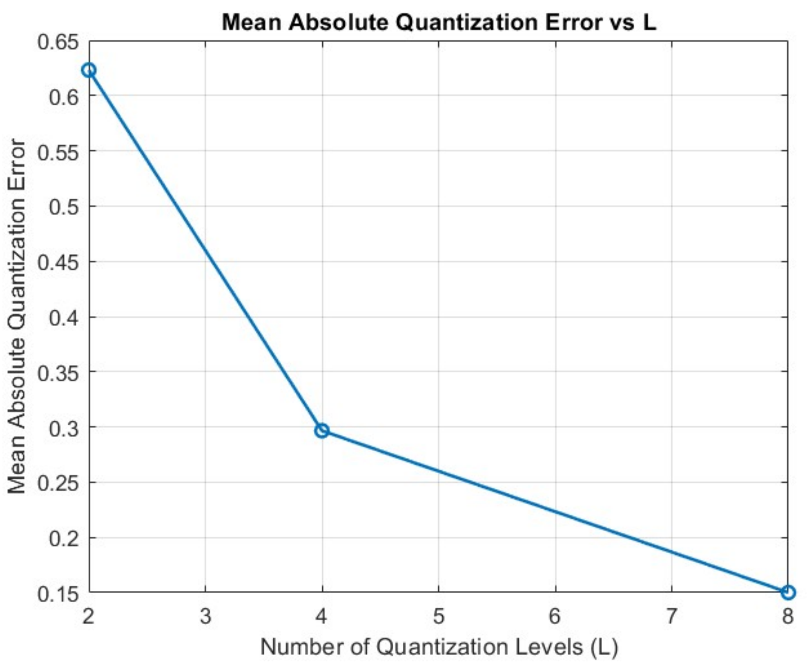
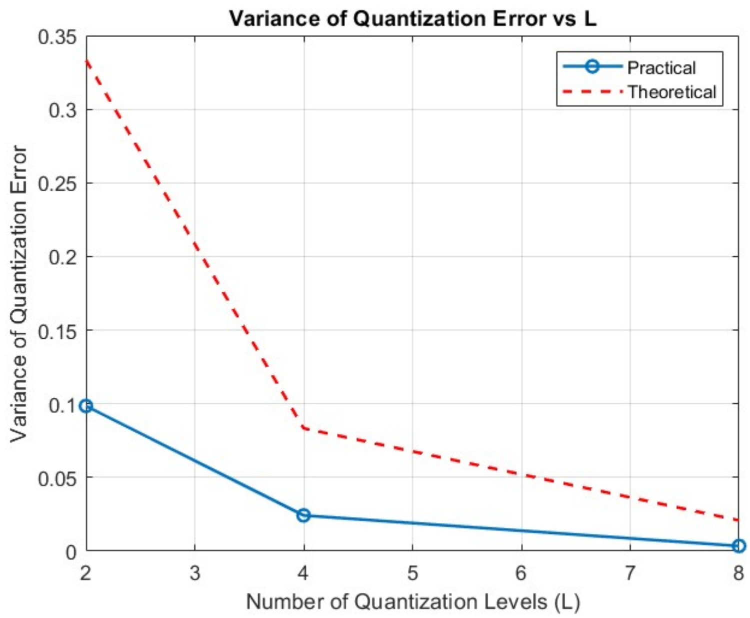
| | | |
|---|---|---|
| Mahmoud Hazem Mansour | 58-4029 | T2 |
| Alaa Ahmed | 58-9492 | T2 |
| Abdelrahman Elabyad | 58-4464 | T2 |

# COMM502- COMMUNICATION THEORY- FINAL LAB PROJECT 2024

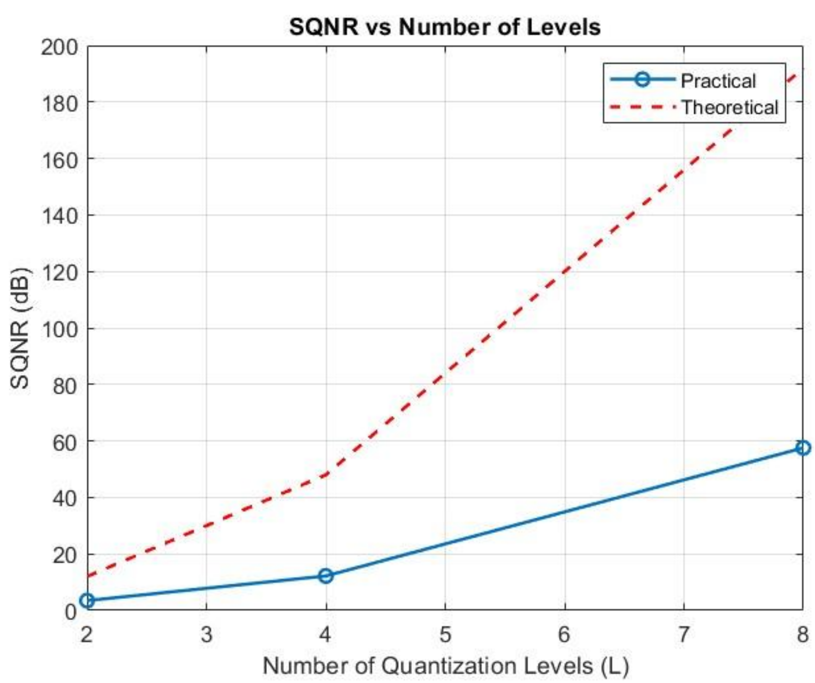
**Comment on 3:** As the number of quantization levels increases, the MAQE decreases.



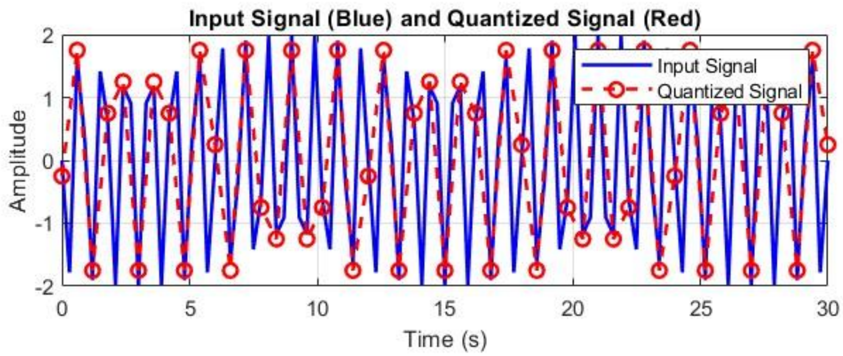

**Comment on 4:** Both practical and theoretical variance decrease as LLL increases.




**Comment on 5:** Theoretical SQNR increases linearly with the logarithm of LLL. Practical SQNR also increases with LLL, but at a slower rate compared to the theoretical SQNR.



## The input and output signals




## Comment on 12:

<u>2 ways that can enhance the approximation of the output figure.</u>

1-Increase Quantization Levels: Use more levels to make the output signal closer to the input.

2-Minimize Compression Loss: Adjust the compression algorithm to improve signal quality.

<u>The reason why there is difference between the input and the output signals.</u>

Quantization Error: Converting continuous values to discrete levels causes errors, seen as deviations in the red (output) signal.

# CODE

```
% Parameters
a = 2;
b = 9;
fs = 2; % Sampling rate
t = 0:0.3:30; % Time vector

% Input signal x(t)
x_t = a * sin(0.5 * b * pi * t);

% Sampling: Take 1 sample every Ns samples (Continious to discrete)
Ns = 2; % Adjust this as needed
[Xs,ts]=Sampling(x_t,t,Ns);
% Quantization levels (L = 2, 4, 8)
L_values = [2, 4, 8];
Vmin = min(Xs);
Vmax = max(Xs);

mean_abs_error = zeros(1, length(L_values));
variance_error_p = zeros(1, length(L_values));
```

```
variance_error_theo = zeros(1, length(L_values));

SQNR_practical = zeros(1, length(L_values));

SQNR_theoretical = zeros(1, length(L_values));


% Outer loop for levels L

for idx = 1:length(L_values)% indx (index) is to access each index in the arrays

   L = L_values(idx);

   Delta = (Vmax - Vmin) / L; % Step size

   Qlevels = Vmin + Delta/2 : Delta : Vmax - Delta/2; % Quantization levels


   % Quantize the signal

   Xq = zeros(size(Xs));

   % loop on the Xs array and subtrct each value from the given xq to find the
smallest error

   for i = 1:length(Xs)

      [min_value, nearest_idx] = min(abs(Xs(i) - Qlevels)); % Find nearest level

      Xq(i) = Qlevels(nearest_idx); % Assign quantized value

   end


   % Calculate errors

   abs_error = abs(Xs - Xq);

   % Mean absolute error

   mean_abs_error(idx) = mean(abs_error);

   % Variance of quantization error

   variance_error_p(idx) = var(abs_error);
```

```
    variance_error_theo(idx)=(Delta^2)/12;
    % SQNR Calculations
    signal_power = mean(Xs.^2);
    noise_power = mean((Xs - Xq).^2);
    % Practical SQNR
    SQNR_practical(idx) = signal_power / noise_power;
    % Theoretical SQNR (for uniform quantization)
    SQNR_theoretical(idx) =3*(L^2);
end
% Plot Input Signal and Quantized Signal on the Same Figure
figure;
subplot(2, 1, 1);
plot(t, x_t, 'b', 'LineWidth', 1.5);
hold on;
plot(ts, Xq, 'r--o', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Amplitude');
title('Input Signal (Blue) and Quantized Signal (Red)');
legend('Input Signal', 'Quantized Signal');
grid on;

% Plot Mean Absolute Quantization Error vs Number of Levels
figure;
plot(L_values, mean_abs_error, '-o', 'LineWidth', 1.5);
```

```
xlabel('Number of Quantization Levels (L)');

ylabel('Mean Absolute Quantization Error');

title('Mean Absolute Quantization Error vs L');

grid on;


% Plot Variance (Practical and Theoretical) of Quantization Error vs Number of
Levels

figure;

plot(L_values, variance_error_p, '-o', 'LineWidth', 1.5);

hold on;

plot(L_values, variance_error_theo, '--r', 'LineWidth', 1.5); % Theoretical variance

xlabel('Number of Quantization Levels (L)');

ylabel('Variance of Quantization Error');

title('Variance of Quantization Error vs L');

legend('Practical', 'Theoretical');

grid on;


% Plot SQNR (Practical and Theoretical) vs Number of Levels

figure;

plot(L_values, SQNR_practical, '-o', 'LineWidth', 1.5);

hold on;

plot(L_values, SQNR_theoretical, '--r', 'LineWidth', 1.5);

xlabel('Number of Quantization Levels (L)');

ylabel('SQNR (dB)');

title('SQNR vs Number of Levels');
```

```
legend('Practical', 'Theoretical');
grid on;

Qlevels
% Normalize to decimal encoding from 0 to 7
decimal_encoded = 0:length(Qlevels)-1;

% Find the index of each quantized value in the quant_levels array
[~, encoded_signal] = ismember(Xq, Qlevels);
% The encoded_signal now contains decimal values from 0 to 7
%applying Huffman source coding on the quantized signal
msg=encoded_signal;
% hsitcounts take the message and number of bins
prob=histcounts(msg,length(unique(msg)))./length(msg);
unique_msg= unique(msg);
i=-log2(prob);
H=sum(prob.*i);
L=sum(prob.*ceil(i));

[dictionary, avg]=huffmandict(unique_msg,prob);
encode=huffmanenco(msg,dictionary);
decoded_msg=huffmandeco(encode,dictionary);
% Check if decoded message is the same as the original message
disp('Is the decoded message equal to the original message?');
```

```
disp(isequal(msg, decoded_msg));

%Calculate Compression Efficiency
original_size = length(msg);
encoded_size = length(encode);

compression_efficiency = H/L;

k=length(unique_msg);

% Calculate Compression Rate
compression_rate = log2(k)/avg;

% Display Results
fprintf('Compression Efficiency: %.2f%%\n', compression_efficiency * 100);
fprintf('Compression Rate: %.2f\n', compression_rate);
```

# Values

Compression Efficiency: 82.35%

Compression Rate: 1.08