Orkun İbrahim Kök

22003656

EEE431 Assignment #1

Part a)

Derivation for this part can be found at the end of the Appendix.

Part b)

The pdf of the Z, $f_Z(z)$ is found as follows:

$$f_Z(z) = \frac{1}{2} e^{|-z|}$$

For 100 samples of X and Y, the realization versus original pdf plot is given below.

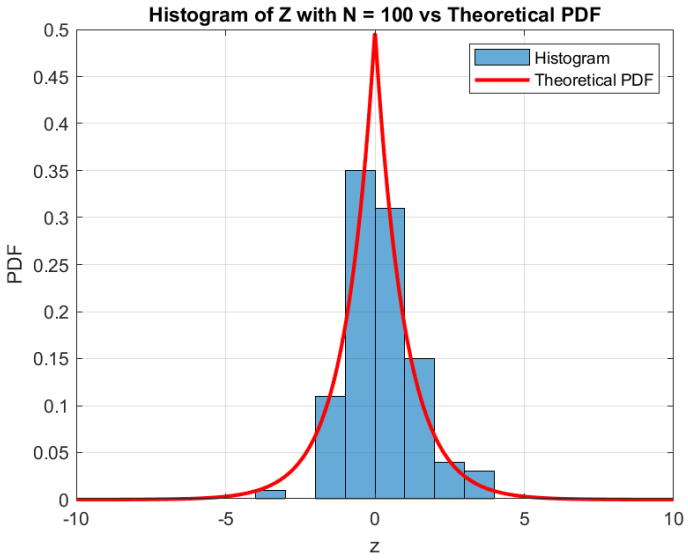

Fig. 1: 100 realization plot

The red curve is the pdf of original Z, and blue histogram shows the realization values.

Below plot shows the realizations for 100000 samples.

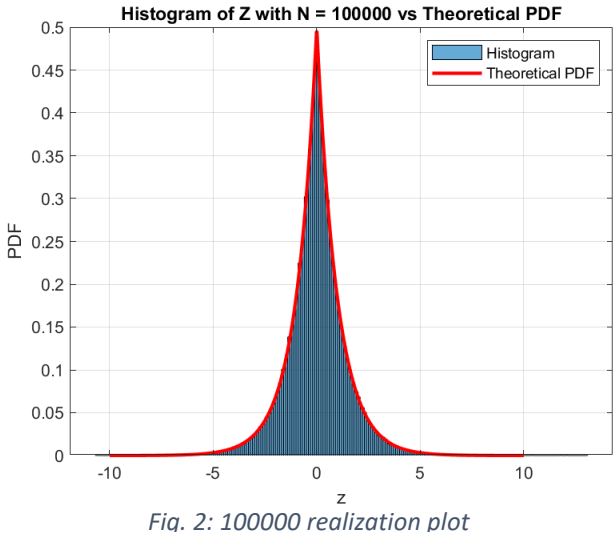

Fig. 2: 100000 realization plot

As expected, as the number of realizations increase, histogram starts fit into the original pdf curve.

Part c)

Because Z is a Laplace distribution with parameter 1, the average power $E[Z^2]$ is calculated as 2. However, using the 100000 realizations of Z, average power is measured as 1.9805.

Empirical $E[Z^2] \approx 1.9805$

Theoretical $E[Z^2]$: 2

Part d)

For uniform quantization, first min and max values are determined. I determined Z_{min} = -4 and Z_{max} = 4, because I observed that after these values, pdf values are very close to zero, so no contribution will come from these values. After, because leves are specified as 8, I found the step size (Δ).

$$\Delta = \frac{Z_{max} - Z_{min}}{N} = \frac{8}{8} = 1$$

Also, because uniform quantization is about midpoints, quantization levels are calculated as follows:

$$a_i = Z_{min} + \left(i - \frac{1}{2}\right) * \Delta \; ; where \; i = 1, 2, 3, \dots, N$$

Quantizaiton levels:

-3.5000 -2.5000 -1.5000 -0.5000 0.5000 1.5000 2.5000 3.5000

To determine input sample z's, and in which bin do they fall into (also assigning the nearest bin of course), the following equation is used.

$$i = round\left(\frac{z - Z_{min}}{\Delta}\right) + 1 \; ; where \; if \; i < 1, i = 1 \quad if \; i > N, i = N$$

For the quantization error, which is denoted as e = z – ź. The error power is $E[e^2]$, and signal power is $E[Z^2]$ as stated before.

$$SQNR = 10 \log_{10}(\frac{E[Z^2]}{E[e^2]})$$

Average quantization error power: 0.3822

SQNR (dB): 7.15 dB

Part e)

Because Laplace distribution is non-uniform distribution, the quantized output Z_Q will have higher probability near center levels and lower probability at extreme levels.

The equation for calculating the PMF values is given below.

$$\hat{P}(\tilde{Z} = l_i) = \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}\{\tilde{z}_j = l_i\}$$

Where 1{.} denotes the indicator function

l_i = i^{th} quantization value (also known as a_i)

N = 8 levels

The most basic explanation can be, counting how many times each quantization level appears in the quantized output and dividing by the total number of samples.

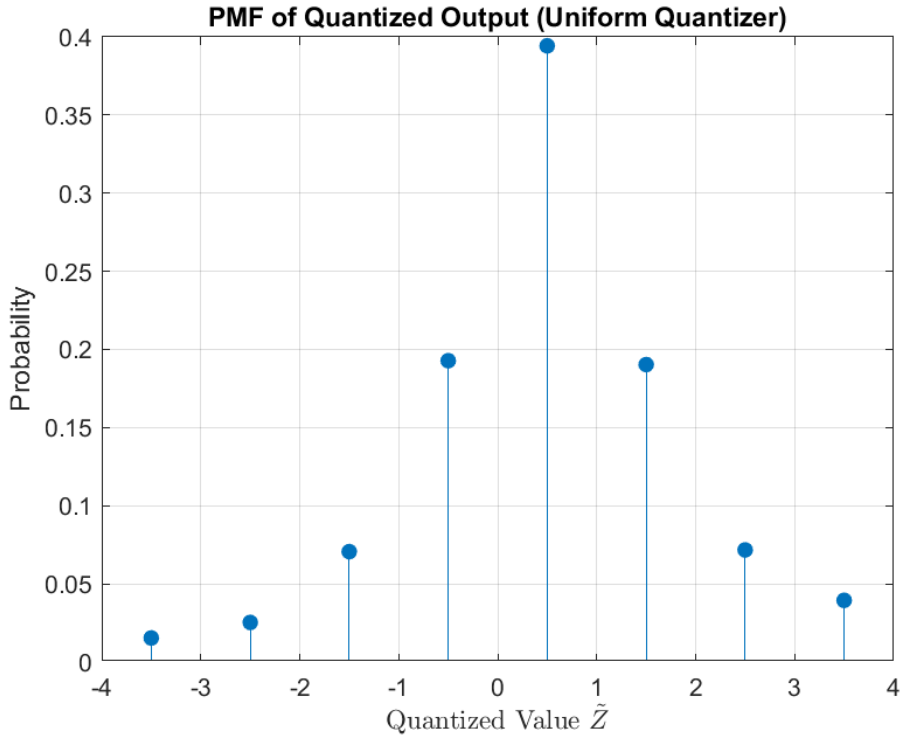

Fig. 3: PMF of Uniform quantized Z

As expected, higher probability around center levels and lower probability at extreme levels.

Part f)

CDF of Z, $F_Z(z)$ is calculated as follows:

$$F_Z(z) = \begin{cases} \frac{1}{2}e^z \; if \; z < 0 \\ 1 - \frac{1}{2}e^z \; if \; z > 0 \end{cases}$$

The Inverse of the CDF, $F_Z^{-1}(z)$ is calculated as follows:

$$F_Z{}^{-1}(u) = \begin{cases} \ln(2u) \; if \;\; 0 < u < 0.5 \\ -\ln(2(1-u)) \; if \;\; 0.5 < u < 1 \end{cases}$$

Part g)

In this part, I implemented a non-uniform PCM quantizer using the companding technique, which transforms the non-uniform input distribution into a uniformly distributed signal before applying a uniform quantizer.

The compressor function g(z) is chosen to be the CDF of the Laplace(0,1) distribution, $F_Z(z)$, which maps the input z into the interval [0,1]. Then I applied a uniform quantizer in this compressed domain with N=8 equally spaced levels.

After quantization in the compressed domain, we apply the inverse of the CDF, $F_Z^{-1}(u)$ as the expander to map the quantized values back to the original domain. The result is a non-uniform quantizer.

As a result, the quantization regions vs quantization points plot can be seen in below figure.

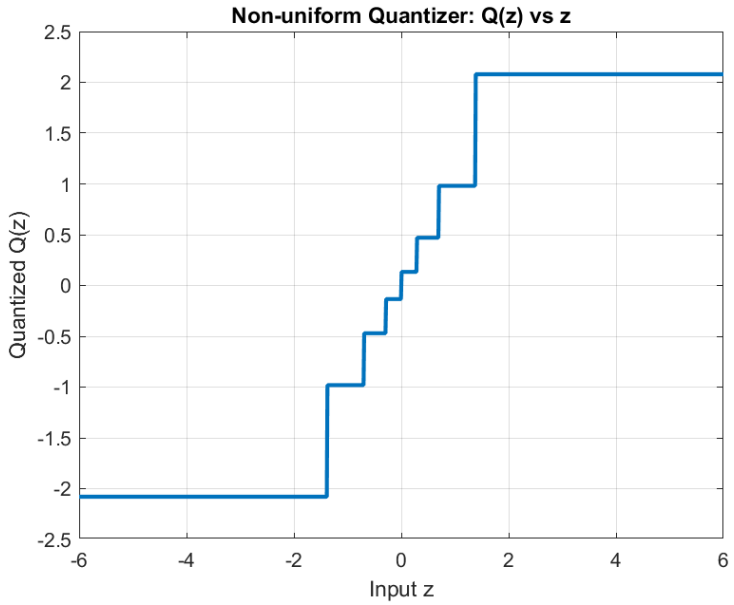

Fig. 4: Non-uniform PCM Quantization regions vs Quantization points

Part h)

In this part, the results of non-uniform PCM quantization are compared to the ones with uniform quantization results. First of all, The SQNR value for non-uniform quantization is greater than uniform quantization SQNR value with equal levels. This is related with average quantization error power. As expected, the error power in non-uniform quantization is less compared to uniform quantization. Therefore, SQNR is greater in non-uniform quantization compared to uniform quantization.

Average quantization error power (non-uniform): 0.2850

SQNR for non-uniform quantization: 8.42 dB

In part d), the results are as follows:

Average quantization error power: 0.3822

SQNR (dB): 7.15 dB

As expected, error decreased, as a result SQNR increased.

Part i)

The non-uniform quantized version will have a more evenly distributed PMF compared to uniform quantization PMF.

The below equation is for calculating the PMF values

$$P(\tilde{Z} = r_i) = \int_{F_Z^{-1}(u_{i-1})}^{F_Z^{-1}(u_i)} f_Z(z)\, dz$$

$F_Z^{-1}(u)$ is the inverse CDF used as expander, and u_i's are i/N values.

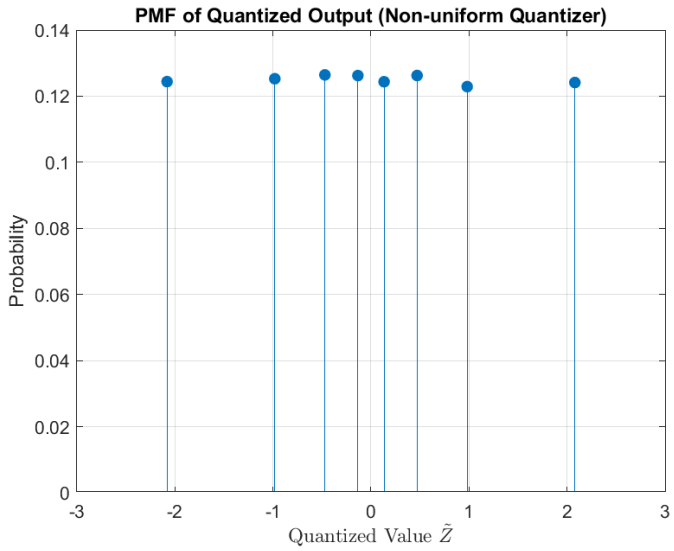

Fig. 5: PMF of non-uniform Quantization

The below plot shows the comparison between PMF values for uniform and non-uniform quantization values of Z.

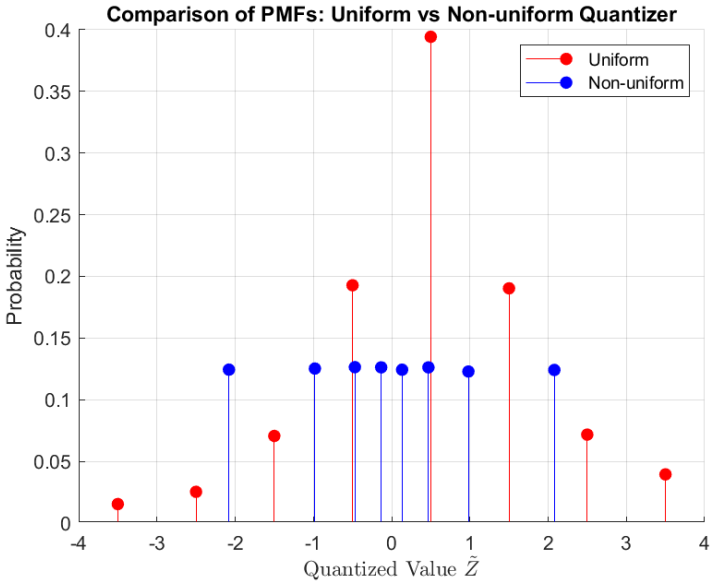

Fig. 6: PMF comparison between uniform and non-uniform quantization

Part j)

In this part, I implemented the Lloyd-Max algorithm to compute an optimal non-uniform quantizer for the Laplace distribution. The Lloyd-Max algorithm iteratively refines the quantization thresholds and reconstruction levels to minimize mean squared error (MSE).

I began with an initial guess of equally spaced reconstruction levels within a defined range [−a,a]. At each iteration, I updated the decision thresholds as the midpoints between adjacent reconstruction levels and the reconstruction levels (centroids) as the expected value of Z within each quantization region, weighted by the source probability density.

This process was repeated until convergence (i.e., changes in reconstruction levels fell below a small threshold). The final output is a set of optimal thresholds and reconstruction levels that minimize quantization distortion for the given source PDF.

1. Guess initial set of representative levels $\hat{x}_q \quad q = 0,1,2,\ldots,M-1$
2. Calculate decision thresholds

$$t_q = \frac{1}{2}\left(\hat{x}_{q-1} + \hat{x}_q\right) \quad q = 1,2,\ldots,M-1$$

3. Calculate new representative levels

$$\hat{x}_q = \frac{\int_{t_q}^{t_{q+1}} x \cdot f_X(x)dx}{\int_{t_q}^{t_{q+1}} f_X(x)dx} \quad q = 0,1,\ldots,M-1$$

4. Repeat 2. and 3. until no further distortion reduction

Fig. 7: Iterative Algorithm

Converged after 45 iterations.

Final Decision Thresholds:

-6.0000 -3.0168 -1.6389 -0.7077 0.0000 0.7077 1.6389 3.0167 6.0000

Final Reconstruction Levels:

-3.8585 -2.1751 -1.1026 -0.3127 0.3127 1.1026 2.1751 3.8584

These are the MATLAB results.

Part k)

Because Lloyd-Max quantization minimizes the MSE, the average error power is low compared to uniform and non-uniform quantization. Therefore, SQNR value will be the highest among all three quantization techniques.

Lloyd-Max Quantization Results:

Quantization Error Power: 0.2076

SQNR (dB): 9.80 dB

---- SQNR Comparison ----

Uniform Quantizer (Part d): 7.15 dB

Compander-based Quantizer (Part h): 8.42 dB

Lloyd-Max Quantizer (Part i): 9.80 dB

As expected, SQNR value increased, due to the decrease in MSE, so as the average error power.

Part l)

In Lloyd Max quantization, PMF is similar to the non-uniform quantization PMF, however, in high density regions a more even distribution is present and a better adaptation of it holds for low density regions as well.

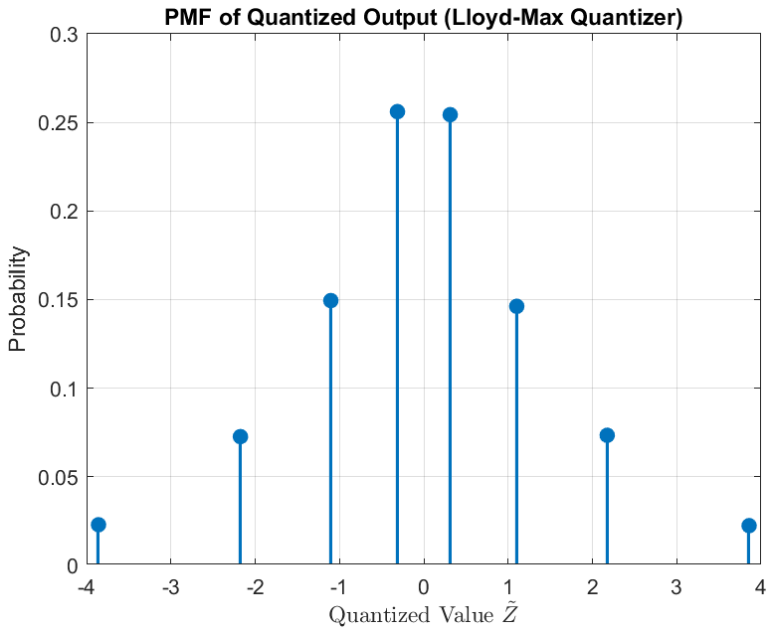

Fig. 8: PMF of Lloyd-Max quantization

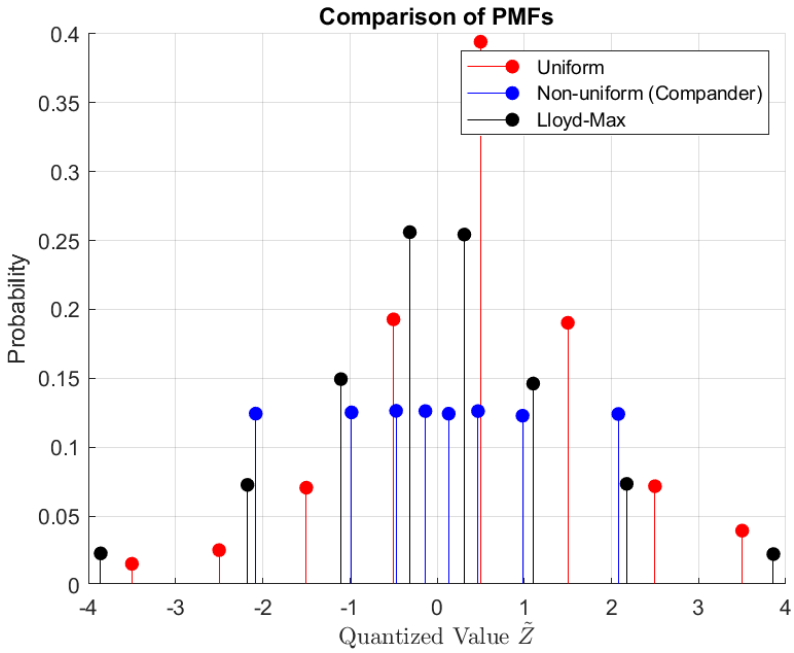

Fig. 9: PMF Comparison of all three quantization techniques

The Lloyd-Max quantizer gives a PMF similar to uniform quantizer however a more evenly distributed version of it. Also, Lloyd-Max quantizer aims to match the quantization resolution to source distribution as well as non-uniform quantizer. However, Lloyd-Max quantizer is better than non-uniform quantizer in that aspect, so it has the lowest MSE among all three quantizer.

Part m)

The Lloyd-Max algorithm is optimal in theory for minimizing quantization error when the source distribution is fully known and the number of levels is fixed. However, in practical scenarios, it may not always outperform other techniques due to assumptions about the source, sensitivity to outliers, and limitations of scalar quantization. Additionally, simpler methods like companding or uniform quantization may offer better trade-offs between performance and implementation complexity, especially when the source distribution is unknown or time-varying.

Because Lloyd-Max requires the exact probability density function (PDF) of the source it may not outperform other techniques.

Appendix

```
%% part b
rng(5);
%100 realizations of x and y
mu = 1;
N1 = 100;
X1 = exprnd(mu, [N1, 1]);
Y1 = exprnd(mu, [N1, 1]);
Z1 = X1 - Y1;

%100000 realizations of x and y
N2 = 100000;
X2 = exprnd(mu, [N2, 1]);
Y2 = exprnd(mu, [N2, 1]);
Z2 = X2 - Y2;

% Theoretical PDF function
fz = @(z) 0.5 * exp(-abs(z));
z_vals = linspace(-10, 10, 1000);
pdf_vals = fz(z_vals);

% Plot for 100 samples
figure;
histogram(Z1, 'Normalization', 'pdf');
hold on;
plot(z_vals, pdf_vals, 'r', 'LineWidth', 2);
title('Histogram of Z with N = 100 vs Theoretical PDF');
xlabel('z'); ylabel('PDF');
legend('Histogram', 'Theoretical PDF');
grid on;

% Plot for 100000 samples
figure;
histogram(Z2, 'Normalization', 'pdf');
hold on;
plot(z_vals, pdf_vals, 'r', 'LineWidth', 2);
title('Histogram of Z with N = 100000 vs Theoretical PDF');
xlabel('z'); ylabel('PDF');
legend('Histogram', 'Theoretical PDF');
grid on;

%% part c
Z2_squared_avg = mean(Z2.^2);
fprintf("Empirical E[Z^2] ≈ %.4f\n", Z2_squared_avg);
theoretical_power = 2;
fprintf("Theoretical E[Z^2]: %.4f\n", theoretical_power);
%The second moment of Z i.e. E[Z^2], is calculated as 2. (Second moment of
%Laplace distribution with scale 1).

%% part d
%I need to make Z2 a column vector for matlab to not give any errors
Z2 = Z2(:);
%level and min-max values
N = 8;
Zmin = -4;
Zmax = 4;
%because Laplace distribution is very close to 0 after -4 and 4, i choose
%the boundaries as -4 and 4.
```

```
delta = (Zmax - Zmin) / N;
levels = Zmin + delta/2 : delta : Zmax - delta/2;

%which bin samples z falls into
indices = round((Z2 - Zmin) / delta) + 1;
indices(indices < 1) = 1;
indices(indices > N) = N;

% Quantized version Z
Zq = levels(indices);
Zq = Zq(:);

% Quantization error and SQNR calculations
e = Z2 - Zq;
error_power = mean(e.^2);
signal_power = mean(Z2.^2);
SQNR_dB_uni = 10 * log10(signal_power / error_power);
disp(levels);
fprintf("Average quantization error power: %.4f\n", error_power);
fprintf("SQNR (dB): %.2f dB\n", SQNR_dB_uni);

%% part e
%Count how many times each quantization level appears.
%Divide those counts by the total number of samples to get probabilities.
%Plot the probabilities as a bar graph
% Compute PMF of quantized values

% Ensure Zq is a column
Zq = Zq(:);
% Count occurrences of each quantized level
pmf_uni = zeros(1, length(levels));
for i = 1:length(levels)
    pmf_uni(i) = sum(Zq == levels(i));
end

% Normalize to get probabilities
pmf_uni = pmf_uni / length(Zq);

figure;
stem(levels, pmf_uni, 'filled');
title('PMF of Quantized Output (Uniform Quantizer)');
xlabel('Quantized Value $\tilde{Z}$', 'Interpreter', 'latex');
ylabel('Probability');
grid on;

%% part f
% CDF of Laplace(0,1)
Fz = @(z) (z < 0) .* (0.5 * exp(z)) + (z >= 0) .* (1 - 0.5 * exp(-z));

% Inverse CDF
Fz_inv = @(u) (u < 0.5) .* log(2*u) + (u >= 0.5) .* (-log(2*(1 - u)));

%% part g
% Define CDF and inverse CDF of Laplace(0,1)
Fz = @(z) (z < 0) .* (0.5 * exp(z)) + (z >= 0) .* (1 - 0.5 * exp(-z));
Fz_inv = @(u) (u < 0.5) .* log(2*u) + (u >= 0.5) .* (-log(2*(1 - u)));

% Quantizer parameters
N = 8;
```

```
delta = 1 / N;

% Uniform quantizer in compressed domain (values in [0, 1])
% Define quantizer levels (midpoints)
u_levels = delta/2 : delta : 1 - delta/2;

% Create input z values for plotting
z_vals = linspace(-6, 6, 1000);
u_vals = Fz(z_vals);  % Compress

% Perform uniform quantization in [0,1]
% Map compressed values to bin indices
bin_indices = min(N, max(1, ceil(u_vals / delta)));
u_quantized = u_levels(bin_indices);  % Quantized compressed values

% Expand back to non-uniform quantized values
z_quantized = Fz_inv(u_quantized);

% Final quantizer output: Q(z)
Qz = z_quantized;

% Plot Q(z) vs z
figure;
plot(z_vals, Qz, 'LineWidth', 2);
xlabel('Input z');
ylabel('Quantized Q(z)');
title('Non-uniform Quantizer: Q(z) vs z');
grid on;

%% part h
% Use Z2 as your original Laplace samples
Z2 = Z2(:);  % Ensure it's a column vector

% Define CDF and inverse CDF of Laplace(0,1)
Fz = @(z) (z < 0) .* (0.5 * exp(z)) + (z >= 0) .* (1 - 0.5 * exp(-z));
Fz_inv = @(u) (u < 0.5) .* log(2*u) + (u >= 0.5) .* (-log(2*(1 - u)));

% Quantizer parameters
N_levels = 8;
delta = 1 / N_levels;
u_edges = 0 : delta : 1;
u_levels = delta/2 : delta : 1 - delta/2;

% Step 1: Compress Z2 using CDF
u_vals = Fz(Z2);

% Step 2: Uniform quantization in compressed domain
[~, bin_indices] = histc(u_vals, u_edges);
bin_indices(bin_indices < 1) = 1;
bin_indices(bin_indices > N_levels) = N_levels;
u_quantized = u_levels(bin_indices);

% Step 3: Expand back to get quantized version of Z2
Z2_companded_quantized = Fz_inv(u_quantized);
Z2_companded_quantized = Z2_companded_quantized(:);
% Step 4: Calculate quantization error and SQNR
e = Z2 - Z2_companded_quantized;
error_power = mean(e.^2);
signal_power = mean(Z2.^2);
```

```
SQNR_dB_nonuni = 10 * log10(signal_power / error_power);

% Display results
fprintf('Average quantization error power (non-uniform): %.4f\n', error_power);
fprintf('SQNR for non-uniform quantization: %.2f dB\n', SQNR_dB_nonuni);

%% part i
% Ensure output is a column
Z2_companded_quantized = Z2_companded_quantized(:);

% Define reconstruction levels again
N = 8;
delta = 1 / N;
u_levels = delta/2 : delta : 1 - delta/2;
reconstruction_levels = Fz_inv(u_levels);  % Same as in previous step

% Count occurrences
pmf_nonuniform = zeros(1, N);
for i = 1:N
    pmf_nonuniform(i) = sum(Z2_companded_quantized == reconstruction_levels(i));
end

% Normalize
pmf_nonuniform = pmf_nonuniform / length(Z2_companded_quantized);

% Plot PMF
figure;
stem(reconstruction_levels, pmf_nonuniform, 'filled');
title('PMF of Quantized Output (Non-uniform Quantizer)');
xlabel('Quantized Value $\tilde{Z}$', 'Interpreter', 'latex');
ylabel('Probability');
grid on;

figure;
hold on;

stem(levels, pmf_uni, 'r', 'filled', 'DisplayName', 'Uniform');
stem(reconstruction_levels, pmf_nonuniform, 'b', 'filled', 'DisplayName', 'Non-
uniform');

xlabel('Quantized Value $\tilde{Z}$', 'Interpreter', 'latex');
ylabel('Probability');
title('Comparison of PMFs: Uniform vs Non-uniform Quantizer');
legend('show');
grid on;

%% part j
% Parameters
M = 8;                          % Number of quantization levels
max_iter = 100;                 % Max iterations
tol = 1e-6;                     % Convergence tolerance
range = 6;                      % Support of Z
z = linspace(-range, range, 10000);  % Fine grid for numerical integration
pdf_z = @(z) 0.5 * exp(-abs(z));     % Laplace(0,1) PDF

% Initialize reconstruction levels (e.g. linearly spaced)
xq = linspace(-range * 0.9, range * 0.9, M);

% Lloyd-Max Iteration
```

```
for iter = 1:max_iter
    % Step 1: Compute thresholds
    t = zeros(1, M+1);
    t(2:M) = 0.5 * (xq(1:M-1) + xq(2:M));
    t(1) = -range;
    t(end) = range;

    xq_new = zeros(1, M);

    % Step 2: Compute centroids
    for q = 1:M
        idx = (z >= t(q)) & (z < t(q+1));
        num = trapz(z(idx), z(idx) .* pdf_z(z(idx)));
        den = trapz(z(idx), pdf_z(z(idx)));
        xq_new(q) = num / den;
    end

    % Step 3: Check for convergence
    if max(abs(xq_new - xq)) < tol
        fprintf('Converged after %d iterations.\n', iter);
        break;
    end

    xq = xq_new;
end

% Final thresholds and levels
thresholds = t;
reconstruction_levels = xq;

% Display results
fprintf('\nFinal Decision Thresholds:\n');
disp(thresholds);

fprintf('Final Reconstruction Levels:\n');
disp(reconstruction_levels);

% Plot the quantization function Q(z)
figure;
z_plot = linspace(-range, range, 1000);
Qz = zeros(size(z_plot));

for q = 1:M
    Qz(z_plot >= thresholds(q) & z_plot < thresholds(q+1)) =
reconstruction_levels(q);
end

plot(z_plot, Qz, 'LineWidth', 2);
xlabel('Input z');
ylabel('Quantized Q(z)');
title('Lloyd-Max Quantization Function');
grid on;

%% part k
% Ensure Z2 is a column vector
Z2 = Z2(:);
N = length(Z2);

%Lloyd-Max Quantizer steps
```

```
Z2_lloyd_quantized = zeros(N, 1);
%Calculation of decision thresholds
for i = 1:length(thresholds)-1
    in_region = Z2 >= thresholds(i) & Z2 < thresholds(i+1);
    Z2_lloyd_quantized(in_region) = reconstruction_levels(i);
end

% Handle edge case for Z2 == thresholds(end)
Z2_lloyd_quantized(Z2 == thresholds(end)) = reconstruction_levels(end);

% Computation of quantization error and SQNR
error_lloyd = Z2 - Z2_lloyd_quantized;
error_power_lloyd = mean(error_lloyd.^2);
SQNR_lloyd_dB = 10 * log10(signal_power / error_power_lloyd);

% Display results
fprintf('\nLloyd-Max Quantization Results:\n');
fprintf('Quantization Error Power: %.4f\n', error_power_lloyd);
fprintf('SQNR (dB): %.2f dB\n', SQNR_lloyd_dB);

fprintf('\n---- SQNR Comparison ----\n');
fprintf('Uniform Quantizer (Part d):       %.2f dB\n', SQNR_dB_uni);
fprintf('Compander-based Quantizer (Part h): %.2f dB\n', SQNR_dB_nonuni);
fprintf('Lloyd-Max Quantizer (Part i):     %.2f dB\n', SQNR_lloyd_dB);

%% part l
Z2_lloyd_quantized = Z2_lloyd_quantized(:);
reconstruction_levels = reconstruction_levels(:)';

% Initialize PMF
pmf_lloyd = zeros(1, length(reconstruction_levels));

% Count occurrences of each quantized level
for i = 1:length(reconstruction_levels)
    pmf_lloyd(i) = sum(Z2_lloyd_quantized == reconstruction_levels(i));
end

% Normalize to get probabilities
pmf_lloyd = pmf_lloyd / length(Z2_lloyd_quantized);

% Plot the PMF
figure;
stem(reconstruction_levels, pmf_lloyd, 'filled', 'LineWidth', 1.5);
xlabel('Quantized Value $\tilde{Z}$', 'Interpreter', 'latex');
ylabel('Probability');
title('PMF of Quantized Output (Lloyd-Max Quantizer)');
grid on;

figure;
hold on;
stem(levels, pmf_uni, 'r', 'filled', 'DisplayName', 'Uniform');
stem(Fz_inv(u_levels), pmf_nonuniform, 'b', 'filled', 'DisplayName', 'Non-uniform
(Compander)');
stem(reconstruction_levels, pmf_lloyd, 'k', 'filled', 'DisplayName', 'Lloyd-Max');
xlabel('Quantized Value $\tilde{Z}$', 'Interpreter', 'latex');
ylabel('Probability');
title('Comparison of PMFs');
legend('show');
grid on;
```

$$F_Z(z) = \mathbb{P}(Z < z) = \mathbb{P}(X - Y < z)$$

$z > 0$ case

$$\mathbb{P}(X-Y<z) = \int_0^{\infty}\int_0^{z+y} e^{-x}\cdot e^{-y}\,dx\,dy$$

$$\int_0^{x=z+y} e^{-x}\,dx = -e^{-x}\Big|_0^{z+y} = \boxed{= -e^{-(z+y)} + 1}$$

$$\int_0^{\infty} e^{-y}\cdot\left(-e^{-(z+y)}+1\right)dy = \int_0^{\infty}\left(-e^{-(z+2y)} + e^{-y}\right)dy$$

$$\Rightarrow \frac{e^{-(z+2y)}}{2} - e^{-y}\Big|_0^{\infty} = \boxed{1 - \frac{e^{-z}}{2}} \qquad z>0$$

$z < 0$ case

$$\mathbb{P}(X-Y<z) = \int_{-z}^{\infty}\int_0^{z+y} e^{-x}\cdot e^{-y}\,dx\,dy$$

$$-e^{-x}\Big|_0^{z+y} = -e^{-(z+y)} + 1$$

$$\Rightarrow \int_0^{\infty}\left(-e^{-(z+2y)} + e^{-y}\right)dy = \frac{e^{-(z+2y)}}{2} - e^{-y}\Big|_{-z}^{\infty}$$

$$= -\frac{e^{z}}{2} + e^{z} = \boxed{\frac{e^{z}}{2}} \qquad z<0$$

$$f_Z(z) = \begin{cases} \frac{e^z}{2}, & z<0 \\ 1-\frac{e^{-z}}{2}, & z>0 \end{cases}$$

$$\frac{d}{dz}F_Z(z) = f_Z(z)$$

$$f_Z(z) = \begin{cases} \frac{e^z}{2}, & z<0 \\ \frac{e^{-z}}{2}, & z>0 \end{cases}$$