Digital Communications Project (1)

3rd Year Comm. | Spring 2025

T32 Members:

NAME	SECTION	ID
Abdallah Essam Muhammed	2	9220479
Hussien Moustafa Amin	2	9221585
Ahmed Wagdy Mohy	1	9220120
Omar Ahmed	3	9220518
Mohand Hassan Aly	4	9221293

Dr. Nafea

Eng. Muhammed Khaled

Table of Contents

Role of each member 5

Problem description 5

Introduction 6

Control flags 7

Generation of data 7

Creation of unipolar ensemble 8

Unipolar Graph 9

Creation of polar NRZ ensemble 9

Polar NRZ Graph 10

Creation of polar RZ ensemble 10

Polar RZ Graph 11

Getting the cell arrays ready to calculate the statistical mean and autocorrelation 12

Q1: Calculating the statistical mean (Hand Analysis) 13

Plotting the statistical mean 14

Hand Analysis: Calculating the statistical autocorrelation 15

Unipolar NRZ statistical autocorrelation 15

Polar NRZ statistical autocorrelation 16

Polar RZ statistical autocorrelation 17

Plotting the statistical autocorrelation 18

Q2: Is the process stationary? 19

Q4: Computing the time mean and autocorrelation of one waveform 20

Time Mean 20

Time Mean Graph 21

Time Autocorrelation 22

Time Autocorrelation Graph 23

Q5: Is the random process ergodic? 24

Plotting the PSD of the ensemble 26

Q6: What is the bandwidth of the transmitted signal? 29

Appendix: Full MATLAB code 30

Table of Figures

FIGURE 1. SDR FLOW 5
FIGURE 2. TX/RX PATHS 6
FIGURE 3. UNIPOLAR REALIZATIONS 9
FIGURE 4. POLAR NRZ REALIZATIONS 10
FIGURE 5. POLAR RZ REALIZATIONS 11
FIGURE 6. UNIPOLAR OUTPUT 12
FIGURE 7. POLAR NRZ OUTPUT 13
FIGURE 8. POLAR RZ OUTPUT 13
FIGURE 9. UNIPOLAR STATISTICAL MEAN 14
FIGURE 10. POLAR NRZ STATISTICAL MEAN 14
FIGURE 11. POLAR RZ STATISTICAL MEAN 14
FIGURE 12. UNIPOLAR STATISTICAL AUTOCORRELATION 18
FIGURE 13. POLAR NRZ STATISTICAL AUTOCORRELATION 18
FIGURE 14. POLAR RZ STATISTICAL AUTOCORRELATION 18
FIGURE 15. UNIPOLAR TIME MEAN 21
FIGURE 16. POLAR NRZ TIME MEAN 22
FIGURE 17. POLAR RZ TIME MEAN 22
FIGURE 18. UNIPOLAR TIME AUTOCORRELATION 23
FIGURE 19. POLAR NRZ TIME AUTOCORRELATION 23
FIGURE 20. POLAR RZ TIME AUTOCORRELATION 24
FIGURE 21. UNIPOLAR MEAN 24
FIGURE 22. POLAR NRZ MEAN 24
FIGURE 23. POLAR RZ MEAN 25
FIGURE 24. UNIPOLAR AUTOCORRELATION 25
FIGURE 25. POLAR NRZ AUTOCORRELATION 25
FIGURE 26. POLAR RZ AUTOCORRELATION 25
FIGURE 27. PSD - UNIPOLAR NRZ 27
FIGURE 28. PSD - POLAR RZ 27
FIGURE 29. PSD - POLAR NRZ 28

Role of each member

NAME	ROLE
Abdallah Essam Muhammed	MATLAB Code
Mohand Hassan Aly	Code & Report Reviewing
Ahmed Wagdy Mohy	MATLAB Code
Omar Ahmed	Data Collection & Report Reviewing
Hussien Moustafa Amin	Report Writing

Problem description

Software-Defined Radio (SDR) transforms how radio systems are designed and operated by shifting many traditional hardware functions into software. SDR allows both the transmission and reception processes to be fully controlled by code, reducing reliance on physical hardware components. This shift brings greater flexibility and adaptability, enabling radio systems to support different communication standards and protocols through software updates rather than hardware changes. SDR's flexibility extends beyond wireless systems and can also be applied to wired communication using line coding techniques.

Conventional radio systems face several challenges due to their reliance on fixed hardware components. These systems are often designed for specific purposes, making it difficult to adapt them for new communication protocols or evolving standards. Hardware upgrades are costly and time-consuming, limiting the flexibility of these systems.

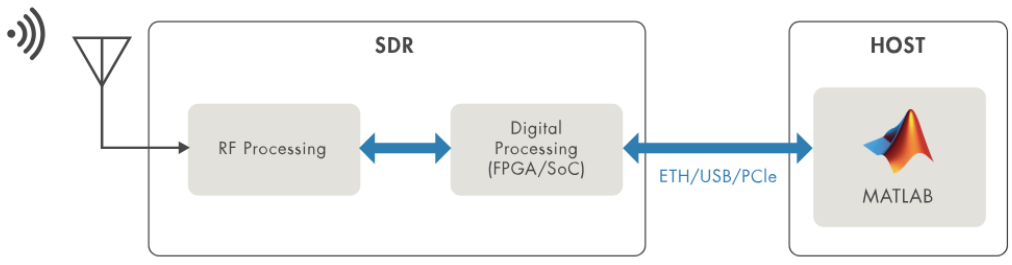

Figure 1. SDR Flow

SDR addresses these problems by replacing hardware-dependent processing with software-defined processing. This transition turns hardware challenges into software challenges, where system behavior can be changed or upgraded through simple software modifications rather than physical redesigns. The key challenge is to ensure that SDR systems can achieve high performance and reliability while maintaining the flexibility and adaptability that software-based systems offer.

Introduction

Traditional radio systems rely heavily on analog circuitry or a combination of analog and digital components to handle signal transmission and reception. These systems are often rigid, with hardware designed for specific tasks, making it difficult to modify or upgrade them.

Software-Defined Radio (SDR) changes this approach by moving the signal processing tasks closer to the antenna and handling them primarily through software. In an SDR system, software defines the transmitted signals and demodulates received signals, offering much greater flexibility than traditional radios.

SDR is widely used in wireless communication, but its principles can also be applied to wired communication systems, where antennas are replaced by cables, and line coding techniques manage data transmission

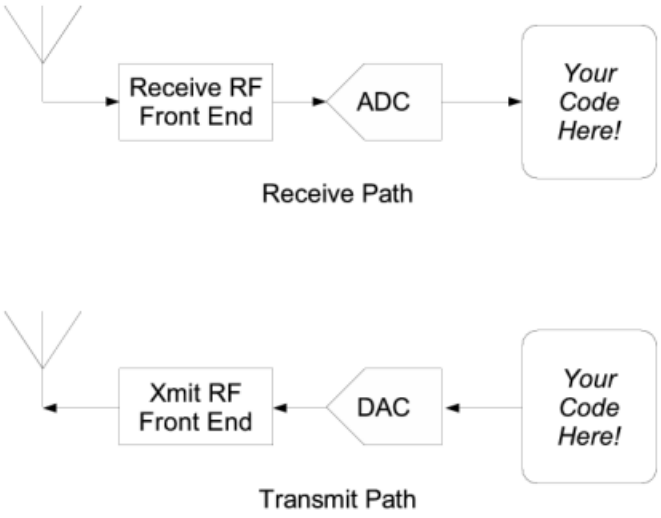

Figure 2. Tx/Rx Paths

Control flags

The analog signal waveforms are passed through an **ADC,** where **7 samples** are taken per bit. According to the **Nyquist criterion** ($f_s \geq 2f_m$), a minimum of **2 samples** per bit is sufficient for accurate signal sampling.

However, **oversampling (7 samples)** is performed to facilitate easier signal reconstruction and improve the accuracy of the digital representation.

CONTROL FLAG	FLAG
# BITS	100
# REALIZATIONS	500
# SAMPLES PER BIT	7 samples/bit
TIME SHIFT	0:6

Generation of data

Explanation of the Process

We aim to generate **500 realizations** of random binary data, where each realization consists of **100 bits**.

1. **Bit Duration and Sampling Rate**
    - Each bit will last for **70 msec**.
    - The ADC will take samples at intervals of **10 msec**.
    - Since a single bit lasts for 70 msec and the sampling period is 10 msec, each bit will be represented by **7 samples** (70 msec ÷ 10 msec).
2. **Generating Random Data**
    - We will generate random binary data (values of 0 or 1).
    - Each realization will include an additional bit (making it 101 bits) to account for any transition or alignment effects during processing.
3. **Random Transmission Delay**
    - After generating the line codes, a random transmission delay will occur.

- This delay will be represented as a random number of samples between **0 and 6**.
- Since each sample is 10 msec, the delay will correspond to a random duration between **0 and 60 msec**.

To apply the random initial time shift, an additional bit is added when generating the random binary data.

```
global bits_count;
global realizations_count;

% Definition of scaling factor, #bits and #realizations
scale = 4;
bits_count = 100;
realizations_count = 500;

% Generate random binary data and transmission delays

% Additional bit added here
binary_data = randi([0, 1], realizations_count, bits_count + 1);

delays = randi([0, 6], realizations_count, 1);
```

Creation of unipolar ensemble

```
% Unipolar Encoding

encoded_unipolar = binary_data * scale;
repeated_unipolar = repelem(encoded_unipolar, 1, 7);
adjusted_unipolar = zeros(realizations_count, bits_count * 7);
for k = 1:realizations_count
    adjusted_unipolar(k, :) = repeated_unipolar(k, delays(k) + 1:delays(k) +
bits_count * 7);
end
```

The code performs unipolar encoding by multiplying the generated binary data (1 or 0) by a scaling factor $\boldsymbol{A}$ to produce a transmitted signal where 1 is represented by $\boldsymbol{A}$ and 0 is represented by 0.

The encoded signal is then repeated 7 times to simulate transmission at a higher sampling rate.

Finally, a transmission delay is applied by shifting the repeated signal based on a randomly generated delay value.

Unipolar Graph

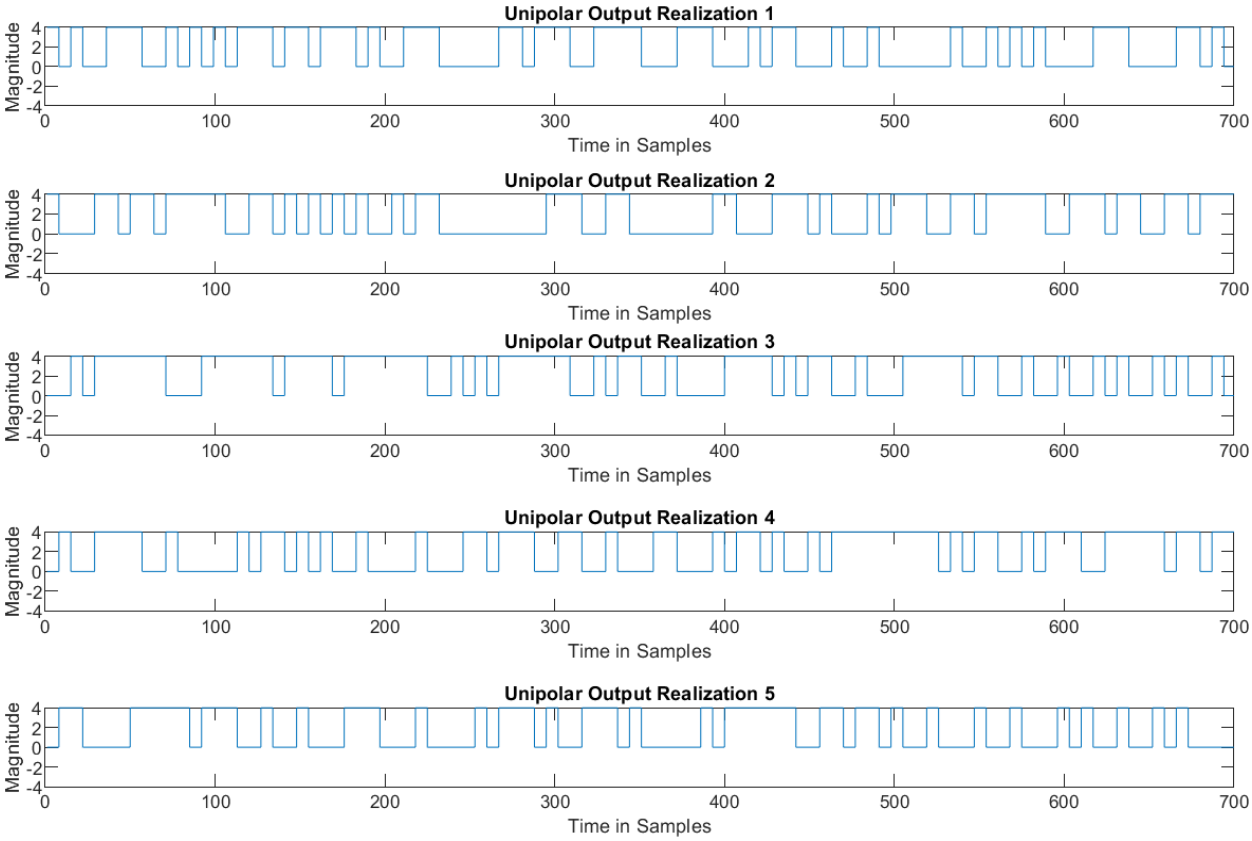

Figure 3. Unipolar Realizations

Creation of polar NRZ ensemble

```
% Polar NRZ Encoding

encoded_polar_nrz = ((2 * binary_data) - 1) * scale;
repeated_polar_nrz = repelem(encoded_polar_nrz, 1, 7);
adjusted_polar_nrz = zeros(realizations_count, bits_count * 7);

for k = 1:realizations_count
    adjusted_polar_nrz(k, :) = repeated_polar_nrz(k, delays(k) + 1:delays(k) +
bits_count * 7);
end
```

The code performs polar NRZ encoding by converting the binary data into values of $\boldsymbol{A}$ and $-\boldsymbol{A}$. This is achieved by multiplying the data by 2, subtracting 1, and then multiplying by the scaling factor $\boldsymbol{A}$. The encoded signal is then repeated 7 times to simulate transmission at a higher sampling rate.

Polar NRZ Graph

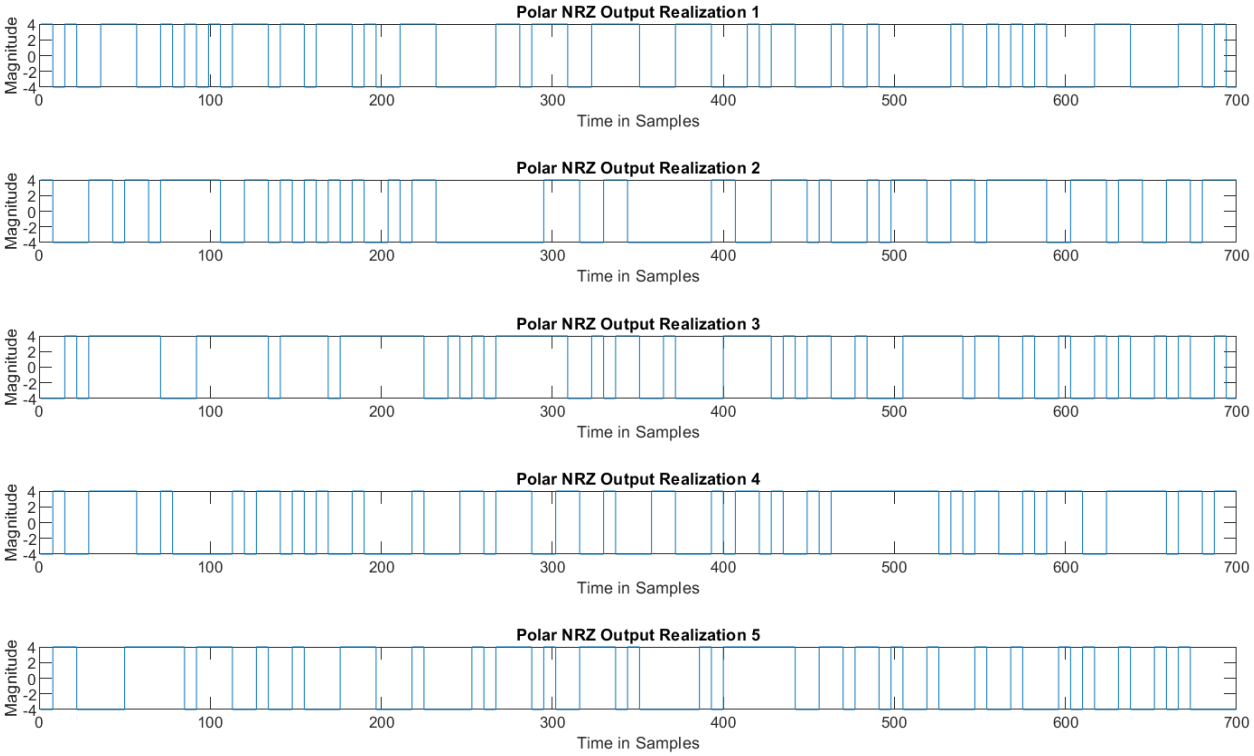

Figure 4. Polar NRZ Realizations

Creation of polar RZ ensemble

```
% Polar RZ Encoding
repeated_polar_rz = repelem(encoded_polar_nrz, 1, 4);
adjusted_polar_rz = zeros(realizations_count, bits_count * 7 + 7);
final_polar_rz = zeros(realizations_count, bits_count * 7);

for k = 1:realizations_count
    n = 0;
    for j = 1:4:bits_count*4 +4
        adjusted_polar_rz(k, j + 3 * n:j + 3 + 3 * n) = repeated_polar_rz(k, j:j +
3);
        n = n + 1;
    end
end

for k = 1:realizations_count
    final_polar_rz(k, :) = adjusted_polar_rz(k, delays(k) + 1:delays(k) +
bits_count * 7);
end
```

The code performs polar RZ encoding by converting the binary data into values of $\boldsymbol{A}$ and $-\boldsymbol{A}$ for half the time period. The encoded signal is repeated 4 times to simulate partial transmission.

The signal is then adjusted so that only the first half of each bit period contains the encoded value, while the rest remains zero.

Finally, a transmission delay is applied by shifting the signal based on a randomly generated delay value.

Polar RZ Graph

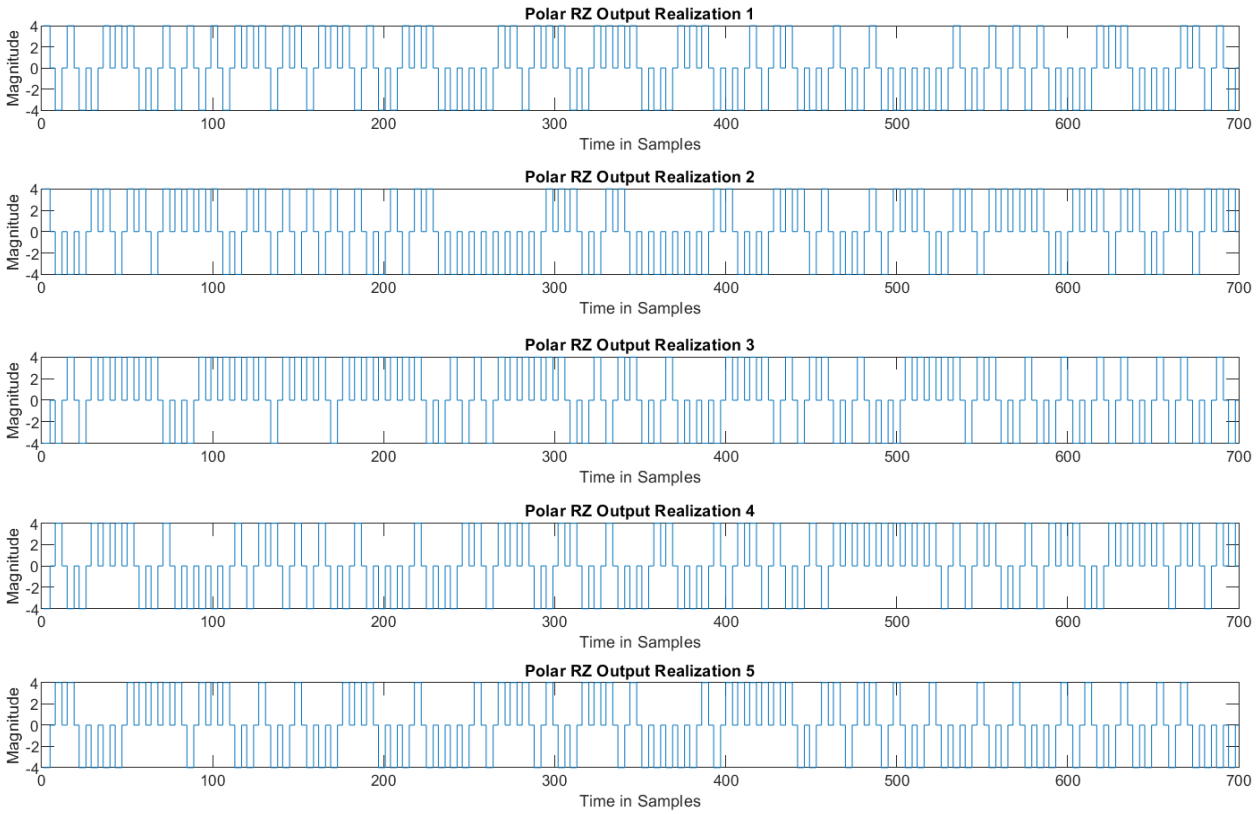

Figure 5. Polar RZ Realizations

Getting the cell arrays ready to calculate the statistical mean and autocorrelation

```
% Unipolar
for k = 1:realizations_count
    adjusted_unipolar(k,:)=repeated_unipolar(k,delays(k)+1:delays(k)+bits_count*7);
end


% Polar NRZ
for k = 1:realizations_count
  adjusted_polar_nrz(k,:)=repeated_polar_nrz(k,delays(k)+1:delays(k)+bits_count*7);
end

% Polar RZ
for k = 1:realizations_count
    final_polar_rz(k,:)=adjusted_polar_rz(k,delays(k)+1:delays(k)+bits_count*7);
end
```

To prepare the cell array for mean and autocorrelation calculations, the code extracts 700 elements (corresponding to num_bits * 7) after the time delay *td* for each realization. This process is applied to the unipolar, polar NRZ, and polar RZ encoded signals to align them correctly for further analysis.

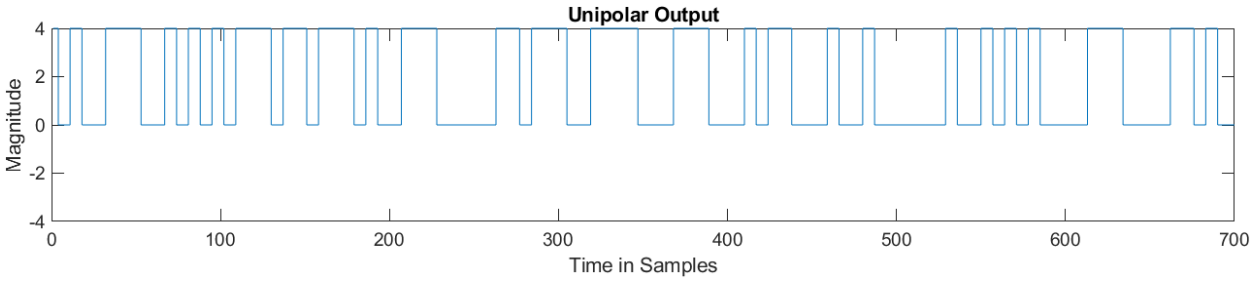

Figure 6. Unipolar Output

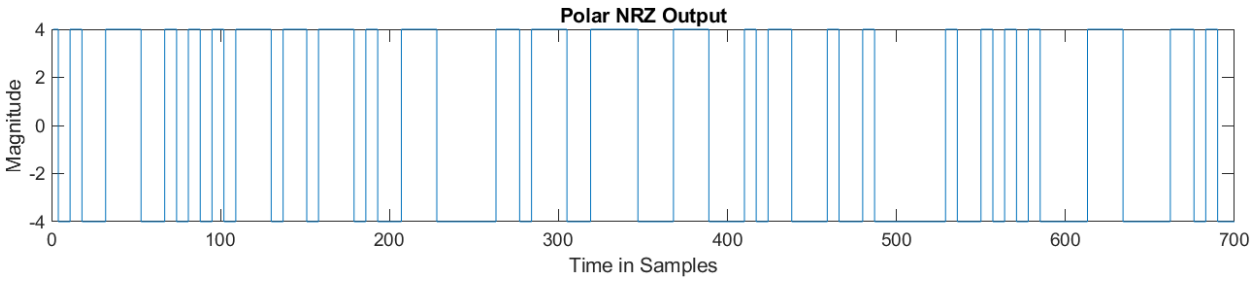

Figure 7. Polar NRZ Output

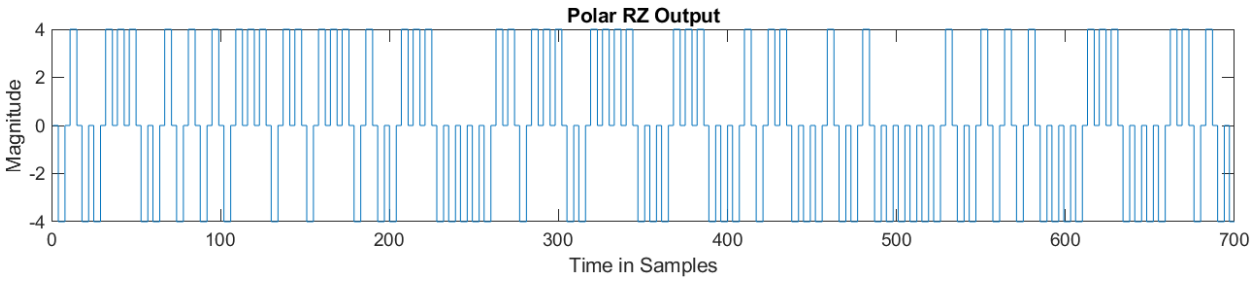

Figure 8. Polar RZ Output

Q1: Calculating the statistical mean (Hand Analysis)

Statistical mean can be calculated using the formula

$$\overline{\boldsymbol{x(t)}} = \boldsymbol{E}\big(\boldsymbol{x(t)}\big) = \sum \boldsymbol{P_i} \times \boldsymbol{x_i(t)}$$

LINE CODE	VALUE
UNIPOLAR NRZ	$\bar{x} = \frac{1}{2}(A) + \frac{1}{2}(0) = \frac{4}{2} = \mathbf{2}$
POLAR NRZ	$\bar{x} = \frac{1}{2}(-A) + \frac{1}{2}(A) = \mathbf{0}$
POLAR RZ	$\bar{x} = \frac{2}{7}(-A) + \frac{2}{7}(A) + \frac{3}{7}(0) = \mathbf{0}$

Plotting the statistical mean

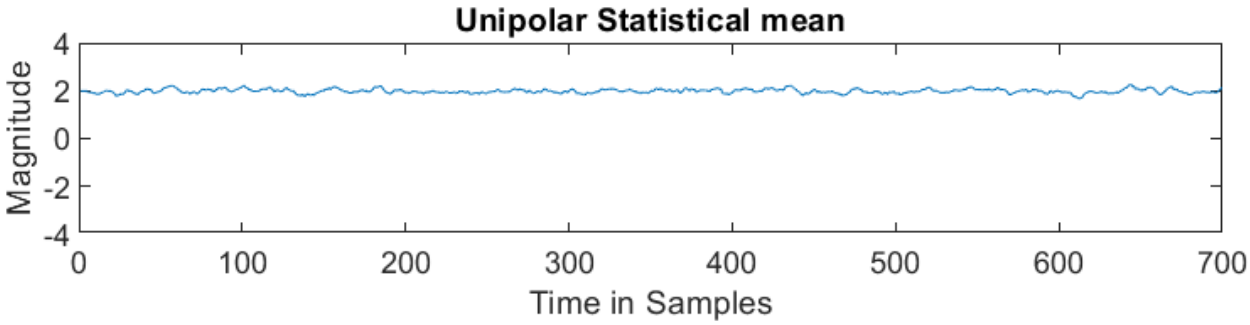

Figure 9. Unipolar Statistical Mean

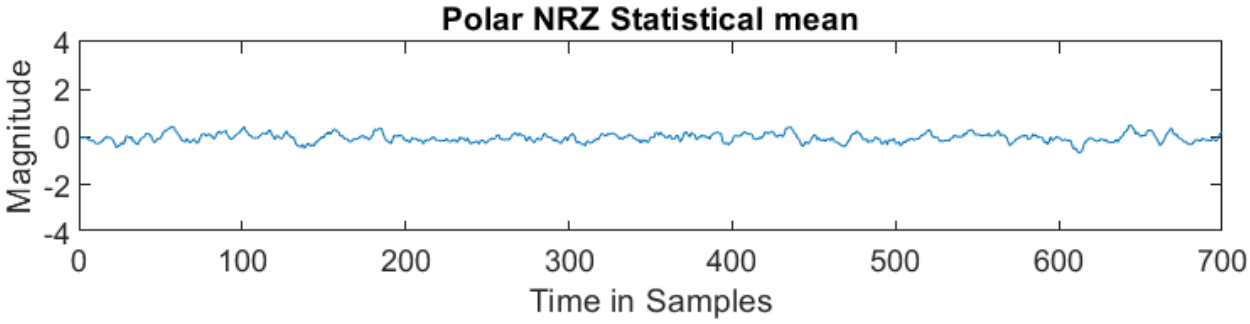

Figure 10. Polar NRZ Statistical Mean

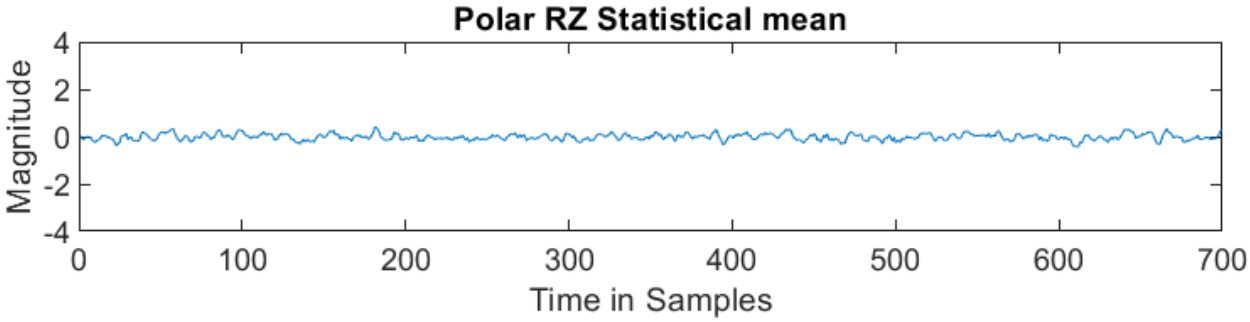

Figure 11. Polar RZ Statistical Mean

As shown in the figures, the statistical mean aligns with the results obtained from the hand analysis equations. When $\boldsymbol{A = 4}$, the mean value for the unipolar NRZ signal is $\frac{A}{2} = \frac{4}{2} = \mathbf{2}$, while the mean values for the polar NRZ and RZ signals are **zero**.

Hand Analysis: Calculating the statistical autocorrelation

Unipolar NRZ statistical autocorrelation

$R_x(\tau) = E(X(t) \times X(t+\tau))$

$E\big(X(t) \times X(t+\tau)\big)$

$= A^2 \times P(A,A) + 0 \times A \times P(0,A) + A \times 0 \times P(A,0) + 0 \times 0 \times P(0,0)$

$= A^2 \times P(A,A)$

$P(transaction) = P(t_1 < t_d + T < t_2) = \int_{t_1-T}^{t_2-T} \frac{1}{T} dt = \frac{\tau}{T}$

$P(no\ transaction) = 1 - \ P(transaction) = 1 - \frac{\tau}{T}$

$P(A,A) = P(X(t+\tau) = A \mid X(t) = A) \times p(X(t) = A)$

$P(X(t+\tau) = A | X(t) = A)$

$= P(no\ transaction) + P(transaction) \times p(X(t+\tau) = A)$

$= 1 - \frac{\tau}{2T}$

$P(A,A) = \left(1 - \frac{\tau}{2T}\right) \times \frac{1}{2}$

$For\ (\tau < T) \Rightarrow\ R_X(\tau) = \frac{A^2}{2} \times (1 - \frac{\tau}{2T})$

$For\ (\tau > T) \Rightarrow R_X(\tau) = A^2 \times p(A,A) = A^2 \times \frac{1}{2 \times 2} = \frac{A^2}{4}$

$Therefore,$

$R_X(\tau) = \frac{A^2}{2} \times \left(1 - \frac{\tau}{2T}\right) \ \rightarrow (|\tau| < T_b)$

$R_X(\tau) = \frac{A^2}{4} \qquad\qquad \rightarrow (|\tau| > T_b)$

$\boldsymbol{R_X(0) = 8}$
$\boldsymbol{R_X(\infty) = 4}$

Polar NRZ statistical autocorrelation

$R_x(\tau) = E(X(t) \times X(t+\tau))$

$E\big(X(t) \times X(t+\tau)\big)$

$= A^2 \times P(A,A) - A \times A \times P(-A,A) + A \times -A \times P(A,-A) - A \times (-A) \times P(-A,-A)$

$= A^2 \times P(A,A) + A^2 \times P(-A,-A) - A^2 \times P(A,-A) - A^2 \times P(-A,A)$

$P(transaction) = P(t_1 < t_d + T < t_2) = \int_{t_1-T}^{t_2-T} \frac{1}{T} dt = \frac{\tau}{T}$

$P(no\ transaction) = 1 - \ P(transaction) = 1 - \frac{\tau}{T}$

$P(A,A) = P(-A,-A) = P(X(t+\tau) = A \mid X(t) = A) \times p(X(t) = A)$

$P(X(t+\tau) = A | X(t) = A)$

$= P(no\ transaction) + P(transaction) \times p(X(t+\tau) = A) = 1 - \frac{\tau}{2T}$

$P(A,A) = P(-A,-A) = \left(1 - \frac{\tau}{2T}\right) \times \frac{1}{2}$

$P(A,-A) = P(-A,A) = P(X(t+\tau) = A | X(t) = \ -A) \times P(X(t) = -A))$

$P(X(t+\tau) = A | X(t) = \ -A) = P(transaction) \times P(X(t+\tau) = A) = \frac{1}{2} \times \frac{\tau}{T}$

$P(A,-A) = P(-A,A) = \frac{1}{2} \times \frac{\tau}{T} \times \frac{1}{2}$

$For\ (\tau < T) \Rightarrow \ R_X(\tau) = A^2 \times (1 - \frac{\tau}{T})$

$For\ (\tau > T) \Rightarrow \ R_X(\tau) = A^2 \times \frac{1}{4} + A^2 \times \frac{1}{4} - A^2 \times \frac{1}{4} - A^2 \times \frac{1}{4} = 0$

$Therefore,$

$R_X(\tau) = \ A^2 \times \left(1 - \frac{\tau}{T}\right) \ \rightarrow (|\tau| < T_b)$

$R_X(\tau) = 0 \qquad\qquad \rightarrow (|\tau| > T_b)$

$$\boldsymbol{R_X(0) = 16}$$

$$\boldsymbol{R_X(\infty) = 0}$$

Polar RZ statistical autocorrelation

$\boldsymbol{For\ (\tau < T_b)}$

$$E\big(X(t) \times X(t+\tau)\big) = A^2 \times P(A.A) + (-A^2) \times P(A.-A) + (-A^2) \times P(-A.A) + A^2 \times P(-A.A)$$
$$= 2 \times A^2 \times P(A.A) - 2 \times A^2 \times P(A.-A)$$

$$P(A.A) = P(X(t+\tau) = A \mid (X(t) = A) \times P(X(t) = A)$$

$$P(X(t+\tau) = A \mid (X(t) = A)) = P(no\ change) + P(change) \times P(X(t+\tau) = A)$$

$$P(change) = P(t < T_a < t+\tau) = \int_t^{t+\tau} \frac{7}{4T} dt = \frac{7\tau}{4T}$$

$$P(no\ change) = 1 - P(change) = 1 - \frac{7\tau}{4T}$$

$$P(X(t+\tau) = A \mid (X(t) = A)) = 1 - \frac{7\tau}{8T}$$

$$P(A.A) = P(-A.-A) = \frac{1}{2} \times \left(1 - \frac{7\tau}{8T}\right)$$

$$P(A.-A) = P(X(t+\tau) = A \mid (X(t) = -A)) \times P(X(t) = -A)$$

$$P(X(t+\tau) = A \mid (X(t) = -A)) = P(change) \times P(X(t+\tau) = A)$$

$$P(X(t+\tau) = A \mid (X(t) = -A)) = \frac{1}{2} \times \frac{7\tau}{4T} = \frac{7\tau}{8T}$$

$$P(A.-A) = P(-A.A) = \frac{1}{2} \times \frac{7\tau}{8T} = \frac{7\tau}{16T}$$

$\boldsymbol{For\ (\tau > T_b)}$

$$E\big(X(t) \times X(t+\tau)\big) = A^2 \times \frac{1}{4} + -A^2 \times \frac{1}{4} + A^2 \times \frac{1}{4} = 0$$

$Therefore,$

$$R_X(\tau) = \frac{4}{7} \times A^2 \times \left(1 - \frac{7 \times \tau}{4 \times T}\right) \qquad \rightarrow (|\tau| < T_b)$$

$$R_X(\tau) = 0 \qquad \rightarrow (|\tau| > T_b)$$

$$\boldsymbol{R_X(0) = \frac{4}{7} \times 16 = 9.1428}$$

$$\boldsymbol{R_X(\infty) = 0}$$

Plotting the statistical autocorrelation

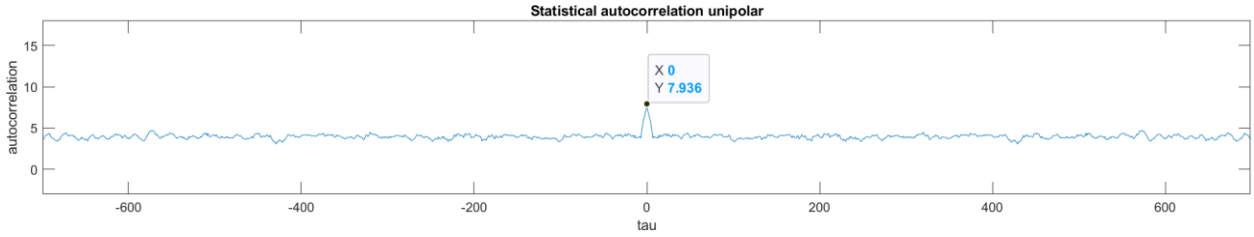

Figure 12. Unipolar Statistical Autocorrelation

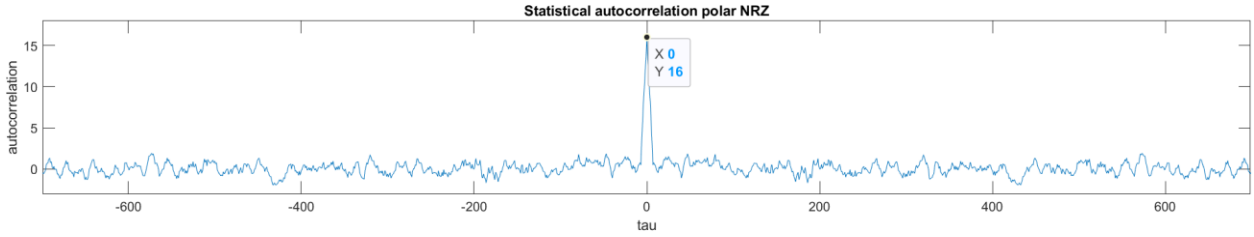

Figure 13. Polar NRZ Statistical Autocorrelation

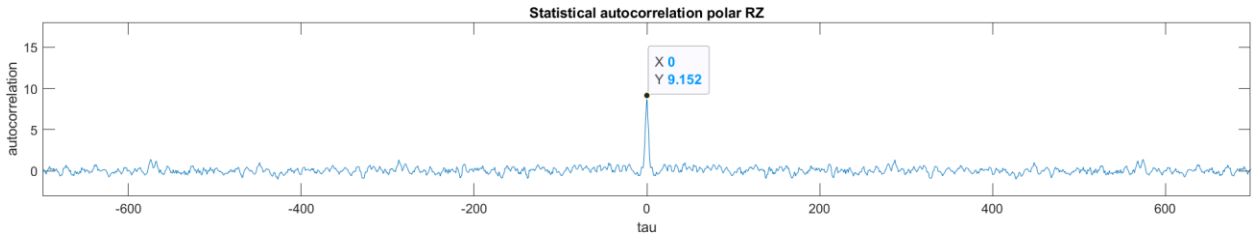

Figure 14. Polar RZ Statistical Autocorrelation

The autocorrelation reaches its maximum at a time difference $\boldsymbol{\tau = 0}$ because a signal correlates most strongly with itself at the same time instance. As $\boldsymbol{\tau}$ increases, the autocorrelation decreases until it becomes approximately constant.

- **Unipolar NRZ:** The autocorrelation becomes constant after 7 samples, which corresponds to the bit duration. The constant value is approximately 4, with a maximum value at $\tau = 0$ of $R_x(0) \approx 8$.
- **Polar NRZ:** The autocorrelation becomes constant after 7 samples (the bit duration), with a constant value of zero and a maximum at $\boldsymbol{\tau = 0}$ of $R_x(0) \approx 16$.
- **Polar RZ:** The autocorrelation becomes constant after 4 samples (the bit duration), with a constant value of zero and a maximum at $\boldsymbol{\tau = 0}$ of $R_x(0) \approx 9.1428$.

Q2: Is the process stationary?

Yes, the process is stationary.

Based on the hand analysis and the graph, we observe the following:

1. **Statistical Mean:**
    - The mean is almost constant over time.

TYPE	STATISTICAL MEAN
UNIPOLAR	$\frac{A}{2}$
POLAR NRZ	0
POLAR RZ	0

2. **Statistical Autocorrelation:**
    - The statistical autocorrelation is only a function of the time difference $\boldsymbol{\tau}$.

CONDITION	UNIPOLAR	POLAR NRZ	POLAR RZ
$\|\tau\| > T$	$\frac{A^2}{4}$	**0**	**0**
$\|\tau\| < T$	$\frac{A^2}{2} \times (1 - \frac{\tau}{2T})$	$A^2 \times (1 - \frac{\tau}{T})$	$\frac{4}{7} \times A^2 \times (1 - \frac{7\tau}{4T})$

Since the process satisfies the conditions of wide-sense stationarity (WSS), we can conclude that **Unipolar NRZ, Polar NRZ, and Polar RZ** are WSS.

Q4: Computing the time mean and autocorrelation of one waveform

Time Mean

Time mean is calculated using the formula

$$\widetilde{x(t)} = \frac{1}{N}\sum_{n=1}^{N} x[n]$$

TYPE	VALUE
UNIPOLAR	$\widetilde{x(t)} = \frac{1}{T} \times \left[\frac{T}{2} \times A + \frac{T}{2} \times 0\right] = \frac{A}{2} = 2$
POLAR NRZ	$\widetilde{x(t)} = \frac{1}{T} \times \left[\frac{T}{2} \times A + \frac{T}{2} \times -A\right] = 0$
POLAR RZ	$\widetilde{x(t)} = \frac{1}{T} \times \left[\frac{T}{4} \times A + \frac{T}{4} \times -A\right] = 0$

```
% Calculate the statistical mean for each line code
mean_unipolar = zeros(1, bits_count * 7);
mean_polar_nrz = zeros(1, bits_count * 7);
mean_polar_rz = zeros(1, bits_count * 7);

for k = 1:bits_count * 7
    mean_unipolar(k) = sum(adjusted_unipolar(:, k)) / realizations_count;
    mean_polar_nrz(k) = sum(adjusted_polar_nrz(:, k)) / realizations_count;
    mean_polar_rz(k) = sum(final_polar_rz(:, k)) / realizations_count;
end


% Calculate mean across time for each realization
time_mean_unipolar = zeros(1, realizations_count);
time_mean_polar_nrz = zeros(1, realizations_count);
time_mean_polar_rz = zeros(1, realizations_count);

for k = 1:realizations_count
    time_mean_unipolar(k) = sum(adjusted_unipolar(k, :)) / (bits_count * 7);
    time_mean_polar_nrz(k) = sum(adjusted_polar_nrz(k, :)) / (bits_count * 7);
```

```
    time_mean_polar_rz(k) = sum(final_polar_rz(k, :)) / (bits_count * 7);
end
```

The code calculates the **statistical mean** and the **mean across time** for the three different line codes.

1. **Statistical Mean Calculation:**
    - Three arrays *(stat_unipolar_mean, stat_polar_NRZ_mean, and stat_polar_RZ_mean)* are initialized to store the statistical mean values over time.
    - The for loop iterates through each time step (num_bits * 7) and computes the statistical mean by summing the signal values across all realizations and dividing by the total number of realizations (num_realizations).
2. **Mean Across Time Calculation:**
    - Three arrays *(time_unipolar_mean, time_polar_NRZ_mean, and time_polar_RZ_mean)* are initialized to store the mean values for each realization over time.
    - The second for loop iterates through each realization and computes the mean by summing the signal values over time and dividing by the total number of time steps (num_bits * 7).

Time Mean Graph

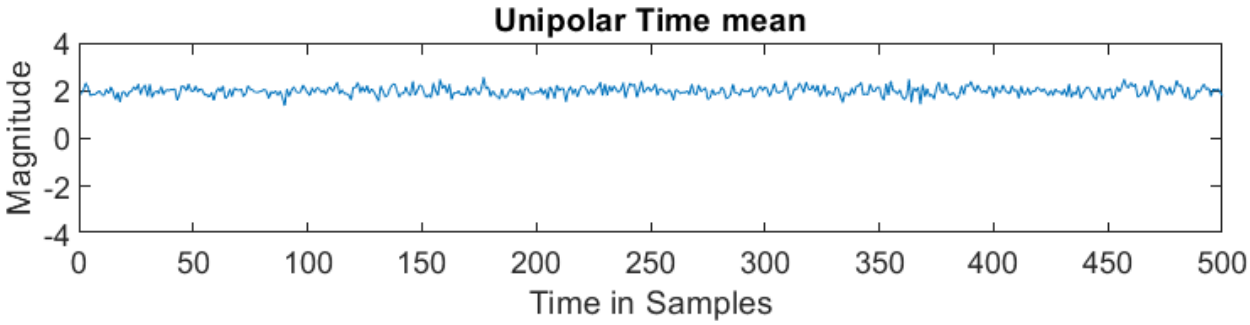

Figure 15. Unipolar Time Mean

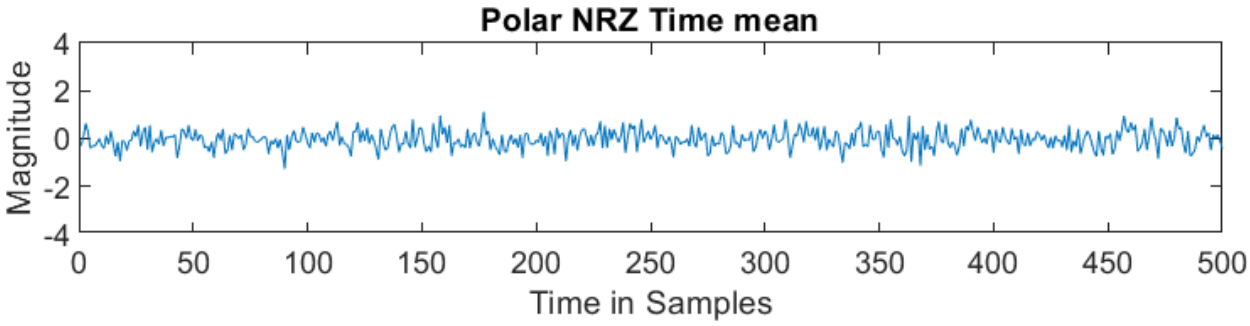

Figure 16. Polar NRZ Time Mean

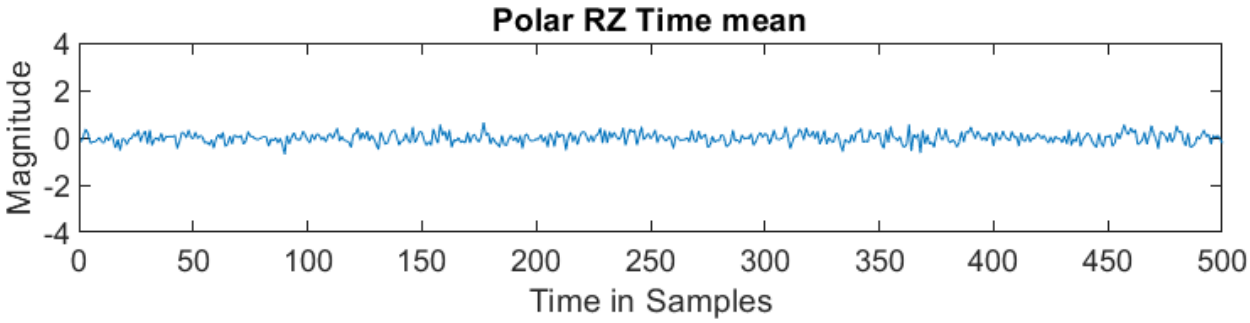

Figure 17. Polar RZ Time Mean

Time Autocorrelation

Time autocorrelation is calculated using the formula

$$R_X(t) = < x(t) \times x(t+\tau) >= \frac{1}{N} \times \sum_n x(n) \times x(n+\tau)$$

```
% Function to calculate time-domain autocorrelation

function result = compute_time_autocorr(x)
    global bits_count;
    time_autocorr = zeros(1, bits_count * 7);
    for tau = 0:bits_count * 7 - 1
        sum_time_autocorr = 0;
        for counter = 1:(bits_count * 7 - tau)
            sum_time_autocorr=sum_time_autocorr+(x(1,counter)*x(1, (counter+tau)));
        end
        time_autocorr(tau + 1) = (1/(bits_count * 7 - tau)) * sum_time_autocorr;
    end
    result = [fliplr(time_autocorr(2:end)) time_autocorr];
end
```

The function ***calc_time_autocorr*** calculates the **time-domain autocorrelation** of a signal.

1. **Initialization:**
    - The function takes an input signal x.
    - It defines a global variable num_bits to determine the size of the signal window.
    - An array time_autocorr is initialized to store the autocorrelation values for each time lag $\boldsymbol{\tau} \rightarrow \boldsymbol{tau}$.
2. **Autocorrelation Calculation:**
    - The outer for loop iterates through each possible time lag τ from 0 to (num_bits*7 – 1).
    - For each τ, the inner for loop multiplies the signal value at position counter with the signal value at $position\ counter + \tau$ and accumulates the result in *sum_time_autocorr.*
    - The autocorrelation value at each τ is computed by dividing the accumulated sum by the number of valid terms (num_bits * 7 - τ).

Time Autocorrelation Graph

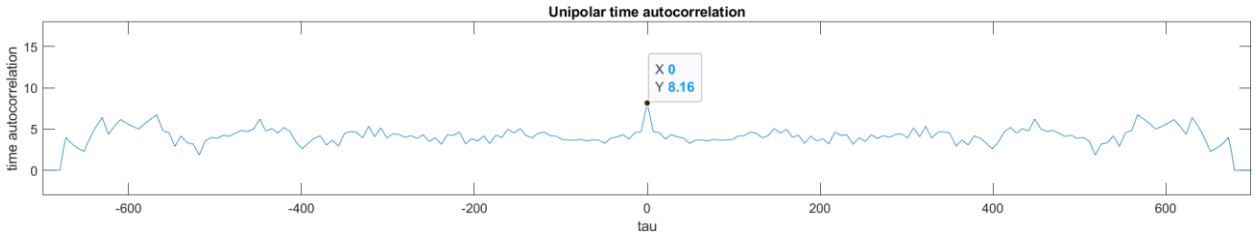

Figure 18. Unipolar Time Autocorrelation

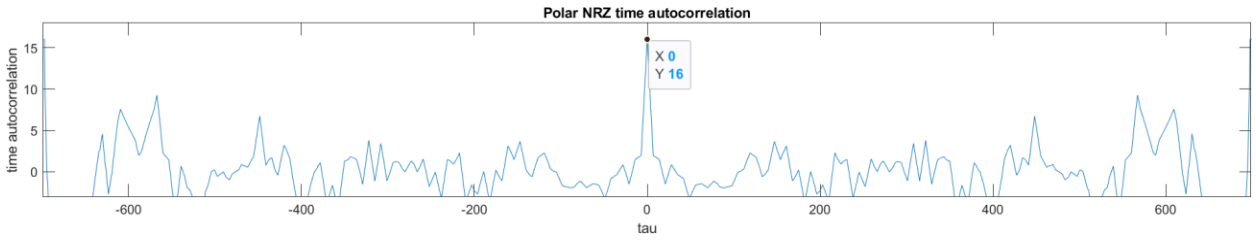

Figure 19. Polar NRZ Time Autocorrelation

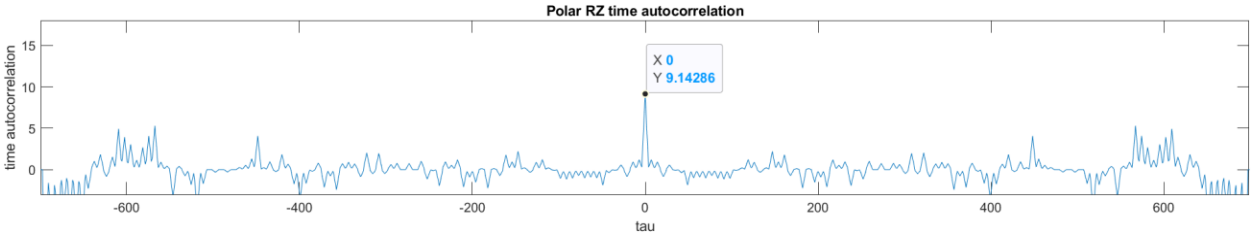

Figure 20. Polar RZ Time Autocorrelation

As shown in the graphs, the time autocorrelation is closely same as the ensemble autocorrelation.

Q5: Is the random process ergodic?

Yes, the process is ergodic.

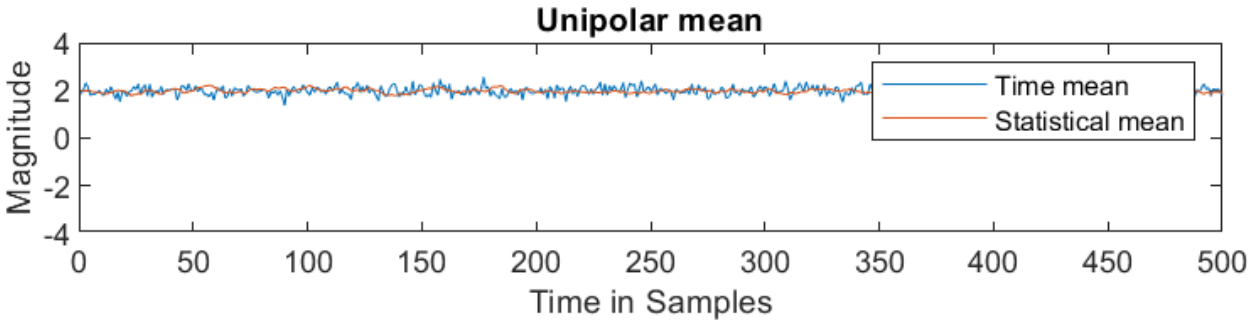

Figure 21. Unipolar Mean

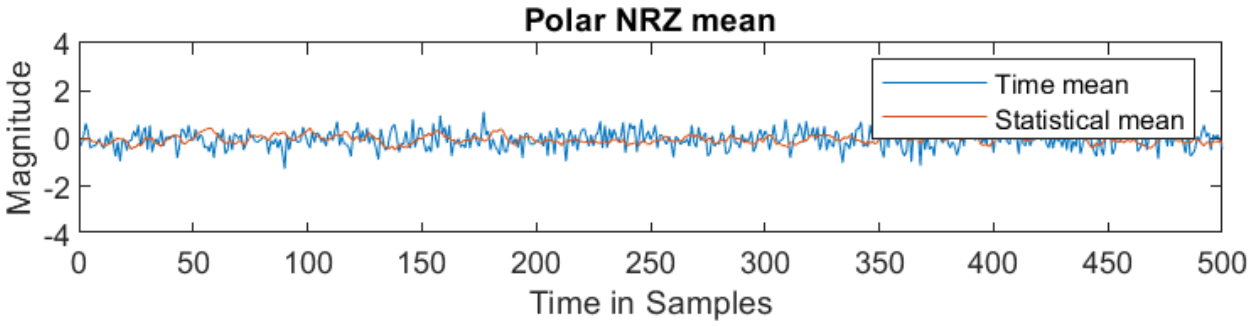

Figure 22. Polar NRZ Mean

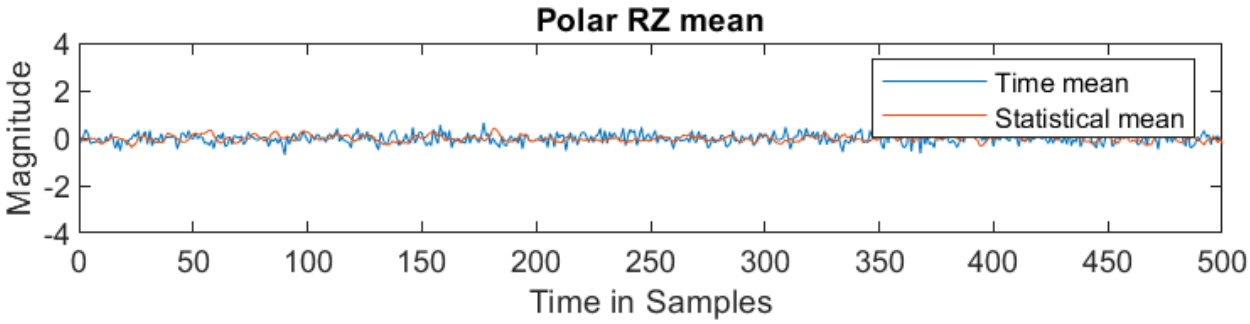

Figure 23. Polar RZ Mean

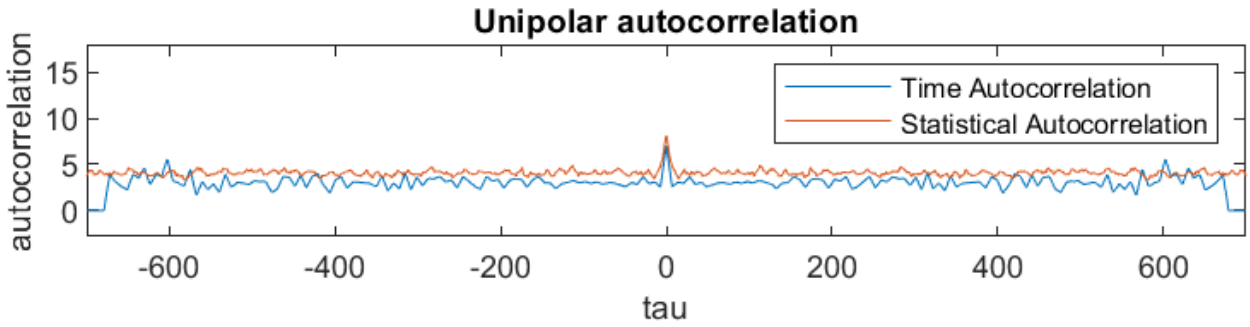

Figure 24. Unipolar Autocorrelation

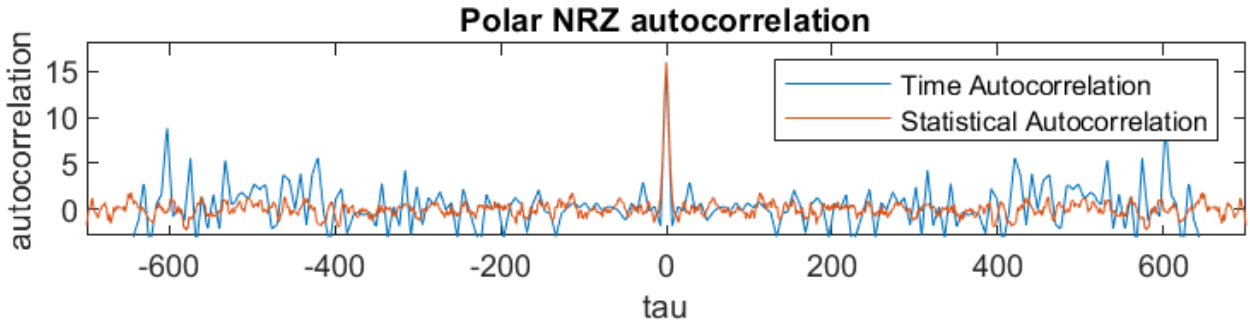

Figure 25. Polar NRZ Autocorrelation

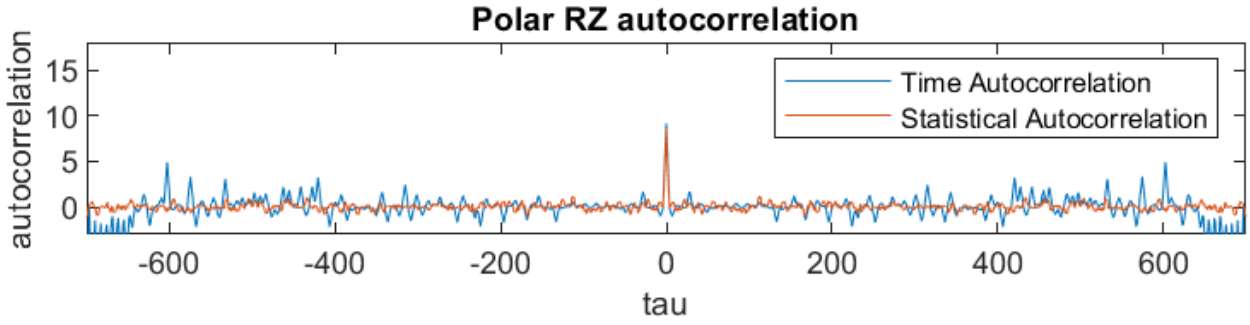

Figure 26. Polar RZ Autocorrelation

- The time mean is similar to the statistical mean, and both remain approximately constant.
- The time autocorrelation is similar to the ensemble autocorrelation.

Since these conditions satisfy the definition of ergodicity, we can conclude that Unipolar NRZ, Polar NRZ, and Polar RZ **are ergodic processes**. Any minor differences between the time and ensemble mean or autocorrelation will disappear as the number of bits approaches the number of realizations.

Plotting the PSD of the ensemble

PSD can be calculated using Fourier transform:

$$\boldsymbol{PSD = FT(R_X(\tau))}$$

TYPE	$\boldsymbol{S_{XX}(w)}$
UNIPOLAR	$\frac{A^2}{4} \times (T \times sinc^2\left(\frac{wT}{2}\right) + \delta(t))$
POLAR NRZ	$A^2 \times T \times sinc^2\left(\frac{wT}{2}\right)$
POLAR RZ	$A^2 \times \frac{16}{49} \times T \times sinc^2\left(\frac{wT}{2} \times \frac{4}{7}\right)$

```
% Calculate Power Spectral Density
sample_rate = 100; % Sampling frequency (Hz)

fft_polar_nrz = abs(fft(autocorr_polar_nrz));
fft_unipolar = abs(fft(autocorr_unipolar));
fft_polar_rz = abs(fft(autocorr_polar_rz));

N = length(autocorr_polar_nrz);  % Number of samples
freq_axis = (-N/2:N/2-1) * sample_rate/N;
```

The code calculates the Power Spectral Density (PSD) for the three types of line codes.

It starts by defining the sampling frequency (fs) as 100 Hz. The code then computes the Fast Fourier Transform (*FFT*) of the autocorrelation functions *(polar_NRZ_autocorr, unipolar_autocorr, polar_RZ_autocorr)* and takes the absolute value to get the magnitude of the frequency components. The length of the autocorrelation sequence (N) is determined, which is used to generate the frequency axis (f) ranging from $\frac{-f_s}{2}$ to $\frac{f_s}{2}$ in steps of $\frac{f_s}{N}$.

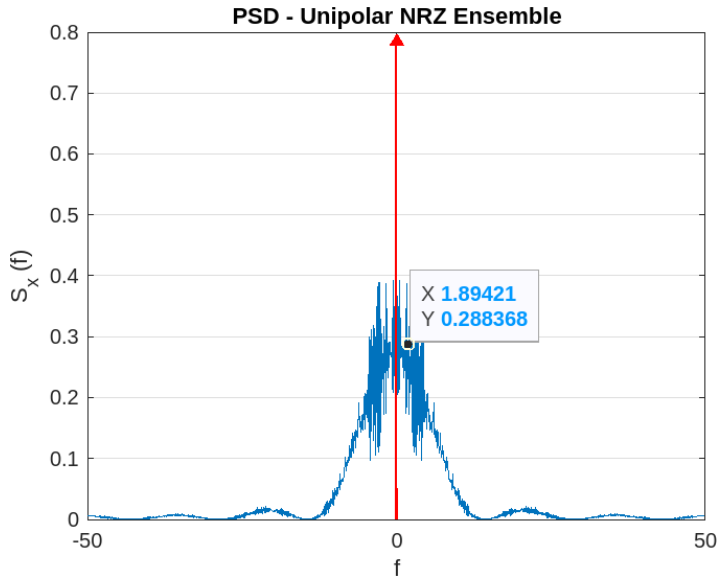

Figure 27. PSD - Unipolar NRZ

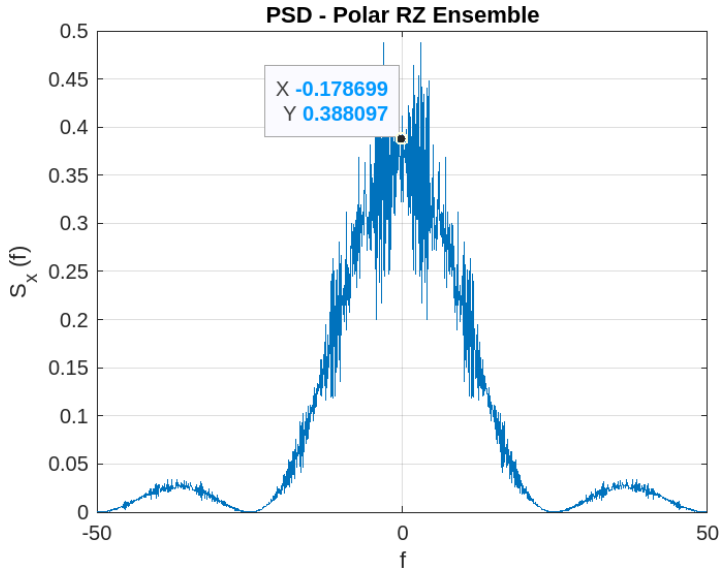

Figure 28. PSD - Polar RZ

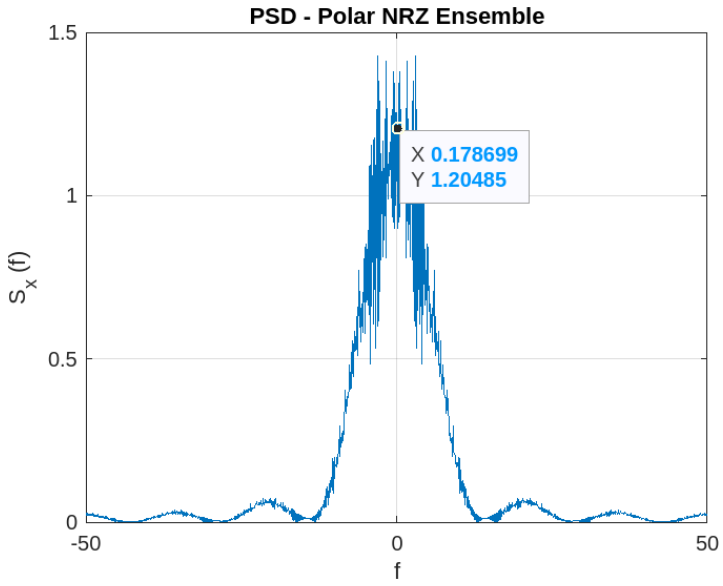

Figure 29. PSD - Polar NRZ

So, from the previous graphs and equations we can conclude:

TYPE	CALCULATED @ $f = 0$	SIMULATED @ $f = 0$
UNIPOLAR	$\frac{A^2}{4} \times T = \mathbf{0.28}$	$\mathbf{0.288}$
POLAR NRZ	$A^2 \times T = \mathbf{1.12}$	$\mathbf{1.204}$
POLAR RZ	$A^2 \times \frac{16}{49} \times T = \mathbf{0.365}$	$\mathbf{0.388}$

After analyzing the FFT output, the amplitudes of the PSD were larger than expected.

To correct this, we normalized the PSD amplitudes by dividing by the sampling frequency (f_s) to obtain accurate values.

The resulting amplitudes and bandwidth were acceptable compared to the expected values.

However, the accuracy can be increased by increasing the number of realizations or samples, as demonstrated in the previous section. A higher number of samples results in time statistics that more closely approximate those of the entire ensemble. Also increasing the number of realizations improves statistical accuracy, leading to better alignment between time statistics and ensemble statistics.

Q6: What is the bandwidth of the transmitted signal?

From equations and the plotted graphs, we can conclude:

TYPE	CALCULATED BW	SIMULATED
UNIPOLAR	$bit\ rate = \frac{1}{T_b} = \mathbf{14.285\ Hz}$	$\mathbf{14.1887\ Hz}$
POLAR NRZ	$bit\ rate = \frac{1}{T_b} = \mathbf{14.285\ Hz}$	$\mathbf{14.3317\ Hz}$
POLAR RZ	$bit\ rate = \frac{1}{T_b} \times \frac{7}{4} = \mathbf{25\ Hz}$	$\mathbf{25.1966\ Hz}$

Appendix: Full MATLAB code

```
close all;
clear;
clc;

global bits_count;
global realizations_count;

scale = 4;
bits_count = 100;
realizations_count = 500;

binary_data = randi([0, 1], realizations_count, bits_count + 1);
delays = randi([0, 6], realizations_count, 1);

encoded_unipolar = binary_data * scale;
repeated_unipolar = repelem(encoded_unipolar, 1, 7);
adjusted_unipolar = zeros(realizations_count, bits_count * 7);
for k = 1:realizations_count
    adjusted_unipolar(k, :) = repeated_unipolar(k, delays(k) + 1:delays(k) +
bits_count * 7);
end

encoded_polar_nrz = ((2 * binary_data) - 1) * scale;
repeated_polar_nrz = repelem(encoded_polar_nrz, 1, 7);
adjusted_polar_nrz = zeros(realizations_count, bits_count * 7);
for k = 1:realizations_count
    adjusted_polar_nrz(k, :) = repeated_polar_nrz(k, delays(k) + 1:delays(k) +
bits_count * 7);
end

repeated_polar_rz = repelem(encoded_polar_nrz, 1, 4);
adjusted_polar_rz = zeros(realizations_count, bits_count * 7 + 7);
final_polar_rz = zeros(realizations_count, bits_count * 7);
for k = 1:realizations_count
    n = 0;
    for j = 1:4:bits_count*4 +4
        adjusted_polar_rz(k, j + 3 * n:j + 3 + 3 * n) = repeated_polar_rz(k, j:j +
3);
        n = n + 1;
    end
end
for k = 1:realizations_count
    final_polar_rz(k, :) = adjusted_polar_rz(k, delays(k) + 1:delays(k) +
bits_count * 7);
end

figure;
for k = 1:5
    subplot(5, 1, k)
    stairs(repeated_unipolar(k, 1:bits_count * 7))
    title(sprintf("Unipolar Output Realization %d", k));
    xlabel("Time in Samples");
```

```
    ylabel("Magnitude");
    axis([0 bits_count * 7 -4 4]);
end

figure;
for k = 1:5
    subplot(5, 1, k)
    stairs(adjusted_polar_rz(k, 1:bits_count * 7))
    title(sprintf("Polar RZ Output Realization %d", k));
    xlabel("Time in Samples");
    ylabel("Magnitude");
    axis([0 bits_count * 7 -4 4]);
end

figure;
for k = 1:5
    subplot(5, 1, k)
    stairs(repeated_polar_nrz(k, 1:bits_count * 7))
    title(sprintf("Polar NRZ Output Realization %d", k));
    xlabel("Time in Samples");
    ylabel("Magnitude");
    axis([0 bits_count * 7 -4 4]);
end

figure;
subplot(3, 1, 1)
stairs(adjusted_unipolar(1, 1:bits_count * 7))
title("Unipolar Output");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 bits_count * 7 -4 4]);

subplot(3, 1, 2)
stairs(adjusted_polar_nrz(1, 1:bits_count * 7))
title("Polar NRZ Output");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 bits_count * 7 -4 4]);

subplot(3, 1, 3)
stairs(final_polar_rz(1, 1:bits_count * 7))
title("Polar RZ Output");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 bits_count * 7 -4 4]);

mean_unipolar = zeros(1, bits_count * 7);
mean_polar_nrz = zeros(1, bits_count * 7);
mean_polar_rz = zeros(1, bits_count * 7);

for k = 1:bits_count * 7
    mean_unipolar(k) = sum(adjusted_unipolar(:, k)) / realizations_count;
    mean_polar_nrz(k) = sum(adjusted_polar_nrz(:, k)) / realizations_count;
    mean_polar_rz(k) = sum(final_polar_rz(:, k)) / realizations_count;
end
```

```
time_mean_unipolar = zeros(1, realizations_count);
time_mean_polar_nrz = zeros(1, realizations_count);
time_mean_polar_rz = zeros(1, realizations_count);

for k = 1:realizations_count
    time_mean_unipolar(k) = sum(adjusted_unipolar(k, :)) / (bits_count * 7);
    time_mean_polar_nrz(k) = sum(adjusted_polar_nrz(k, :)) / (bits_count * 7);
    time_mean_polar_rz(k) = sum(final_polar_rz(k, :)) / (bits_count * 7);
end

figure;
subplot(3, 1, 1)
plot(mean_unipolar)
title("Unipolar Statistical mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 bits_count * 7 -4 4]);

subplot(3, 1, 2)
plot(mean_polar_nrz)
title("Polar NRZ Statistical mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 bits_count * 7 -4 4]);

subplot(3, 1, 3)
plot(mean_polar_rz)
title("Polar RZ Statistical mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 bits_count * 7 -4 4]);

figure;
subplot(3, 1, 1)
plot(time_mean_unipolar)
title("Unipolar Time mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 realizations_count -4 4]);

subplot(3, 1, 2)
plot(time_mean_polar_nrz)
title("Polar NRZ Time mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 realizations_count -4 4]);

subplot(3, 1, 3)
plot(time_mean_polar_rz)
title("Polar RZ Time mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 realizations_count -4 4]);
```

```
figure;
subplot(3, 1, 1)
plot(time_mean_unipolar)
hold on;
plot(mean_unipolar)
legend("Time mean", "Statistical mean")
title("Unipolar mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 realizations_count -4 4]);

subplot(3, 1, 2)
plot(time_mean_polar_nrz)
hold on;
plot(mean_polar_nrz)
legend("Time mean", "Statistical mean")
title("Polar NRZ mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 realizations_count -4 4]);

subplot(3, 1, 3)
plot(time_mean_polar_rz)
hold on;
plot(mean_polar_rz)
legend("Time mean", "Statistical mean")
title("Polar RZ mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 realizations_count -4 4]);

% Modified autocorrelation plots with symmetric x-axis
tau_max = bits_count * 7 - 1;
tau_axis = -tau_max:tau_max;

autocorr_unipolar = compute_stat_autocorr(adjusted_unipolar);
autocorr_polar_nrz = compute_stat_autocorr(adjusted_polar_nrz);
autocorr_polar_rz = compute_stat_autocorr(final_polar_rz);

time_autocorr_unipolar = compute_time_autocorr(adjusted_unipolar);
time_autocorr_polar_nrz = compute_time_autocorr(adjusted_polar_nrz);
time_autocorr_polar_rz = compute_time_autocorr(final_polar_rz);

figure;
subplot(3, 1, 1)
plot(tau_axis, autocorr_unipolar)
axis([-tau_max tau_max -3 18])
title('Statistical autocorrelation unipolar');
xlabel('tau');
ylabel('autocorrelation');

subplot(3, 1, 2)
plot(tau_axis, autocorr_polar_nrz)
axis([-tau_max tau_max -3 18])
title('Statistical autocorrelation polar NRZ');
```

```
xlabel('tau');
ylabel('autocorrelation');

subplot(3, 1, 3)
plot(tau_axis, autocorr_polar_rz)
axis([-tau_max tau_max -3 18])
title('Statistical autocorrelation polar RZ');
xlabel('tau');
ylabel('autocorrelation');

figure;
subplot(3, 1, 1)
plot(tau_axis, time_autocorr_unipolar)
axis([-tau_max tau_max -3 18])
title('Unipolar time autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');

subplot(3, 1, 2)
plot(tau_axis, time_autocorr_polar_nrz)
axis([-tau_max tau_max -3 18])
title('Polar NRZ time autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');

subplot(3, 1, 3)
plot(tau_axis, time_autocorr_polar_rz)
axis([-tau_max tau_max -3 18])
title('Polar RZ time autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');

figure;
subplot(3, 1, 1)
plot(tau_axis, time_autocorr_unipolar)
hold on;
plot(tau_axis, autocorr_unipolar)
legend("Time Autocorrelation", "Statistical Autocorrelation")
axis([-tau_max tau_max -3 18])
title('Unipolar autocorrelation');
xlabel('tau');
ylabel('autocorrelation');

subplot(3, 1, 2)
plot(tau_axis, time_autocorr_polar_nrz)
hold on;
plot(tau_axis, autocorr_polar_nrz)
legend("Time Autocorrelation", "Statistical Autocorrelation")
axis([-tau_max tau_max -3 18])
title('Polar NRZ autocorrelation');
xlabel('tau');
ylabel('autocorrelation');

subplot(3, 1, 3)
plot(tau_axis, time_autocorr_polar_rz)
```

```
hold on;
plot(tau_axis, autocorr_polar_rz)
legend("Time Autocorrelation", "Statistical Autocorrelation")
axis([-tau_max tau_max -3 18])
title('Polar RZ autocorrelation');
xlabel('tau');
ylabel('autocorrelation');

sample_rate = 100; % Sampling frequency (Hz)

fft_polar_nrz = abs(fft(autocorr_polar_nrz));
fft_unipolar = abs(fft(autocorr_unipolar));
fft_polar_rz = abs(fft(autocorr_polar_rz));

N = length(autocorr_polar_nrz);  % Number of samples
freq_axis = (-N/2:N/2-1) * sample_rate/N;

figure;
subplot(3, 1, 1);
plot(freq_axis, fftshift(fft_unipolar)/sample_rate);
title('PSD - Unipolar NRZ Ensemble');
xlabel('f');
ylabel('Sx(f)');
ylim([0 10]);

subplot(3, 1, 2);
plot(freq_axis, fftshift(fft_polar_nrz)/sample_rate);
title('PSD - Polar NRZ Ensemble');
xlabel('f');
ylabel('Sx(f)');
ylim([0 3]);

subplot(3, 1, 3);
plot(freq_axis, fftshift(fft_polar_rz)/sample_rate);
title('PSD - Polar RZ Ensemble');
xlabel('f');
ylabel('Sx(f)');
ylim([0 1]);

function result = compute_stat_autocorr(x)
    global realizations_count;
    global bits_count;
    stat_autocorr = zeros(1, bits_count * 7);
    for k = 1:bits_count * 7
        stat_autocorr(k) = (1/realizations_count) * sum(x(:, 1) .* x(:, k));
    end
    result = [fliplr(stat_autocorr(2:end)) stat_autocorr];
end

function result = compute_time_autocorr(x)
    global bits_count;
    time_autocorr = zeros(1, bits_count * 7);
    for tau = 0:bits_count * 7 - 1
        sum_time_autocorr = 0;
        for counter = 1:(bits_count * 7 - tau)
```

```
            sum_time_autocorr = sum_time_autocorr + (x(1, counter) * x(1, (counter
+ tau)));
        end
        time_autocorr(tau + 1) = (1/(bits_count * 7 - tau)) * sum_time_autocorr;
    end
    result = [fliplr(time_autocorr(2:end)) time_autocorr];
end
```