Communication and Electronics ystems

ECE34: Digital Communications

Major Task

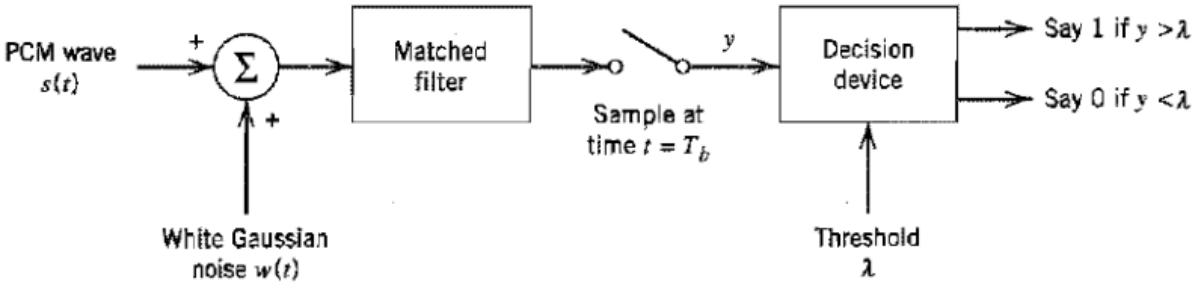
ID	Name	Contribution
21P0177	Omar Osama Fathi	Encoding the signals
21P0407	Ahmed Mostafa Gomaa	Adding the noise
21P0140	Julia Rami Emil Amir	Matched filters and Calculating threshold and BER

Submitted to :

Dr. Hussein Abdel Atty Elsayed Mohamed

Eng. Mariz Ashraf

This MATLAB code generates a random bit stream, encodes it using three different encoding techniques (Polar NRZ PCM, Polar RZ PCM, and 4-Ary PCM with Gray Coding), and displays the results. Here's a detailed explanation of the code (we will follow this system):



Code Breakdown

1. Initialization

```
clear
clc
```

- Clears the MATLAB workspace and command window to ensure a fresh start for execution.

---

2. Initialization and Parameters

```
% Number of bits to generate
Num_bits = 100000;
Bits = [0, 1];
```

- Num_bits: Defines the length of the bit stream to generate.
- Bits: Represents the possible values in the binary bit stream (0 and 1).

---

3. Time Vectors

```
% Generate random bit stream Tb = 1sec
time_NRZ = 0:1:Num_bits;
time_RZ = 0:0.5:Num_bits;
time_Ary = 0:1:Num_bits;
```

- **time_NRZ**: Assumes one sample per bit.
- **time_RZ**: Assumes two samples per bit (used for RZ encoding).
- **time_Ary**: Assumes two bits per symbol (used for 4-Ary encoding).

---

4. Random Bit Stream Generation

```
Stream_bits = datasample(Bits, Num_bits);
```

- Generates a random sequence of 0s and 1s, forming the bit stream of length Num_bits.
- datasample: Randomly samples from the Bits array.

5. Eb/No Values

```
%Values of Eb/No
EbN0_dB = -10 : 2 : 6;
```

- Specifies the range of Eb/N0 (Energy per bit to noise power density ratio) in decibels for further analysis.

---

6. Polar NRZ PCM Encoding

```
% Polar NRZ PCM Encoding
Encode_NRZ = Stream_bits * 2 - 1;
```

- Encodes each bit as follows:
  - 0 → -1
  - 1 → 1
- This results in a bipolar signal where bits are mapped to amplitudes of ±1.

```
Encode_NRZ_plot = Encode_NRZ;
Encode_NRZ_plot(length(Stream_bits) + 1) = -1;
```

- Prepares the encoded signal for plotting by appending an extra point for clarity.

---

7. Polar RZ PCM Encoding

```
% Polar RZ PCM Encoding
Encode_RZ_plot = zeros(1, Num_bits * 2 + 1);
for i = 1:1:Num_bits
    Encode_RZ_plot(2 * i - 1) = Encode_NRZ(i);
end
Encode_RZ = Encode_RZ_plot(1:end - 1);
```

- Encodes the signal using Polar RZ PCM:
  - Each bit has two samples:
    - First half: Amplitude matches the corresponding NRZ value.
    - Second half: Returns to zero.
- The Encode_RZ_plot array is double the size of the original bit stream (due to two samples per bit).

---

8. 4-Ary PCM Encoding

```
% 4-Ary PCM Encoding
Four_Ary_plot = zeros(1, Num_bits);
for j = 2:2:Num_bits
    if (Stream_bits(j - 1) == 0 && Stream_bits(j) == 0)
        Four_Ary_plot(j) = -3;
         Four_Ary_plot(j - 1) = -3;
    elseif (Stream_bits(j - 1) == 0 && Stream_bits(j) == 1)
        Four_Ary_plot(j) = -1;
        Four_Ary_plot(j - 1) = -1;
    elseif (Stream_bits(j - 1) == 1 && Stream_bits(j) == 1)
        Four_Ary_plot(j) = 1;
        Four_Ary_plot(j - 1) = 1;
    elseif (Stream_bits(j - 1) == 1 && Stream_bits(j) == 0)
```

```
        Four_Ary_plot(j) = 3;
        Four_Ary_plot(j - 1) = 3;
    end
end
```

- Uses two consecutive bits to determine the signal amplitude based on the following mapping:
  - 00 → -3
  - 01 → -1
  - 11 → 1
  - 10 → 3
- The same amplitude is assigned to both bits in the pair to preserve alignment in the time domain.

```
Encode_4_Ary = Four_Ary_plot;
Four_Ary_plot(end + 1) = -3;
```

- Prepares the encoded signal for plotting by appending an extra point.

---

9. Output Summary

```
% Display generated and encoded information
fprintf('Random bit stream of %d bits generated and encoded using:\n',
Num_bits);
fprintf('1. Polar NRZ PCM\n2. Polar RZ PCM\n3. 4-Ary PCM with Gray
Coding\n');
```

- Prints a summary of the generated bit stream and encoding methods used.

---

10. Plotting Section

This section visualizes the encoded signals and their constellations.

10.1. Polar NRZ PCM Signal Plot

```
% Plotting Section
% Plot Polar NRZ PCM
figure;
subplot ( 3 , 1 , 1 ) ;
stairs(time_NRZ, Encode_NRZ_plot, 'b', 'LineWidth', 2.5);
grid on;
box on;
title('Polar NRZ PCM', 'FontName', 'Times New Roman', 'FontWeight',
'bold');
ylabel('Amplitude', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
xlabel('Time', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
ylim([-1.5 1.5]);
xlim([0 10]);
```

- **Purpose**: Plots the Polar NRZ PCM signal over time.
- **stairs function**: Draws a stepwise graph to represent the discrete nature of digital signals.
- **Time in NRZ PCM**: assume the T_b = 1 sec so the period for every bit in the plotting encode 1 sec

10.2. Polar RZ PCM Signal Plot

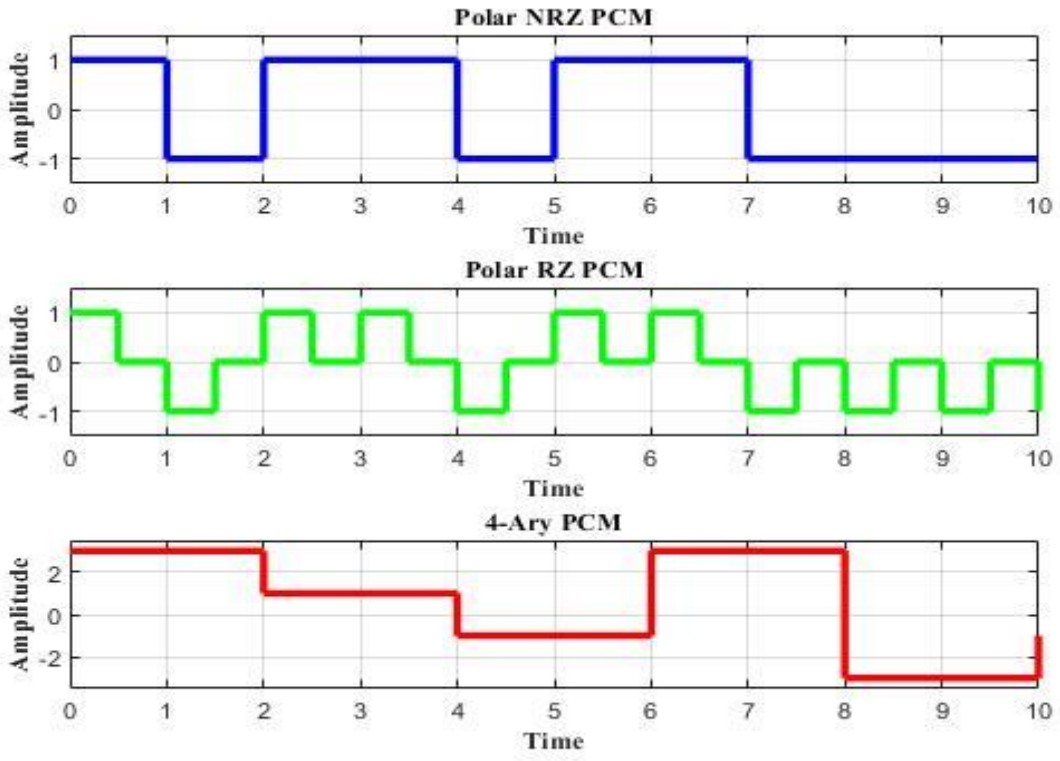
```
% Plot Polar RZ PCM
subplot ( 3 , 1 , 2 ) ;
stairs(time_RZ, Encode_RZ_plot, 'g', 'LineWidth', 2.5);
grid on;
box on;
title('Polar RZ PCM', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
ylabel('Amplitude', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
xlabel('Time', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
ylim([-1.5 1.5]);
xlim([0 10]);
```

- **Purpose**: Visualizes the Polar RZ PCM encoding.
- **Time in RZ PCM**: assume the T_s = 0.5 sec but the periad of one cycle for one bit is 1sec
- **Green line ('g')**: Indicates the encoded signal.

10.3. 4-Ary PCM Signal Plot

```
% Plot 4-Ary PCM
subplot ( 3 , 1 , 3 ) ;
stairs(time_Ary, Four_Ary_plot, 'r', 'LineWidth', 2.5);
grid on;
box on;
title('4-Ary PCM', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
ylabel('Amplitude', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
xlabel('Time', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
ylim([-3.5 3.5]);
xlim([0 10]);
```

- **Purpose**: Displays the signal corresponding to 4-Ary PCM.



- **Key Difference**: Encodes two bits per symbol using Gray coding, with amplitudes of -3, -1, 1, and 3 and T_s in this modulation is 2sec.

---

11. Constellation Diagrams

This section visualizes the signal mapping in the real-imaginary plane using rule:$\sqrt{E/Ts}$.

11.1. Polar NRZ PCM Constellation

```
% Plot Constellation
% Plot Polar NRZ PCM
figure;
subplot( 3 , 1 , 1 );
plot(Encode_NRZ, zeros(size(Encode_NRZ)), '.', 'LineWidth', 100);
title('Polar NRZ PCM Constallation', 'FontName', 'Times New Roman',
'FontWeight', 'bold');
ylabel('Imagine', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
xlabel('Real', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
grid on;
box on;
```

- **Purpose**: Plots the constellation for Polar NRZ PCM.
- **Key Details**:
  - X-axis: Represents the real component (A) (±1 for NRZ).
  - Y-axis: Represents the imaginary component (always 0).

11.2. Polar RZ PCM Constellation

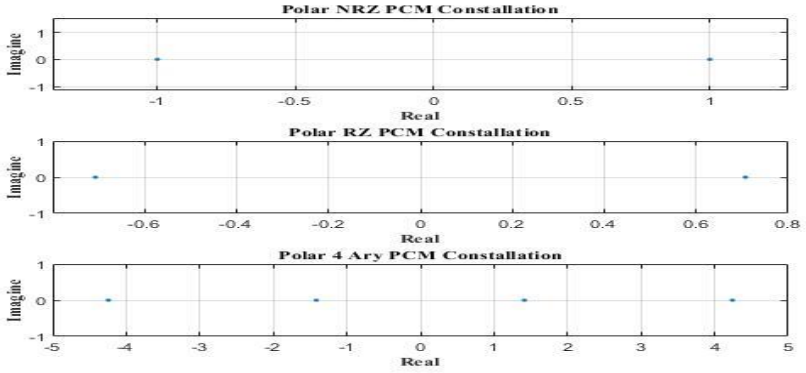
```
% Plot Polar RZ PCM
subplot( 3 , 1 , 2 );
plot(Encode_RZ(1:2:end) / sqrt(2), zeros(size(Encode_RZ(1:2:end))), '.',
'LineWidth', 100);
title('Polar RZ PCM Constallation', 'FontName', 'Times New Roman',
'FontWeight', 'bold');
ylabel('Imagine', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
xlabel('Real', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
grid on;
box on;
```

- **Purpose**: Displays the Polar RZ PCM constellation.
- **Mapping**:
  - Real values scaled by $A/\sqrt{2}$ to account for the RZ encoding.

11.3. 4-Ary PCM Constellation

```
% Plot Polar 4-Ary PCM
subplot( 3 , 1 , 3 );
plot(Encode_4_Ary(1:2:end) * sqrt(2), zeros(size(Encode_4_Ary(1:2:end))),
'.', 'LineWidth', 100);
title('Polar 4 Ary PCM Constallation', 'FontName', 'Times New Roman',
'FontWeight', 'bold');
ylabel('Imagine', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
xlabel('Real', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
grid on;
box on;
```

- **Purpose**: Plots the 4-Ary PCM constellation $A\sqrt{2}$



---

12. Probability Calculation

```
% Probabilty of 0 and 1
Count0 = sum(Stream_bits == 0);
Probability0 = Count0 / length(Stream_bits);
Probability1 = 1 - Probability0;
```

- **Count0**: Counts the number of zeros in the bit stream.
- **Probability0**: Probability of a 0 in the stream.
- **Probability1**: Probability of a 1 in the stream (calculated as 1 - Probability0).

Purpose:

- Provides statistical information about the distribution of 0s and 1s in the generated bit stream to use it in EQ. the optimum threshold ($P_0 \neq P_1$).

---

13. Transmitting Signals Through AWGN Channel

The code iterates over various **Eb/N0** values to simulate the effect of Additive White Gaussian Noise (AWGN) on the encoded signals.

13.1. Noise Power Calculation

```
% Transmit through AWGN Channel
for EbN0_dB_f = -10 : 2 : 6
% Calculate noise power NRZ
    EbN0_Power = 10^(EbN0_dB_f/10);
    noise_power_NRZ = 1 / (EbN0_Power);

    % Calculate noise power RZ
    noise_power_RZ = 1 / (EbN0_Power);

    % Calculate noise power 4-Ary
    noise_power_4Ary = 1 / (EbN0_Power);
```

- **EbN0_Power**: Converts Eb/N0 from dB to linear scale.

13.2. Adding AWGN

```
% Add AWGN to NRZ PCM
```

```
    noise_NRZ = sqrt(noise_power_NRZ / 2) * randn(size(Encode_NRZ));
    received_NRZ = Encode_NRZ + noise_NRZ;


% Add AWGN to RZ PCM
    noise_RZ = sqrt(noise_power_RZ) * randn(size(Encode_RZ));
    received_RZ = Encode_RZ + noise_RZ;
```

- `% Add AWGN to 4-Ary PCM`
- `    noise_Ary = sqrt(noise_power_4Ary / 4) * randn(size(Encode_4_Ary));`
- `    received_Ary = Encode_4_Ary + noise_Ary;`
- **AWGN (Additive White Gaussian Noise)**:
  - Generated using randn (standard normal distribution).
  - Scaled by Noise Power $\sqrt{Noise\ Power/2}$ to match the required variance.
  - Added to the respective encoded signals (NRZ, RZ, 4-Ary).

---

14. Threshold Calculation

```
% Optimum Threshold NRZ
    Threshold_NRZ = (noise_power_NRZ / 4) * log10(Probability0 /
Probability1);
```

- **Purpose**: Computes the optimum threshold for decoding Polar NRZ PCM.

---

15. Plotting the Received Signals

15.1. Received NRZ PCM Signal

```
% Plot diagrams for received signals
    figure;
    subplot(3, 1, 1);
    stairs(time_NRZ, received_NRZ_plot, 'b', 'LineWidth', 2.5);
    hold on;
    plot(time_NRZ, Threshold_NRZ * ones(size(time_NRZ)));
    title(['Received NRZ PCM (Eb/N0 = ', num2str(EbN0_dB_f), ' dB)'],
'FontName', 'Times New Roman', 'FontWeight', 'bold');
    ylabel('Amplitude', 'FontName', 'Times New Roman', 'FontWeight',
'bold');
    xlabel('Time', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    xlim([0 10]);
    grid on;
    box on;
```

- **Purpose**: Visualizes the received NRZ PCM signal with noise.
- **Threshold Overlay**: Highlights the decoding threshold.

15.2. Received RZ PCM Signal

```
subplot(3, 1, 2);
    stairs(time_RZ, received_RZ_plot, 'g', 'LineWidth', 2.5);
    hold on;
```

```
    plot(time_RZ, zeros(size(time_RZ)));
    title(['Received RZ PCM (Eb/N0 = ', num2str(EbN0_dB_f), ' dB)'],
'FontName', 'Times New Roman', 'FontWeight', 'bold');
    ylabel('Amplitude', 'FontName', 'Times New Roman', 'FontWeight',
'bold');
    xlabel('Time', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    xlim([0 10]);
    grid on;
    box on;
```

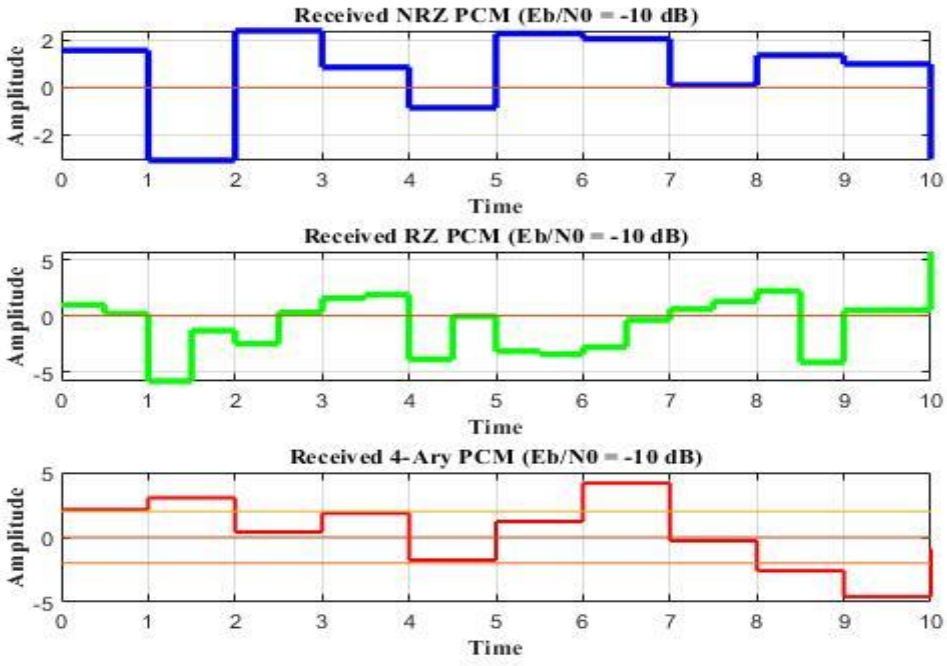
- **Purpose**: Displays the received RZ PCM signal with noise.
- **Zero Line**: Indicates the midpoint for decoding.

15.3. Received 4-Ary PCM Signal

```
subplot(3, 1, 3);
    stairs(time_Ary, received_Ary_plot, 'r', 'LineWidth', 1.5);
    hold on;
    plot(time_Ary, zeros(size(time_Ary)));
    plot(time_Ary, 2 * ones(size(time_Ary)));
    plot(time_Ary, -2 * ones(size(time_Ary)), 'Color', [1, 0.41, 0.16]);
    title(['Received 4-Ary PCM (Eb/N0 = ', num2str(EbN0_dB_f), ' dB)'],
'FontName', 'Times New Roman', 'FontWeight', 'bold');
    ylabel('Amplitude', 'FontName', 'Times New Roman', 'FontWeight',
'bold');
    xlabel('Time', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    xlim([0 10]);
    grid on;
    box on;
```

- **Purpose**: Plots the noisy 4-Ary PCM signal.
- **Overlayed Lines**:
    - Amplitudes ±2 and 0 represent decision thresholds.

Example of these plots (we have like this example at every E_b/N_o in code) :



16. Constellation Plots

16.1. NRZ PCM Constellation

```
% Plot Constellation
    % Plot Polar NRZ PCM
    figure;
    subplot( 3 , 1 , 1 );
    plot(received_NRZ, zeros(size(received_NRZ)), '.', 'LineWidth', 100);
    title(['Received Constellation NRZ PCM (Eb/N0 = ', num2str(EbN0_dB_f),
' dB)'], 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    ylabel('Imagine', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    xlabel('Real', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    grid on;
    box on;
```

- **Purpose**: Visualizes the NRZ PCM constellation post noise addition.
- **Key Details**:
    - X-axis: Signal amplitudes (A).
    - Y-axis: Imaginary part (0).

16.2. RZ PCM Constellation

```
% Plot Polar RZ PCM
    subplot( 3 , 1 , 2 );
    plot(received_RZ(1:2:end) / sqrt(2), zeros(size(received_RZ(1:2:end))),
'.', 'LineWidth', 100);
    title(['Received Constellation RZ PCM (Eb/N0 = ', num2str(EbN0_dB_f), '
dB)'], 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    ylabel('Imagine', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    xlabel('Real', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    grid on;
    box on;
```

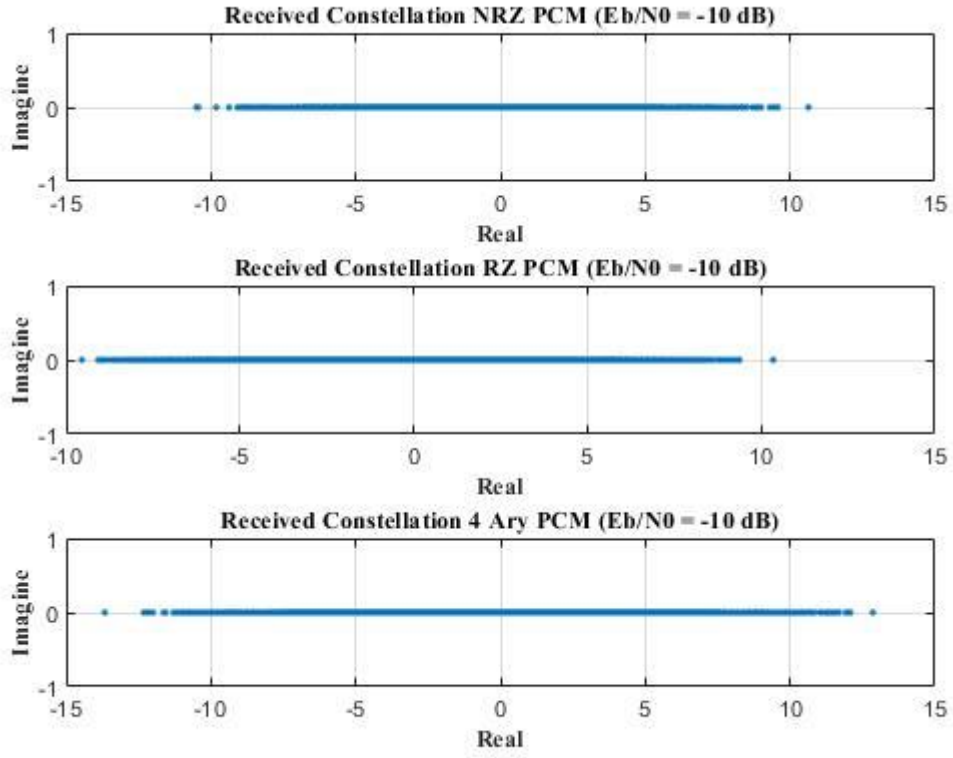
- **Purpose**: Plots the RZ PCM constellation with noise effects.
- **Mapping**:
    - X-axis: Real part scaled by $A/\sqrt{2}$
    - Y-axis: Imaginary part (0).

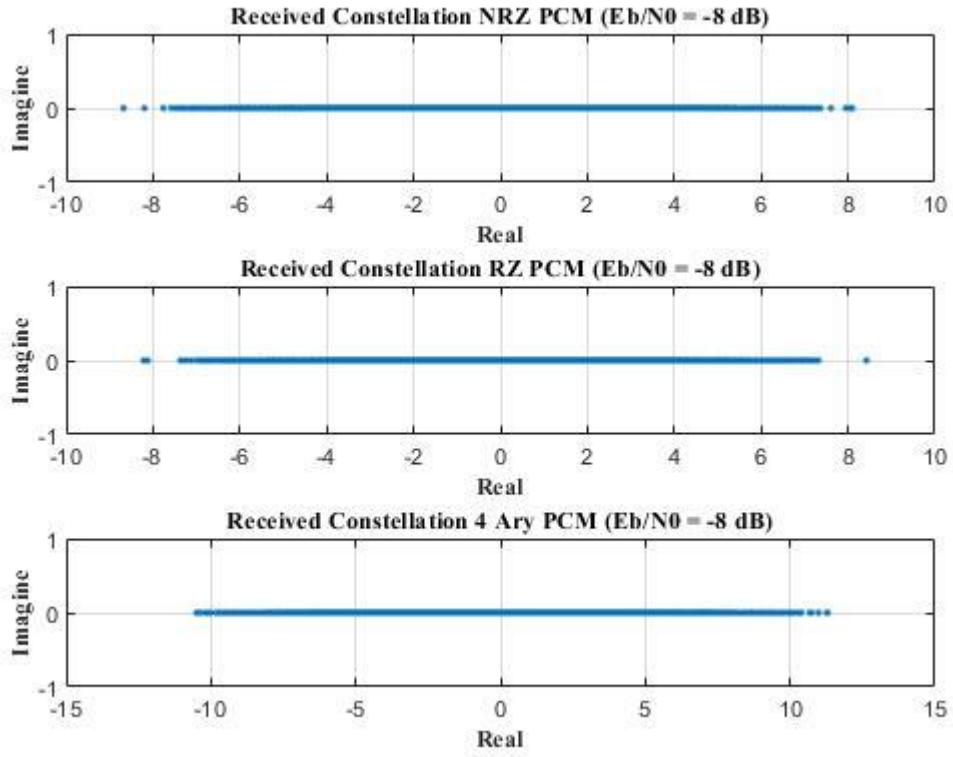
16.3. 4-Ary PCM Constellation

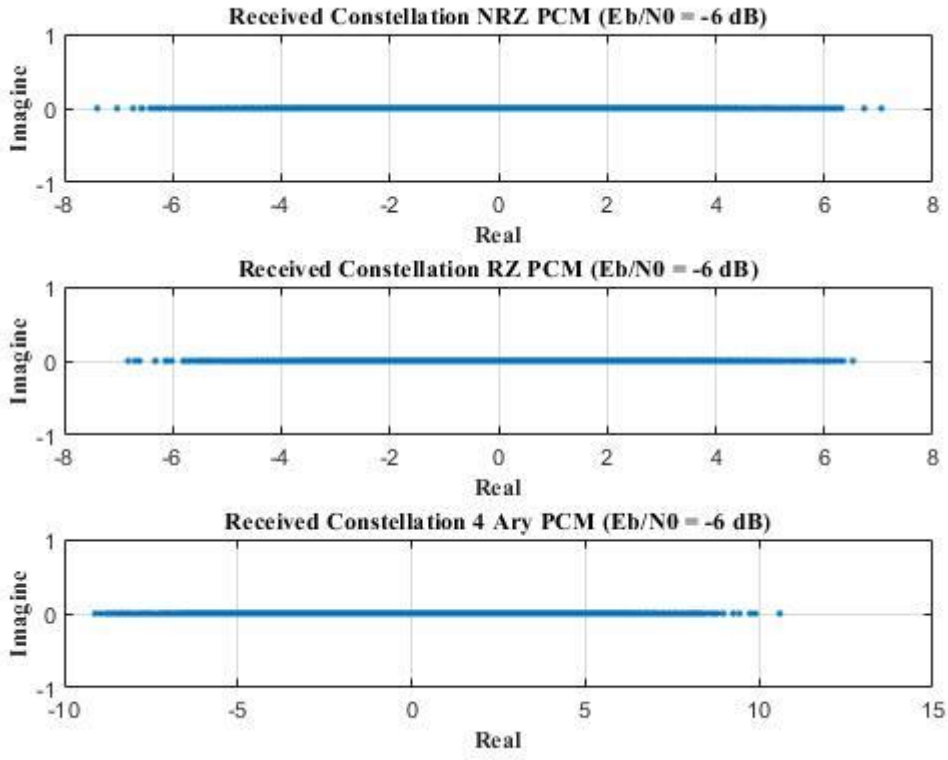
```
% Plot Polar 4-Ary PCM
    subplot( 3 , 1 , 3 );
    plot(received_Ary(1:2:end) * sqrt(2),
zeros(size(received_Ary(1:2:end))), '.', 'LineWidth', 100);
    title(['Received Constellation 4 Ary PCM (Eb/N0 = ',
num2str(EbN0_dB_f), ' dB)'], 'FontName', 'Times New Roman', 'FontWeight',
'bold');
    ylabel('Imagine', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    xlabel('Real', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
    grid on;
    box on;
```

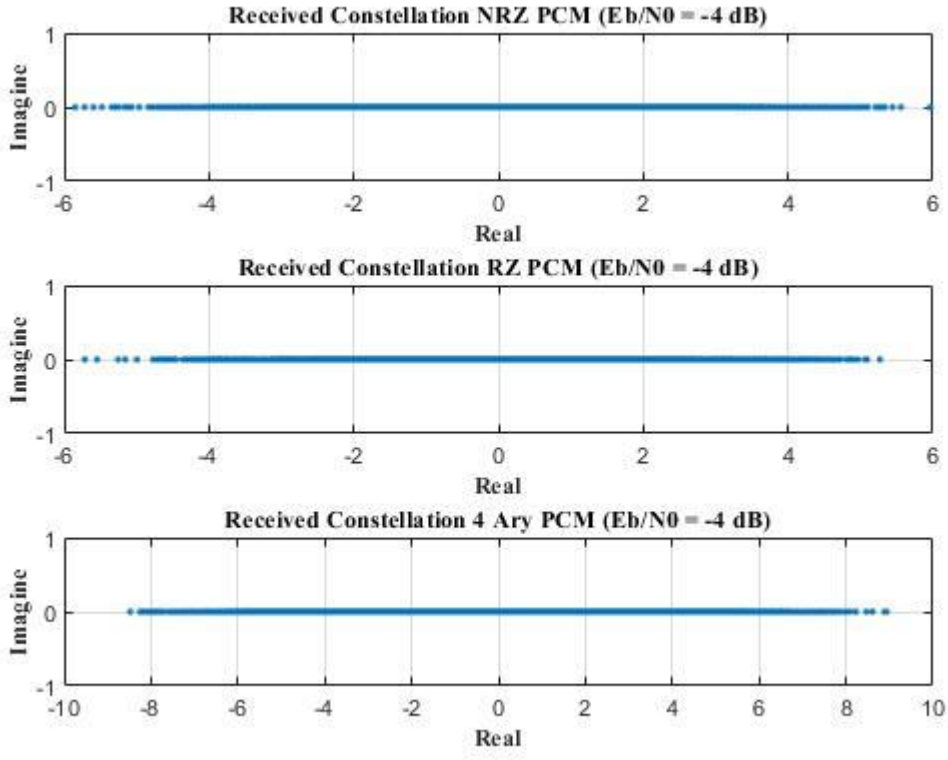
- **Purpose**: Visualizes the 4-Ary PCM constellation post noise addition.
- **Mapping**:

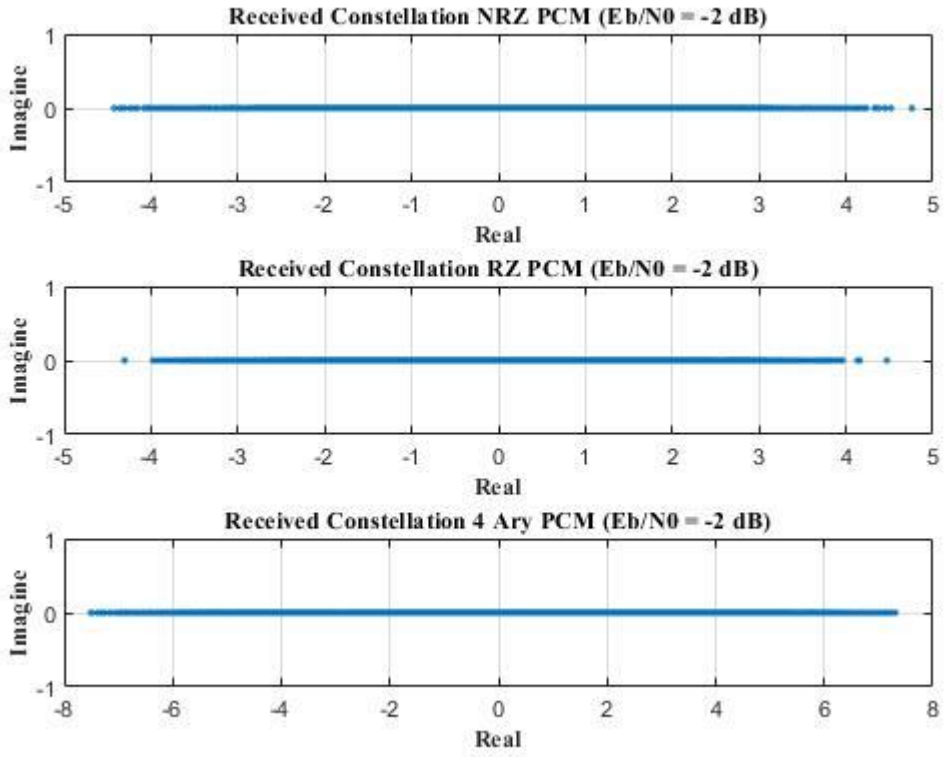
- Amplitudes are scaled by $A\sqrt{2}$ for better clarity.

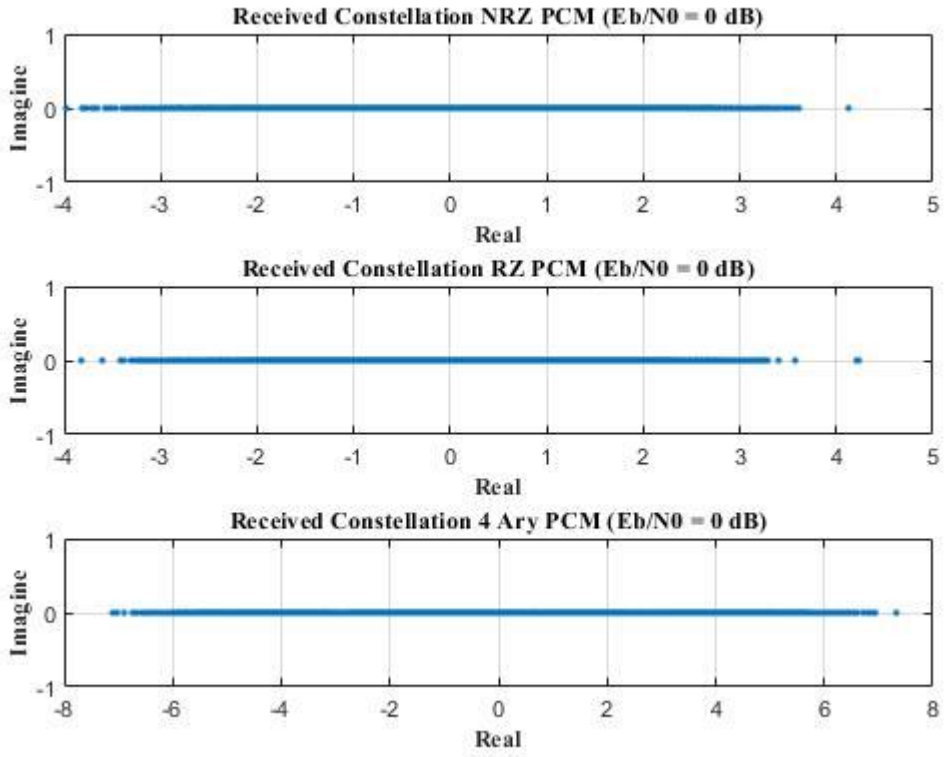


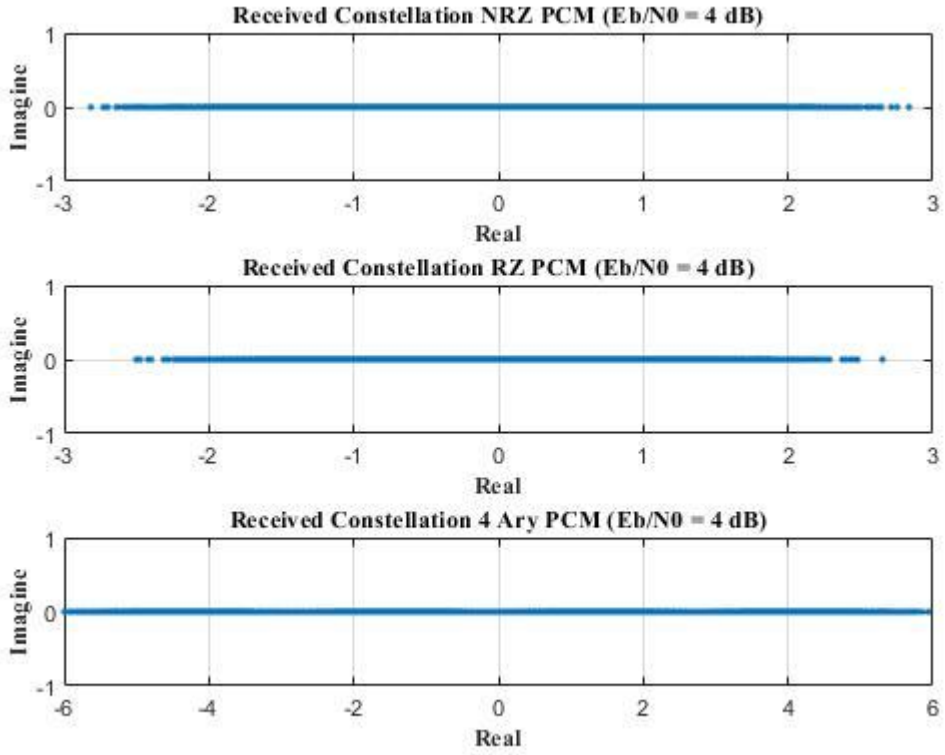


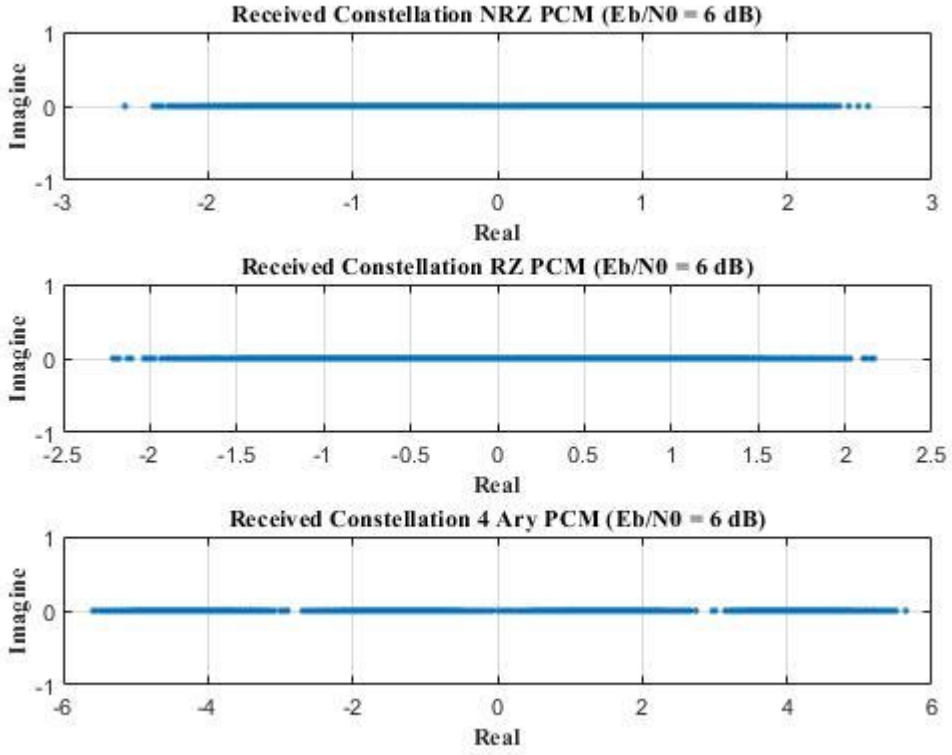

Received Constellation NRZ PCM (Eb/N0 = -6 dB)
Imagine
Real
Received Constellation RZ PCM (Eb/N0 = -6 dB)
Imagine
Real
Received Constellation 4 Ary PCM (Eb/N0 = -6 dB)
Imagine
Real

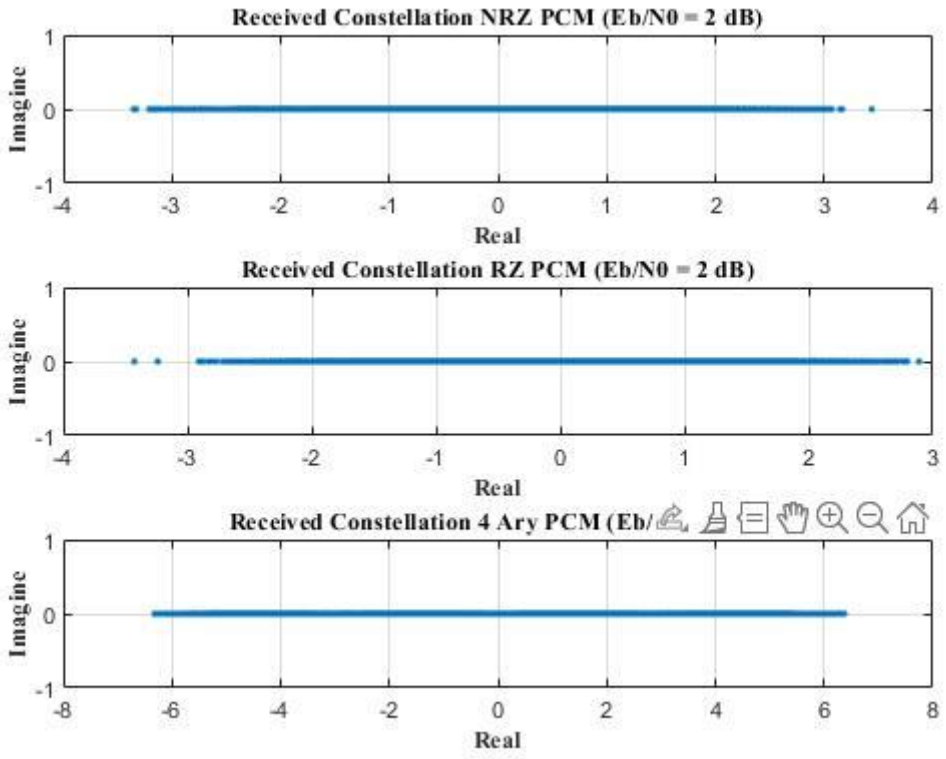

Received Constellation NRZ PCM (Eb/N0 = -4 dB)
Imagine
Real
Received Constellation RZ PCM (Eb/N0 = -4 dB)
Imagine
Real
Received Constellation 4 Ary PCM (Eb/N0 = -4 dB)
Imagine
Real


Received Constellation NRZ PCM (Eb/N0 = -2 dB)
Imagine
Real
Received Constellation RZ PCM (Eb/N0 = -2 dB)
Imagine
Real
Received Constellation 4 Ary PCM (Eb/N0 = -2 dB)
Imagine
Real


Received Constellation NRZ PCM (Eb/N0 = 0 dB)
Imagine
Real
Received Constellation RZ PCM (Eb/N0 = 0 dB)
Imagine
Real
Received Constellation 4 Ary PCM (Eb/N0 = 0 dB)
Imagine
Real


Received Constellation NRZ PCM (Eb/N0 = 4 dB)
Imagine
Real
Received Constellation RZ PCM (Eb/N0 = 4 dB)
Imagine
Real
Received Constellation 4 Ary PCM (Eb/N0 = 4 dB)
Imagine
Real


Received Constellation NRZ PCM (Eb/N0 = 6 dB)
Imagine
Real
Received Constellation RZ PCM (Eb/N0 = 6 dB)
Imagine
Real
Received Constellation 4 Ary PCM (Eb/N0 = 6 dB)
Imagine
Real



we will notice the constellation diagrams appears as cloud noise.

---

17. Matched Filtering

Matched Filtering is a method used to maximize the signal-to-noise ratio (SNR) of a received signal. It is performed by convolving the received signal with a time-reversed version of the transmitted signal.

17.1 Create the Matched Filters

```
% Create the matched filter
    % NRZ PCM
    matched_filter_NRZ = fliplr(Encode_NRZ);
    % RZ PCM
    matched_filter_RZ = fliplr(Encode_RZ);
    % 4-Ary PCM
    matched_filter_ARY = fliplr(Encode_4_Ary);
```

- **fliplr**: Reverses the signal in time, which is a key property of matched filtering.
- Separate matched filters are created for NRZ, RZ, and 4-Ary PCM encoding.

17.2 Apply the Matched Filters

```
% Apply the matched filter
    % NRZ PCM
    output_filter_NRZ = conv(received_NRZ, matched_filter_NRZ, 'same');
```

```
    % RZ PCM
    output_filter_RZ = conv(received_RZ, matched_filter_RZ, 'same');
    % 4-Ary PCM
    output_filter_ARY = conv(received_Ary, matched_filter_ARY, 'same');
```

- **conv**: Convolves the received signal with its matched filter.
- **'same'**: Ensures the output has the same length as the input signal.

---

18. Signal Thresholding

Thresholding determines whether a bit is interpreted as 0 or 1 (or specific values for 4-Ary encoding).

18.1 Signal Threshold for NRZ

```
% Signal after threshold NRZ
    received_NRZ_threshold = (received_NRZ > Threshold_NRZ) * 1 +
(received_NRZ < Threshold_NRZ) * -1;
```

- For NRZ PCM, a binary decision is made:
  - If the value is greater than the threshold, it is interpreted as 1.
  - Otherwise, it is interpreted as -1.

18.2 Signal Threshold for RZ

```
% Signal after threshold RZ threshold = 0
    received_RZ_threshold = (received_RZ(1:2:end) > 0) * 1 +
(received_RZ(1:2:end) < 0) * -1;
```

- For RZ PCM, sampling occurs only at the middle of each bit period (1:2:end).
- Similar to NRZ, binary decisions are made based on whether the value is greater or less than 0.

18.3 Signal Threshold for 4-Ary

```
    % Signal after threshold 4-Ary threshold = 0, 2 , 2
    received_Ary_threshold = (received_Ary > 0) .* ((received_Ary > 2) * 3
+ (received_Ary < 2) * 1) + (received_Ary < 0) .* ((received_Ary > -2) * -1
+ (received_Ary < -2) * -3);
```

- For 4-Ary PCM, thresholds are set at:
  - 0 and ±2.
- The thresholds map received values to their corresponding 4-Ary levels:
  - -3, -1, 1, 3.

---

19. Bit Error Rate (BER) Calculation

The **BER** quantifies the percentage of incorrectly decoded bits.

19.1 BER for NRZ

```
%Bit Error Rate NRZ
```

```
    BER_NRZ = sum (received_NRZ_threshold ~= Encode_NRZ) /
length(received_NRZ_threshold);
    BER_VS_No_NRZ(6 - (EbN0_dB_f / -2)) = BER_NRZ;
```

- Computes the proportion of mismatched bits between the decoded and transmitted signals.
- **BER_VS_No_NRZ** stores the BER for NRZ at each **Eb/N0**.

19.2 BER for RZ

```
%Bit Error Rate RZ
    BER_RZ = sum (received_RZ_threshold ~= Encode_RZ(1:2:end)) /
length(received_RZ_threshold);
    BER_VS_No_RZ(6 - (EbN0_dB_f / -2)) = BER_RZ;
```

- Similar to NRZ, but uses the mid-bit samples for comparison (Encode_RZ(1:2:end)).

19.3 BER for 4-Ary

```
%Bit Error Rate 4-Ary
    BER_Ary = sum (received_Ary_threshold ~= Encode_4_Ary) /
length(received_Ary_threshold);
    BER_VS_No_Ary(6 - (EbN0_dB_f / -2)) = BER_Ary;
```

- For 4-Ary PCM, the BER is calculated by comparing decoded values to transmitted levels (Encode_4_Ary).

---

20. Theoretical BER Calculation

```
%Bit Error Rate Theoritical NRZ
    BER_Theoritical = (Probability0 * 0.5 * erfc((1 + Threshold_NRZ) /
((EbN0_Power)^-.5))) + (Probability1 * 0.5 * erfc((1 - Threshold_NRZ) /
((EbN0_Power)^-0.5)));
    BER_Theoritical_VS_No(6 - (EbN0_dB_f / -2)) = BER_Theoritical;
```

- **erfc**: Computes the complementary error function for Gaussian noise.
- Combines probabilities of errors for 0 and 1 bit levels.

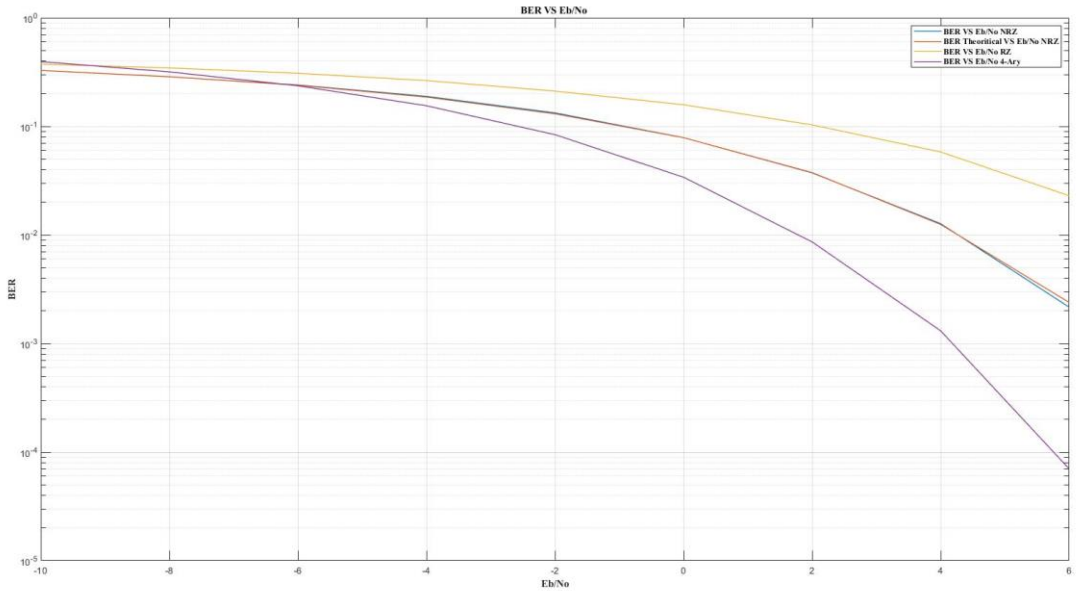
---

21. Plotting BER vs Eb/N0

```
% Plot BER VS Eb/no and BER theoritical
figure;
semilogy(EbN0_dB, BER_VS_No_NRZ, 'LineWidth', 1);
hold on
semilogy(EbN0_dB, BER_Theoritical_VS_No, 'LineWidth', 1);
hold on
semilogy(EbN0_dB, BER_VS_No_RZ, 'LineWidth', 1);
hold on
semilogy(EbN0_dB, BER_VS_No_Ary, 'LineWidth', 1);
title('BER VS Eb/No ', 'FontName', 'Times New Roman', 'FontWeight',
'bold');
ylabel('BER', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
xlabel('Eb/No', 'FontName', 'Times New Roman', 'FontWeight', 'bold');
legend('BER VS Eb/No NRZ', 'BER Theoritical VS Eb/No NRZ', 'BER VS Eb/No
RZ', 'BER VS Eb/No 4-Ary', 'FontName', 'Times New Roman', 'FontWeight',
'bold');
ylim([10^-5  1]);
```

```
xlim([-10 6]);
```

- **Purpose**: Visualizes the BER for:
  - NRZ PCM
  - RZ PCM
  - 4-Ary PCM
  - Theoretical BER for NRZ
- **semilogy**: Plots the BER on a logarithmic scale for better visibility of low error rates.



This graph shows the Energy of the 4 – Ary is the greater then NRZ PCM and the RZ PCM is the smallest Energy so the effect of noise on it is great.

The graph of BER for NRZ practical using code is same the the graph of BER for NRZ theoritical.