**Department of Computer Engineering**



**Cairo University**

**Faculty of 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 Attia | 1 | 20 |

Contents

[**Part 1:…………** 4](#_Toc129967225)

[**Comment:** 4](#_Toc129967226)

[**Part 2:…………** 5](#_Toc129967227)

[**Comment:** 5](#_Toc129967228)

[**Part 3:…………** 6](#_Toc129967229)

[**Comment:** 6](#_Toc129967230)

[**Part 4:…………** 7](#_Toc129967231)

[**Comment:** 7](#_Toc129967232)

[**Part 5:…………** 8](#_Toc129967233)

[**Comment:** 8](#_Toc129967234)

[**Part 6:…………** 9](#_Toc129967235)

[**Comment:** 9](#_Toc129967236)

[**Index:** 10](#_Toc129967237)

Figures

[Figure 1 Fig 4](#_Toc129905445)

[Figure 2 fig 5](#_Toc129905446)

[Figure 3 Fig 6](#_Toc129905447)

[Figure 4 Fig 7](#_Toc129905448)

[Figure 5 Fig 8](#_Toc129905449)

[Figure 6 Fig 9](#_Toc129905450)

# **Part 1:** **Implementation of the uniform scalar quantizer function**

Text

Description automatically generatedFigure 1 Fig

## **Comment:**

*Idea of the function:*

* In case of m=0 “mid-rise”:
  + We used the following equation:
  + For example, for in\_val = -2∆ => qind = -1
  + Then, we add to qind the absolute of the min value in the qind + 1 to make sure that the minimum level in qind = 1 so qind will equal 1
  + For example, for in\_val = 2∆ => qind = 3
  + Then, after line 16: qind 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:
  + For example, for n\_bit = 2, ∆= 3, in\_val = -6 and X\_max = 6 => qind = 0 + 1 =1
  + For example, for n\_bit = 2, ∆= 3, in\_val = 6 and X\_max = 6 => qind = 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.

# Text Description automatically generated**Part 2: Implementation of uniform scaler de-quantizer function**

Figure 2 fig

## **Comment:**

* First, we divided the output levels using ∆ and the value of “m.”
* When m=0 “mid-rise”:
  + Factor of ∆/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.**

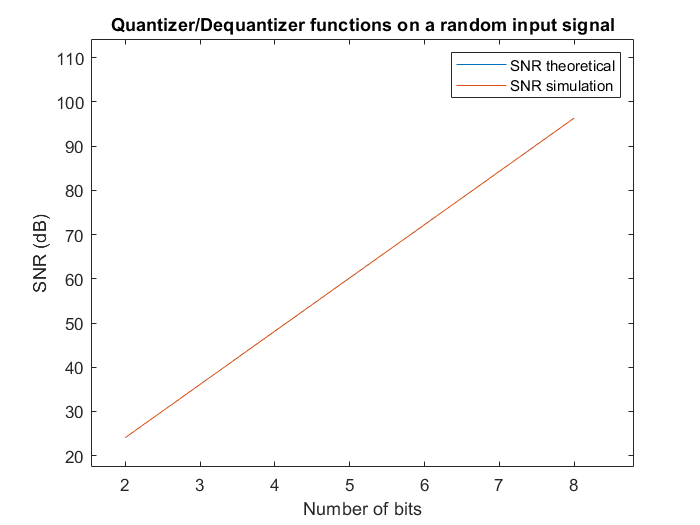
****

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.**

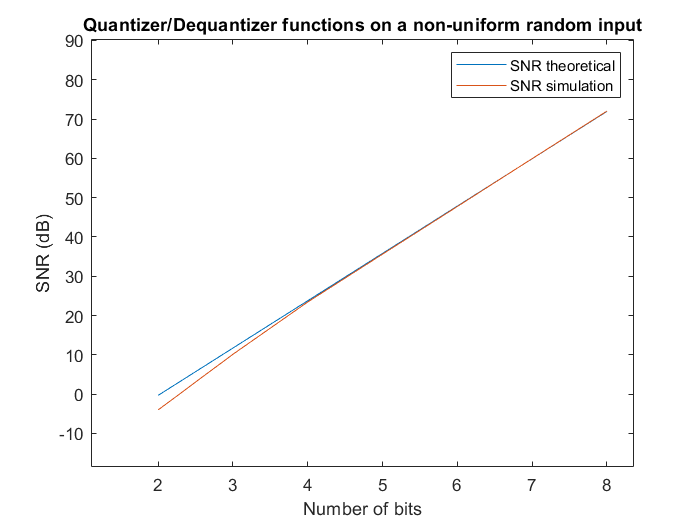


Figure 5 Fig

## **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**

Figure 6 Fig

## **Comment:**

# **Index:**

% =============================================================================

% 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

----------------------------------------------------------------------------------------

% =============================================================================

% 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'});

% =============================================================================

% 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'});