EEE 431 Spring 2024/25 MATLAB Assignment 1

a)

We want to compute $P(Z<z) = F(z) = P(X-Y<z)$

Case 1 → $z<0$, $P(X-Y<z) = \int_{-z}^{\infty}\int_{0}^{z+y} e^{-x}e^{-y}\,dx\,dy$

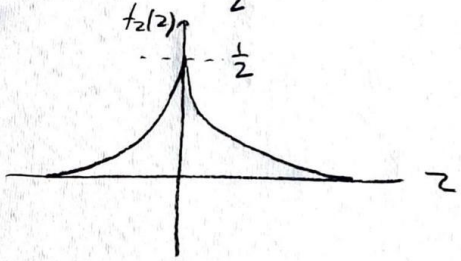
$= \int_{-z}^{\infty} e^{-y}\left(-e^{-x}\Big|_0^{z+y}\right) = \int_{-z}^{\infty}\left(-e^{-2y-z}+e^{-y}\right)dy = \frac{e^{-z}}{2}\cdot e^{-2y}\Big|_{-z}^{\infty} + \left(e^{-y}\Big|_{-z}^{\infty}\right)$

$= -\frac{e^{z}}{2} + e^{z} = \frac{e^{z}}{2}$

case 2 → $z>0$, $\int_0^{\infty}\int_0^{z+y} e^{-x}e^{-y}\,dx\,dy = \frac{e^{-z}}{2}\cdot e^{-2y}\Big|_0^{\infty} - e^{-y}\Big|_0^{\infty} = 1-\frac{e^{-z}}{2}$

$$F(z) = \begin{cases} \frac{1}{2}e^{z}, & z<0 \\ 1-\frac{1}{2}e^{-z} & z\geq 0 \end{cases}, \quad f_z(z) = \frac{dF(z)}{dz}, \quad f_Z(z) = \begin{cases} \frac{1}{2}e^{z} & z<0 \\ \frac{1}{2}e^{-z} & z\geq 0 \end{cases}$$

$$f_Z(z) = \frac{1}{2}\cdot e^{-|z|} \quad \text{for } -\infty < z < \infty.$$



b) The samples X and Y are generated by using inverse transform. After generating X nad Y sample Z is determined by simply X-Y. The X, Y sampels ar egenerated as following.

$$F_X(x) = 1 - e^{-x}$$

$$F_X(x)^{-1} = -\ln(1-U)\,, U\sim uniform(0,1)$$

Since U is uniformly distributed, 1-U is also uniform hence the final transformation necessary to generate samples is:

$$F_X(x)^{-1} = \ln(U)$$

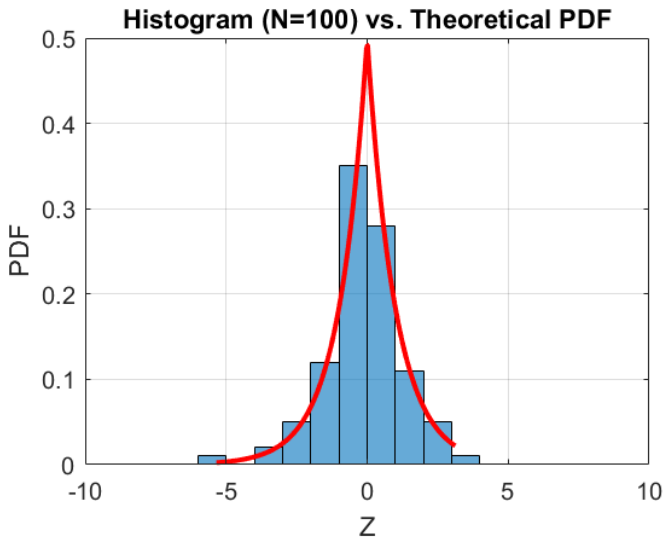


Figure 1 Histogram of realizations of Z with 100 samples

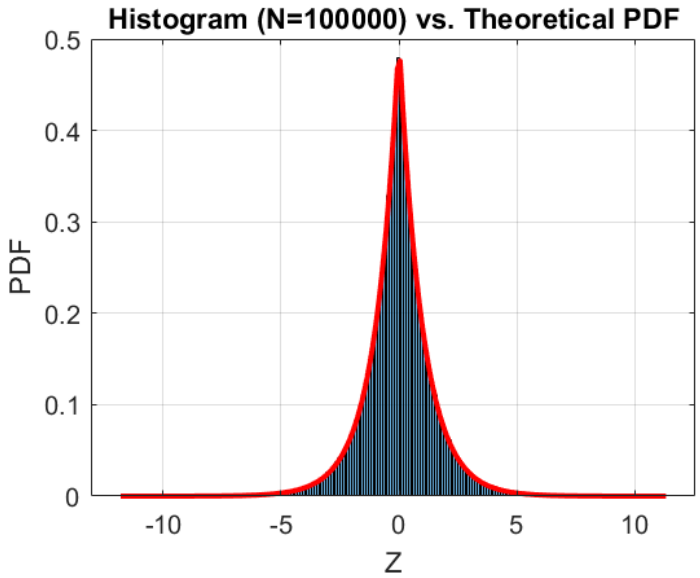


Figure 2 Histogram of realizations of Z with 100000 samples

After observing Figure 1 and Figure 2 it is obvious that in the 100000 sample case the samples tightly follow the pdf of Z compared to the 100 sample case. The empirical distribution tends to converge the theoratical distribution as sample size increases.

c) The theoratical value for average source power is computed as following.

$$E[Z] = E[X] - E[Y] = 0$$

$$Var(Z) = Var(X) + Var(Y) = 2$$

$$E[Z^2] = Var(Z) + E[Z]^2 = 2$$

By using MATLAB, the empirical mean is found by using mean function which can be observed in the following figure. The empirical mean is significantly close to the theoratical mean for 100000 samples.

```
Empirical mean of Z^2 (N=100000) = 2.018235
```

Figure 3 Demonstration of empirical mean

d) In this part it is desired to perform uniform quantization with N=8 levels and compute related SQNR. In uniform quantization the $\tilde{Z}$ value that minimizes MSE is the E[Z] which corresponds to mid-points of boundires. The lower and upper bounds of the finite interval for Z is determined by the minimum and maximum sampled values of Z and throughout the assignment Z is generated only one time to be consistent in comparisons of quantization methods. For the uniform quantization boundires (9 points) and reconstruction values (8 points) are provided below. Also, after computationally obtaining quantization errors, average power of quantization error and relevant SQNR in db can be observed in Figure 4.

$boundries = \{$ -11.3396 -8.6490 -5.9584 -3.2678 -0.5772 2.1134 4.8040 7.4946 10.1852$\}$

$\tilde{Z} = \{$ -9.9943 -7.3037 -4.6131 -1.9225 0.7681 3.4587 6.149 8.8399$\}$

```
Average power of quantization error = 0.7839
SQNR (dB) = 4.0667 dB
```

Figure 4 Average power of quantization error and SQNR in dB

As it can be observed the quantizer performs poorly in with this approach, the quantizer should be optimized such that the outer bins between the zmin and zmax interval will maximize the SQNR. In other words a1(outer bin) should be determined with trial and error in the following equation.

$$D = \int_{-\infty}^{a_1} \left[x - (a_1 - \Delta/2)\right]^2 f_X(x)dx + \sum_{i=1}^{N-2} \int_{a_1+(i-1)\Delta}^{a_1+i\Delta} \left[x - (a_1 + i\Delta - \Delta/2)\right]^2 f_X(x)dx$$
$$+ \int_{a_1+(N-2)\Delta}^{\infty} \left[x - (a_1 + (N-2)\Delta + \Delta/2)\right]^2 f_X(x)dx$$

In MATLAB, a1 is user defined and now boundries are determined as 2 large intervals from zmin to negative a1, a1 to zmax and between –a1 and a1 is seperated into 6 equal reigons just as in the equation above, when a1 = 5 the highest SQNR is achieved. New boundries, reconstuction levels and, SQNR value can be observed as following.

$boundries = \{$ -11.3841 -5.0000 -3.3333 -1.6667 0 1.6667 3.3333 5.0000 13.0455$\}$
$Q(z) = \{$ -8.1920 -4.1667 -2.5000 -0.8333 0.8333 2.5000 4.1667 9.0228$\}$

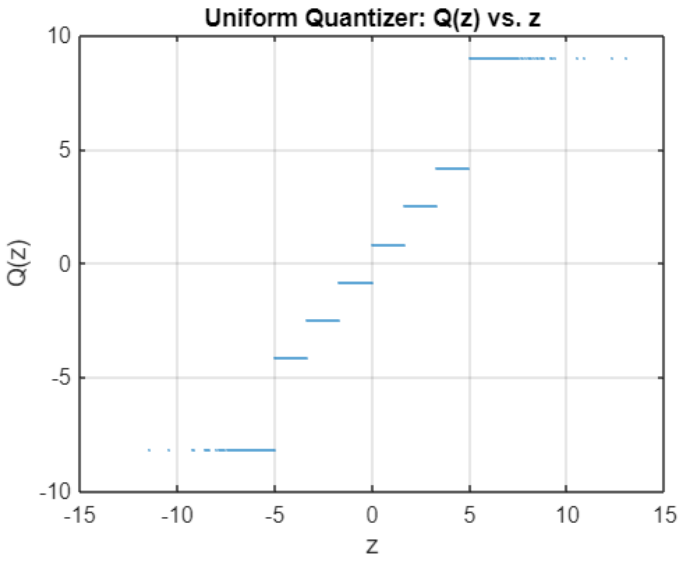


Figure 5 Vizualization of boundries and reconstruction values for optimized uniform quantizer

```
Average power of quantization error = 0.3085
SQNR (dB) = 8.1039 dB
```

Figure 6 Average power of quantization error and SQNR in Db of optimized uniform quantizer

e) Based on the occurance frequency the pmf can be observed in Figure 5. The empirical pmf of uniform quantizer makes sense because the boundires are uniformly seperated and the reconstruction levels are distanced equally, since empirical distirbution tends to be distributed by the Laplacian pdf as it can be observed the frequency of quantization values around 0 is higher.

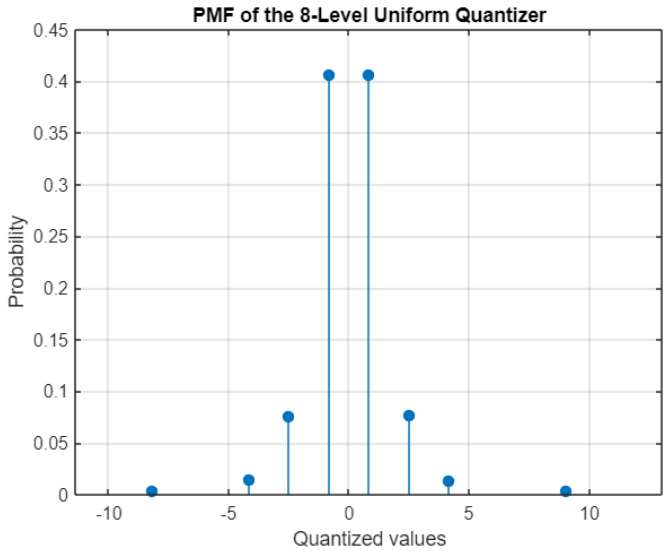


Figure 7 PMF of the 8-level optimized Uniform Quantizer

f) For the compander, we should first implement $F_Z(z)$ and $F_Z(z)^{-1}$ in MATLAB. Before implementing in MATLAB $F_Z(z)$ and $F_Z(z)^{-1}$ are derived analytically. $F_Z(z)$ is already derived in part a so the expressions for the inverse function will be provided.

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

g) In this part the methadology is as following. First Z values are mapped to uniform u (uniform between [0,1] due to CDF transformation) domain (compressor). Afterwards 8-level uniform quantization is applied and as the last step u values are mapped back to Z domain (expander). The boundries, reconstruction levels and Q(z) vs z plots can be observed accordingly.

$boundries =$ { -12.2436 -1.3863 -0.6931 -0.2877 -0.0000 0.2877 0.6931 1.3862 11.2030}
Q(z)= {-2.0794 -0.9808 -0.4700 -0.1335 0.1335 0.4700 0.9808 2.0793}

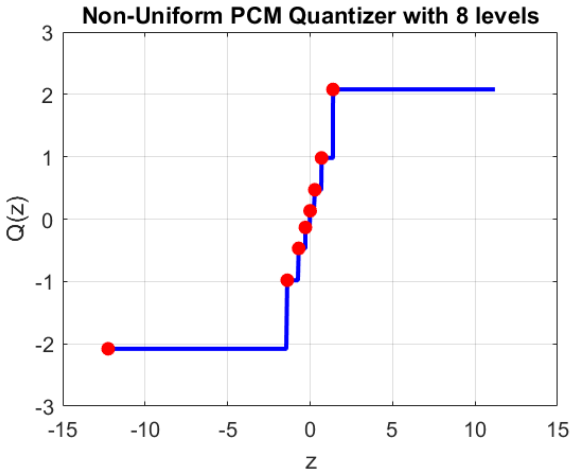


Figure 8 Vizualization of boundries and reconstruction values for non-uniform PCM

h) The average power of quantization error and related SQNR in db can be observed in Figure 7. When compared with the SQNR and average power of quantization error values of the uniform quantizer and optimized uniform quantizer, the non-uniform PCM quantizer performs better with less average power of quantization error and higher SQNR. However it is beneficial to mention that there are no significant difference when compared to optimized uniform quantizer. The improvement is a result of non-uniformly seperated reconstruction levels and boundries. From the values above and Figure 6 it is clear to see that boundries are shorter around z=0 which decreases the quantization error by assigning the frequent elements to more sensitive reconstruction values.

```
Average power of quantization error = 0.279865
SQNR (dB) = 8.504078 dB
```

Figure 9 Average power of quantization error and related SQNR in dB

i)The pmf of non-uniform PCM quantizer can be observed in Figure 8 which almost has a uniform distribution. As explained previosly a compression and expression operation is performed to Z samples. As a result of this, a improvement in quantization can be observed which is explained by the denser quantization values around 0. The uniform quantization in u domain verifies that after expansion the lowest and highest quantization reigons are large with smaller quantization reigons

around 0. It is known that Z tends to have values around 0 when sample size increases hence more frequent values with smaller intervals and larger intervals with less frequent values outputs approximately the same probability.

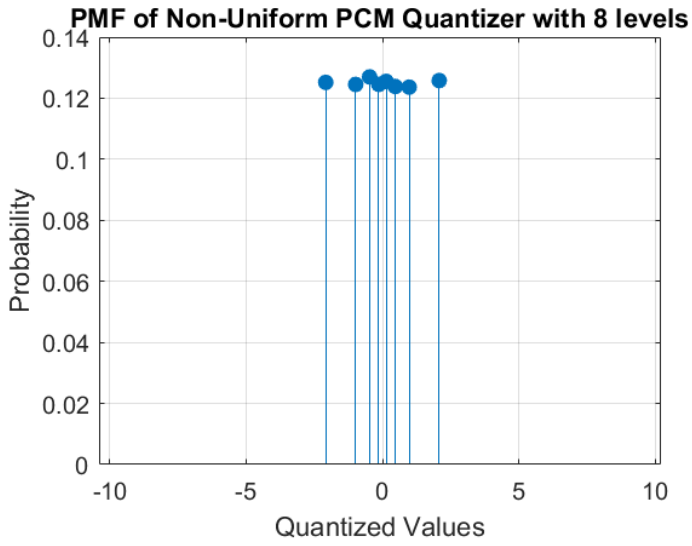


Figure 10 pmf of non-uniform PCM quantizer with 8 levels

j)The Loyd-Max algorithm is an iterative algorithm to determine optimal reconstruction values and boundries. The fundamental idea of the algorithm is to determine boundries and quantization values according to the pdf. Previosly it was mentioned that the value that minimizes MSE is the expected value of the random variable hence, to find the optimal reconstruction level expected value of the specified reigon is calculated and as the second step boundries are updated according to the obtained reconstruction values as following.

$$\widetilde{z_i} = \frac{\int_{b_{i-1}}^{b_i} z f_z(z) dz}{\int_{b_{i-1}}^{b_i} f_z(z) dz}$$

$$b_i = \frac{1}{2}(\widetilde{z_i} + \widetilde{z_{i+1}})$$

For the 8-level Lloyd-Max quantizer implementation in MATLAB, the finite interval is determined to be between minimum value of Z and maximum value of Z which are the same for the previous quantizers to make reasonable observations. The implementation consists of 1 outer for loop and 3 inner for loops. The outer for loop is responsible to iterate through max number of iterations which is specified by the user, the first inner loop is to assign sample Z values to the nearest reigon. The second inner for loop is responsible for taking the expectation of Z samples within the reigon by using mean function which is equalivent to the above integral as described previously. In the final for loop boundries are calculates according to the previously provided equation (mean of two consecutive calculated reconstruction values). It is benefitial to mention that after 50 iterations no improvement in SQNR is observed and the final boundries and reconstruction levels can be observed as following.

$boundries = \{$ -10.3705 -3.4689 -1.8725 -0.8372 -0.0774 0.6716 1.6700 3.2470 9.6884$\}$

$\tilde{Z} = \{$ -4.4581 -2.4797 -1.2652 -0.4092 0.2544 1.0888 2.2512 4.2429$\}$

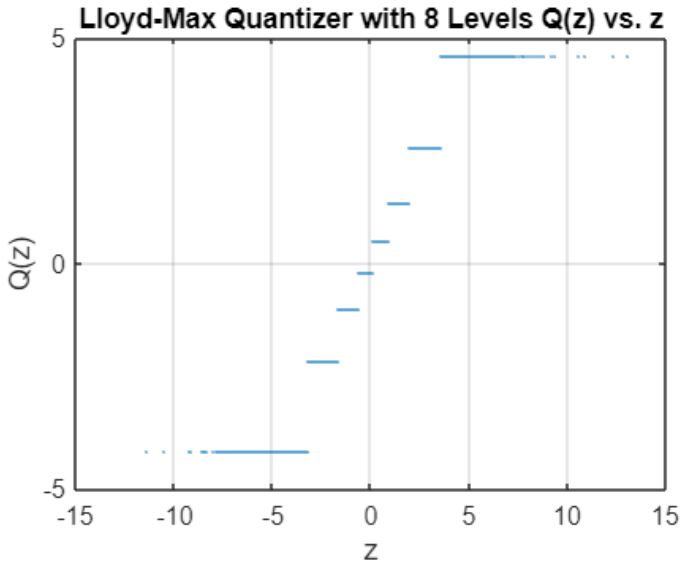


Figure 11 Vizualization of boundries and reconstruction values for Lloyd-Max qauntizer

k)

```
Average power of quantization error = 0.106469
SQNR (dB) = 12.703089 dB
```

Figure 9 Average power of quantization error and SQNR in Db

Due to it's iterative trait and directly optimizing the MSE of quantization error by taking the pdf into considiration, Lloyd-Max algorithm, achieves the highest SQNR with the minimum average power of quantization error which makes the Lloyd-Max algorithm as the most preferable option for this quantization problem.

l) The pmf of Lloyd-Max quantizer can be observed in Figure 10. As expected the quantization boundries are smaller in terms of lenght around 0. As a result of this the frequent elements are assigned to reconstruction levels near 0. The pmf is different than the compander pmf since there is no transformation to u domain but follows a similler pattern with the uniform quantizer. However due to optimization as it can be observed by comparing Figure 5 and Figure 10, the Lloyd-Max quantizer is more precise. The quantization values are closer to 0.

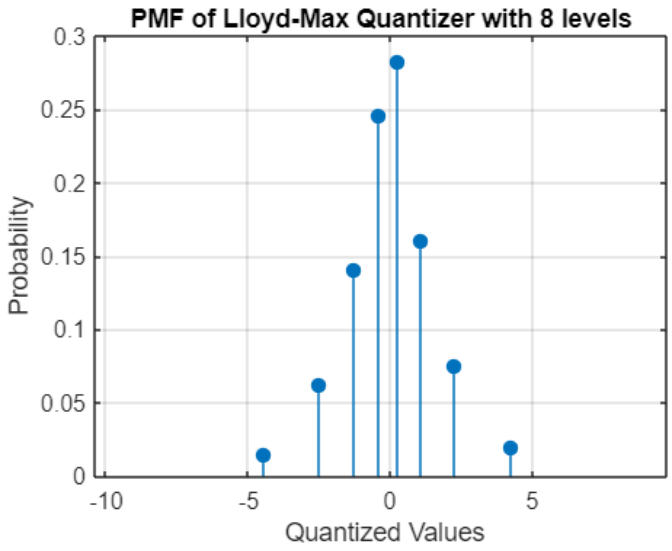


Figure 10 pmf of Lloyd-Max quantizer with 8 levels

m) Lloyd max algorithm aims to minimize the MSE, whereas compander approach try to quantize by projecting the data to an uniform space and than expand it to original space by performing uniform quantization in the uniform space. In real life applications, such as speech processing, in most of the cases perceptual quality is the priority rather than MSE. Hence with the projectio and expansion logic described previously, speech signals are mimiced better by the compander quantizer in some cases. In addition due to its iterative algorithm Lloyd-max consumes considerable amount of power, if the computational efficiency is prioritized, for low variance dsitributions uniform quantizer can represent data sufficiently with low computanial power hence can be preferred over the Lloyd-max quantizer.

APPENDIX

MATLAB Codes:

```
N1 = 100;
X1 = -log(rand(N1,1));
Y1 = -log(rand(N1,1));
Z1 = X1 - Y1;


figure(1);
histogram(Z1, 'Normalization','pdf');
hold on;
x_axis = linspace(min(Z1), max(Z1), 200);
pdf = 0.5 * exp(-abs(x_axis));
%plot(x_axis, pdf, 'r', 'LineWidth', 2);


title('Histogram (N=100) vs. Theoretical PDF');
xlim([-10,10]);
xlabel('Z');
ylabel('PDF');
grid on;


N2 = 100000;
X2 = -log(rand(N2,1));
Y2 = -log(rand(N2,1));
Z2 = X2 - Y2;
figure(2);
histogram(Z2, 'Normalization','pdf');
hold on;
x_axis_2 = linspace(min(Z2), max(Z2), 200);
pdf_2 = 0.5 * exp(-abs(x_axis_2));
%plot(x_axis_2, pdf_2, 'r', 'LineWidth', 2);


title('Histogram (N=100000) vs. Theoretical PDF');
xlabel('Z');
ylabel('PDF');
grid on;


N = 100000;
X = -log(rand(N, 1));
Y = -log(rand(N, 1));
Z = X - Y;
avg_Z_sq = mean(Z.^2);
fprintf('Empirical mean of Z^2 (N=%d) = %f\n', N, avg_Z_sq);
```

```
Z_min = min(Z);
Z_max = max(Z);
a = 5.0;



edges = [Z_min, linspace(-a, a, 7), Z_max];


mid = 0.5 * (edges(1:end-1) + edges(2:end));
disp(edges);
disp(mid);
Z_recons = zeros(size(Z));
for i = 1:N
    idx = find(Z(i) >= edges(1:end-1) & Z(i) < edges(2:end), 1);
    if isempty(idx)
        idx = 8;
    end
    Z_recons(i) = mid(idx);
end
figure;
plot(Z, Z_recons, '.', 'MarkerSize', 1);
xlabel('z');
ylabel('Q(z)');
title('Uniform Quantizer: Q(z) vs. z');
grid on;

E = Z - Z_recons;
mse = mean(E.^2);


Z_power = mean(Z.^2);
SQNR = Z_power / mse;
SQNR_dB = 10 * log10(SQNR);
fprintf('Average power of quantization error = %.4f\n', mse);
fprintf('SQNR (dB) = %.4f dB\n', SQNR_dB);


elements = zeros(1, 8);


%e
for i = 1:N

    idx = find(mid == Z_recons(i), 1);
    elements(idx) = elements(idx) + 1;
end
```

```
p = elements / N;
figure(3);
stem(mid, p, 'filled');
xlim([Z_min, Z_max]);
xlabel('Quantized values');
ylabel('Probability');
title('PMF of the 8-Level Uniform Quantizer');
grid on;


functionVals = 0;
y_min = FZ(Z_min);
y_max = FZ(Z_max);




edges_y = linspace(y_min, y_max, 9);


midpoints_y = 0.5*(edges_y(1:end-1) + edges_y(2:end));




num_plot_points = 100000;
z_val = linspace(Z_min, Z_max, num_plot_points);
q_z = zeros(size(z_val));



for i = 1:num_plot_points
    y_val = FZ(z_val(i));

    y_val = max(y_min, min(y_max, y_val));



    idx = find((y_val >= edges_y(1:end-1)) & (y_val < edges_y(2:end)), 1);
    if isempty(idx)

        idx = 8;
    end
    y_mid = midpoints_y(idx);



    q_z(i) = FZ_inv(y_mid);
end
figure(4);
figure;
plot(z_val, q_z, 'b-', 'LineWidth', 2);
```

```
xlabel('z');
ylabel('Q(z)');
title('Non-Uniform PCM Quantizer with 8 levels');
grid on; hold on;


z_bound = zeros(1,9);
for k = 1:9
    z_bound(k) = FZ_inv( max(y_min, min(y_max, edges_y(k))) );
end


z_mid = zeros(1,8);
for k = 1:8
    z_mid(k) = FZ_inv(midpoints_y(k));
end
disp(z_bound);
disp(z_mid);


plot(z_bound(1:end-1), z_mid, 'ro','MarkerFaceColor','r');
q_z_compander = zeros(size(Z));
for i = 1:length(Z)
    y_val = FZ(Z(i));
    y_val = max(y_min, min(y_max, y_val));

    idx = find(y_val >= edges_y(1:end-1) & y_val < edges_y(2:end), 1);
    if isempty(idx)
        idx = 8;
    end
    y_mid = midpoints_y(idx);

    q_z_compander(i) = FZ_inv(y_mid);
end


E_compander = Z - q_z_compander;
mse_compander = mean(E_compander.^2);


Z_power = mean(Z.^2);
SQNR_compander = Z_power / mse_compander;
SQNR_compander_dB = 10 * log10(SQNR_compander);


fprintf('Average power of quantization error = %f\n', mse_compander);
fprintf('SQNR (dB) = %f dB\n', SQNR_compander_dB);


unique_vals = unique(q_z_compander);


elements_compander = zeros(1, length(unique_vals));
```

```
for i = 1:N
    idx = find(unique_vals == q_z_compander(i), 1);
    elements_compander(idx) = elements_compander(idx) + 1;
end


p_compander = elements_compander / N;


figure(6);
stem(unique_vals, p_compander, 'filled');
xlim([Z_min, Z_max]);
xlabel('Quantized Values');
ylabel('Probability');
title('PMF of Non-Uniform PCM Quantizer with 8 levels');
grid on;
```

```
function comp = FZ(z_in)
    comp = zeros(size(z_in));
    for i = 1:length(z_in)
        if z_in(i) < 0
            comp(i) = 0.5 * exp(z_in(i));
        else
            comp(i) = 1 - 0.5 * exp(-z_in(i));
        end
    end
end


function expand = FZ_inv(p_in)
    expand = zeros(size(p_in));
    for i = 1:length(p_in)
        if p_in(i) < 0.5
            expand(i) = log(2 * p_in(i));
        else
            expand(i) = -log(2 * (1 - p_in(i)));
        end
    end
end



quant_int = 8;
max_iter = 50;


Z_min = min(Z);
Z_max = max(Z);
```

```
bound = linspace(Z_min, Z_max, quant_int+1);


recons = 0.5 * (bound(1:end-1) + bound(2:end));


prev_b = bound;
prev_r = recons;


for iter = 1:max_iter

    region_z = zeros(size(Z));

    for j = 1:length(Z)
        idx = find(Z(j) >= bound(1:end-1) & Z(j) < bound(2:end), 1);
        if isempty(idx)
            idx = quant_int;
        end
        region_z(j) = idx;
    end

    for k = 1:quant_int
        Z_in_region = Z(region_z == k);
        if ~isempty(Z_in_region)
            recons(k) = mean(Z_in_region);
        end

    end


    for k = 2:quant_int
        bound(k) = 0.5 * (recons(k-1) + recons(k));
    end
    bound(1) = Z_min;
    bound(end) = Z_max;



    prev_b = bound;
    prev_r = recons;
end


fprintf('Final boundaries b:\n');
disp(bound);
fprintf('Final reconstruction levels r:\n');
disp(recons);
```

```
q_z_LM = zeros(size(Z));
for i = 1:length(Z)
    idx = find(Z(i) >= bound(1:end-1) & Z(i) < bound(2:end), 1);
    if isempty(idx)
        idx = quant_int;
    end
    q_z_LM(i) = recons(idx);
end
plot(Z, q_z_LM, '.', 'MarkerSize', 1);
xlabel('z');
ylabel('Q(z)');
title('Lloyd-Max Quantizer with 8 Levels Q(z) vs. z');
grid on;


E_LM = Z - q_z_LM;
mse_LM = mean(E_LM.^2);
average_power_LM = mean(Z.^2);
SQNR_LM = average_power_LM / mse_LM;
SQNR_LM_dB = 10 * log10(SQNR_LM);


fprintf('Average power of quantization error = %f\n', mse_LM);
fprintf('SQNR (dB) = %f dB\n', SQNR_LM_dB);


elements_LM = zeros(1, quant_int);
for i = 1:length(q_z_LM)
    idx = find(recons == q_z_LM(i), 1);
    elements_LM(idx) = elements_LM(idx) + 1;
end
p_LM = elements_LM / length(q_z_LM);


figure;
stem(recons, p_LM, 'filled');
xlim([Z_min, Z_max]);
xlabel(' Quantized Values');
ylabel('Probability');
title(sprintf('PMF of Lloyd-Max Quantizer with 8 levels'));
grid on;
```