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	Section	B. N
Peter Atef Fathi	1	19
Beshoy Morad Atya	1	20

Contents

Figures

Part 1: Implementation of the uniform scalar quantizer function

```
1 % ============================================================================
2 % Uniform scalar quantizer
3 % ============================================================================
4 % in_val is a vector with the original samples
5 % n_bits is the number of bits available to quantize one sample in the quantizer
6 % xmax is the maximum value in the original vector
7 % m = 0 defines a "midrise" quantizer, and m = 1 gives a "midtread" quantizer
8 % q_ind is a vector with indexes of the chosen quantization level
9 function q_ind = UniformQuantizer(in_val, n_bits, xmax, m)
10     L = 2 ^ n_bits;
11     Delta = 2 * xmax / L;
12 
13     if (m == 0)
14         % midrise
15         q_ind = floor((in_val + Delta) / Delta);
16         q_ind = q_ind + abs(min(q_ind)) + 1;
17     else
18         % midtread
19         q_ind = round((in_val + xmax) / Delta) + 1;
20     end
21     q_ind(q_ind >= L) = L;
22 
23 end
```

Figure 1 Fig

Comment:

<u>*Idea of the function:*</u>

- In case of m=0 “mid-rise”:
  - We used the following equation: $q_{ind} = \left\lfloor \frac{in_{val} + \Delta}{\Delta} \right\rfloor$
  - For example, for in_val = -2Δ => q_{ind} = -1
  - Then, we add to q_{ind} the absolute of the min value in the q_{ind} + 1 to make sure that the minimum level in q_{ind} = 1 so q_{ind} will equal 1
  - For example, for in_val = 2Δ => q_{ind} = 3
  - Then, after line 16: q_{ind} will be 5 and this the usage of line 21 to eliminate all levels larger than L to be L.
- In case of m=1 “mid-tread”:
  - We used the following equation: $q_{ind} = round\left(\frac{in_{val} + Xmax}{\Delta}\right) + 1$
  - For example, for n_bit = 2, Δ= 3, in_val = -6 and X_max = 6 => q_{ind} = 0 + 1 =1
  - For example, for n_bit = 2, Δ= 3, in_val = 6 and X_max = 6 => q_{ind} = 4 + 1 =5 and here we can see the usage of line 21 to eliminate all levels larger than L to be L so that q_ind will be 4 instead of 5.

Part 2: Implementation of uniform scaler de-quantizer function

```
% ============================================================================
% Uniform scalar de-quantizer
% ============================================================================
% q_ind is a vector with indexes of the chosen quantization level
% n_bits is the number of bits available to quantize one sample in the quantizer
% xmax is the maximum value in the original vector
% m = 0 defines a "midrise" quantizer, and m = 1 gives a "midtread" quantizer
function deq_val = UniformDequantizer(q_ind, n_bits, xmax, m)
    L = 2 ^ n_bits;
    Delta = 2 * xmax / L;
    output_level = ((1 - m) * Delta/2) - xmax : Delta : ((1 - m) * Delta/2) + xmax;

    deq_val = output_level(q_ind);
end
```

Figure 2 fig

Comment:

- First, we divided the output levels using Δ and the value of "m."
- When m=0 "mid-rise":
    - Factor of Δ/2 is added to x-max as the starting and ending points of the levels.
- When m=1 "mid-tread":
    - The output levels' range from negative x-max to positive x-max.
- Finally, we map the indexes to the value of the level.

Part 3: Testing the quantizer/de-quantizer functions on a deterministic input (ramp)

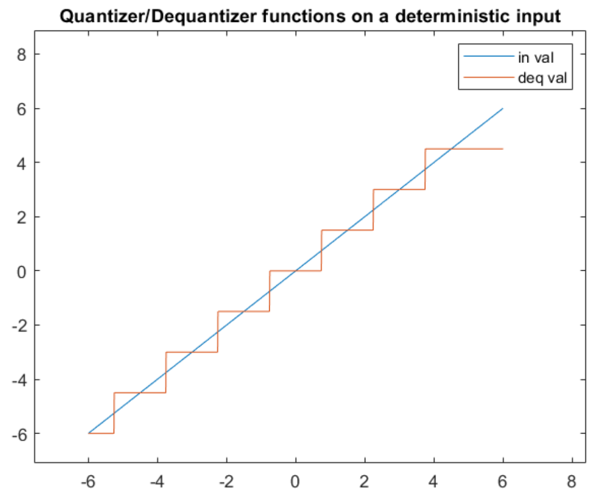
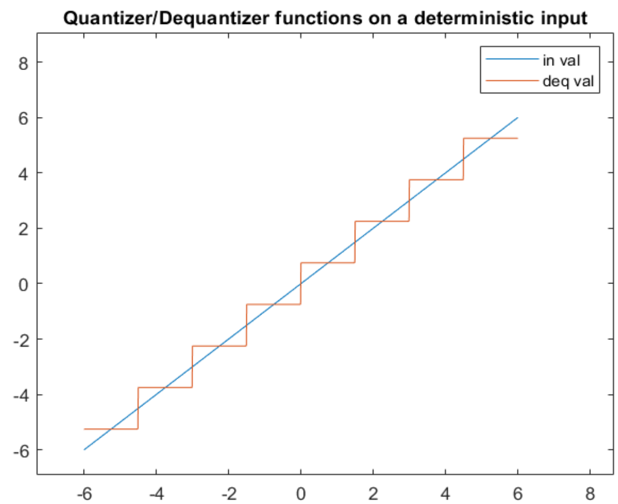
M = 1 "mid-tread"	M = 0 "mid-rise"
	

Figure 3 Fig

Comment:

- As shown for m = 0 "mid-rise" figure, there is no "level 0" and the levels are: ±0.75V, ±2.25V, ±3.75V, ±5.25V
- for m= 1 "mid-tread" there is "level 0"
- Also, in case of mid-tread we tried to increase the value of SNR by using 2^n levels instead of (2^n) – 1 by shifting one level down so the levels become: 0, ±1.5V, ±3V, ±4.5V, -6V.
- It's noticed the quantization error decreases with the increase of number of bits.

Part 4: Testing your input on a random input signal.

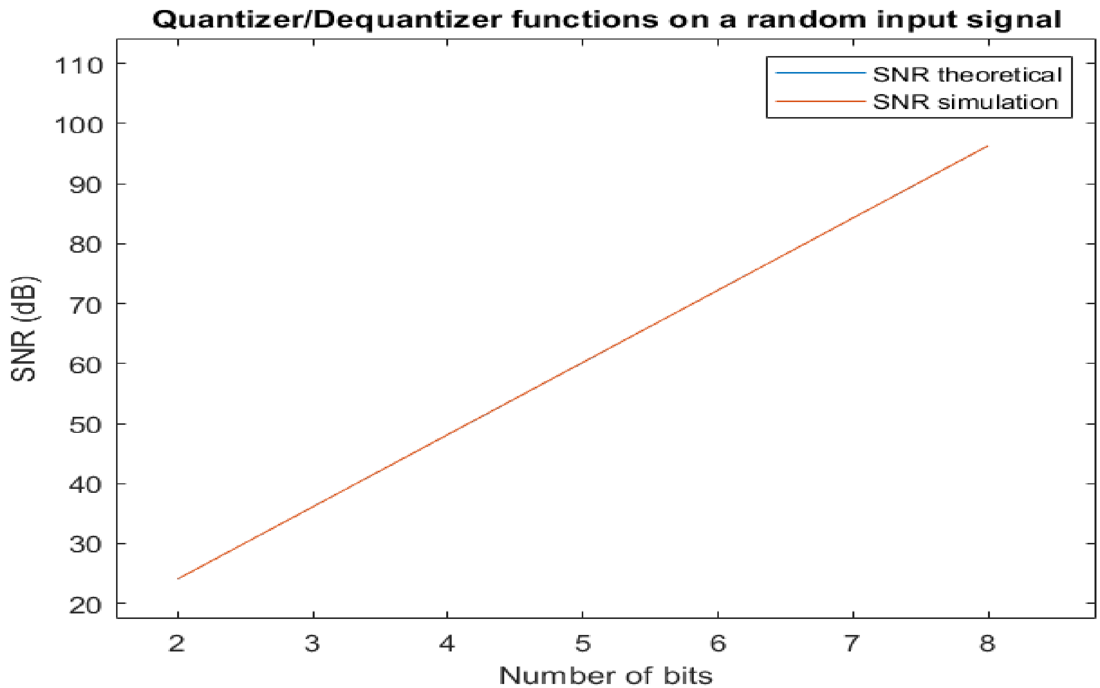


Figure 4 Fig

Comment:

- The graph tells us that the SNR simulation and theoretical are almost the same because the input signal was independent and identically distributed (i.i.d) continuous uniform random variables.

Part 5: Testing the uniform quantizer on a non-uniform random input.

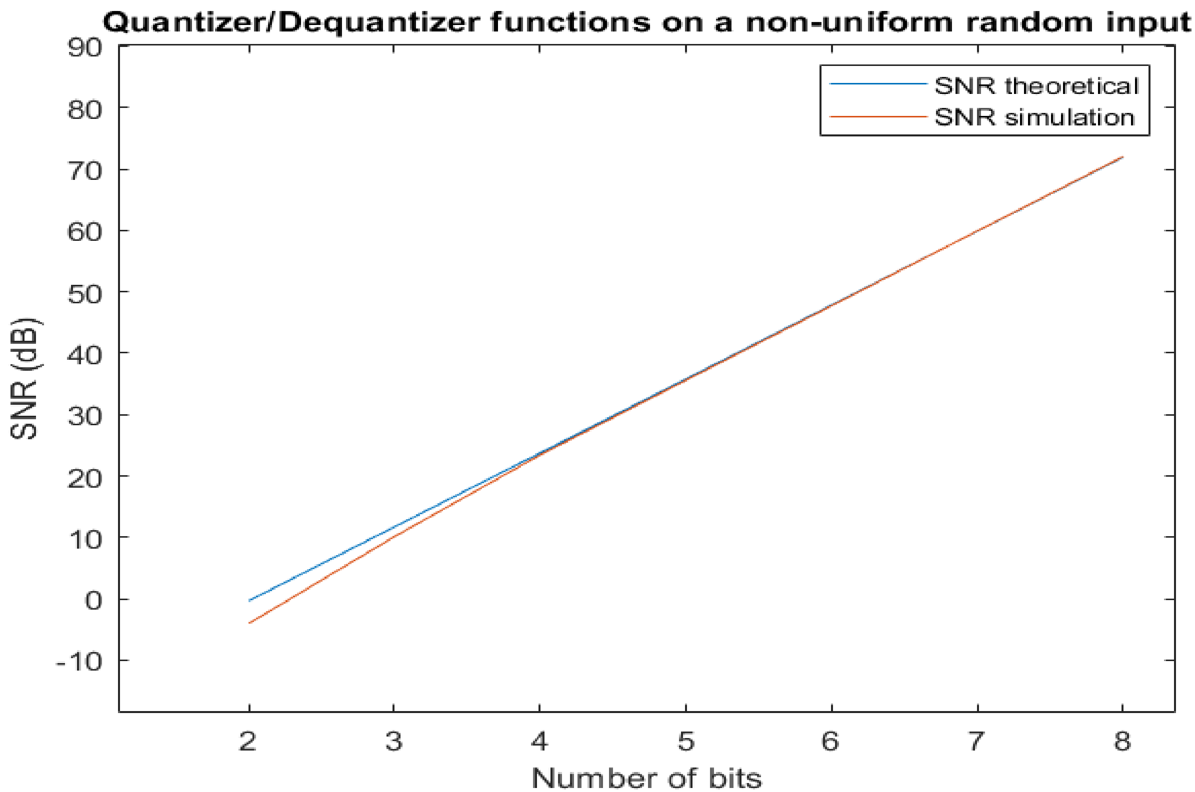
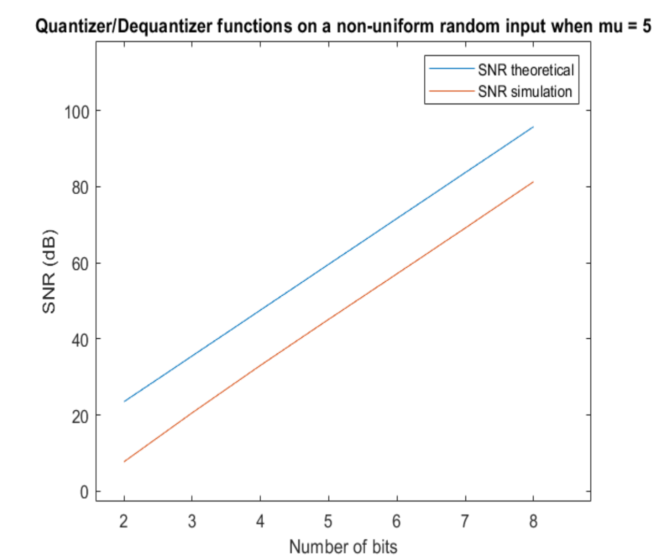


Figure 5 Fig

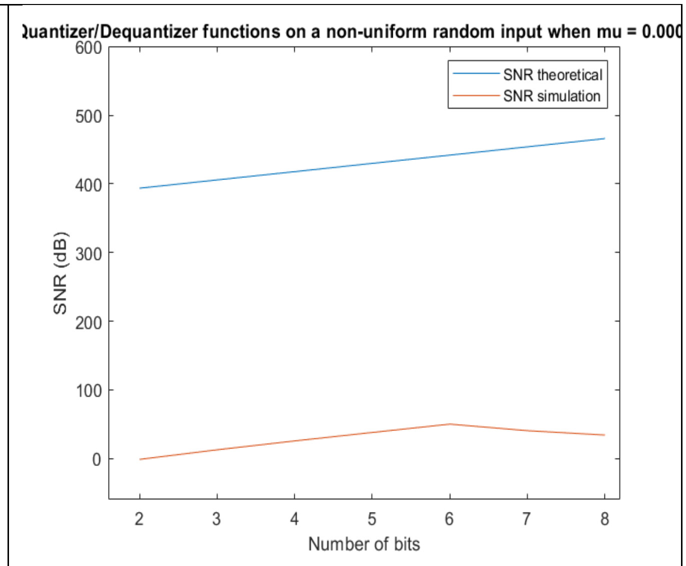
Comment:

- As the signal follows an exponential distribution, there were some values which are noticeably smaller than most of the signal values which leads to large quantization error while using uniform quantizer which leads to small SNR.
- What we claimed is supported by the graph for small bits (e.g., n = 2 and n=3) and for large number of bits, the simulation and theoretical SNR are almost the same.

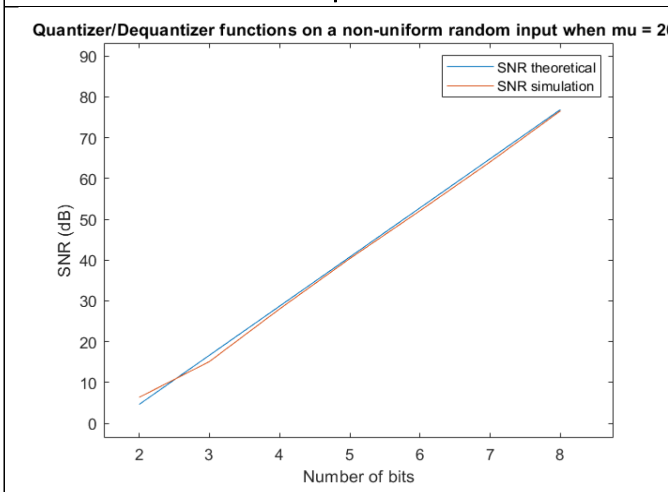
Part 6: Quantization of the non-uniform signal using a non-uniform μ law quantizer.



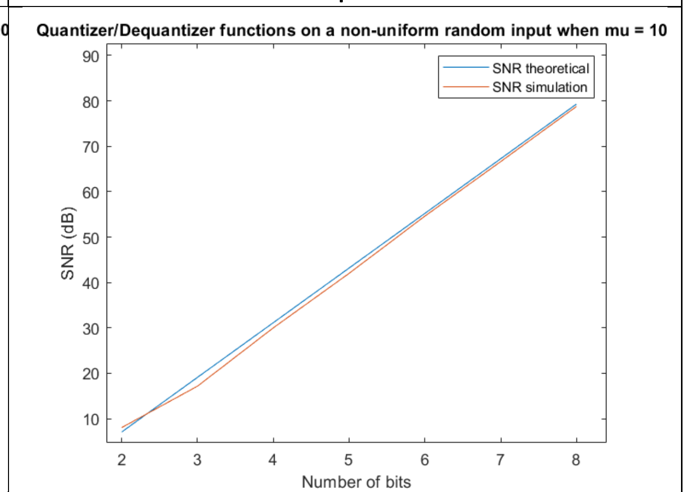
For μ = 5



For μ = 0



For μ = 200



For μ = 100

Figure 6 Fig

Comment:

It's noticed that with the increase of μ we get simulation SNR near to the theoretical SNR.

Index:

```matlab
% ============================================================================
% Point1: Uniform scalar quantizer
% ============================================================================
% in_val is a vector with the original samples
% n_bits is the number of bits available to quantize one sample in the quantizer
% xmax is the maximum value in the original vector
% m = 0 defines a "midrise" quantizer, and m = 1 gives a "midtread" quantizer
% q_ind is a vector with indexes of the chosen quantization level
function q_ind = UniformQuantizer(in_val, n_bits, xmax, m)
    L = 2 ^ n_bits;
    Delta = 2 * xmax / L;

    if (m == 0)
        % midrise
        q_ind = floor((in_val + Delta) / Delta);
        q_ind = q_ind + abs(min(q_ind)) + 1;
    else
        % midtread
        q_ind = round((in_val + xmax) / Delta) + 1;
    end
    q_ind(q_ind >= L) = L;

end
```

```matlab
% ============================================================================
% Point2: Uniform scalar de-quantizer
% ============================================================================
% q_ind is a vector with indexes of the chosen quantization level
% n_bits is the number of bits available to quantize one sample in the quantizer
% xmax is the maximum value in the original vector
% m = 0 defines a "midrise" quantizer, and m = 1 gives a "midtread" quantizer
function deq_val = UniformDequantizer(q_ind, n_bits, xmax, m)
    L = 2 ^ n_bits;
    Delta = 2 * xmax / L;
    output_level = ((1 - m) * Delta/2) - xmax : Delta : ((1 - m) * Delta/2) + xmax;

    deq_val = output_level(q_ind);
end
```

```
% ================================================================================
% Point3: Test the quantizer/dequantizer functions on a deterministic input
% ================================================================================
% variables
in_val = -6 : 0.01 : 6;
n_bits = 3;
xmax = 6;

% mid-rise
m = 0;
% mid-tread
% m = 1;

% functions calls
q_ind = UniformQuantizer(in_val, n_bits, xmax, m);
deq_val = UniformDequantizer(q_ind, n_bits, xmax, m);

% plot
figure
plot(in_val, in_val);
hold on
plot(in_val, deq_val);
title('Quantizer/Dequantizer functions on a deterministic input');
legend({'in val','deq val'});
```

```
% ===============================================================================
% Point4: Test the quantizer/ dequantizer functions on a random input signal
% ===============================================================================
% variables
n_bits = 2 : 1 : 8;
xmax = 5;
m = 0;

SNR_theoretical = zeros(1, length(n_bits));
SNR_simulation = zeros(1, length(n_bits));

% Loop 100 times to get the average SNR
for j = 1 : 100
    in_val = unifrnd(-5, 5, 1, 10000);

    for i = 1 : length(n_bits)
        q_ind = UniformQuantizer(in_val, n_bits(i), xmax, m);
        deq_val = UniformDequantizer(q_ind, n_bits(i), xmax, m);

        quantization_error = in_val - deq_val;

        E_quantization_error = mean(quantization_error.^2);
        E_input = mean(in_val.^2);

        SNR_simulation(i) = SNR_simulation(i) + mag2db(E_input / E_quantization_error);

        L = 2 ^ n_bits(i);
        SNR_theoretical(i) = SNR_theoretical(i) + mag2db(E_input *
((3*(L^2))/(xmax^2)));
    end
end

% Get the average SNR
SNR_simulation = SNR_simulation / 100;
SNR_theoretical = SNR_theoretical / 100;

% plot
plot(n_bits, SNR_theoretical);
hold on
plot(n_bits, SNR_simulation);
title('Quantizer/Dequantizer functions on a random input signal');
xlabel('Number of bits');
ylabel('SNR (dB)');
legend({'SNR theoretical','SNR simulation'});
```

```matlab
% =============================================================================
% Point5: Test the uniform quantizer on a non-uniform random input
% =============================================================================
% variables
n_bits = 2 : 1 : 8;
m = 0;

sgn = 2 * randi([0 1], 1, 10000) - 1;
in_val = sgn .* exprnd(1, 1, 10000);

xmax = max(abs(in_val));

SNR_theoretical = zeros(1, length(n_bits));
SNR_simulation = zeros(1, length(n_bits));

for i = 1 : length(n_bits)
    q_ind = UniformQuantizer(in_val, n_bits(i), xmax, m);
    deq_val = UniformDequantizer(q_ind, n_bits(i), xmax, m);

    quantization_error = in_val - deq_val;

    E_quantization_error = mean(quantization_error.^2);
    E_input = mean(in_val.^2);

    SNR_simulation(i) = mag2db(E_input / E_quantization_error);

    L = 2 ^ n_bits(i);
    SNR_theoretical(i) = mag2db(E_input * ((3*(L^2))/(xmax^2)));
end

% plot
plot(n_bits, SNR_theoretical);
hold on
plot(n_bits, SNR_simulation);
title('Quantizer/Dequantizer functions on a non-uniform random input');
xlabel('Number of bits');
ylabel('SNR (dB)');
legend({'SNR theoretical','SNR simulation'});
```

```
% ==========================================================================
% Point6: Test quantizer on the non-uniform signal using a non-uniform? μ-law quantizer
% ==========================================================================
% variables
n_bits = 2 : 1 : 8;
m = 0;
mu = 0.000001;
n = 10000;
sign = 2 * randi([0 1], 1, n) - 1;
magnitude = exprnd(1, 1, n);
x = sign .* magnitude;
xmax = max(abs(x));
x_norm = x / xmax;

SNR_theoretical = zeros(1, length(n_bits));
SNR_simulation = zeros(1, length(n_bits));

%compress
y = sign .* (log(1 + mu * abs(x_norm)) / log(1 + mu));

for i = 1 : length(n_bits)
    q_ind = UniformQuantizer(y, n_bits(i), max(y), m);
    deq_val = UniformDequantizer(q_ind, n_bits(i), max(y), m);
    %expander
    z = sign .* (((1 + mu).^ abs(deq_val) - 1) / mu);
    %denormailze
    z_final = z * xmax;

    quantization_error = x - z_final;

    E_quantization_error = mean(quantization_error.^2);
    E_input = mean(x.^2);

    SNR_simulation(i) = mag2db(E_input / E_quantization_error);
    L = 2 ^ n_bits(i);
    SNR_theoretical(i) = mag2db(((3*(L^2))/((log(1+mu))^2)));
end

% plot
plot(n_bits, SNR_theoretical);
hold on
plot(n_bits, SNR_simulation);
title('Quantizer/Dequantizer functions on a non-uniform random input when mu =
0.000001');
xlabel('Number of bits');
ylabel('SNR (dB)');
legend({'SNR theoretical','SNR simulation'});
```