Cairo University
Faculty of Engineering

Department of Computer Engineering

0

ELC 325B – Spring 2023

Digital Communications

Assignment #1

Quantization

Submitted to

Dr. Mai

Dr. Hala

Eng. Mohamed Khaled

Submitted by

Name	Sec	BN
Karim Mahmoud Kamal	2	10
Mohamed Walid Fathy	2	20

Contents

Figures

Part 1: Uniform Quantizer

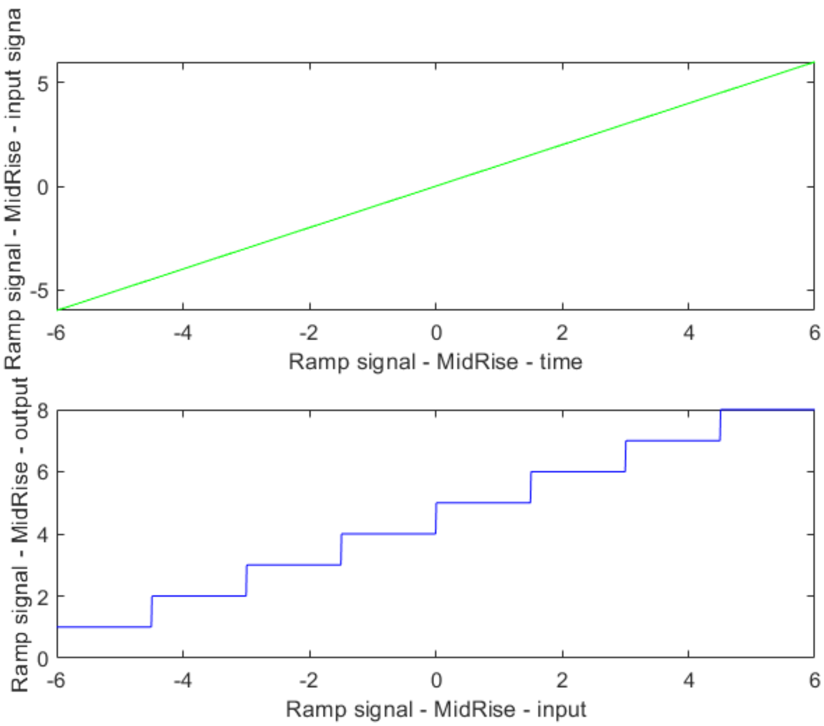


*Figure 1 **Midrise** Uniform Quantizer*

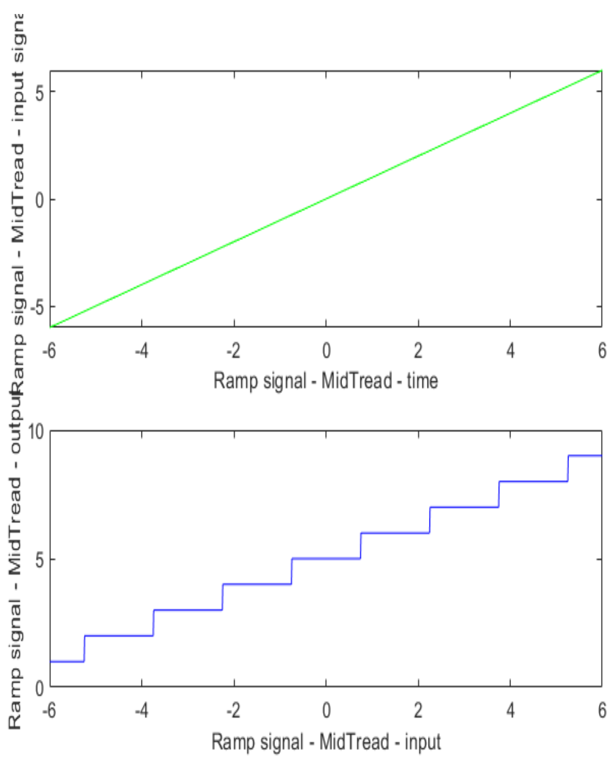


*Figure 2 **Midtread** Uniform Quantizer - **9 Levels***

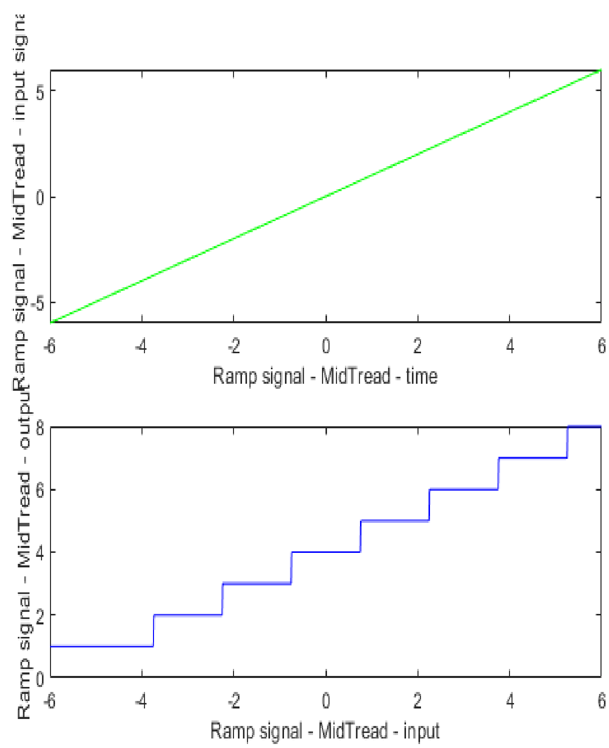


*Figure 3 **Midtread** Uniform Quantizer - 8 Levels*

Comment:

The output of the Quantizer is the indices of the quantization levels starting from 1.

Fig.1 shows the **Midrise (m = 0)** Uniform Quantizer, it quantizes the signal with number of levels = 2^number_of_bits.

The Quantization is done by approximating each point to its nearest level.

The output signal is rising at zero in time.

Fig.2 and Fig.3 shows the **Midtread (m=1)** Uniform Quantizer, it also quantizes the signal with number of levels = 2^number_of_bits.

The Quantization is done by approximating each point to its nearest level.

The output signal is parallel to the x-axis at zero in time.

inputs: number of bits = 3 , xmax = 6 , x=-6:0.01:6

- **Note on Quantization Levels Choice:**
    - Fig.1: MidRise with **8** Levels (4 below zero and 4 above zero).
    - Fig.2: MidTread with **9** Levels (4 below zero and 4 above zero).
    - Fig.3: MidTread with **8** Levels (we assume that positive levels are more than negative levels by one as there exists a level at zero).

Part 2: Uniform De-Quantizer

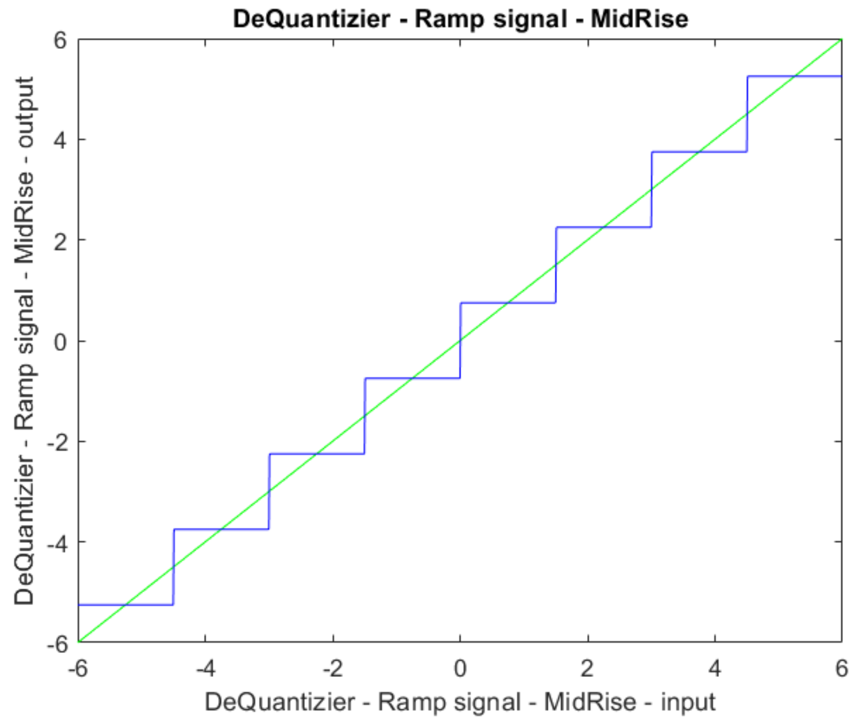
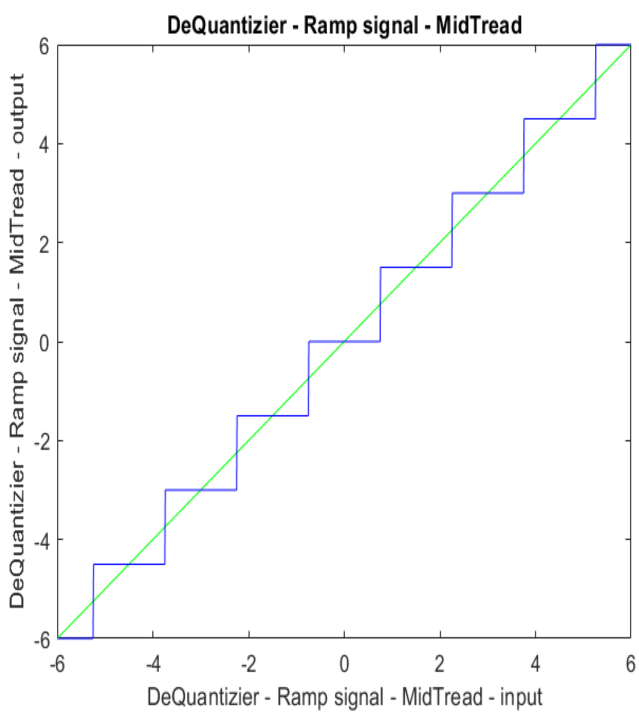


Figure 4 ***Midrise*** *Uniform De-Quantizer*



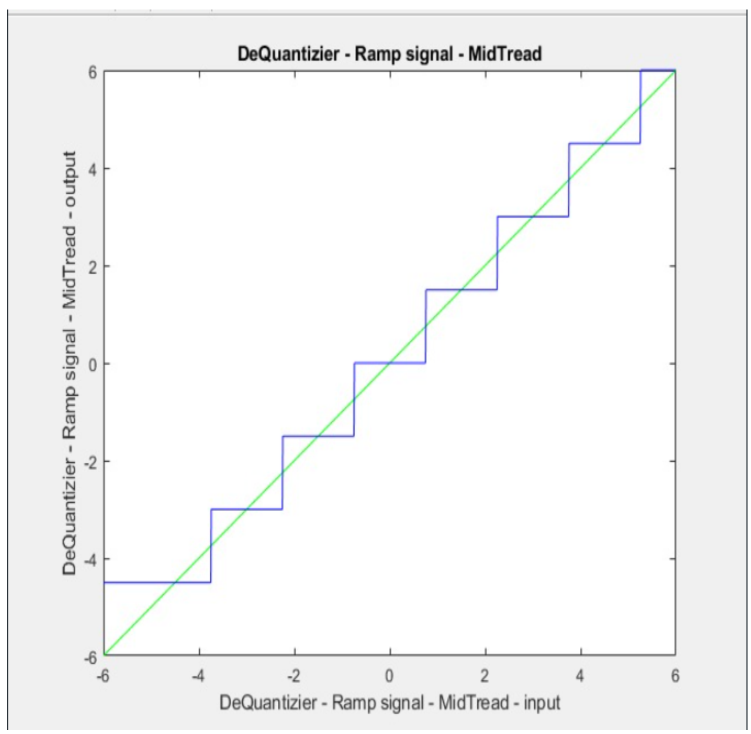


Figure 5 ***Midtread*** *Uniform De-Quantizer -* ***9*** *levels*

Figure 6 ***Midtread*** *Uniform De-Quantizer* ***8*** *levels*

Comment:

The De-Quantizer output is mapping the indices of quantized levels to the values between min and max voltage levels.

Fig.4 shows the MidRise uniform De-quantizer.

Fig.5 & Fig.6 show the Midtread uniform De-quantizer.

The only difference is how the output signal looks around zero (in Midrise the quantizied level is vertical at zero & in Midtread the quantizied level is horizontalat zero).

The explanation is the same as with the quantizers.

- Fig.4: **MidRise** with **8** Levels (4 below zero and 4 above zero). Levels starts from -5.25 and ends at 5.25 with step size = 2* x_max / number of levels = 1.5.
- Fig.5: **MidTread** with **9** Levels (4 below zero and 4 above zero). Levels starts from -6 and ends at 6 with step size = 2* x_max / number of levels = 1.5.
- Fig.6: **MidTread** with **8** Levels (we assume that positive levels are more than negative levels by one as there exist a level at zero). Levels starts from -4.5 and ends at 6 with step size = 2* x_max / number of levels = 1.5.

Part 3: Quantizing/Dequantizing with ramp signal

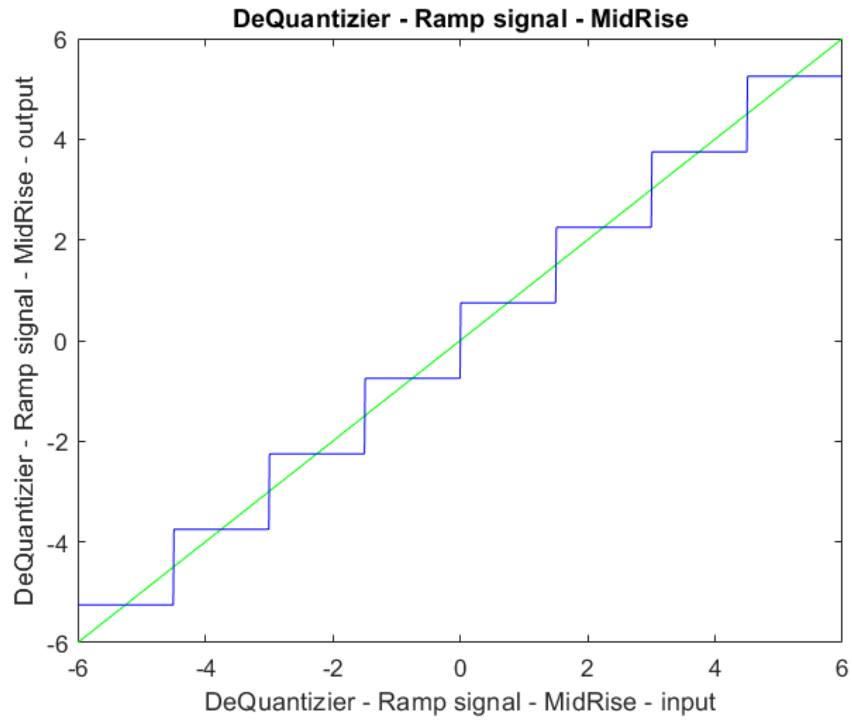
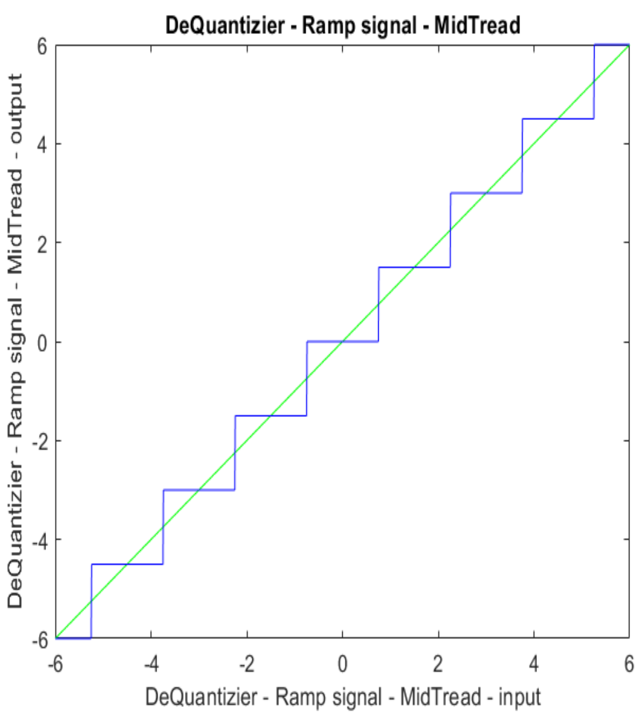


*Figure 4 **Midrise** Uniform De-Quantizer*



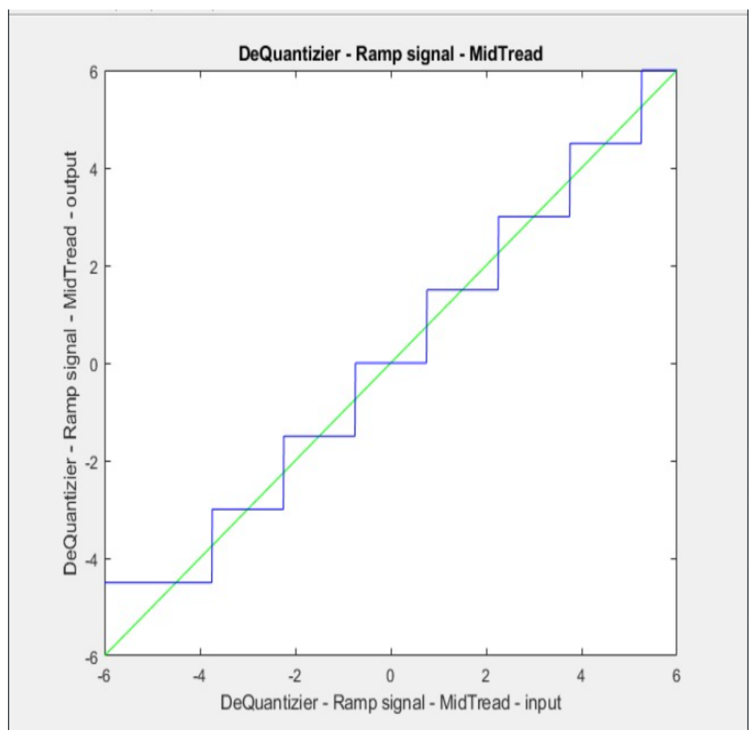


*Figure 5 **Midtread** Uniform De-Quantizer - **9** levels*

*Figure 6 **Midtread** Uniform De-Quantizer **8** levels*

Comment:

The De-Quantizer output is mapping the indices of quantized levels to the values between min and max voltage levels.

Fig.4 shows the MidRise uniform De-quantizer.

Fig.5 & Fig.6 show the Midtread uniform De-quantizer.

The only difference is how the output signal looks around zero (in Midrise the quantizied level is vertical at zero & in Midtread the quantizied level is horizontalat zero).

The explanation is the same as with the quantizers.

- Fig.4: **MidRise** with **8** Levels (4 below zero and 4 above zero). Levels starts from -5.25 and ends at 5.25 with step size = 2* x_max / number of levels = 1.5.
- Fig.5: **MidTread** with **9** Levels (4 below zero and 4 above zero). Levels starts from -6 and ends at 6 with step size = 2* x_max / number of levels = 1.5.
- Fig.6: **MidTread** with **8** Levels (we assume that positive levels are more than negative levels by one as there exist a level at zero). Levels starts from -4.5 and ends at 6 with step size = 2* x_max / number of levels = 1.5.

Part 4: Random Input Signal Test

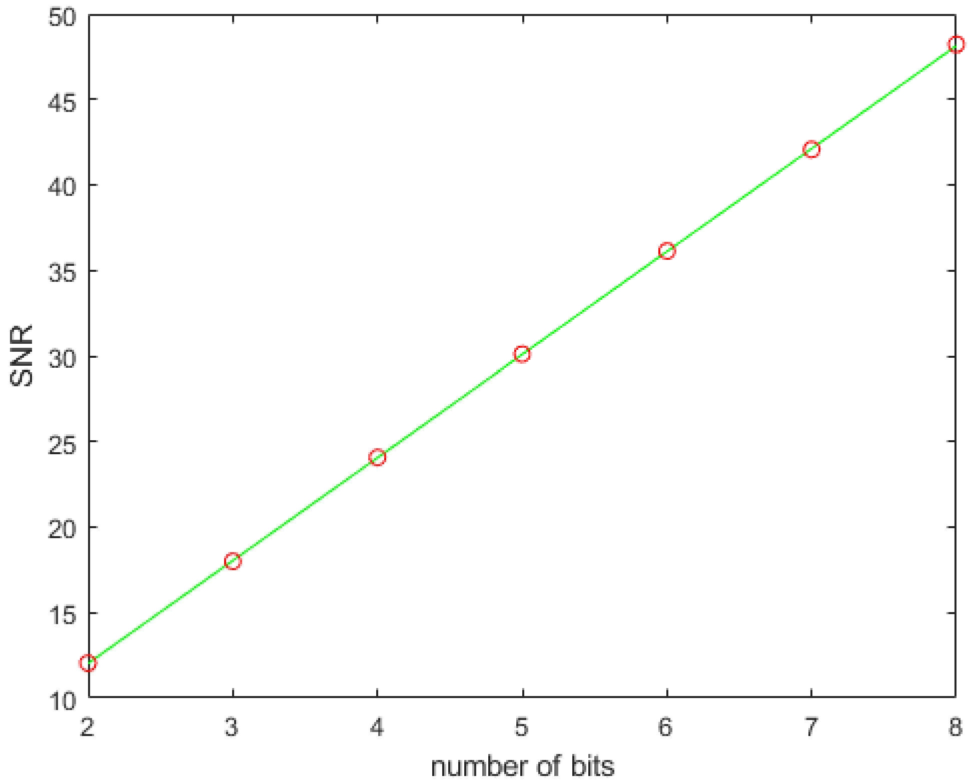


Figure 7 Random Input Signal - Theoretical and Simulation SNR

Comment:

- The Red Points are the Simulation Results.
- The Green Line is the Theoretical Results.

lower_bound = -5 , upper_bound = 5, number of random variables = 10000;

- we generate n i.i.d. uniform random variables between lower_bound and upper_bound using this equation ⇒ **lower_bound + (upper_bound - lower_bound).*rand(number_of_R_V ,1)**

- Case: **midrise**(m = 0)

- We Calculate **SNR_Simulation** = power of the input signal / power of the error.
- **The power** = sum of samples of input signal / number of samples ==> calculated using mean function.
- Calculate **SNR_Theoritically** = ((3*number of levels*number of level) / (max level*max level)) * power of the input signal.

we found that **SNR_Simulation & SNR_Theoritically are identical (shown in graph)**

Part 5: Uniform Quantizer with a Non-Uniform Random Input

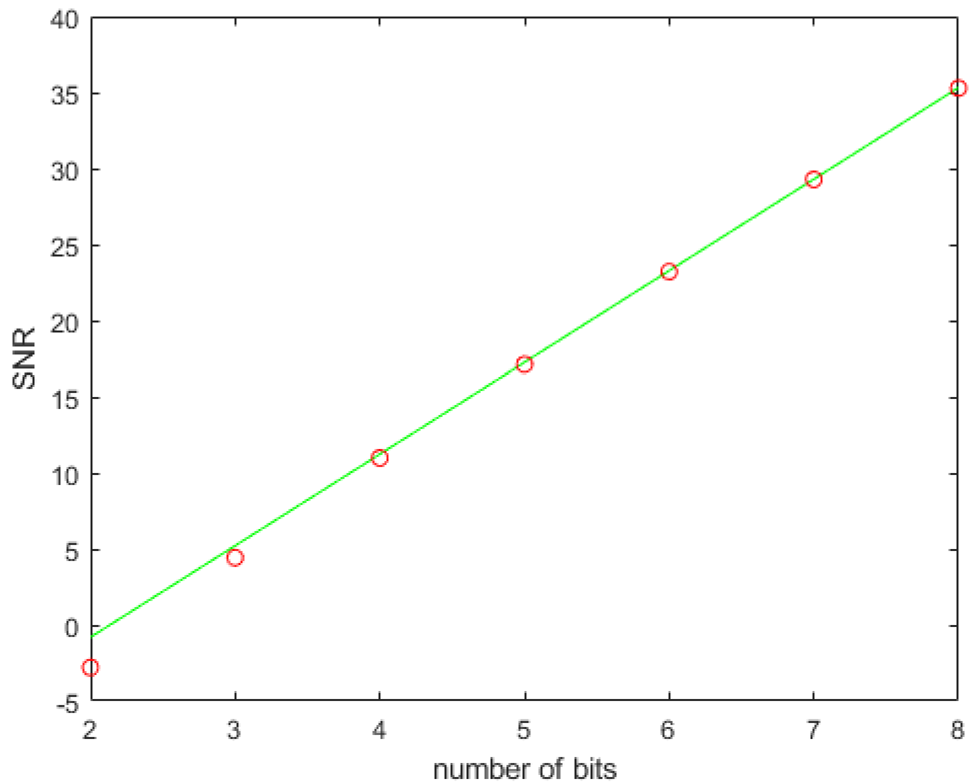


Figure 8 Uniform Quantizer on a Non-Uniform Random Input - Theoretical and Simulation SNR

Comment:

- The Red Points are the Simulation Results.
- The Green Line is the Theoretical Results.
- Equal probabilities between positive and negative with 0.5 each:
- sign = 2 * randi([0 1] , 1 , number_of_R_V) - 1;
- Exponential distribution using exprnd
- magnitude = exprnd(1, 1, number_of_R_V);
- **Midrise**: m = 0;
- Calculate **SNR_Simulation** = power of the input signal / power of the error
- **The power** = sum of samples of input signal / number of samples ==> calculated using mean function
- Calculate **SNR_Theoritically** = ((3*number of levels*number of level) / (max level*max level)) * power of the input signal

we found that **SNR_Simulation** & **SNR_Theoritically** are **identical** (shown in graph) starting from number of bits = **5** till the **end** and there is a **little difference** when number of bits = **2,3,4**

Part 6: Non Uniform Random Signal Using Non Uniform Mu

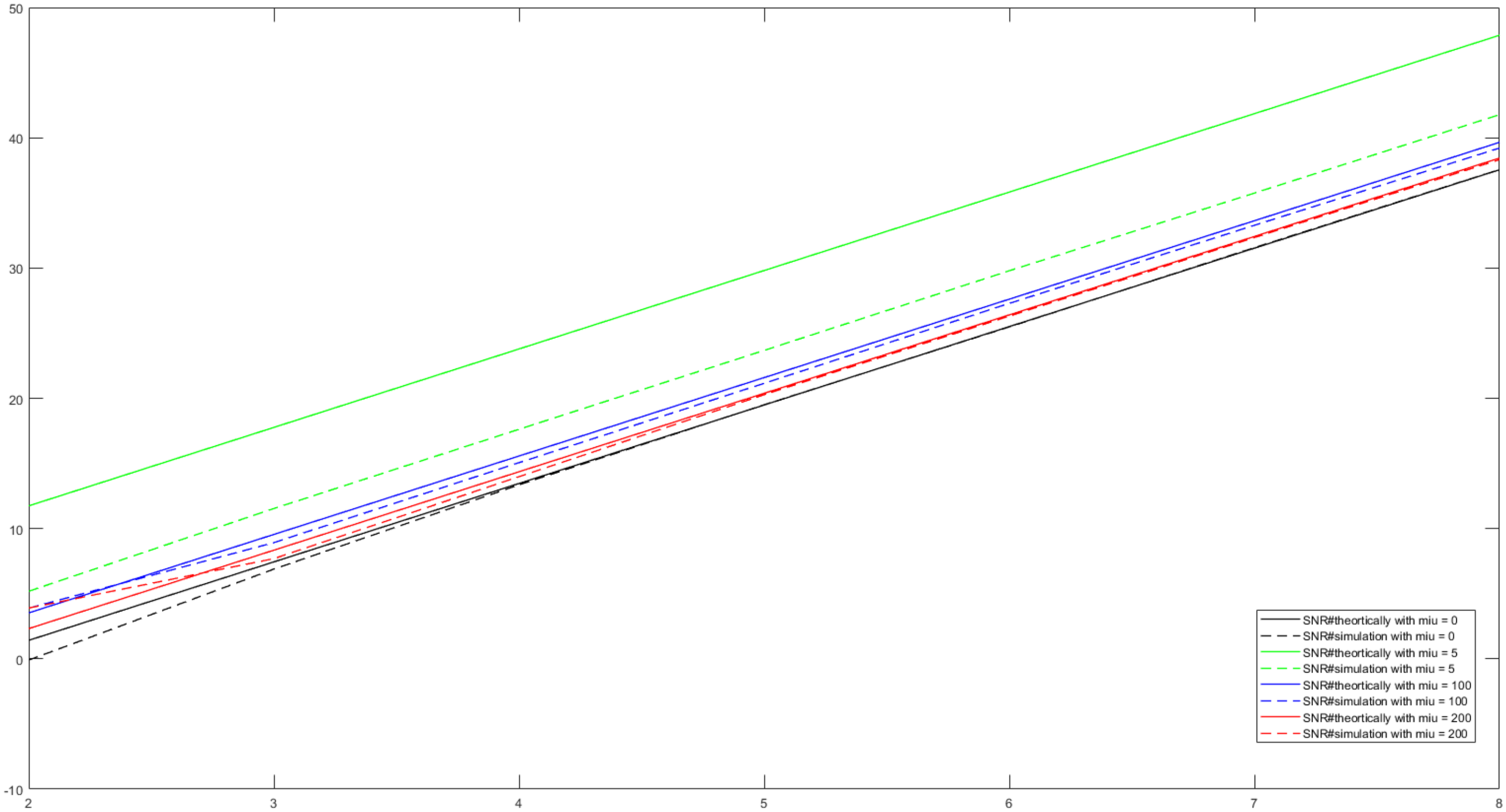


Figure 9 Non Uniform Random Signal Using Non Uniform Mu

Comment:

Equal probabilities between postitive and negative with 0.5 each:

- sign = 2 * randi([0 1] , 1 , number_of_R_V) - 1;
- Exponential distribution using exprnd
- magnitude = exprnd(1, 1, number_of_R_V);
- Midrise: m = 0;
- Calculate **SNR_Simulation** = power of the input signal / power of the error
- **The power** = sum of samples of input signal / number of samples ==> calculated using mean function
- Calculate **SNR_Theoritically** = ((3*number of levels*number of level) / (max level*max level)) * power of the input signal

<u>steps:</u>

1. normalize the input signal
2. compress the function
3. do quantization and dequantization on the compressed signal
4. expand the signal
5. denormalize the signal
6. calculate SNR theoretically and simulation

<u>Note:</u>

We notice that the 2 graphs in uniform quantization (part 5) and when mu = 0 (part 6) are identical both theoretically and simulation.

Index: The Code

```
clc;

close all;

clear all;


%--------------------------------Uniform Quantizier----------------------%


%implement a ramp signal

x = -6:0.01:6;

n_bits = 3;

xmax = 6;

m = 0;


y_before_quantization=x;


%--------------------------------Uniform Quantizier - Mid Rise----------------------%


%quantize the signal using midRise

y_after_quantization_midRise = UniformQuantizer(x, n_bits, xmax, m);


figure(1)

subplot(2,1,1)

title('Ramp signal - MidRise - time domain')
```

```
plot(x,y_before_quantization,'-g')
xlabel('Ramp signal - MidRise - time')
ylabel('Ramp signal - MidRise - input signal')

subplot(2,1,2)
title('Ramp signal - MidRise')
plot(x,y_after_quantization_midRise,'-b')
xlabel('Ramp signal - MidRise - input')
ylabel('Ramp signal - MidRise - output')



%--------------------------------Uniform Quantizier - Mid Tread----------------------%

%quantize the signal using midtread
m=1;
y_after_quantization_midTread= UniformQuantizer(x, n_bits, xmax, m);

figure(2)
subplot(2,1,1)
title('Ramp signal - MidTread - time domain')
plot(x,y_before_quantization,'-g')
xlabel('Ramp signal - MidTread - time')
ylabel('Ramp signal - MidTread - input signal')
```

```
subplot(2,1,2)

title('Ramp signal - MidTread')

plot(x,y_after_quantization_midTread,'-b')

xlabel('Ramp signal - MidTread - input')

ylabel('Ramp signal - MidTread - output')


%-------------------------------------Uniform DeQuantizier---------------------------------%


%implement a ramp signal

x = -6:0.01:6;

n_bits = 3;

xmax = 6;

m = 0;


y_before_dequantization=x;


%--------------------------------Uniform DeQuantizier - Mid Rise----------------------%


%quantize the signal using midRise

y_after_dequantization_midRise = UniformDequantizer(y_after_quantization_midRise, n_bits, xmax, m);
```

```
figure(3)

subplot(1,1,1)

title('DeQuantizier - Ramp signal - MidRise - time domain')

plot(x,y_before_dequantization,'-g')

xlabel('DeQuantizier - Ramp signal - MidRise - time')

ylabel('DeQuantizier - Ramp signal - MidRise - input signal')

hold on;

%subplot(2,1,2)

title('DeQuantizier - Ramp signal - MidRise')

plot(x,y_after_dequantization_midRise,'-b')

xlabel('DeQuantizier - Ramp signal - MidRise - input')

ylabel('DeQuantizier - Ramp signal - MidRise - output')



%--------------------------------Uniform DeQuantizier - Mid
Tread----------------------%


%Dequantize the signal using midtread

m=1;

y_after_dequantization_midTread=
UniformDequantizer(y_after_quantization_midTread, n_bits, xmax, m);


figure(4)

subplot(1,1,1)
```

```
title('DeQuantizier - Ramp signal - MidTread - time domain')

plot(x,y_before_dequantization,'-g')

xlabel('DeQuantizier - Ramp signal - MidTread - time')

ylabel('DeQuantizier - Ramp signal - MidTread - input signal')

hold on;

%subplot(2,1,2)

title('DeQuantizier - Ramp signal - MidTread')

plot(x,y_after_dequantization_midTread,'-b')

xlabel('DeQuantizier - Ramp signal - MidTread - input')

ylabel('DeQuantizier - Ramp signal - MidTread - output')


%--------------------------------Random Signal - Requirement #4----------------------%


lower_bound = -5;

upper_bound = 5;

number_of_R_V = 10000;


% generate n i.i.d. uniform random variables between lower_bound and
upper_bound

x = lower_bound + ( upper_bound - lower_bound ).*rand( number_of_R_V ,1) ;

y = x;


n_bits = 2:1:8;

xmax = 5;
```

```
%midrise
m = 0;

% intial value for the two arrays
SNR_simulation = zeros(size(n_bits));
SNR_theortically = zeros(size(n_bits));

% number of levels
number_of_levels = 2 .^ n_bits;

%calculate the power of the input signal
power = mean(x .^ 2);

for i = 1:length(n_bits)

   %midrise - Quantizer and DeQuantizer
   quantizer_input = UniformQuantizer(x, n_bits(i), xmax, m) ;
   quantizer_output = UniformDequantizer(quantizer_input , n_bits(i), xmax,
m);

   % calculate the error
   q_error = x - quantizer_output;
```

```
   %Calculate SNR - Simulation

   SNR_simulation(i) = 10*log10(mean(x .^ 2) / mean(q_error .^ 2));


   %Calculate SNR - Theoritically

   SNR_theortically(i) = 10 * log10(power / (((xmax) .^ 2) / (3 *
((number_of_levels(i) .^ 2)))));


end


disp(SNR_simulation);

disp(SNR_theortically);


figure(5)

subplot(1,1,1)

title('Req-4 : simulated & actual SNR')

plot(n_bits,SNR_theortically,'-g', n_bits, SNR_simulation, 'bo')

xlabel('n-bits')

ylabel('SNR')



%---------------------------------Non Uniform Random Signal - Requirement
#5-----------------------%


rng('default');
```

```
number_of_R_V = 10000;


% +ve || -ve with equal probability, each = 0.5
sign = 2 * randi( [0 1] , 1 , number_of_R_V ) - 1;


% Exponential distribution
magnitude = exprnd(1, 1, number_of_R_V);


x = sign .* magnitude;


n_bits = 2:1:8;
xmax = max(abs(x));


%midrise
m = 0;


% intial value for the two arrays
SNR_simulation = zeros(size(n_bits));
SNR_theortically = zeros(size(n_bits));


% number of levels
number_of_levels = 2 .^ n_bits;
```

```
%calculate the power of the input signal

power = mean(x .^ 2);


for i = 1:length(n_bits)


   %midrise - Quantizer and DeQuantizer

   quantizer_input = UniformQuantizer(x, n_bits(i), xmax, m) ;

   quantizer_output = UniformDequantizer(quantizer_input , n_bits(i), xmax,
m);


   % calculate the error

   q_error = x - quantizer_output;


   %Calculate SNR - Simulation

   SNR_simulation(i) = 10*log10(mean(x .^ 2) / mean(q_error .^ 2));


   %Calculate SNR - Theoritically

   SNR_theortically(i) = 10 * log10(power / (((xmax) .^ 2) / (3 *
((number_of_levels(i) .^ 2)))));

end


figure(6)

subplot(1,1,1)

title('Req-5 : simulated & actual SNR')
```

```
plot(n_bits,SNR_theortically,'-g', n_bits, SNR_simulation, 'bo')

xlabel('n-bits')

ylabel('SNR')



%--------------------------------Non Uniform Random Signal Using Non Uniform Miu - Requirement #6-----------------------%

%array od Mius

miu = [0 ,5, 100, 200];


figure(7)

plotting_colors = ['k' , 'g' ,   'b' ,  'r'];

plots = zeros(size(n_bits));


for j = 1:length(miu)


  % intial value for the two arrays

  SNR_simulation = zeros(size(n_bits));

  SNR_theortically = zeros(size(n_bits));


  for i = 1:length(n_bits)


     xmax = max(abs(x));

     x_n=x/xmax;
```

```
if (miu(j) > 0)
   y = sign .* (log(1+miu(j)*abs(x_n))/log(1+miu(j)));
else
   y = x_n;
end


ymax = max(abs(y));
quantizer_input = UniformQuantizer(y, n_bits(i), ymax, m) ;  %midrise
quantizer_output = UniformDequantizer(quantizer_input , n_bits(i), ymax,
m);


if (miu(j) > 0)
   z = sign .*(((1+miu(j)).^abs(quantizer_output)-1)/miu(j));
else
   z =  quantizer_output;
end
de_comp = z * xmax;


q_error = abs(x - de_comp);
SNR_simulation(i) = 10*log10(mean(x .^ 2) / mean(q_error .^ 2));


if (miu(j) > 0)
```

```
        SNR_theortically(i) = 10 * log10 ((3*(number_of_levels(i) .^ 2))/( (log(1 +
miu(j))) .^ 2) );

      else

        SNR_theortically(i) = 10 * log10(power / (((xmax) .^ 2) / (3 *
((number_of_levels(i) .^ 2)))));

      end

   end

   % SNR_theortically SNR Vs number of bits

   plots(j) = plot(n_bits, SNR_theortically,  sprintf('%s-' , plotting_colors(j))  ,
'LineWidth', 1);

   hold on

   % SNR_simulation SNR Vs number of bits

   plots(j + 1) = plot(n_bits, SNR_simulation, sprintf('%s--' , plotting_colors(j)) ,
'LineWidth', 1);


end


disp(length(plots));

legend('SNR#theortically with miu = 0','SNR#simulation with miu = 0' ,
'SNR#theortically with miu = 5', 'SNR#simulation with miu = 5',
'SNR#theortically with miu = 100', 'SNR#simulation with miu = 100',
'SNR#theortically with miu = 200', 'SNR#simulation with miu = 200');

legend show

%--------------------------------Uniform Quantizier Function----------------------%


function q_ind = UniformQuantizer(in_val, n_bits, xmax, m)
```

```
% number of levels
L = 2^n_bits;

% step size
delta = (2*xmax) /L;
disp(['delta : ', num2str(delta)]);

% the quantization levels depend on midRise or MidTread
if ( m == 0 )
   q_levels = delta/2 - xmax : delta : delta/2 + xmax;
else
   q_levels = -xmax : delta : xmax;
end

disp(['Levels : ', num2str(q_levels)]);

% initialize quantized levels output
q_ind = zeros(size(in_val), 'int32');

for i = 1:length(in_val)
   % find the closest reconstruction level to the input sample
   [~, q_ind(i)] = min(abs(in_val(i) - q_levels));
```

```
  end
end

%--------------------------------Uniform DeQuantizier Function----------------------%

function deq_val = UniformDequantizer(q_ind, n_bits, xmax, m)

  % number of levels
  L = 2^n_bits;

  % step size
  delta = (2*xmax)/L;
  disp(['delta : ', num2str(delta)]);

  % the quantization levels depend on midRise or MidTread
  if ( m == 0 )
    q_levels = delta/2 - xmax : delta : delta/2 + xmax;
  else
    q_levels = -xmax : delta : xmax;
  end
  disp(['Levels : ', num2str(q_levels)]);

  % initialize quantized levels output
```

```
    deq_val = zeros(size(q_ind), 'double');


    for i = 1:length(q_ind)
     deq_val(i) = q_levels(q_ind(i));
    end
end
```