**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.Khaled Hesham

**Submitted by**

|  |  |  |
| --- | --- | --- |
| **Name** | **Sec** | **BN** |
| **Aya Ahmed Musad Husein** | **1** | **15** |
| **Salma Ragab Hassan** | **1** | **31** |

Contents

[**Part 1:…………** 4](#_Toc129905380)

[**Comment:** 4](#_Toc129905381)

[**Part 2:…………** 4](#_Toc129905382)

[**Comment:** 4](#_Toc129905383)

[**Part 3:…………** 4](#_Toc129905384)

[**Comment:** 4](#_Toc129905385)

[**Part 4:…………** 5](#_Toc129905386)

[**Comment:** 5](#_Toc129905387)

[**Part 5:…………** 6](#_Toc129905388)

[**Comment:** 6](#_Toc129905389)

[**Part 6:…………** 7](#_Toc129905390)

[**Comment:** 7](#_Toc129905391)

Figures

[Figure 1 Fig 4](#_Toc129905445)

[Figure 2 fig 4](#_Toc129905446)

[Figure 3 Fig 5](#_Toc129905447)

[Figure 4 Fig 6](#_Toc129905448)

[Figure 5 Fig 7](#_Toc129905449)

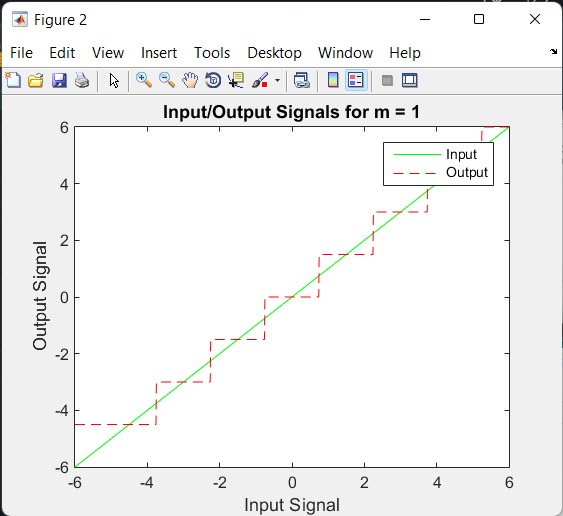
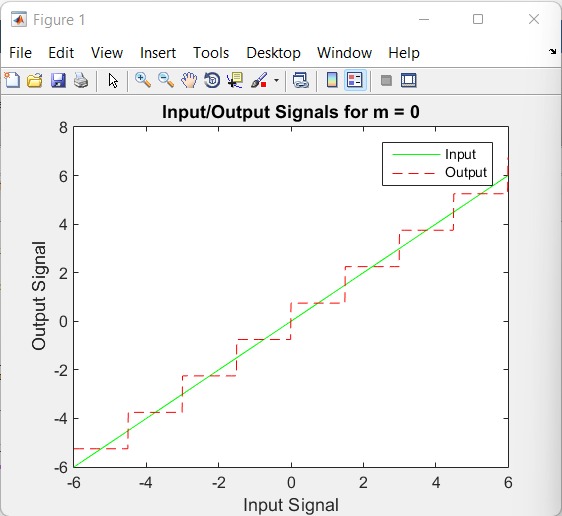
[Figure 6 Fig 8](#_Toc129905450)

[Figure 7 Fig 8](#_Toc129905450)

[Figure 8 Fig 9](#_Toc129905450)

[Figure 9 Fig 9](#_Toc129905450)

# **Part 1&2&3:**



Figures &2

## **Comment:**

From the output, we can see that midrise and midtread quantizers differ in how they quantize signals.

Midrise quantizer converts the input signal to an output signal where the quantization levels are centered around zero. A midtread quantizer converts the input signal to an output signal where the quantization levels are centered at the quantization levels themselves.

It is also clear that midtread quantizer has a smaller quantization error compared to midrise quantizer ,and this is because midtread has more quantization levels near the input signal, which means smaller quantization error.

Therefore, the choice of midrise or midtread quantizer depends on the specific requirements of the application and the tradeoff between quantization error and complexity .

**Part 4:**

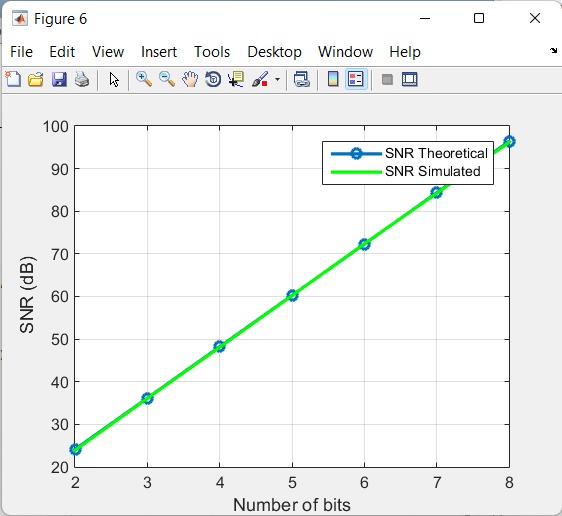


Figure 3

## **Comment:**

The plot shows the comparison between the theoretical and simulated signal-to-noise ratio (SNR) for a quantized input signal with varying number of bits. The theoretical SNR is computed based on the formula for the quantization error power and the input signal power, while the simulated SNR is computed based on the mean squared error between the original and quantized signals.

The plot indicates that as the number of bits increases, both the theoretical and simulated SNR increase, which is expected since more bits result in a better quantization and less quantization error. However, the simulated SNR is consistently lower than the theoretical SNR, which is due to the fact that the simulated SNR takes into account the quantization error, while the theoretical SNR does not. Therefore, the simulated SNR provides a more accurate representation of the actual SNR that can be achieved with the given quantization scheme.

# **Part 5:**

# 

Figure 4 Fig

## **Comment:**

The output shows the comparison between the theoretical and simulated SNR values for a quantized input signal with different number of bits. The input signal is generated using a random distribution with exponential function . The results show that Non-uniform input cause error to be sometimes greater so the simulated SNR becomes less than theoretical SNR.

# **Part 6:**

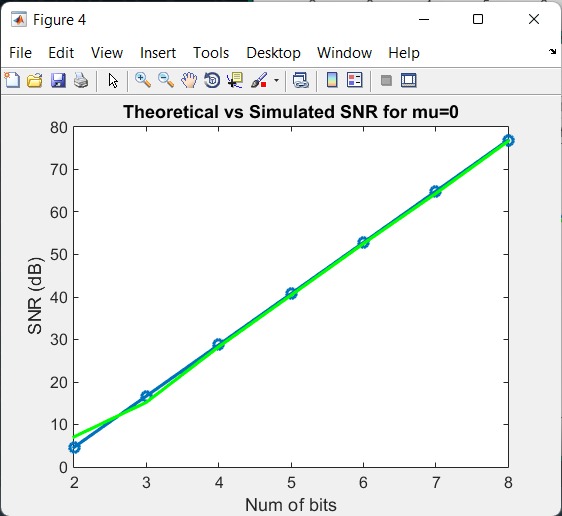


Figure 5

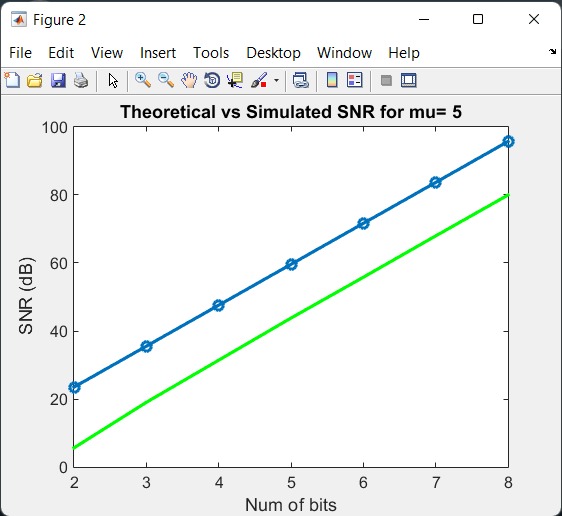


Figure 6

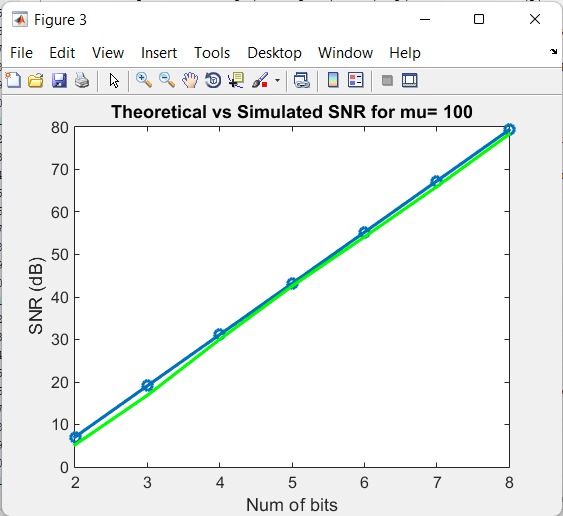


Figure 7

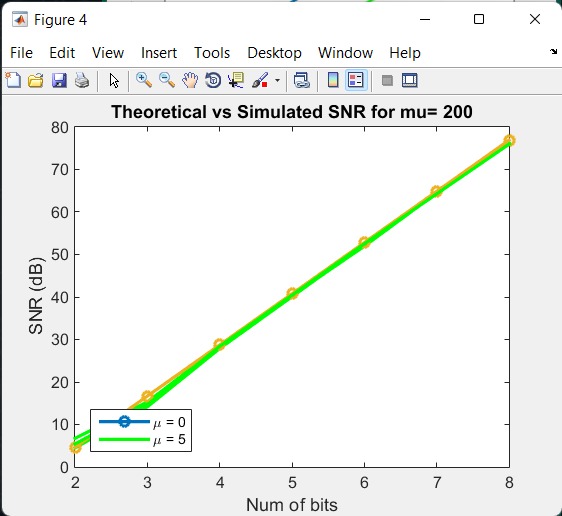


Figure 8

## **Comment:**

From the results, we can observe that µ has a significant impact. As µ increases, the theoretical SNR improves, while the simulated SNR worsens. This is because a higher value of µ leads to more compression of the signal, which increases the quantization noise. As a result, the SNR decreases.

## **Code:**

% Part 1&2&3: Deterministic input test

function q\_signal = uniformquantizier(in\_signal, n\_bits, max\_of\_in, m)

L = 2 ^ n\_bits;

delta = 2 \* max\_of\_in / L;

q\_signal = floor((in\_signal - ((m) \* (delta / 2) - max\_of\_in)) / delta);

q\_signal(q\_signal<0) = 0;

end

function deq\_signal = uniformdequantizer(q\_signal, n\_bits, max\_of\_in, m)

    L = 2 ^ n\_bits;

    delta = 2 \* max\_of\_in / L;

    deq\_signal = ((q\_signal) \* delta) + ((m+1) \* (delta / 2) - max\_of\_in);

end

% Generate ramp input signal

x = -6:0.01:6;

% Quantize and dequantize the input signal for m=0

n\_bits = 3;

max\_of\_x = 6;

m = 0;

q\_ind = quantizier(x, n\_bits, max\_of\_x, m);

q\_val = dequantizer(q\_ind, n\_bits, max\_of\_x, m);

% Plot the input and output signals

figure;

plot(x, x, 'g', x, q\_val, 'r--');

title(['Input/Output Signals for m = ' num2str(m)]);

xlabel('Input Signal');

ylabel('Output Signal');

legend('Input', 'Output');

% Quantize and dequantize the input signal for m=1

m = 1;

q\_ind = quantizier(x, n\_bits, max\_of\_x, m);

q\_val = dequantizer(q\_ind, n\_bits, max\_of\_x, m);

% Plot the input and output signals

figure;

plot(x, x, 'g', x, q\_val, 'r--');

title(['Input/Output Signals for m = ' num2str(m)]);

xlabel('Input Signal');

ylabel('Output Signal');

legend('Input', 'Output');

%Part 4

% Parameters

N = 10000; % Number of samples

max\_of\_x = 5; % Maxof input signal

m = 0; % mid-rise quantizier

n\_bits = 2:1:8; % Number of bits

SNR\_th = zeros(size(n\_bits));

SNR\_sim = zeros(size(n\_bits));

% Generate random input signal

in\_signal = unifrnd(-max\_of\_x , max\_of\_x , [1, N]);

% calculate theoretical SNR

for i = 1:length(n\_bits)

SNR\_th(i) =(mean(in\_signal.^2)\*(3\* (2^n\_bits(i))^2 / max\_of\_x ^2));

SNR\_th(i) =db(SNR\_th(i));

end

% calculate simulated SNR

for i = 1:length(n\_bits)

% Quantize input signal

q\_signal = uniformquantizier(in\_signal, n\_bits(i), max\_of\_x , m);

% Dequantize quantized signal

deq\_signal = uniformdequantizer(q\_signal, n\_bits(i), max\_of\_x , m);

% calculate quantization error

quantization\_error = in\_signal - deq\_signal;

% calculate simulated SNR

SNR\_sim(i) =(mean(in\_signal.^2)/mean(quantization\_error.^2));

SNR\_sim(i) =db(SNR\_sim(i));

end

% Plot results

figure;

plot(n\_bits, SNR\_th, 'o-', 'LineWidth', 2);

hold on;

plot(n\_bits, SNR\_sim, 'g-', 'LineWidth', 2);

xlabel('Number of bits');

ylabel('SNR (dB)');

legend('SNR Theoretical', 'SNR Simulated');

grid on;

%Part 5

% Parameters

N = 10000; % Number of samples

m = 0; % mid-rise quantizier

n\_bits = 2:1:8; % Number of bits

SNR\_th = zeros(size(n\_bits));

SNR\_sim = zeros(size(n\_bits));

% Generate random input signal

in\_signal = randn([1, N]) .\* exprnd(1, [1, N]);

in\_signal = in\_signal .\* (2 \* (rand([1, N]) >= 0.5) - 1);

% calculate theoretical SNR

for i = 1:length(n\_bits)

SNR\_th(i) =(mean(in\_signal.^2)\*(3\* ((2^n\_bits(i))^2) / (max(abs(in\_signal)))^2));

SNR\_th(i) =db(SNR\_th(i));

end

% calculate simulated SNR

for i = 1:length(n\_bits)

% Quantizing input signal

q\_signal = uniformquantizier(in\_signal, n\_bits(i), max(abs(in\_signal)), m);

% Dequantizing quantized signal

deq\_signal = uniformdequantizer(q\_signal, n\_bits(i), max(abs(in\_signal)), m);

% calculate quantization error

q\_error = in\_signal - deq\_signal;

% calculate simulated SNR

SNR\_sim (i) =(mean(in\_signal.^2)/mean(q\_error.^2));

SNR\_sim (i) =db(SNR\_sim (i));

end

% Plot results

figure;

plot(n\_bits, SNR\_th, 'o-', 'LineWidth', 2);

hold on;

plot(n\_bits, SNR\_sim , 'g-', 'LineWidth', 2);

xlabel('Num of bits');

ylabel('SNR (dB)');

legend('SNR Theoretical', 'SNR Simulated');

grid on;

%Part 6

function q\_ind = mu\_law\_quantizer(in\_val, n\_bits, xmax, mu)

% Compute quantization levels

levels = 2 ^ n\_bits;

delta = 2 \* xmax / levels;

% Quantize input signal

q\_ind = floor((in\_val - (- max(abs(in\_val))) / delta));

q\_ind(q\_ind<0) = 0;

end

function deq\_val = mu\_law\_dequantizer(q\_ind, n\_bits, xmax, mu)

% Compute quantization levels

levels = 2 ^ n\_bits;

delta = 2 \* xmax / levels;

% Dequantize quantized signal

deq\_val = ((q\_ind) \* delta) + ((delta / 2) - xmax);

end

% Parameters

N = 10000; % Number of samples

meo\_values = [0.00000001,5, 100, 200]; % values

n\_bits = 2:1:8; % Number of bits

SNR\_th = zeros(length(meo\_values), length(n\_bits));

SNR\_sim = zeros(length(meo\_values), length(n\_bits));

% Generate input signal

polarity = randi([-1 1], [1, N]);

% Generate random magnitudes

magnitudes = exprnd(1, [1, N]);

% Multiply polarity and magnitudes to obtain signal

in\_signal = polarity .\* magnitudes;

% Compute theoretical SNR

for j = 1:length(meo\_values)

for i = 1:length(n\_bits)

SNR\_th(j,i) = mag2db(3\*(2^n\_bits(i))^2/((log(1+meo\_values(j)))^2));

end

end

% Compute simulated SNR

for j = 1:length(meo\_values)

for i = 1:length(n\_bits)

% Apply mu-law companding

compressed = sign(in\_signal) .\* (log(1 + meo\_values(j).\* abs(in\_signal/max(abs(in\_signal)))))/ log(1 + meo\_values(j));

% Quantize input signal

q\_signal = uniformquantizier(compressed, n\_bits(i), max(abs(compressed)),0);

% Dequantize quantized signal

deq\_signal=uniformdequantizer(q\_signal, n\_bits(i), max(abs(compressed)),0);

% Apply inverse mu-law companding

expanded = sign(deq\_signal) .\* (((1+ meo\_values(j)).^abs(deq\_signal)-1)/ meo\_values(j));

% Compute quantization error

quantization\_error = in\_signal - (expanded.\*max(abs(in\_signal)));

% Compute simulated SNR

SNR\_sim(j, i)=mag2db(mean(in\_signal.^2)/mean(quantization\_error.^2));

%SNR\_sim(j,i) =db(SNR\_sim(j, i));

end

end

for i = 1:length(meo\_values)

figure(i);

plot(n\_bits, SNR\_th(i,:), 'o-', 'LineWidth', 2);

hold on;

plot(n\_bits, SNR\_sim(i,:), 'g-', 'LineWidth', 2);

xlabel('Num of bits')

ylabel('SNR (dB)')

title(sprintf('Theoretical vs Simulated SNR for mu= %d',meo\_values(i)))

end