in dequantization:

value = level * delta - max

and in mid rise we add (delta/2)

value = level * delta - max + delta/2

and in mid tread we add (delta)

value = level * delta - max + delta

Part 3:Q3

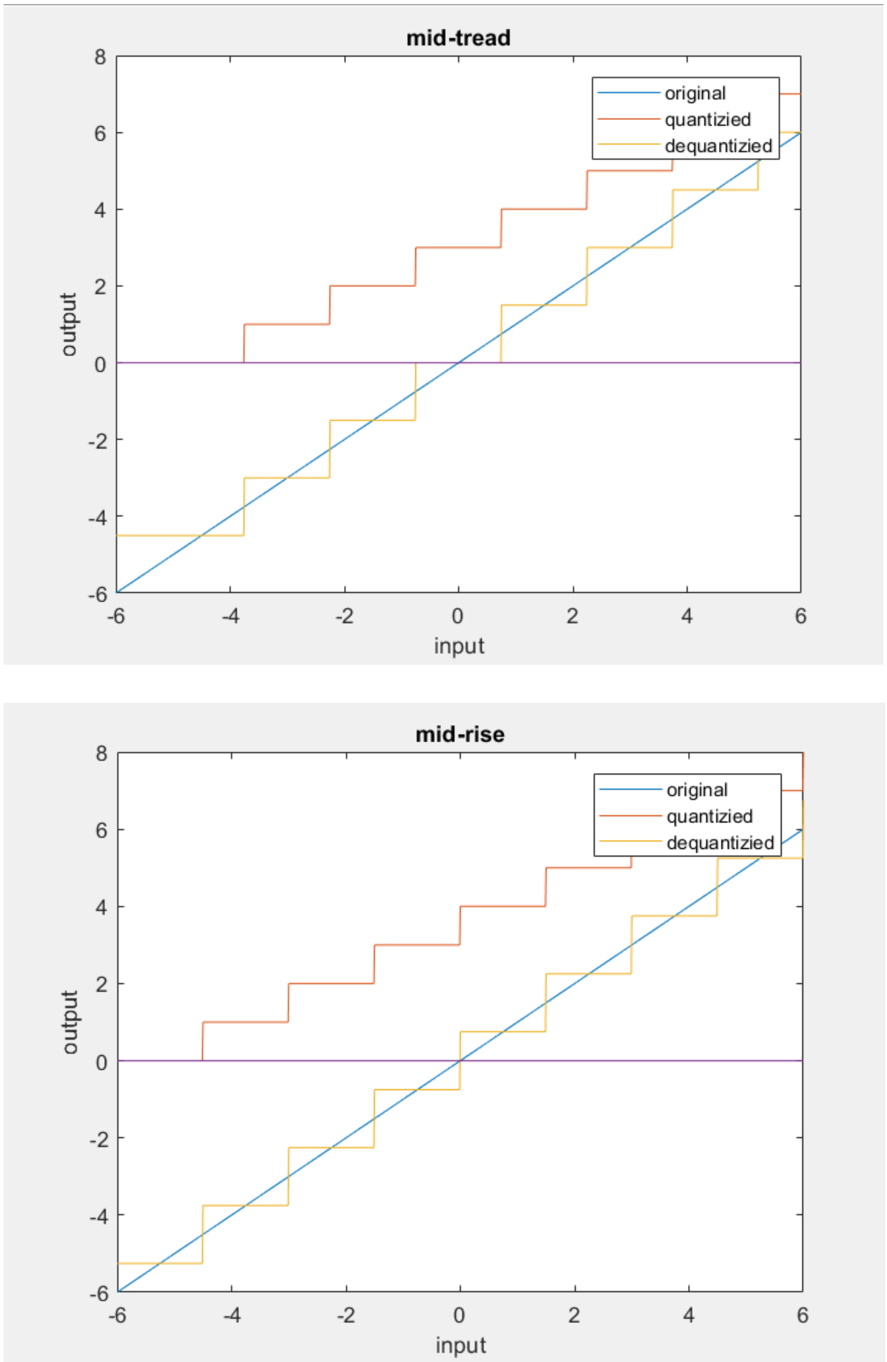

Figure 3 Fig

Comment:

here we are going to test our quantizer and de-quantizer in this figures

Part 4:Q4

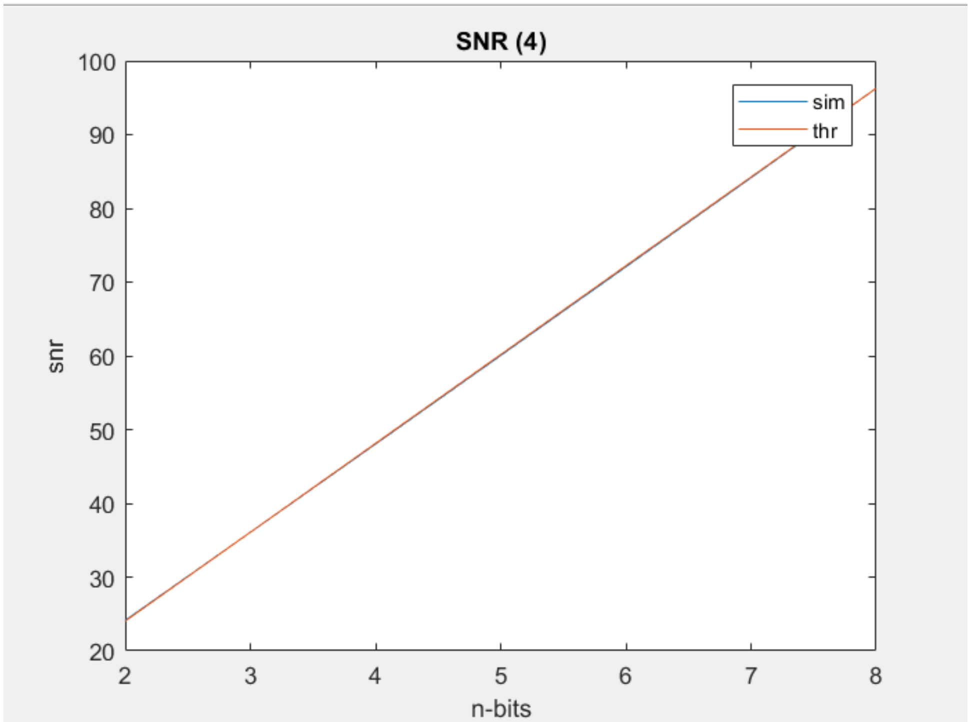

Figure 4 Fig

Comment:

When we use uniform data to quantize and de quantize, the difference between the snr simulation and snr theoretical is very small and almost zero for all numbers of bits.

We used these equations to calculate SNR:

Simulation SNR equation:

sim_snr = [sim_snr, mean(x.^2)/mean(error.^2)];

Theoretical SNR equation:

scale=(3*((2^n_bits)^2))/(xmax^2);
thr_snr = [thr_snr, scale*mean(x.^2)];

Part 5:Q5

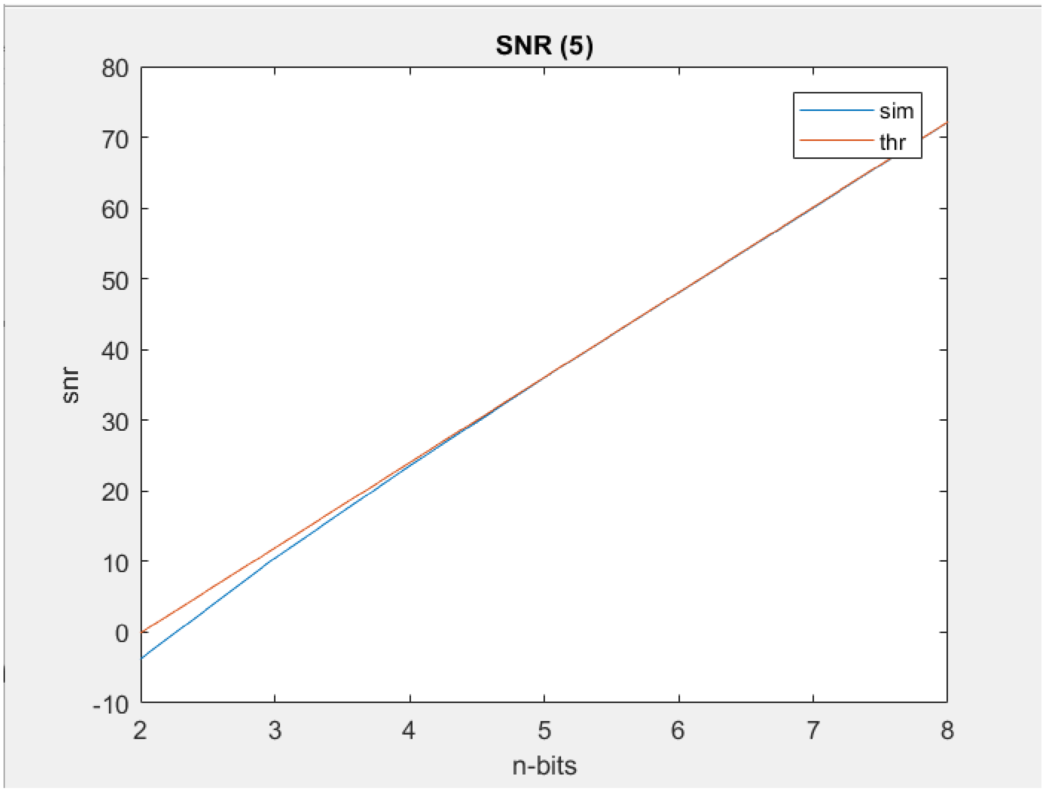

Figure 5 Fig

Comment:

when we use a random exponential data to the uniform quantizer and uniform

de- quantizer the difference between the snr simulation and snr theoretical is high at the small number of bits and the difference decreases when the number of bits increases.

Part 6:Q6

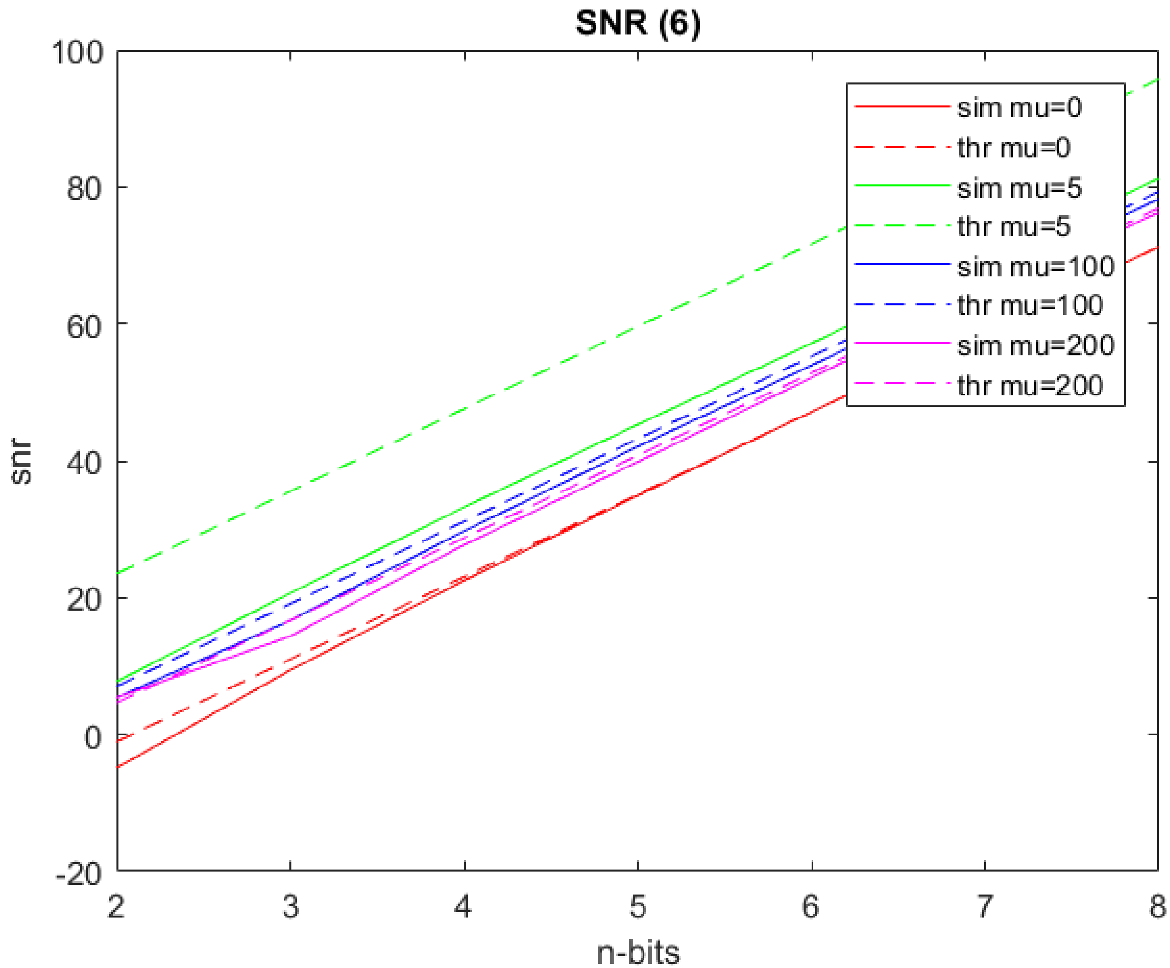

Figure 6 Fig

Comment:

here we are using the non uniform quantizer for random data to improve the graph in req 5 to make the difference between snr theoretical and simulation small

but it is depending on the value of mu chosen

it is very clear in the figure when we use mu for mu law quantizer when mu increases the difference between theoretical and simulation decreases and at mu=0 it is equivalent to requirement 5(uniform)

The steps that we are doing :

1-generate random data with random sign(and the random sign with prob 1/2)

2-compress the data before entering the quantizer using this equation

```
function y = Compression(x, u, sign)
y=sign .* (log(1+u*abs(x))/log(1+u));
end
```

3-enter the uniform quantizer

4-enter the uniform de-quantizer

5-expansion the data using this equation

```
function y = Expansion(x, u, sign)
y= sign .*(((1+u).^abs(x)-1)/u);
end
```

We used these equations to calculate SNR:

Same previous equation for simulation SNR.

for theoretical SNR:

```
scale=(3*((2^n_bits)^2));
thr_snr = [thr_snr,scale/((log(1+mu))^2)];
```

Index: Code

```
clear;
close all;

% -----------------Requirement 3------------------
x =-6:0.01:6;
n_bits = 3;
xmax = 6;

% mid-tread
m = 1;
y=UniformQuantizer(x,n_bits,xmax,m);
y_deq=UniformDequantizer(y,n_bits, xmax, m);
plotFourOutput(x,x,x,y,x,y_deq,x,x.*0,"mid-tread","input","output")
legend('original','quantizied','dequantizied');

% mid-rise
m=0;
y=UniformQuantizer(x,n_bits,xmax,m);
y_deq=UniformDequantizer(y,n_bits,xmax,m);
plotFourOutput(x,x,x,y,x,y_deq,x,x.*0,"mid-rise","input","output")
legend('original','quantizied','dequantizied');


% ------------------Requirement 4---------------------
sim_snr=[];
thr_snr=[];
x = random('Uniform',-5,5,1,10000);
xmax=max(abs(x));
m=0;

for n_bits= 2:1:8
y=UniformQuantizer(x,n_bits,xmax,m);
y_deq = UniformDequantizer(y, n_bits, xmax, m);

error=abs(x-y_deq);
sim_snr = [sim_snr, mean(x.^2)/mean(error.^2)];
scale=(3*((2^n_bits)^2))/(xmax^2);
thr_snr = [thr_snr, scale*mean(x.^2)];
```

```
end

n_bits= 2:1:8;
plotTwoOutput(n_bits,mag2db(sim_snr),n_bits,mag2db(thr_snr),"SNR(4)","n-bits","snr")
legend('sim','thr');

% ------------------Requirement 5---------------------
sim_snr=[];
thr_snr=[];
size = [1 10000];
x_exp = exprnd(1,size);
sign = (randi([0,1],size)*2)-1;
x = x_exp.*sign;
xmax=max(abs(x));
m=0;

for n_bits= 2:1:8
y=UniformQuantizer(x,n_bits,xmax,m);
y_deq = UniformDequantizer(y, n_bits, xmax, m);

error=abs(x-y_deq);
sim_snr = [sim_snr, mean(x.^2)/mean(error.^2)];
scale=(3*((2^n_bits)^2))/(xmax^2);
thr_snr = [thr_snr, scale*mean(x.^2)];
end
n_bits= 2:1:8;
plotTwoOutput(n_bits,mag2db(sim_snr),n_bits,mag2db(thr_snr),"SNR(5)","n-bits","snr")
legend('sim','thr');

% ------------------Requirement 6---------------------

figure();
x_norm=x/xmax;
c={"r",'g','b','m'};
i=1;

for mu=[0, 5, 100,200]
sim_snr=[];
thr_snr=[];

if(mu~=0)
```

```
  x_comp = Compression(x_norm,mu,sign);
else
  x_comp=x;
end

ymax=max(abs(x_comp));

for n_bits= 2:1:8
y = UniformQuantizer(x_comp, n_bits,ymax, m);
y_deq = UniformDequantizer(y, n_bits, ymax, m);

if(mu~=0)
y_expand = Expansion(y_deq,mu,sign);
y_deq = y_expand *xmax;
end

error=abs(x-y_deq);
sim_snr = [sim_snr, mean(x.^2)/mean(error.^2)];
if(mu~=0)
scale=(3*((2^n_bits)^2));
thr_snr = [thr_snr,scale/((log(1+mu))^2)];
else
scale=(3*((2^n_bits)^2))/(xmax^2);
thr_snr = [thr_snr, scale*mean(x.^2)];
end
end

n_bits= 2:1:8;
% plotTwoOutput(n_bits,mag2db(sim_snr),n_bits,mag2db(thr_snr),strcat('SNR(6) mu= ',
num2str(mu)),"n-bits","snr")
% legend('sim','thr');

plot(n_bits,mag2db(sim_snr),'-','color',c{i})
hold on
plot(n_bits,mag2db(thr_snr),'--','color',c{i})
title("SNR (6)")
xlabel("n-bits")
ylabel("snr")

i=i+1;
end
```

```
legend('sim mu=0','thr mu=0','sim mu=5','thr mu=5','sim mu=100','thr mu=100','sim mu=200','thr mu=200');



% -----------------------Requirement 1----------------
function q_ind = UniformQuantizer(in_val, n_bits, xmax, m)
   levels = 2 ^ n_bits;
   delta = 2 * xmax / levels;
   q_ind = floor((in_val - ((m) * (delta / 2) - xmax)) / delta);
   q_ind(q_ind<0) = 0;
end

% ----------------------Requirement 2-----------------
function deq_val = UniformDequantizer(q_ind, n_bits, xmax, m)
   levels = 2 ^ n_bits;
   delta = 2 * xmax / levels;
   deq_val = ((q_ind) * delta) + ((m+1) * (delta / 2) - xmax);
end

% ------------------ plotting functions --------------
function plotTwoOutput(x1,y1,x2,y2,label,labelx,labely)
figure();
plot(x1,y1)
hold on
plot(x2,y2)
hold off
title(strcat(label, ""))
xlabel(strcat(labelx, ""))
ylabel(strcat(labely, ""))
end

function plotFourOutput(x1,y1,x2,y2,x3,y3,x4,y4,label,labelx,labely)
figure();
plot(x1,y1)
hold on
plot(x2,y2)
plot(x3,y3)
plot(x4,y4)
hold off
```

```
title(strcat(label, ""))
xlabel(strcat(labelx, ""))
ylabel(strcat(labely, ""))
end

% ------------------- For Requirement 6---------------------
function y = Compression(x, u, sign)
y=sign .* (log(1+u*abs(x))/log(1+u));
end

function y = Expansion(x, u, sign)
y= sign .*(((1+u).^abs(x)-1)/u);
end
```