**Electronics and Electrical Communications Engineering  
Department**

**Faculty of Engineering**

**Cairo University**

**Implementation and Comparative Analysis of Huffman and Fano Source Coding Algorithms**

**ELC4020 “Advanced Communication Systems “**

**4th Year**

**1st Semester - Academic Year 2025/2026**

**Prepared by:**

|  |  |  |
| --- | --- | --- |
| **NAME** | **SECTION** | **ID** |
| **Mohamed Ahmed Abd El Hakam** | **3** | **9220647** |
| **Yousef Khaled Omar Mahmoud** | **4** | **9220984** |

**Submission Date:** 2 November 2025

**Instructor: Eng. Mohamed Khaled**

**Dr. Mohammed Nafie & Dr. Mohamed Khairy**

# Contents

[A. **Contents** 2](#_Toc195652116)

[**B.** **Table of Figures** 3](#_Toc195652117)

[**C.** **Role of Each Member** 4](#_Toc195652118)

[**D.** **Project Description** 5](#_Toc195652119)

[**E.** **Introduction** 5](#_Toc195652120)

[**F.** **Control Flags** 5](#_Toc195652121)

[**G.** **Generation of Data** 6](#_Toc195652122)

[**H.** **polar NRZ ensemble creation** 6](#_Toc195652123)

[**I.** **Uni polar NRZ ensemble creation** 7](#_Toc195652124)

[**J.** **polarRZ ensemble creation** 8](#_Toc195652125)

[**K.** **Random initial time shift** 9](#_Toc195652126)

[**L.** **Getting cell arrays ready to calculate the statistical mean and autocorrelation:** 10](#_Toc195652127)

[**M.** **Questions** 12](#_Toc195652128)

[**1.** **Statistical Mean** 12](#_Toc195652129)

[**1.1.** **Hand Analysis** 12](#_Toc195652130)

[**1.2.**  **Code Snippet** 12](#_Toc195652131)

[**1.3.** **Plotting the Statistical Mean:** 13](#_Toc195652132)

[**2.** **Statistical Autocorrelation** 14](#_Toc195652133)

[**2.1.** **Hand Analysis** 14](#_Toc195652134)

[**2.2.** **Code Snippet** 15](#_Toc195652135)

[**2.3.** **Plotting the statistical autocorrelation** 16](#_Toc195652136)

[**3.** **Is the Process Stationary** 18](#_Toc195652137)

[**4.** **The time mean and autocorrelation function for one waveform** 19](#_Toc195652138)

[**4.1.** **Time Mean** 19](#_Toc195652139)

[**4.3.** **Time Auto Correlation** 22](#_Toc195652140)

[**4.4.** **Time Auto Correlation for one wave form:** 23](#_Toc195652141)

[**5.** **Is The Random Process Ergodic?** 24](#_Toc195652142)

[**6.** **the PSD & Bandwidth of the Ensemble** 26](#_Toc195652143)

[**6.1.** **PSD using fft:** 26](#_Toc195652144)

[**6.2.** **Theoritical PSD:** 28](#_Toc195652145)

[**N.** **References:** 29](#_Toc195652146)

[**O.** **Appendix** 29](#_Toc195652147)

# Table of Figures

[Figure 1 Rx and Tx path 5](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652148)

[Figure 2 ADC Binary Output 6](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652149)

[Figure 3 PolarNRZ Realizations 7](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652150)

[Figure 4 Uni Polar Realizations 8](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652151)

[Figure 5 PolarRZ Realization 9](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652152)

[Figure 6 Realization Shifted 10](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652153)

[Figure 7 Plot of Statistical Mean 13](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652154)

[Figure 8 Statistical Auto Correction plot 16](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652155)

[Figure 9 Statistical Auto Correction plot zoomed 16](#_Toc195652156)

[Figure 10 autocorrelation at two different times 18](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652157)

[Figure 11 Time Mean for Uni Polar 19](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652158)

[Figure 12 Time Mean for Polar NRZ 20](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652159)

[Figure 13 Time Mean for Polar RZ 20](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652160)

[Figure 14 Time Mean Vs Realization 21](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652161)

[Figure 15 Time Auto Correction plot zoomed 23](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652162)

[Figure 16 Time Auto Correction plot 23](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652163)

[Figure 17 Time Mean vs Statistical 24](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652164)

[Figure 18 Time Auto Correlation Vs Statistical 24](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652165)

[Figure 19 PSD plot of the Ensemble 27](file:///D:\project\Digital%20Communication%20Radio\Digital_Communication_Radio\project%201\P1-T25.docx#_Toc195652166)

# Role of Each Member

|  |  |
| --- | --- |
| Role | Name |
| code the Huffman source coding | Youssef Khaled |
| compute the realization calculation | Ahmed Mohamed |
| compute the time calculation | Shahd Hamed |
| Report and Hand Analysis | Mohamed Ahmed |
| Report and Hand Analysis | Omar Ahmed |

# Project Description

Using software radio technique (SDR) to transmit stream of randomness bits through an ideal channel (which performing a small delay) using Matlab. Performing measures and analysis to see the performance of the system through three main line codes (unipolar, polar nrz and polar rz).

هنا ممكن تشرحله انه تشرحله انه  
source code implementqation using matlab

# Introduction

Software radio is a revolutionary approach that brings the programming code directly to the antenna, minimizing reliance on traditional radio hardware as shown in figure 1.

By doing so, it transforms challenges associated with radio hardware into software- related issues. Unlike conventional radios, where signal processing primarily relies on analog circuitry or a combination of analog and digital chips, software radio operates by having software dictate both the transmitted and received waveforms.

هنا ممكن تشرحله انه تشرحله انه  
what’s source coding and why it’s important

This paradigm shift allows for greater flexibility and adaptability in radio systems, as they can be easily reconfigured and optimized through software updates, rather than hardware modifications.

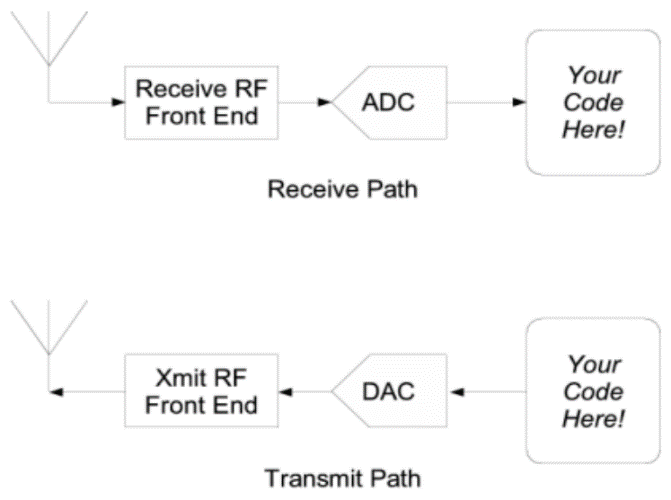


Figure 1 Rx and Tx path

# Control Flags

مفيش المرة ده ممكن تمسحها لو عاوز

|  |  |  |
| --- | --- | --- |
| Flag | Value | Description |
| A | 4 | Amplitude of line code |
| N\_realizations | 500 | Number of waveforms (ensemble size) |
| num\_bits | 101 | Bits per waveform and one extra bit for shifting |
| bit\_duration | 70e-3 | Duration of each bit |
| dac\_interval | 10e-3 | DAC update interval |

# Input Data Symbols

As seen this is the input given

كمل هنا عشان معيش أفكار

# Huffman Source Coding

حط هنا شوية معلومات عنه زي مثلا انه ممكن يوصل ل 100% لو كان عددهم ما لا نهاية

## Hand Analysis

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | P | C | P | C | P | C | P | C | P | C | P | C |
| A | 0.35 | 00 | 0.35 | 00 | 0.35 | 00 | 0.35 | 00 | 0.35 | 1 | 0.65 | 0 |
| B | 0.3 | 01 | 0.3 | 01 | 0.3 | 01 | 0.3 | 01 | 0.35 | 00 | 0.35 | 1 |
| C | 0.2 | 10 | 0.2 | 10 | 0.2 | 10 | 0.2 | 10 | 0.3 | 01 |  |  |
| D | 0.1 | 110 | 0.1 | 110 | 0.1 | 110 | 0.15 | 11 |  |  |  |  |
| E | 0.04 | 1110 | 0.04 | 1110 | 0.05 | 111 |  |  |  |  |  |  |
| F | 0.005 | 11110 | 0.01 | 1111 |  |  |  |  |  |  |  |  |
| G | 0.005 | 11110 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

So here’s the output that we are aiming for.

With the entropy, average length and efficiency:

So by calculating them we can find that:

And

So the overall efficiency is

# Generation of Data



Using the function: “**Randi**” to generate random binary data of size 500x101[3]

(500 waveforms each with 101 bits). This data represents the binary bits that need to be encoded.

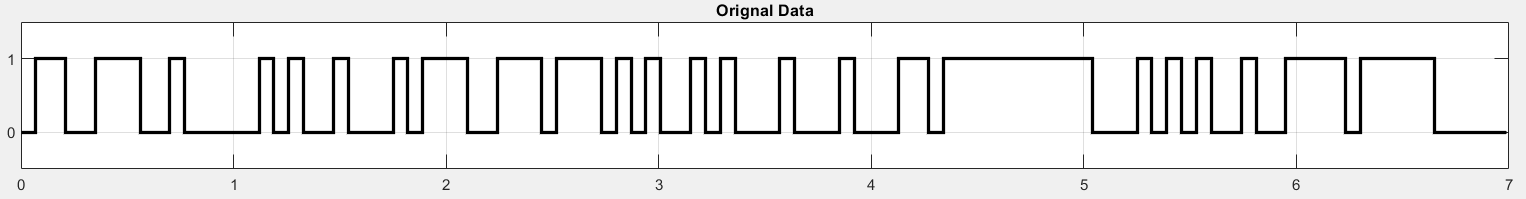
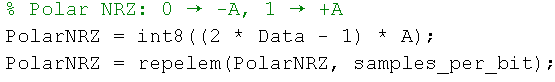


Figure 2 ADC Binary Output

For the line codes we will use this function:

# polar NRZ ensemble creation



* The data consists of 0s and 1s. We converted these values to A and -A respectively.
* Then, we utilized the “**repelem”** function to repeat each element seven times (samples\_num).

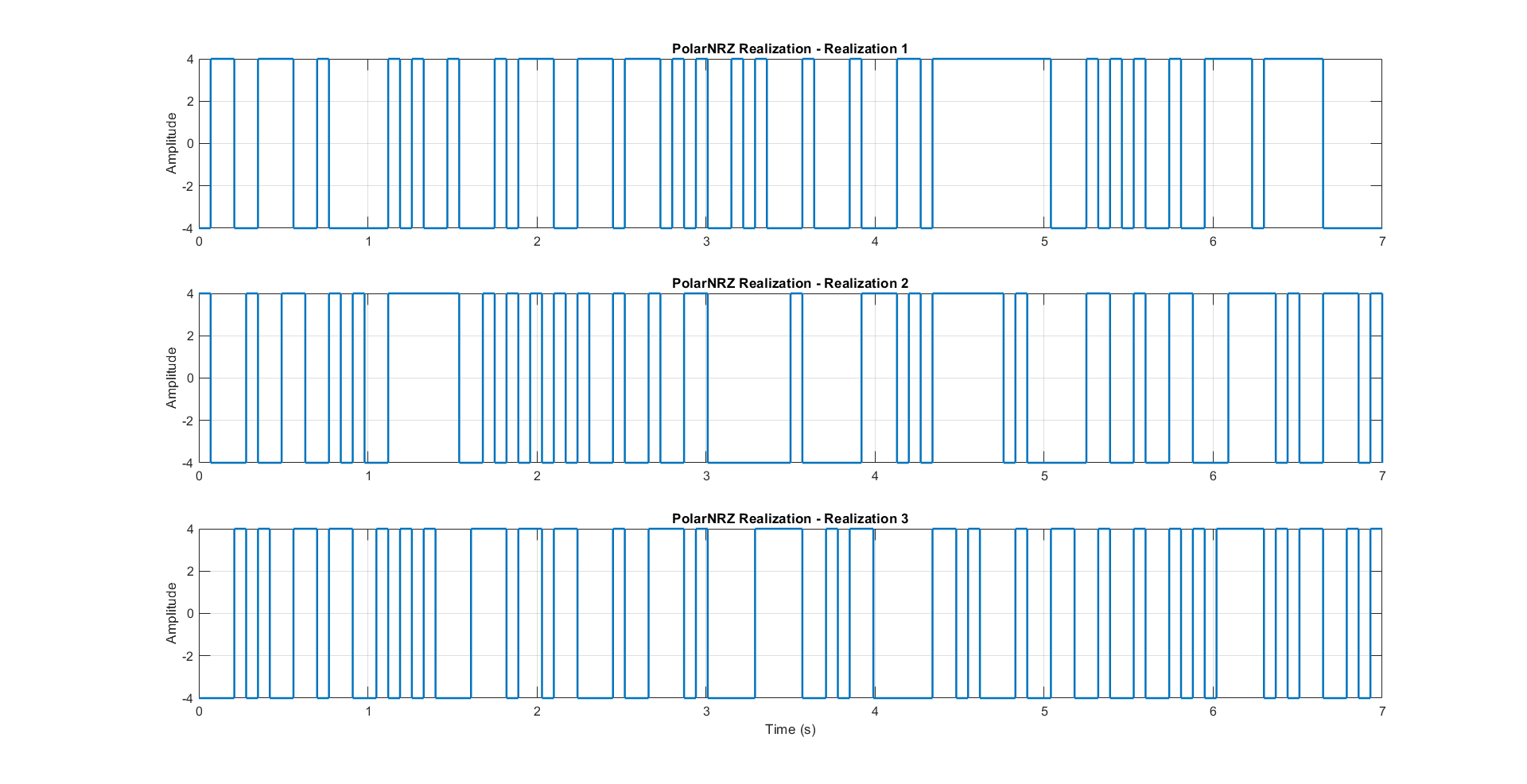


Figure 3 PolarNRZ Realizations

# Uni polar NRZ ensemble creation

* We then generate unipolar NRZ amplitudes along with its realization.
* We convert data (1,0) to 1 →A ,0→ 0 to have uni\_polar\_NRZ.

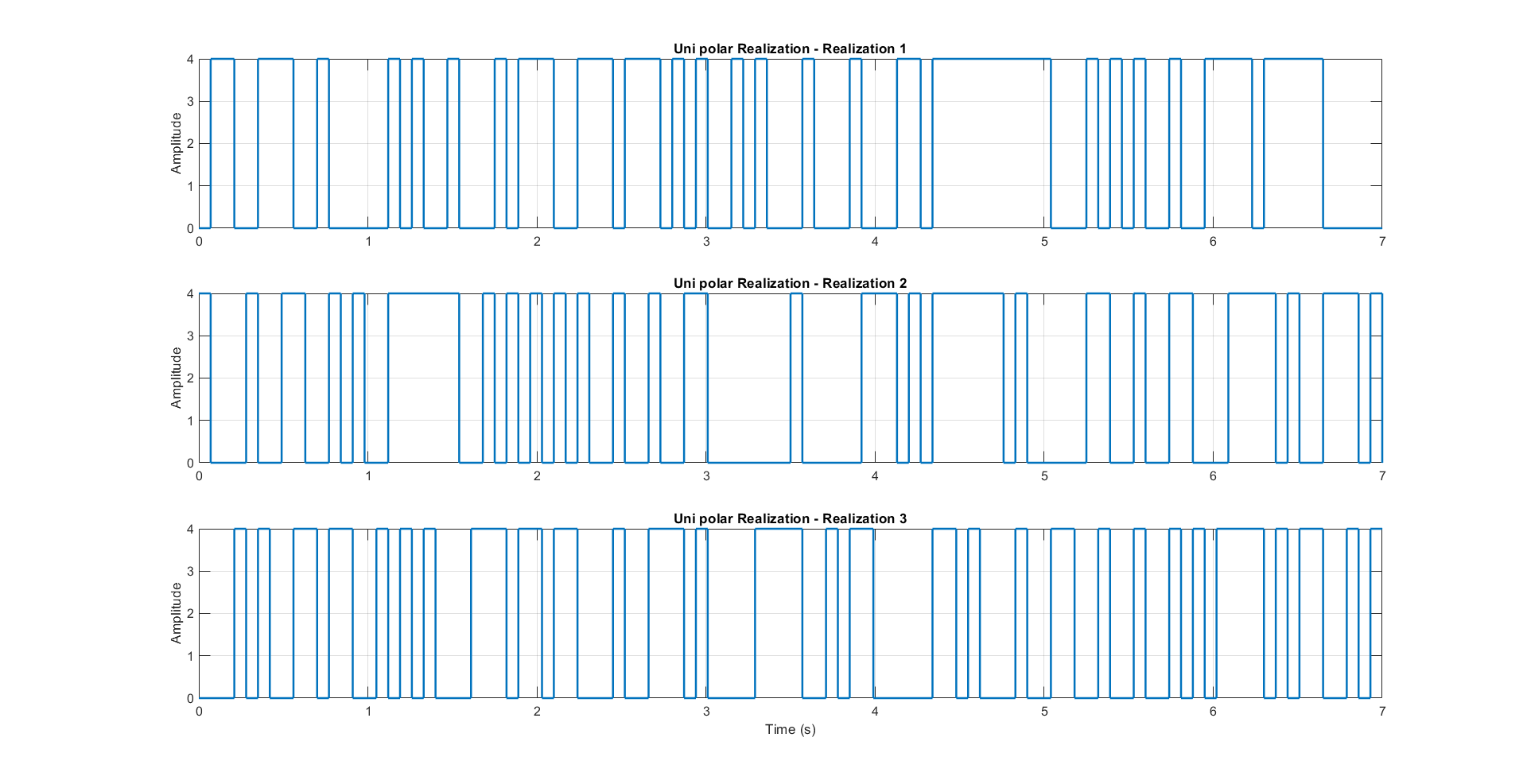
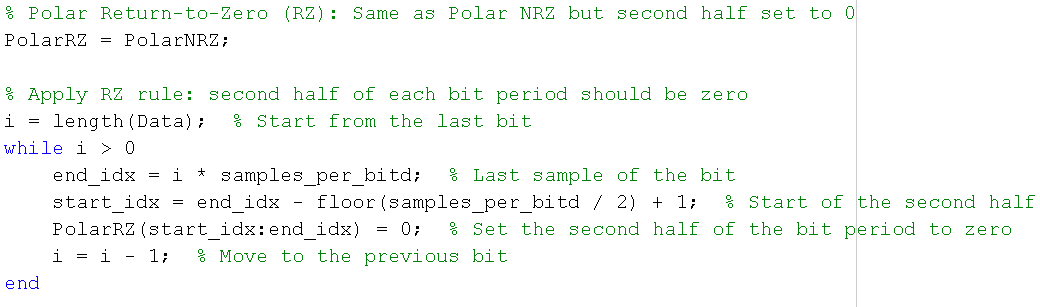


Figure 4 Uni Polar Realizations

# polarRZ ensemble creation



* The data consists of 0s and 1s. We first convert these values to amplitudes:  
  **0 → -A, 1 → +A** (this is the standard **Polar NRZ** encoding).
* Then, we utilized the repelem function to repeat each amplitude value samples\_per\_bit times. This creates a constant level for each bit across its time duration.
* To convert **Polar NRZ** to **Polar Return-to-Zero (RZ)**, we start with the Polar NRZ waveform.
* We apply the RZ rule by modifying the **second half of each bit period**:  
  For every bit, we calculate the index range that corresponds to the second half of its duration and set those values to zero.
* This creates a waveform where the signal returns to zero in the second half of each bit period, while the first half retains the Polar NRZ value (+A or -A).
* The result is a **Polar RZ** line code that has a non-zero level only during the first half of each bit, making it more suitable for synchronization at the receiver.

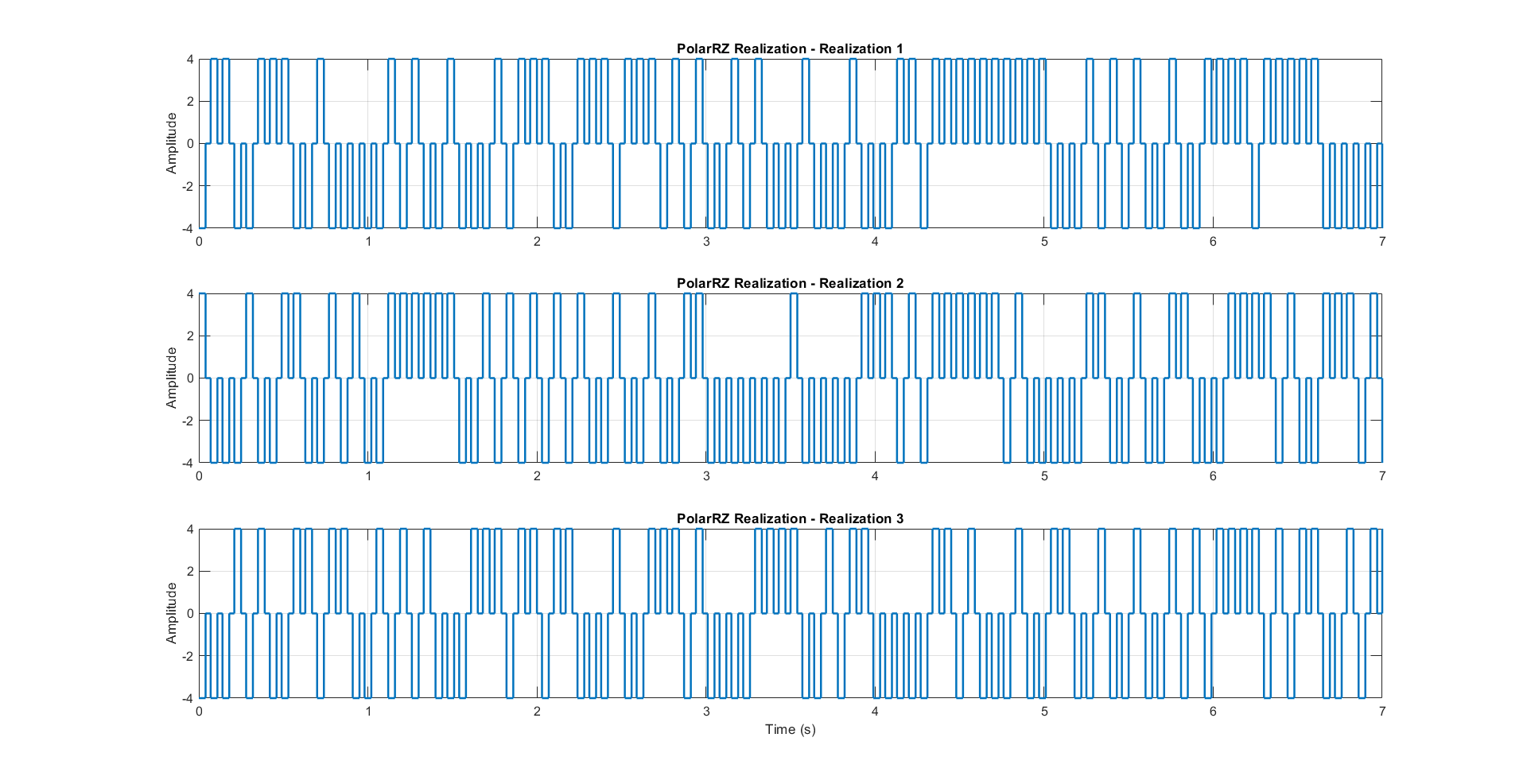
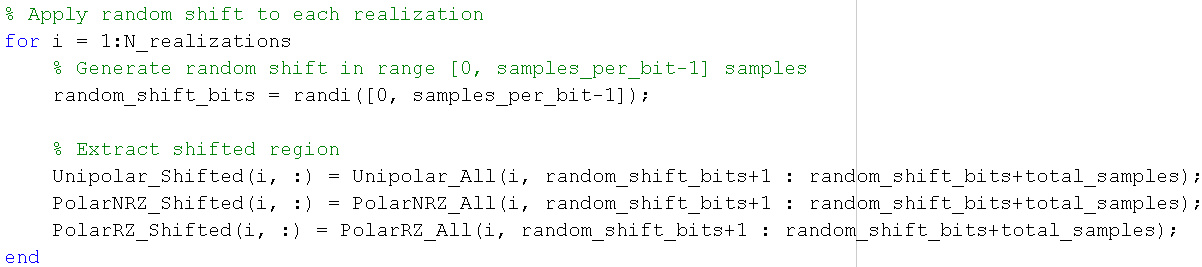


Figure 5 PolarRZ Realization

# Random initial time shift

For the random shift we made this function:

* Generating a single random initial time delay that can range from ‘0’ to ‘6’ samples for each waveform using the function “randi”.
* Then, we utilized the randi function to generate a random number ranging from 0 to 6, which represents the delay or start time, then we take the elements from this random index (start\_indices) to 700+( start\_indices).

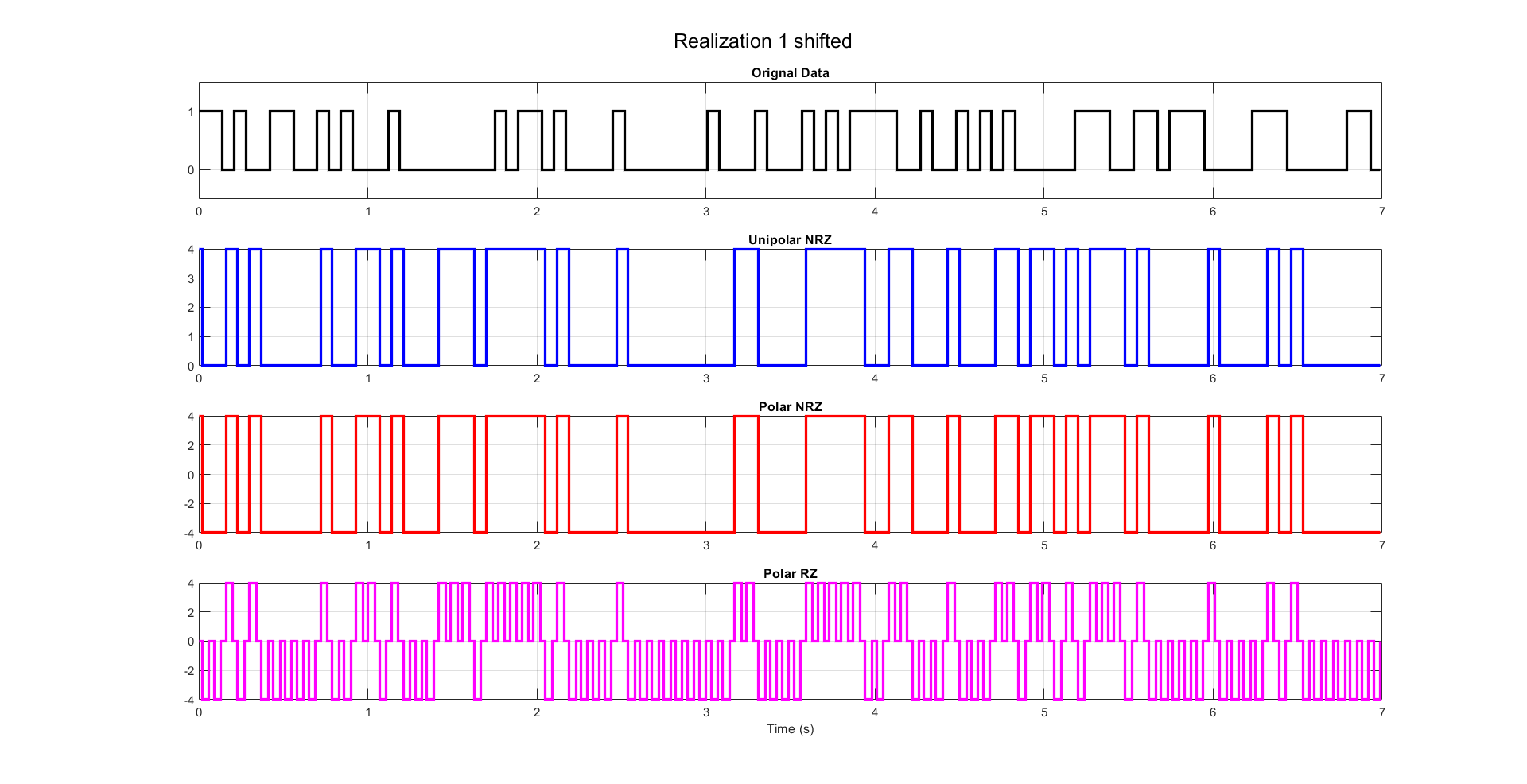
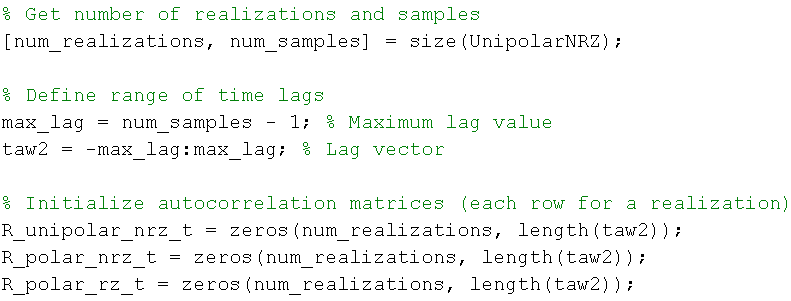


Figure 6 Realization Shifted

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

For the mean the cells are ready, as for the autocorrelation we’re going to use this function:

In which we’re making the array ready by shifting it with tau.



So the array will have the length of max tau which is 700.

# Questions

## Statistical Mean

### Hand Analysis

For the “Statistical Mean” which represents the average of all the realizations at the same time instant, let us consider the first line code method “Unipolar NRZ”

𝜇𝑋(𝑡) = 0 ∗ 0.5 + 4 ∗ 0.5 = 2 (𝐶𝑜𝑛𝑠𝑡𝑎𝑛𝑡 𝑎𝑐𝑟𝑜𝑠𝑠 𝑡𝑖𝑚𝑒)

And in the same matter, we can calculate the “Statistical Mean” for both “Polar NRZ” and “Polar RZ” as following:

𝜇𝑋\_𝑃𝑁𝑅𝑍(𝑡) = 4 ∗ 0.5 + (−4) ∗ 0.5 = 0 (𝐶𝑜𝑛𝑠𝑡𝑎𝑛𝑡 𝑎𝑐𝑟𝑜𝑠𝑠 𝑡𝑖𝑚𝑒).

𝜇𝑋\_𝑃𝑅𝑍(𝑡) = 4 ∗ 0.5 + (−4) ∗ 0.5 = 0 (𝐶𝑜𝑛𝑠𝑡𝑎𝑛𝑡 𝑎𝑐𝑟𝑜𝑠𝑠 𝑡𝑖𝑚𝑒).

### Code Snippet

* The mean is calculated as μ=ΣΧ/N (the sum divided by the number of the elements).

### Plotting the Statistical Mean:

Figure 7 Plot of Statistical Mean

* As expected, polar RZ & NRZ have almost zero mean and the uni polar has mean around 2 Bec its amplitude ranges from 0:4.

## Statistical Autocorrelation

### Hand Analysis

𝑹𝑿(𝝉) = 𝑬[𝑿(𝝉) 𝑿(𝒕 + 𝝉)] = ∑ 𝑿(𝝉) 𝑿(𝒕 + 𝝉) 𝑷(𝑿(𝝉) 𝑿(𝒕 + 𝝉))

* + **For Unipolar NRZ:**

We have 2 cases **(Considering T to be 70ms or 7 samples),**

1. |𝝉| < 𝑻

𝑅𝑋(𝜏) = 𝐸[𝑋(𝜏) 𝑋(𝑡 + 𝜏)]

= 42 ∗ 𝑃(4,4) + 02 ∗ 𝑃(0,0) + 4 ∗ 0 ∗ 𝑃(0,4) + 0 ∗ 4 ∗ 𝑃(4,0)

= 42 ∗ 𝑃(4,4)

𝑃(4,4) = 𝑃(𝑋(𝑡 + 𝜏) = 4 | 𝑋(𝑡) = 4) ∗ 𝑃(𝑋(𝑡) = 4)

𝑃(𝑋(𝑡 + 𝜏) = 4 | 𝑋(𝑡) = 4) = 𝑃(𝑇̅) + 𝑃(𝑇) ∗ 𝑃(𝑋(𝑡 + 𝜏) = 4)

1. |𝝉| > 𝑻

𝑅𝑋(𝜏) = 𝐸[𝑋(𝜏) 𝑋(𝑡 + 𝜏)]

= 42 ∗ 0.5 ∗ 0.5 + 02 ∗ 0.5 ∗ 0.5 + 4 ∗ 0 ∗ 0.5 ∗ 0.5 + 0 ∗ 4 ∗ 0.5 ∗ 0.5

= 42 ∗ 0.5 ∗ 0.5

= 4

* + And using the same flow, we can find that the ACF for “**Polar NRZ**” is
  + And similarly, the ACF for “**Polar RZ**” is

**And as we know:**

𝑇𝑜𝑡𝑎𝑙 𝑃𝑜𝑤𝑒𝑟 = 𝑅𝑋(0) & 𝐷𝐶 𝑃𝑜𝑤𝑒𝑟 = 𝑅𝑋(∞).

𝐴𝐶 𝑃𝑜𝑤𝑒𝑟 = 𝑇𝑜𝑡𝑎𝑙 𝑃𝑜𝑤𝑒𝑟 − 𝐷𝐶 𝑃𝑜𝑤𝑒𝑟.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Unipolar NRZ** | **Polar NRZ** | **Polar RZ** |
| **Total Power** | **8** | **16** | **9.147** |
| **DC Power** | **4** | **0** | **0** |
| **AC Power** | **4** | **16** | **9.147** |

### Code Snippet

**Annotations**

* The Statistical Autocorrelation is created by taking the element-wise product of each column with the first column of a selected matrix of data points, then averaging the resulting column-wise products.
* To guarantee that Autocorr is an even fun we concatenate between the result of fliplr fun & the averages vector before flipping (2:700 to ensure no repeated value at zero).

### Plotting the statistical autocorrelation

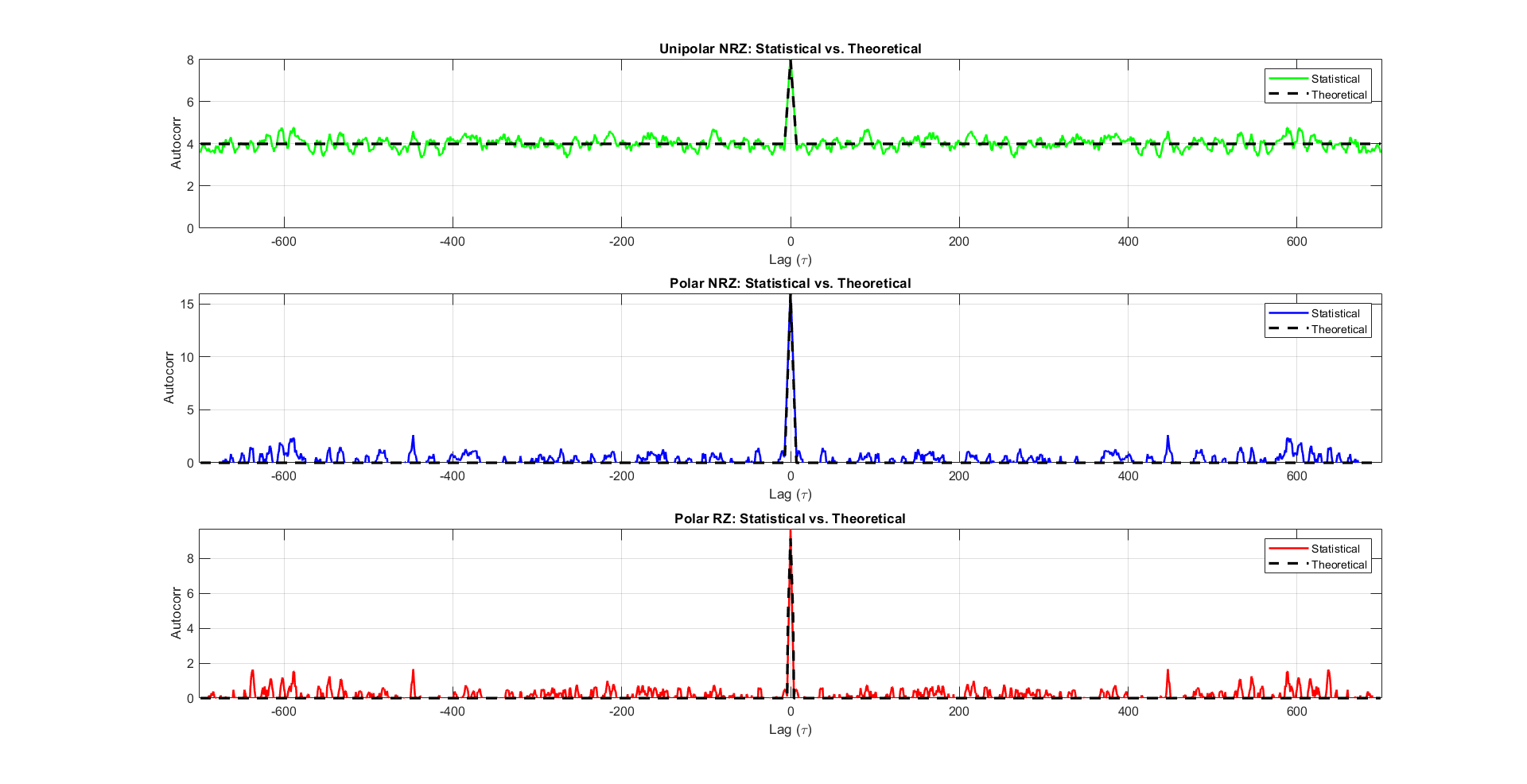
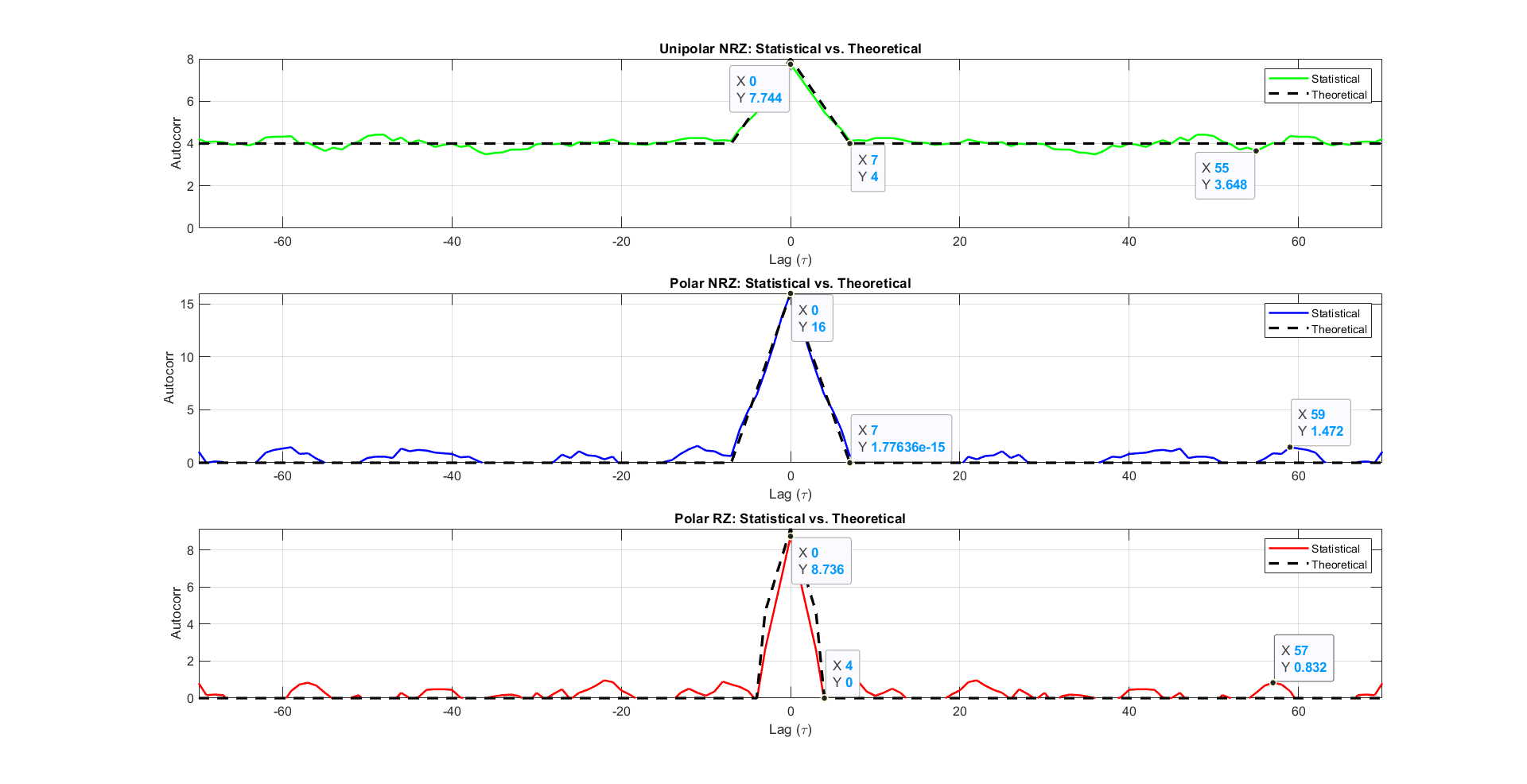


Figure 8 Statistical Auto Correction plot

Figure 9 Statistical Auto Correction plot zoomed

The resulting autocorrelation values are plotted against the corresponding time delays (τ). We observe that at τ = 0 the autocorrelation with the point itself is maximum, indicating perfect correlation.

* **Uni polar:** The autocorrelation becomes constant after 7 samples, as we calculated to be the bit duration and it’s around 4, The maximum at zero equals 7.744 ≈ 8.
* **Polar NRZ:** The autocorrelation becomes constant after 7 samples, as we calculated to be the bit duration and it’s around zero, The maximum at zero equals 16.
* **Polar RZ:** The autocorrelation becomes constant after 4 samples, as we calculated to be the half bit duration and it’s around zero, The maximum at zero equals 8.736 ≈ 9.147.

## Is the Process Stationary

Figure 10 autocorrelation at two different times

* For the **mean**, as shown in section 1 figure 7 the **mean** ≈ **constant with time**.
* For the **autocorrelation**, as shown in figure 9 the **autocorrelation R(t1=1)** ≈ **R(t2=8).**

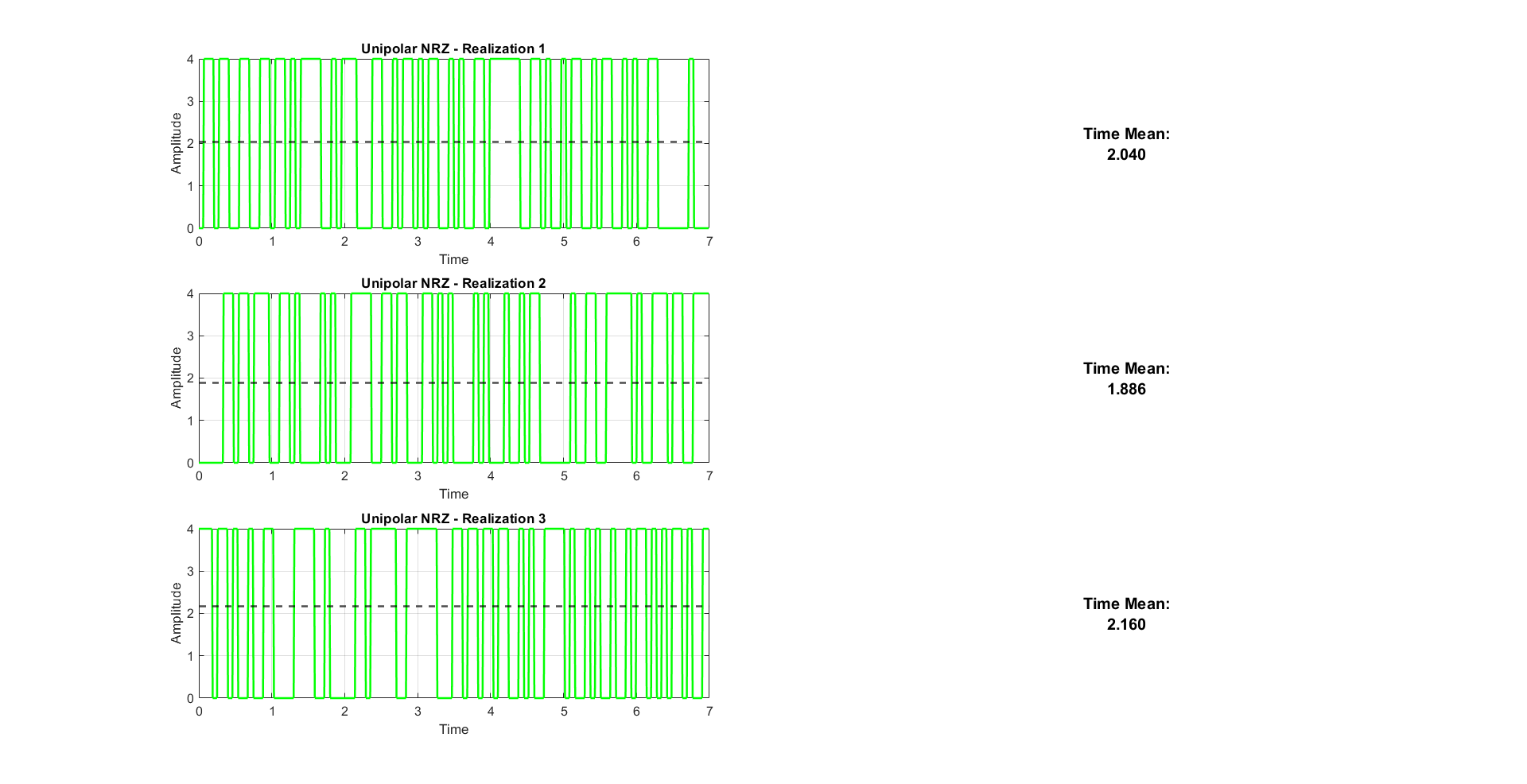
Yes, the process is stationary (WSSP) because the mean is constant function in time as shown in Figure 7 Plot of Statistical Mean and the autocorrelation depends only on the time difference not the absolute time.

## The time mean and autocorrelation function for one waveform

### Time Mean

* We add the values of a realization across time instant then divide by the number of samples (700 sample per realization).

Figure 11 Time Mean for Uni Polar



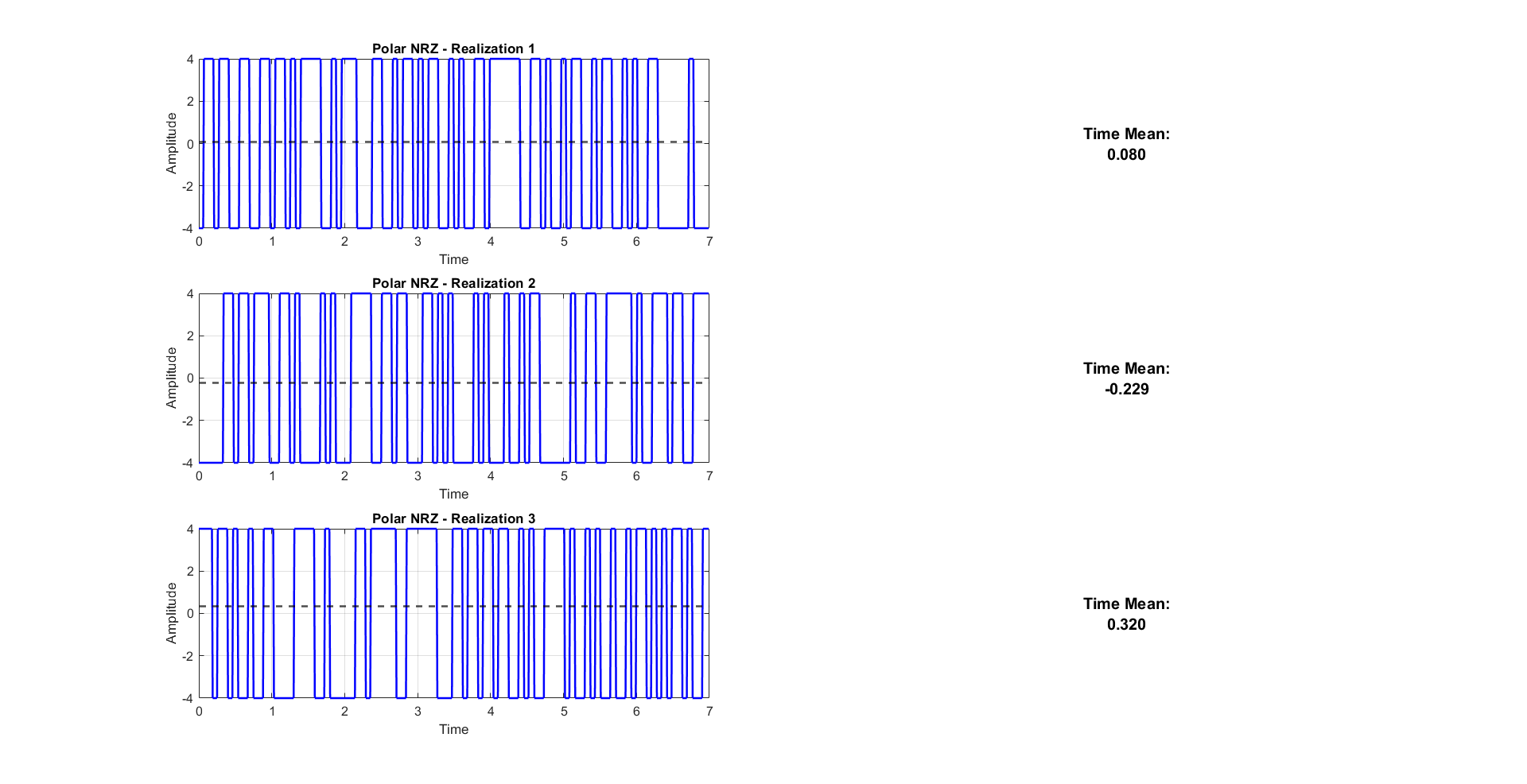
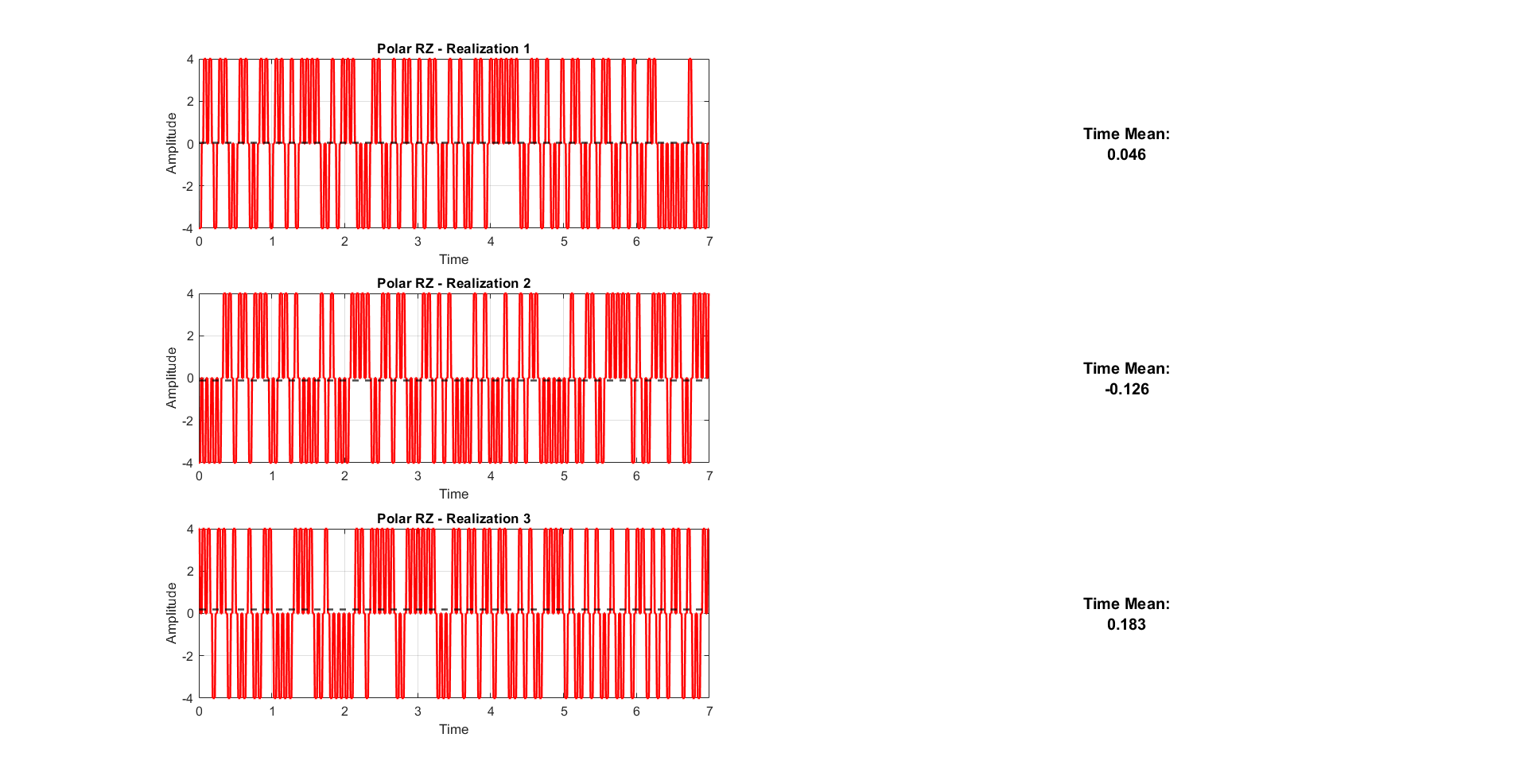
* As expected, polar RZ & NRZ have almost zero mean and the uni polar has mean around 2 Because its amplitude ranges from 0 : 4

Figure 12 Time Mean for Polar NRZ

Figure 13 Time Mean for Polar RZ

### Time Mean Vs Realization

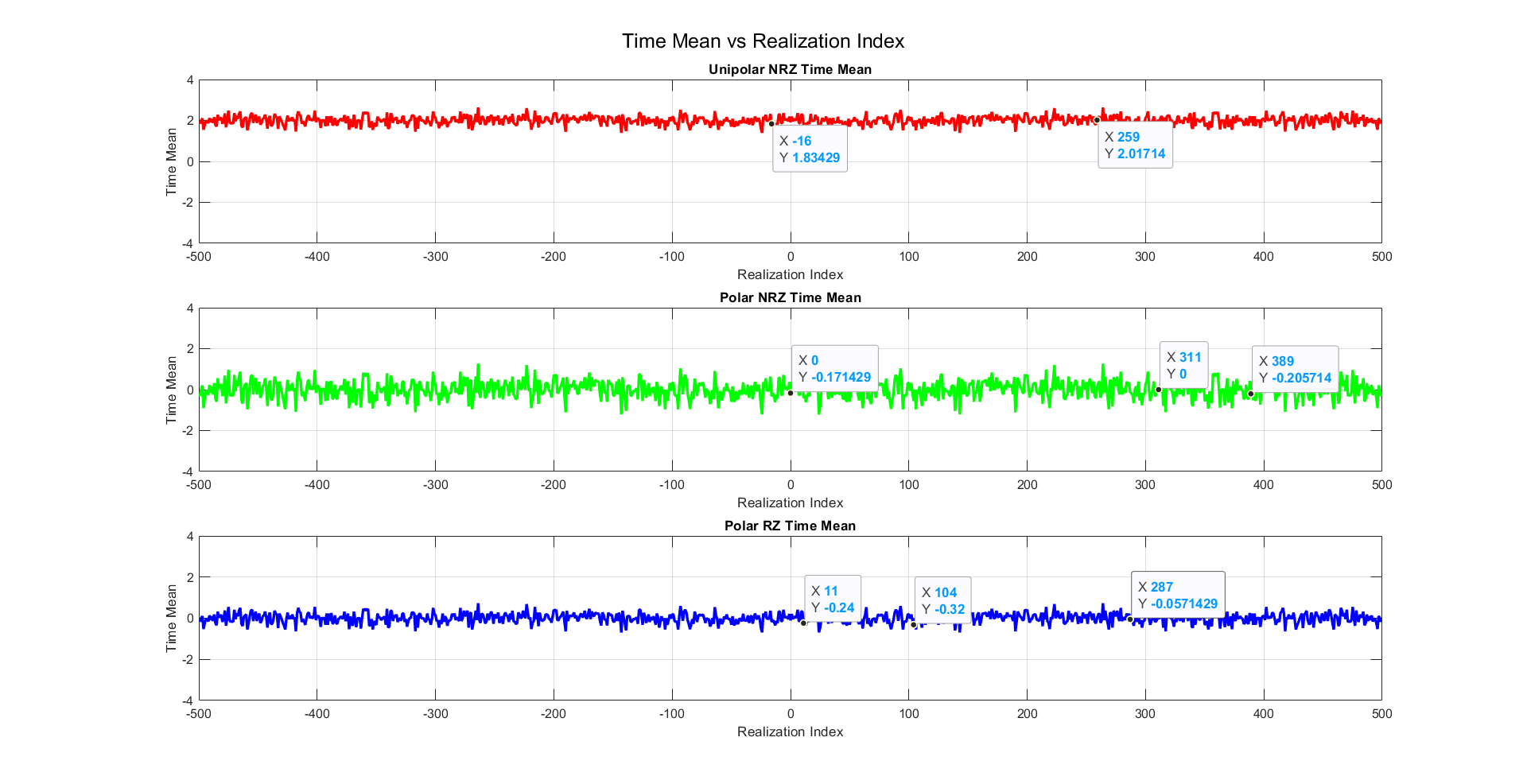
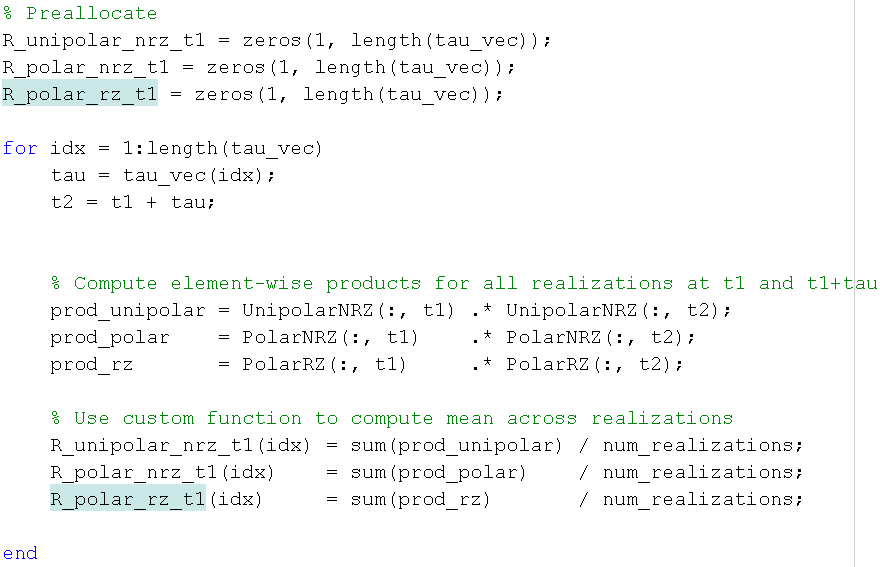


Figure 14 Time Mean Vs Realization

* As expected the time mean is almost equal to the statistical mean.
* Polar RZ & NRZ have almost zero mean and the uni polar has mean around 2.

### Time Auto Correlation

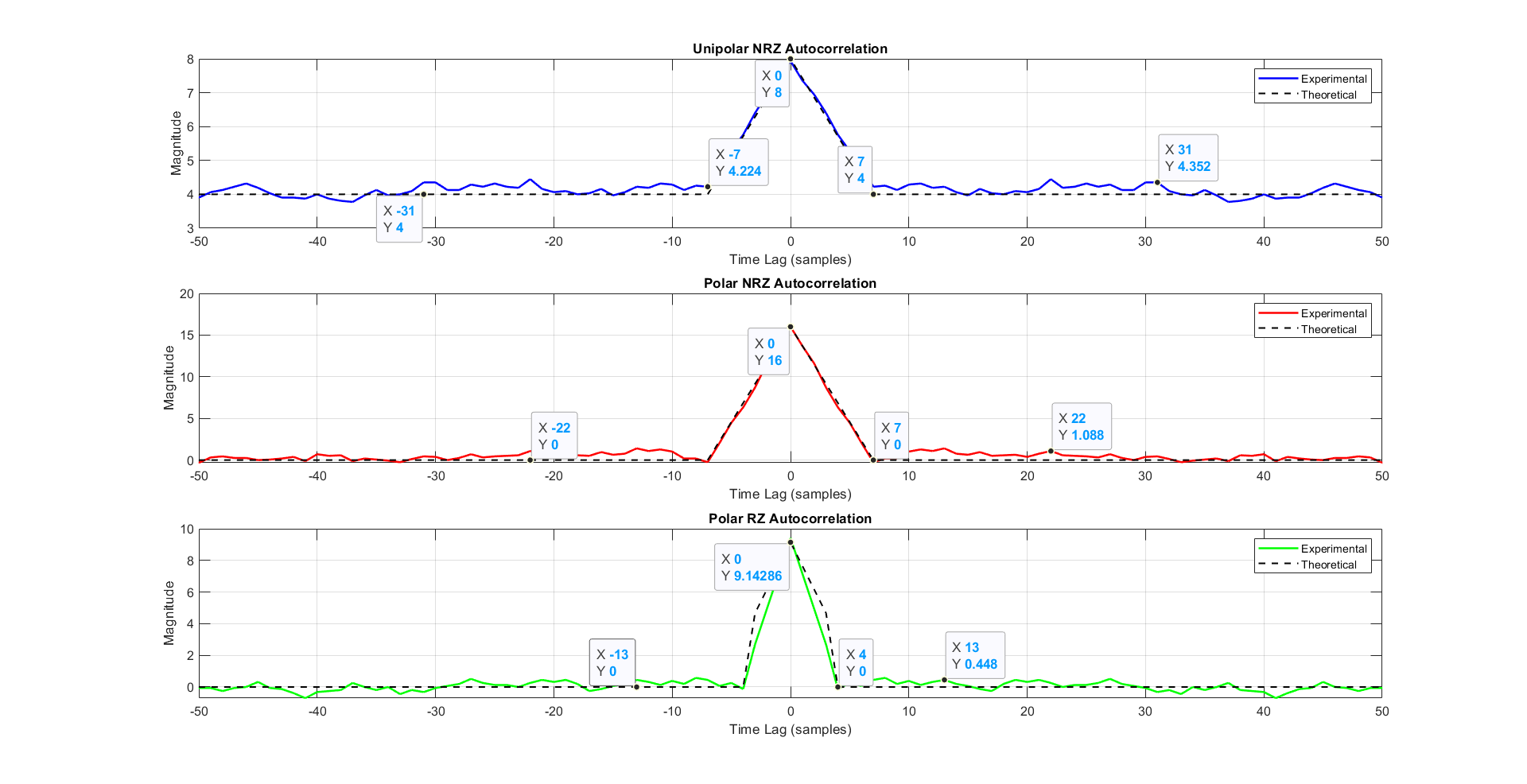
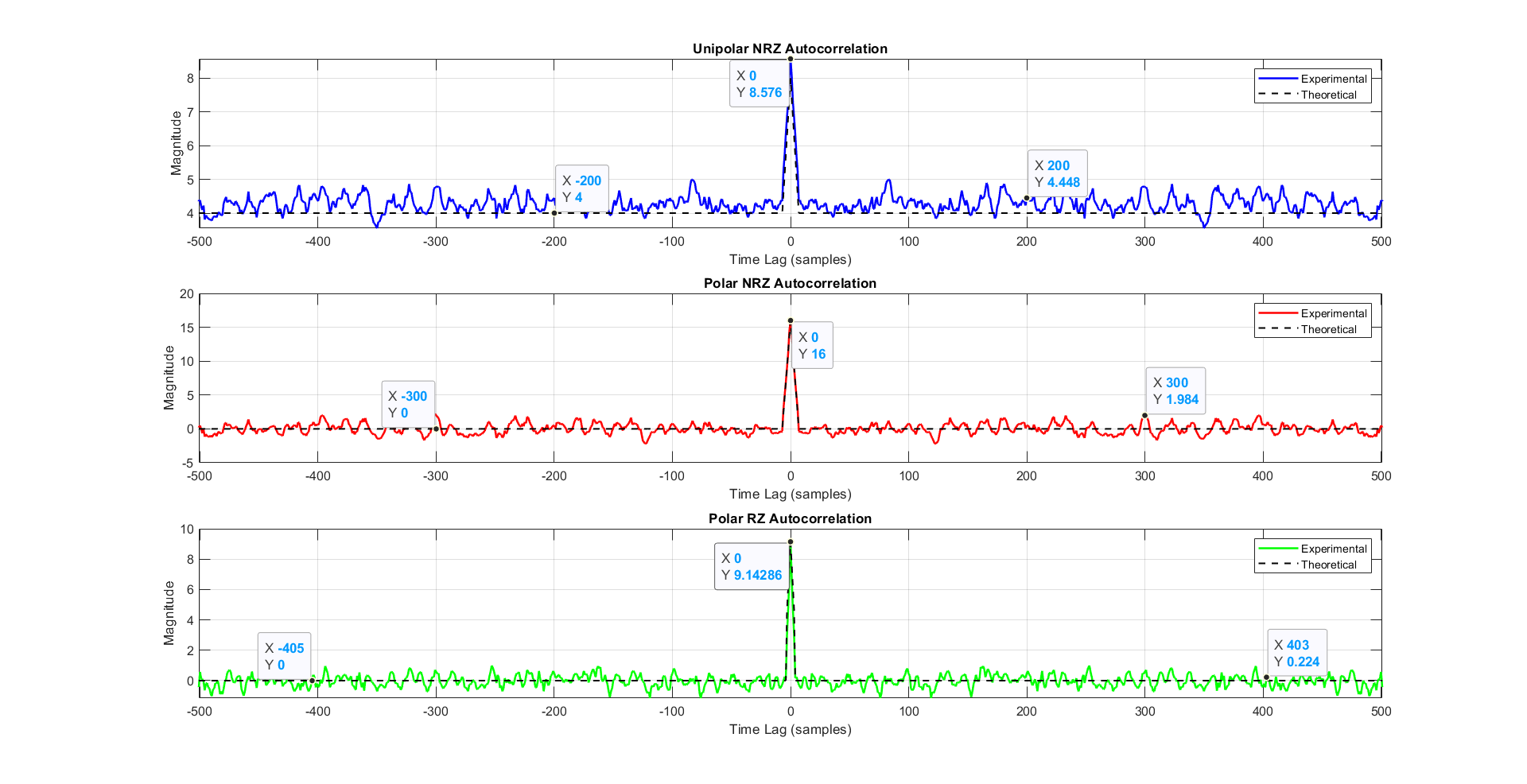
For the time Auto Correlation we’re going to use this function



### Time Auto Correlation for one wave form:

Figure 15 Time Auto Correction plot zoomed

Figure 16 Time Auto Correction plot



As shown in the graphs:

* The time autocorrelation is closely same as the ensemble  
  autocorrelation.
* The autocorrelation function has maximum at 𝛕 = 𝟎 and it is an even function.

## Is The Random Process Ergodic?

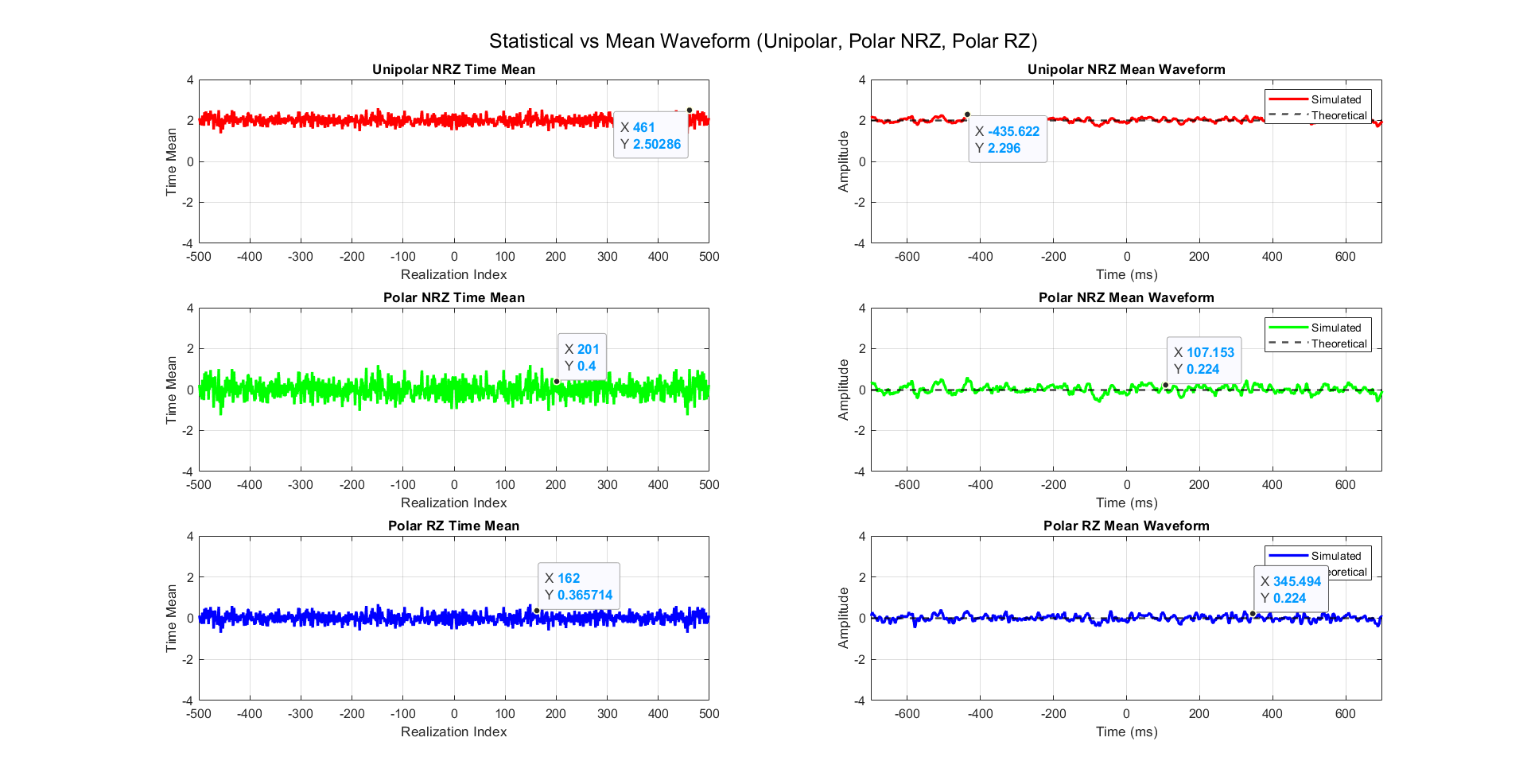


Figure 17 Time Mean vs Statistical

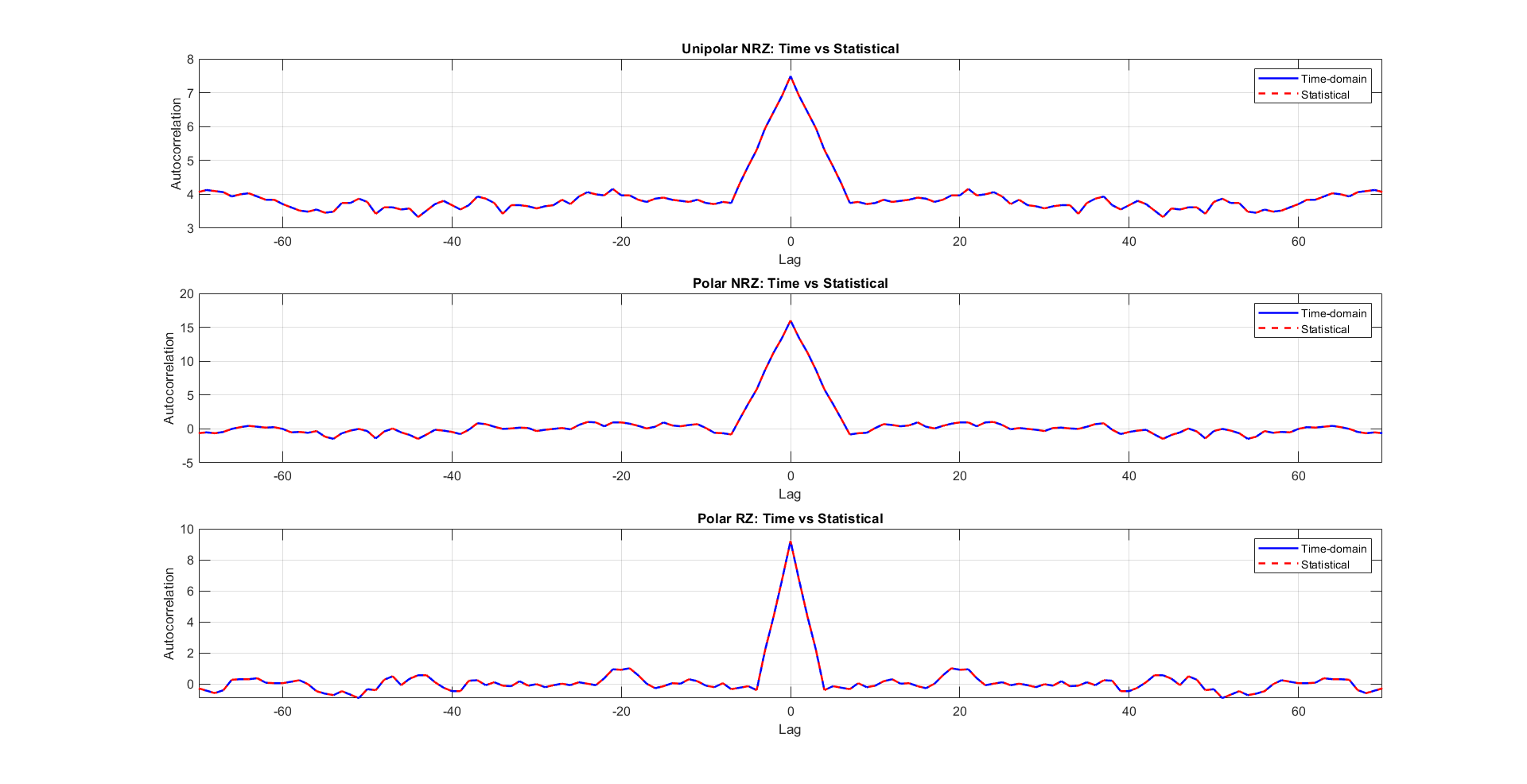


Figure 18 Time Auto Correlation Vs Statistical

* For the **mean**, the Time mean is almost equal to the statistical mean.
* For the **Auto Correlation,** the Time looks almost identical to the statistical.

But, There not fully identical as we ran this code snippet



And the result was **0.5760, so they are almost Identical.**

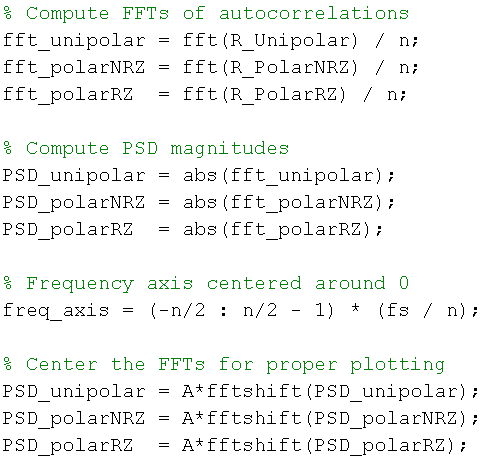
* Yes, because the time mean ≈ the Statistical mean and the time autocorrelation is ≈ the ensemble autocorrelation.

**Then this process is ergodic**

## the PSD & Bandwidth of the Ensemble

### PSD using fft:

For the **PSD,** we are going to use this function:



* We take the Fourier transform of the avg time autocorrelation **= 0.5\*(R(t1)+ R(t2))** then centralize the graph around zero.
* since
* **For the BW**
* the BW is the frequency of the first zero of sinc^2 function (intersection with frequency-axis)

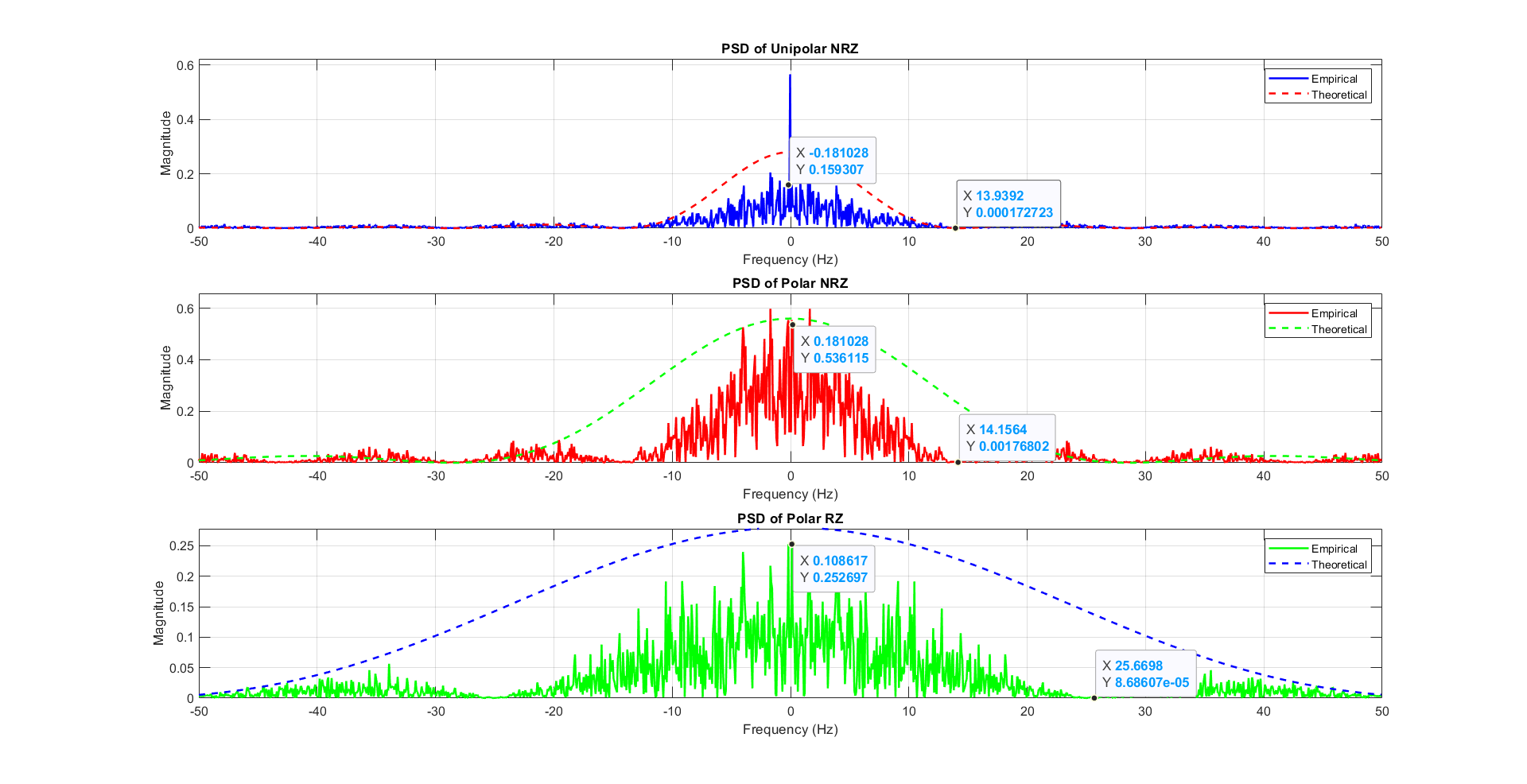


Figure 19 PSD plot of the Ensemble

**Annotations**

* in polar RZ & NRZ : we have sinc^2 function without delta at zero frequency (NO DC)
* in uni polar NRZ : we have sinc^2 function with delta at zero frequency (there is DC)
* BW of the unipolar NRZ & polar NRZ is the bitrate which approximately equal 14 hz
* BW of the polar RZ is the double of bitrate which approximately equal 25.66 hz

### Theoritical PSD:

From **references[1], [2]**, we found out that the PSDs are:

|  |  |
| --- | --- |
| Line Code | PSD |
| Uni Polar | S(f)=A2/4\*Tb\*(sin(πfTb)/πfTb)2 + A2/4\* δ(f) |
| Polar NRZ | S(f)=A2\*Tb\*(sin(πfTb/2)/πfTb/2)2 |
| Polar RZ | S(f)=A2\*Tb\*(sin(πfTb/4)/πfTb/4)2 |

Note that:

* Uni polar has a DC pulse which is noticeable in figure 19
* Polar don’t have the DC pulse
* Polar RZ has double the frequency of Polar NRZ
* A=4, Tb = 70 ms

So by comparing the practical vs theoretical:

|  |  |  |
| --- | --- | --- |
| Line Code | Theoritcal PSD at f=0 | Paractical PSD at f=0 |
| Uni Polar | A2/4\*Tb =0.28 | 0.159 |
| Polar NRZ | A2/2\*Tb =0.56 | 0.536 |
| Polar RZ | A2/4\*Tb =0.28 | 0.252 |

For **BW:**

|  |  |  |
| --- | --- | --- |
| Line Code | Theoritcal BW | Paractical BW |
| Uni Polar | 1/Tb = 14.285 Hz | 13.972 Hz |
| Polar NRZ | 1/Tb = 14.285 Hz | 14.15 Hz |
| Polar RZ | 2/Tb = 28.57 Hz | 25.66 Hz |

# References:

**[1]** B. P. Lathi and Z. Ding, Modern Digital and Analog Communication Systems, 3rd ed. New York, NY, USA: Oxford University Press, 2009, ch. 7.

**[2]** S. R. Iyer, “Line Coding PSD – ECE 4001,” Electronic Duo, Sep. 2017. [Online]. Available: <https://electronicduo.blogspot.com/2017/09/line-coding-psd-ece-4001.html>

**[3]** <https://github.com/youefkh05/Digital_Communication_Radio>

# Appendix

clc;

clear;

close all;

% Parameters

A = 4; % Amplitude

N\_realizations = 500; % Number of waveforms (ensemble size)

num\_bits = 100+1; % Bits per waveform and one extra bit for shifting

bit\_duration = 70e-3; % 70 ms per bit

dac\_interval = 10e-3; % DAC updates every 10 ms

samples\_per\_bit = round(bit\_duration / dac\_interval); % 7 samples per bit

total\_time = num\_bits \* bit\_duration; % Total waveform duration

t = 0:dac\_interval:(total\_time - dac\_interval); % Time vector

% Preallocate matrices for efficiency

Unipolar\_All = zeros(N\_realizations, length(t), 'int8');

PolarNRZ\_All = zeros(N\_realizations, length(t), 'int8');

PolarRZ\_All = zeros(N\_realizations, length(t), 'int8');

% Generate and store 500 realizations

for i = 1:N\_realizations

Data = randi([0, 1], 1, num\_bits, 'int8'); % Random bit sequence

%encode the data

[Unipolar, PolarNRZ, PolarRZ] = generate\_linecodes(Data, A, samples\_per\_bit);

% Store in matrices

Unipolar\_All(i,:) = Unipolar;

PolarNRZ\_All(i,:) = PolarNRZ;

PolarRZ\_All(i,:) = PolarRZ;

% Plot the first realization

if i==1

plot\_linecodes(Data, Unipolar, PolarNRZ, PolarRZ, t, num\_bits-1,'Realization 1');

end

end

% plot first three realization

plot\_realizations(Unipolar\_All, t, 3, 'Uni polar Realization');

plot\_realizations(PolarNRZ\_All, t, 3, 'PolarNRZ Realization');

plot\_realizations(PolarRZ\_All, t, 3, 'PolarRZ Realization');

% Apply random shift to all realizations

[Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted] =...

apply\_random\_shift\_fixed\_size(Unipolar\_All, PolarNRZ\_All, PolarRZ\_All, samples\_per\_bit);

t\_shifted = t(1:length(Unipolar\_Shifted)); % Ensure the time vector matches

%plot after shift

plot\_linecodes(Data, Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted,...

t\_shifted, num\_bits-1, 'Realization 1 shifted');

% Convert to double for accuracy

Unipolar\_Shifted = double(Unipolar\_Shifted);

PolarNRZ\_Shifted = double(PolarNRZ\_Shifted);

PolarRZ\_Shifted = double(PolarRZ\_Shifted);

%calculate the mean acrros time (question 1)

Unipolar\_Mean = calculate\_mean(Unipolar\_Shifted);

PolarNRZ\_Mean = calculate\_mean(PolarNRZ\_Shifted);

PolarRZ\_Mean = calculate\_mean(PolarRZ\_Shifted);

%plot the mean across time

plot\_mean\_waveforms(t\_shifted, Unipolar\_Mean, PolarNRZ\_Mean, PolarRZ\_Mean, 2, 0, 0, A);

% Compute variance for each code line

Unipolar\_Var = calculate\_variance(Unipolar\_Shifted);

PolarNRZ\_Var = calculate\_variance(PolarNRZ\_Shifted);

PolarRZ\_Var = calculate\_variance(PolarRZ\_Shifted);

%plot the vatiance across time

plot\_variance(t\_shifted, Unipolar\_Var, PolarNRZ\_Var, PolarRZ\_Var);

% Determine max\_lag dynamically

max\_lag = size(Unipolar\_Shifted, 2) - 1;

% calculate the autocorrelation (question 3)

[Unipolar\_AutoCorr, PolarNRZ\_AutoCorr, PolarRZ\_AutoCorr] = ...

compute\_stat\_autocorr(Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted, max\_lag);

% plot the autocorrelation

plot\_autocorrelation(Unipolar\_AutoCorr, PolarNRZ\_AutoCorr, PolarRZ\_AutoCorr, max\_lag, 100\*bit\_duration, A);

% Compute time autocorrelation

t1=1;

t2=8;

[R\_unipolar\_t, R\_polar\_nrz\_t, R\_polar\_rz\_t, taw] =...

compute\_time\_autocorr(Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted, t1);

% Plot the time autocorrelation

plot\_time\_autocorrelation(R\_unipolar\_t, R\_polar\_nrz\_t, R\_polar\_rz\_t, taw, max\_lag, 100\*bit\_duration, A);

% Check for Staionary (question 2)

[R\_unipolar\_t2, R\_polar\_nrz\_t2, R\_polar\_rz\_t2, taw2] =...

compute\_time\_autocorr(Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted, t2);

plot\_two\_realizations(R\_unipolar\_t, R\_unipolar\_t2, taw, max\_lag);

% Compute time mean for each line code

Unipolar\_TimeMean = compute\_time\_mean(Unipolar\_Shifted);

PolarNRZ\_TimeMean = compute\_time\_mean(PolarNRZ\_Shifted);

PolarRZ\_TimeMean = compute\_time\_mean(PolarRZ\_Shifted);

% Plot the time mean

plot\_realizations\_with\_mean(t\_shifted, Unipolar\_TimeMean, Unipolar\_Shifted, 'Unipolar NRZ', 'g');

plot\_realizations\_with\_mean(t\_shifted, PolarNRZ\_TimeMean, PolarNRZ\_Shifted, 'Polar NRZ', 'b');

plot\_realizations\_with\_mean(t\_shifted, PolarRZ\_TimeMean, PolarRZ\_Shifted, 'Polar RZ', 'r');

%Plot time mean vs realization

plot\_time\_mean\_vs\_realization(Unipolar\_TimeMean, PolarNRZ\_TimeMean, PolarRZ\_TimeMean, A);

% Check for ergodic (question 5)

plot\_mean\_time\_vs\_statistical(t\_shifted, ...

Unipolar\_Mean, PolarNRZ\_Mean, PolarRZ\_Mean, ...

2, 0, 0, ...

Unipolar\_TimeMean, PolarNRZ\_TimeMean, PolarRZ\_TimeMean, A);

plot\_combined\_autocorrelation(R\_unipolar\_t, R\_polar\_nrz\_t, R\_polar\_rz\_t, ...

taw, max\_lag, 100\*bit\_duration, A, ...

Unipolar\_AutoCorr, PolarNRZ\_AutoCorr, PolarRZ\_AutoCorr);

% Plotting the PSD (Question 6)

[R\_avg\_unipolar, R\_avg\_polar\_nrz, R\_avg\_polar\_rz, tau\_full] = ...

get\_Ravg(Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted, t1, t2);

Ts=bit\_duration/samples\_per\_bit;% Sampling Time

fs = 1/Ts; % Sampling Frequency

[PSD\_unipolar ,PSD\_polarNRZ ,PSD\_polarRZ] =...

plot\_linecode\_psd(R\_avg\_unipolar, R\_avg\_polar\_nrz, R\_avg\_polar\_rz, fs, A, bit\_duration);

%-----------------------Functions----------------------------

function [Unipolar, PolarNRZ, PolarRZ] = generate\_linecodes(Data, A, samples\_per\_bit)

% Ensure input Data is of type int8

Data = int8(Data);

% Convert samples\_per\_bit to double for safe calculations

samples\_per\_bitd = double(samples\_per\_bit);

% Unipolar NRZ: 0 ? 0V, 1 ? A

Unipolar = int8(Data \* A);

Unipolar = repelem(Unipolar, samples\_per\_bit); % Repeat each bit for duration

% Polar NRZ: 0 ? -A, 1 ? +A

PolarNRZ = int8((2 \* Data - 1) \* A);

PolarNRZ = repelem(PolarNRZ, samples\_per\_bit);

% Polar Return-to-Zero (RZ): Same as Polar NRZ but second half set to 0

PolarRZ = PolarNRZ;

% Apply RZ rule: second half of each bit period should be zero

i = length(Data); % Start from the last bit

while i > 0

end\_idx = i \* samples\_per\_bitd; % Last sample of the bit

start\_idx = end\_idx - floor(samples\_per\_bitd / 2) + 1; % Start of the second half

PolarRZ(start\_idx:end\_idx) = 0; % Set the second half of the bit period to zero

i = i - 1; % Move to the previous bit

end

end

function plot\_linecodes(Data, Unipolar, PolarNRZ, PolarRZ, t, num\_bits\_to\_show, plot\_title)

% Ensure num\_bits\_to\_show does not exceed the actual number of bits

num\_samples\_per\_bit = ceil(length(t) / length(Data));

num\_samples\_to\_show = num\_bits\_to\_show \* num\_samples\_per\_bit;

% Trim the signals to display only the required number of bits

t\_show = t(1:num\_samples\_to\_show);

Unipolar\_show = Unipolar(1, 1:num\_samples\_to\_show); % Select row 1 explicitly

PolarNRZ\_show = PolarNRZ(1, 1:num\_samples\_to\_show);

PolarRZ\_show = PolarRZ(1, 1:num\_samples\_to\_show);

% Convert Data into a sample-wise representation for accurate plotting

Data\_show = repelem(Data(1:num\_bits\_to\_show), num\_samples\_per\_bit);

Data\_t = t(1:length(Data\_show)); % Adjust time axis

% Plot the signals

figure;

sgtitle(plot\_title); % Set a title for the entire figure

subplot(4,1,1);

stairs(Data\_t, Data\_show, 'k', 'LineWidth', 2);

title('Orignal Data');

ylim([-0.5, 1.5]); % Keep the binary level range

yticks([0 1]);

yticklabels({'0', '1'});

grid on;

subplot(4,1,2);

stairs(t\_show, Unipolar\_show, 'b', 'LineWidth', 2);

title('Unipolar NRZ');

grid on;

subplot(4,1,3);

stairs(t\_show, PolarNRZ\_show, 'r', 'LineWidth', 2);

title('Polar NRZ');

grid on;

subplot(4,1,4);

stairs(t\_show, PolarRZ\_show, 'm', 'LineWidth', 2);

title('Polar RZ');

grid on;

xlabel('Time (s)');

end

function plot\_realizations(linecode\_all, t, N, title\_prefix)

% plot\_realizations - Plots the first N realizations of a given line code

%

% Inputs:

% linecode\_all : Matrix of size [realizations x time] for a line code

% t : Time vector corresponding to samples

% N : Number of realizations to plot (e.g., 5)

% title\_prefix : Title prefix string (e.g., 'Unipolar NRZ')

%

% Example usage:

% plot\_realizations(Unipolar\_All, t, 5, 'Unipolar NRZ');

figure;

for i = 1:N

subplot(N, 1, i);

stairs(t, linecode\_all(i, :), 'LineWidth', 1.5);

title([title\_prefix ' - Realization ' num2str(i)]);

ylabel('Amplitude');

xlim([0 7]); % Focus on the first 7 seconds

grid on;

if i == N

xlabel('Time (s)');

end

end

end

function [Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted] =...

apply\_random\_shift\_fixed\_size(Unipolar\_All, PolarNRZ\_All, PolarRZ\_All, samples\_per\_bit)

% Define parameters

N\_realizations = size(Unipolar\_All, 1); % 500 realizations

extended\_samples = size(Unipolar\_All, 2); % 707 samples

total\_samples = 700; % Fixed output size

% Initialize shifted matrices

Unipolar\_Shifted = zeros(N\_realizations, total\_samples, 'int8');

PolarNRZ\_Shifted = zeros(N\_realizations, total\_samples, 'int8');

PolarRZ\_Shifted = zeros(N\_realizations, total\_samples, 'int8');

% Apply random shift to each realization

for i = 1:N\_realizations

% Generate random shift in range [0, samples\_per\_bit-1] samples

random\_shift\_bits = randi([0, samples\_per\_bit-1]);

% Extract shifted region

Unipolar\_Shifted(i, :) = Unipolar\_All(i, random\_shift\_bits+1 : random\_shift\_bits+total\_samples);

PolarNRZ\_Shifted(i, :) = PolarNRZ\_All(i, random\_shift\_bits+1 : random\_shift\_bits+total\_samples);

PolarRZ\_Shifted(i, :) = PolarRZ\_All(i, random\_shift\_bits+1 : random\_shift\_bits+total\_samples);

end

end

function mean\_waveform = calculate\_mean(waveform\_matrix)

% Calculates the mean across all realizations without using the mean function

% waveform\_matrix: Matrix where each row is a realization

[num\_realizations, num\_samples] = size(waveform\_matrix); % Get matrix dimensions

mean\_waveform = sum(waveform\_matrix, 1) / num\_realizations; % Sum and divide by count

end

function plot\_mean\_waveforms(t, Unipolar\_Mean, PolarNRZ\_Mean, PolarRZ\_Mean, ...

Unipolar\_Theoretical, PolarNRZ\_Theoretical, PolarRZ\_Theoretical, A)

% Function to plot the mean waveforms of different line codes across time

% with their corresponding theoretical mean values

%

% Inputs:

% t - Time vector (ignored in favor of fixed [-700, 700] mapping)

% Unipolar\_Mean - Simulated mean waveform of Unipolar NRZ

% PolarNRZ\_Mean - Simulated mean waveform of Polar NRZ

% PolarRZ\_Mean - Simulated mean waveform of Polar RZ

% Unipolar\_Theoretical - Theoretical mean value for Unipolar NRZ

% PolarNRZ\_Theoretical - Theoretical mean value for Polar NRZ

% PolarRZ\_Theoretical - Theoretical mean value for Polar RZ

% A - Amplitude limit for y-axis

% Create new time vector from -700 to 700 ms based on length of input

t\_ms = linspace(-700, 700, length(t));

figure;

% --- Unipolar NRZ ---

subplot(3,1,1);

plot(t\_ms, Unipolar\_Mean, 'r', 'LineWidth', 2); hold on;

yline(Unipolar\_Theoretical, '--k', 'LineWidth', 1.5);

xlabel('Time (ms)');

ylabel('Amplitude');

title('Mean of Unipolar NRZ');

grid on;

ylim([-A, A]);

xlim([-length(t)-1, length(t)+1]);

legend('Simulated', 'Theoretical');

% --- Polar NRZ ---

subplot(3,1,2);

plot(t\_ms, PolarNRZ\_Mean, 'g', 'LineWidth', 2); hold on;

yline(PolarNRZ\_Theoretical, '--k', 'LineWidth', 1.5);

xlabel('Time (ms)');

ylabel('Amplitude');

title('Mean of Polar NRZ');

grid on;

ylim([-A, A]);

xlim([-length(t)-1, length(t)+1]);

legend('Simulated', 'Theoretical');

% --- Polar RZ ---

subplot(3,1,3);

plot(t\_ms, PolarRZ\_Mean, 'b', 'LineWidth', 2); hold on;

yline(PolarRZ\_Theoretical, '--k', 'LineWidth', 1.5);

xlabel('Time (ms)');

ylabel('Amplitude');

title('Mean of Polar RZ');

grid on;

ylim([-A, A]);

xlim([-length(t)-1, length(t)+1]);

legend('Simulated', 'Theoretical');

sgtitle('Mean of Different Line Codes with Theoretical Values');

end

function variance\_waveform = calculate\_variance(waveform\_matrix)

% Calculates the variance across all realizations (column-wise)

% waveform\_matrix: Matrix where each row is a realization (num\_realizations x num\_samples)

% Returns: variance\_waveform (1 x num\_samples), representing the variance of each sample point

% Compute the mean using the previously implemented function

mean\_waveform = calculate\_mean(waveform\_matrix);

[num\_realizations, num\_samples] = size(waveform\_matrix); % Get dimensions

% Ensure mean\_waveform is the same size for element-wise subtraction

mean\_waveform = repmat(mean\_waveform, num\_realizations, 1);

% Compute variance manually using the variance formula

variance\_waveform = sum((waveform\_matrix - mean\_waveform).^2, 1) / num\_realizations; % Population variance

end

function plot\_variance(t, Unipolar\_Var, PolarNRZ\_Var, PolarRZ\_Var)

% Plots the variance of different line codes over time

% t: Time vector

% Unipolar\_Var, PolarNRZ\_Var, PolarRZ\_Var: Variance waveforms

figure;

subplot(3,1,1);

plot(t, Unipolar\_Var, 'r', 'LineWidth', 2);

xlabel('Time (s)');

ylabel('Variance');

title('Variance of Unipolar NRZ');

grid on;

subplot(3,1,2);

plot(t, PolarNRZ\_Var, 'g', 'LineWidth', 2);

xlabel('Time (s)');

ylabel('Variance');

title('Variance of Polar NRZ');

grid on;

subplot(3,1,3);

plot(t, PolarRZ\_Var, 'b', 'LineWidth', 2);

xlabel('Time (s)');

ylabel('Variance');

title('Variance of Polar RZ');

grid on;

% Add a super title for clarity

sgtitle('Variance of Different Line Codes');

end

function [Unipolar\_AutoCorr, PolarNRZ\_AutoCorr, PolarRZ\_AutoCorr] =...

compute\_stat\_autocorr(Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted, max\_lag)

% Compute Statistical Autocorrelation for given signals using calculate\_mean

% Inputs:

% Unipolar\_Shifted - Shifted signal matrix for Unipolar NRZ

% PolarNRZ\_Shifted - Shifted signal matrix for Polar NRZ

% PolarRZ\_Shifted - Shifted signal matrix for Polar RZ

% Outputs:

% Unipolar\_AutoCorr - Computed autocorrelation for Unipolar NRZ

% PolarNRZ\_AutoCorr - Computed autocorrelation for Polar NRZ

% PolarRZ\_AutoCorr - Computed autocorrelation for Polar RZ

% Initialize autocorrelation arrays

Unipolar\_AutoCorr = zeros(1, max\_lag + 1);

PolarNRZ\_AutoCorr = zeros(1, max\_lag + 1);

PolarRZ\_AutoCorr = zeros(1, max\_lag + 1);

% Compute mean autocorrelation using calculate\_mean function

for i = 0:max\_lag

Unipolar\_AutoCorr(i+1) = calculate\_mean(Unipolar\_Shifted(:, 1) .\* Unipolar\_Shifted(:, i+1));

PolarNRZ\_AutoCorr(i+1) = calculate\_mean(PolarNRZ\_Shifted(:, 1) .\* PolarNRZ\_Shifted(:, i+1));

PolarRZ\_AutoCorr(i+1) = calculate\_mean(PolarRZ\_Shifted(:, 1) .\* PolarRZ\_Shifted(:, i+1));

end

% Compute symmetric autocorrelation values

Unipolar\_AutoCorr = [fliplr(Unipolar\_AutoCorr), Unipolar\_AutoCorr(2:end)];

PolarNRZ\_AutoCorr = [fliplr(PolarNRZ\_AutoCorr), PolarNRZ\_AutoCorr(2:end)];

PolarRZ\_AutoCorr = [fliplr(PolarRZ\_AutoCorr), PolarRZ\_AutoCorr(2:end)];

end

function plot\_autocorrelation(Unipolar\_AutoCorr, PolarNRZ\_AutoCorr, PolarRZ\_AutoCorr, max\_lag, Tb, A)

% Plots statistical and theoretical autocorrelation for 3 line codes

% Inputs:

% Unipolar\_AutoCorr, PolarNRZ\_AutoCorr, PolarRZ\_AutoCorr - Statistical autocorrelations

% max\_lag - Maximum lag

% Tb - Bit duration

% A - Amplitude

% Time axis for lags

t = -max\_lag:max\_lag;

tau = abs(t); % Use absolute lag for symmetry

x\_limit = max\_lag / 10;

% -------- Theoretical Expressions -------- %

% Unipolar NRZ

Rx\_Unipolar = (tau < Tb) .\* ((A^2 / 2) .\* (1 - ( tau / (2\*Tb)))) + ...

(tau >= Tb) .\* (A^2 / 4);

% Polar NRZ

Rx\_PolarNRZ = (tau < Tb) .\* (A^2 .\* (1 - (tau / Tb)));

% Polar RZ

Rx\_PolarRZ = (tau < Tb/2) .\* ((4/7) \* A^2 .\* (1 - (8 \* tau ./ (7 \* Tb))));

% -------- Plotting -------- %

figure("Name", "Statistical & Theoretical Autocorrelation");

subplot(3,1,1);

plot(t, Unipolar\_AutoCorr, 'g', 'LineWidth', 1.5); hold on;

plot(t, Rx\_Unipolar, '--k', 'LineWidth', 2);

legend('Statistical', 'Theoretical');

xlim([-701, 701]);

ylim([0, inf]);

xlabel("Lag (\tau)");

ylabel("Autocorr");

title("Unipolar NRZ: Statistical vs. Theoretical");

grid on;

subplot(3,1,2);

plot(t, PolarNRZ\_AutoCorr, 'b', 'LineWidth', 1.5); hold on;

plot(t, Rx\_PolarNRZ, '--k', 'LineWidth', 2);

legend('Statistical', 'Theoretical');

xlim([-701, 701]);

ylim([0, inf]);

xlabel("Lag (\tau)");

ylabel("Autocorr");

title("Polar NRZ: Statistical vs. Theoretical");

grid on;

subplot(3,1,3);

plot(t, PolarRZ\_AutoCorr, 'r', 'LineWidth', 1.5); hold on;

plot(t, Rx\_PolarRZ, '--k', 'LineWidth', 2);

legend('Statistical', 'Theoretical');

xlim([-701, 701]);

ylim([0, inf]);

xlabel("Lag (\tau)");

ylabel("Autocorr");

title("Polar RZ: Statistical vs. Theoretical");

grid on;

end

function TimeMean = compute\_time\_mean(waveform\_matrix)

% Computes the time mean for each realization of a given waveform

% Inputs:

% waveform\_matrix - Matrix where each row represents a realization

% Output:

% TimeMean - Column vector containing the time mean for each realization

% Compute time mean for each realization (mean along rows)

TimeMean = sum(waveform\_matrix, 2) / size(waveform\_matrix, 2);

end

function plot\_realizations\_with\_mean(t\_shifted, Signals\_TimeMean, signals\_waveform, signal\_name, color)

% Plots the first 3 realizations of a signal in a 3x2 grid and displays their time means as text.

%

% Inputs:

% t\_shifted - Time vector

% Signals\_TimeMean - Vector of time means (one per realization)

% signals\_waveform - Matrix where each row is a realization

% signal\_name - Name of the signal (string) for labeling

% color - Plot color (e.g., 'g' for green)

figure('Name', [signal\_name, ' - Realizations and Time Mean']);

for i = 1:3

% First Column: Plot the waveform realization

subplot(3,2,(i-1)\*2+1);

plot(t\_shifted, signals\_waveform(i,:), color, 'LineWidth', 1.5); % Plot waveform

hold on;

yline(Signals\_TimeMean(i), '--k', 'LineWidth', 1.5); % Add time mean line

hold off;

xlabel('Time');

ylabel('Amplitude');

title([signal\_name, ' - Realization ', num2str(i)]);

grid on;

% Second Column: Display time mean as a text box

subplot(3,2,(i-1)\*2+2);

axis off; % Hide axes for a clean text display

text(0.5, 0.5, sprintf('Time Mean:\n%.3f', Signals\_TimeMean(i)), ...

'FontSize', 12, 'FontWeight', 'bold', 'HorizontalAlignment', 'center', 'BackgroundColor', 'w');

end

end

function plot\_time\_mean\_vs\_realization(unipolar\_mean, polarNRZ\_mean, polarRZ\_mean, A)

% Function to plot the time mean vs realization index (symmetric around 0)

%

% Inputs:

% - unipolar\_mean: Time mean of Unipolar NRZ (1xN or Nx1 vector)

% - polarNRZ\_mean: Time mean of Polar NRZ (1xN or Nx1 vector)

% - polarRZ\_mean: Time mean of Polar RZ (1xN or Nx1 vector)

% - A: Amplitude limit for y-axis

% Ensure row vectors

unipolar\_mean = unipolar\_mean(:).';

polarNRZ\_mean = polarNRZ\_mean(:).';

polarRZ\_mean = polarRZ\_mean(:).';

N = length(unipolar\_mean);

realization\_indices = -N+1:N-1; % match mirrored length = 2N - 1

% Mirror signals (excluding duplicated center)

unipolar\_mirrored = [fliplr(unipolar\_mean(2:end)), unipolar\_mean];

polarNRZ\_mirrored = [fliplr(polarNRZ\_mean(2:end)), polarNRZ\_mean];

polarRZ\_mirrored = [fliplr(polarRZ\_mean(2:end)), polarRZ\_mean];

figure;

subplot(3,1,1);

plot(realization\_indices, unipolar\_mirrored, 'r', 'LineWidth', 2);

grid on;

xlabel('Realization Index');

ylabel('Time Mean');

title('Unipolar NRZ Time Mean');

ylim([-A, A]);

subplot(3,1,2);

plot(realization\_indices, polarNRZ\_mirrored, 'g', 'LineWidth', 2);

grid on;

xlabel('Realization Index');

ylabel('Time Mean');

title('Polar NRZ Time Mean');

ylim([-A, A]);

subplot(3,1,3);

plot(realization\_indices, polarRZ\_mirrored, 'b', 'LineWidth', 2);

grid on;

xlabel('Realization Index');

ylabel('Time Mean');

title('Polar RZ Time Mean');

ylim([-A, A]);

sgtitle('Time Mean vs Realization Index');

end

function [R\_unipolar\_t1, R\_polar\_nrz\_t1, R\_polar\_rz\_t1, tau\_vec] = ...

compute\_time\_autocorr(UnipolarNRZ, PolarNRZ, PolarRZ, t1)

% Computes time autocorrelation R(t1, tau) at a fixed t1 for tau = 0:700

%

% Inputs:

% UnipolarNRZ, PolarNRZ, PolarRZ - Realizations (each row is a signal)

% t1 - Time index to fix (must be positive and < num\_samples)

%

% Outputs:

% R\_unipolar\_t1, R\_polar\_nrz\_t1, R\_polar\_rz\_t1 - Autocorrelation vectors

% tau\_vec - Vector of lags (tau)

[num\_realizations, num\_samples] = size(UnipolarNRZ);

max\_tau = 690;

tau\_vec = 0:max\_tau;

% Preallocate

R\_unipolar\_t1 = zeros(1, length(tau\_vec));

R\_polar\_nrz\_t1 = zeros(1, length(tau\_vec));

R\_polar\_rz\_t1 = zeros(1, length(tau\_vec));

for idx = 1:length(tau\_vec)

tau = tau\_vec(idx);

t2 = t1 + tau;

% Compute element-wise products for all realizations at t1 and t1+tau

prod\_unipolar = UnipolarNRZ(:, t1) .\* UnipolarNRZ(:, t2);

prod\_polar = PolarNRZ(:, t1) .\* PolarNRZ(:, t2);

prod\_rz = PolarRZ(:, t1) .\* PolarRZ(:, t2);

% Use custom function to compute mean across realizations

R\_unipolar\_t1(idx) = sum(prod\_unipolar) / num\_realizations;

R\_polar\_nrz\_t1(idx) = sum(prod\_polar) / num\_realizations;

R\_polar\_rz\_t1(idx) = sum(prod\_rz) / num\_realizations;

end

end

function plot\_time\_autocorrelation(R\_unipolar, R\_polarNRZ, R\_polarRZ, tau\_vec, max\_lag, Tb, A)

% Plots experimental and theoretical time autocorrelation for each waveform type.

%

% Inputs:

% R\_unipolar, R\_polarNRZ, R\_polarRZ - matrices of time autocorrelation (each row = realization)

% tau\_vec - Vector of lags (non-negative only)

% max\_lag - Maximum lag value (samples) to define axis limits

% Tb - Bit duration

% A - Amplitude

x\_limit = max\_lag / 10;

% Full tau range including negative lags

tau\_full = [-flip(tau\_vec(2:end)), tau\_vec]; % symmetric lags (excluding 0 twice)

% Flip the autocorrelation to complete the negative half

R\_unipolar\_full = [fliplr(R\_unipolar(1,2:end)), R\_unipolar(1,:)];

R\_polarNRZ\_full = [fliplr(R\_polarNRZ(1,2:end)), R\_polarNRZ(1,:)];

R\_polarRZ\_full = [fliplr(R\_polarRZ(1,2:end)), R\_polarRZ(1,:)];

% Theoretical expressions

tau\_sec = tau\_full \* (Tb / 7); % Convert to seconds assuming 10 samples per Tb

% -------- Theoretical Expressions -------- %

Rx\_Unipolar = (abs(tau\_sec) < Tb) .\* ((A^2 / 2) .\* (1 - (abs(tau\_sec) / (2\*Tb)))) + ...

(abs(tau\_sec) >= Tb) .\* (A^2 / 4);

Rx\_PolarNRZ = (abs(tau\_sec) < Tb) .\* (A^2 .\* (1 - abs(tau\_sec) / Tb));

Rx\_PolarRZ = (abs(tau\_sec) < Tb/2) .\* ((4/7) \* A^2 .\* (1 - (8 \* abs(tau\_sec) ./ (7 \* Tb))));

% Plotting

figure('Name', 'Time Autocorrelation');

% Unipolar NRZ

subplot(3,1,1);

plot(tau\_full, R\_unipolar\_full, 'b', 'LineWidth', 1.5); hold on;

plot(tau\_full, Rx\_Unipolar, '--k', 'LineWidth', 1.2);

xlim([-501, 501]);

grid on;

xlabel('Time Lag (samples)'); ylabel('Magnitude');

title('Unipolar NRZ Autocorrelation'); legend('Experimental', 'Theoretical');

% Polar NRZ

subplot(3,1,2);

plot(tau\_full, R\_polarNRZ\_full, 'r', 'LineWidth', 1.5); hold on;

plot(tau\_full, Rx\_PolarNRZ, '--k', 'LineWidth', 1.2);

xlim([-501, 501]);

grid on;

xlabel('Time Lag (samples)'); ylabel('Magnitude');

title('Polar NRZ Autocorrelation'); legend('Experimental', 'Theoretical');

% Polar RZ

subplot(3,1,3);

plot(tau\_full, R\_polarRZ\_full, 'g', 'LineWidth', 1.5); hold on;

plot(tau\_full, Rx\_PolarRZ, '--k', 'LineWidth', 1.2);

xlim([-501, 501]);

grid on;

xlabel('Time Lag (samples)'); ylabel('Magnitude');

title('Polar RZ Autocorrelation'); legend('Experimental', 'Theoretical');

end

function plot\_mean\_time\_vs\_statistical(t, ...

Unipolar\_Mean, PolarNRZ\_Mean, PolarRZ\_Mean, ...

Unipolar\_Theoretical, PolarNRZ\_Theoretical, PolarRZ\_Theoretical, ...

Unipolar\_TimeMean, PolarNRZ\_TimeMean, PolarRZ\_TimeMean, A)

% Plot statistical (time mean vs realization) and mean waveform vs time side by side

%

% Inputs:

% t - Time vector

% \*\_Mean - Simulated mean waveforms

% \*\_Theoretical - Theoretical mean values

% \*\_TimeMean - Time mean for each realization

% A - Amplitude limit for Y-axis

% Ensure row vectors

Unipolar\_TimeMean = Unipolar\_TimeMean(:).';

PolarNRZ\_TimeMean = PolarNRZ\_TimeMean(:).';

PolarRZ\_TimeMean = PolarRZ\_TimeMean(:).';

% Time axis for waveform mean (same as in original)

t\_ms = linspace(-700, 700, length(t));

% Realization indices (symmetric for mirroring)

N = length(Unipolar\_TimeMean);

realization\_indices = -N+1:N-1;

% Mirror time mean data (excluding duplicated center)

unipolar\_mirrored = [fliplr(Unipolar\_TimeMean(2:end)), Unipolar\_TimeMean];

polarNRZ\_mirrored = [fliplr(PolarNRZ\_TimeMean(2:end)), PolarNRZ\_TimeMean];

polarRZ\_mirrored = [fliplr(PolarRZ\_TimeMean(2:end)), PolarRZ\_TimeMean];

figure;

% Unipolar

subplot(3,2,1); % Row 1, Col 1

plot(realization\_indices, unipolar\_mirrored, 'r', 'LineWidth', 2);

xlabel('Realization Index'); ylabel('Time Mean');

title('Unipolar NRZ Time Mean');

ylim([-A, A]); grid on;

subplot(3,2,2); % Row 1, Col 2

plot(t\_ms, Unipolar\_Mean, 'r', 'LineWidth', 2); hold on;

yline(Unipolar\_Theoretical, '--k', 'LineWidth', 1.5);

xlabel('Time (ms)'); ylabel('Amplitude');

title('Unipolar NRZ Mean Waveform');

legend('Simulated', 'Theoretical'); grid on;

ylim([-A, A]); xlim([min(t\_ms), max(t\_ms)]);

% Polar NRZ

subplot(3,2,3);

plot(realization\_indices, polarNRZ\_mirrored, 'g', 'LineWidth', 2);

xlabel('Realization Index'); ylabel('Time Mean');

title('Polar NRZ Time Mean');

ylim([-A, A]); grid on;

subplot(3,2,4);

plot(t\_ms, PolarNRZ\_Mean, 'g', 'LineWidth', 2); hold on;

yline(PolarNRZ\_Theoretical, '--k', 'LineWidth', 1.5);

xlabel('Time (ms)'); ylabel('Amplitude');

title('Polar NRZ Mean Waveform');

legend('Simulated', 'Theoretical'); grid on;

ylim([-A, A]); xlim([min(t\_ms), max(t\_ms)]);

% Polar RZ

subplot(3,2,5);

plot(realization\_indices, polarRZ\_mirrored, 'b', 'LineWidth', 2);

xlabel('Realization Index'); ylabel('Time Mean');

title('Polar RZ Time Mean');

ylim([-A, A]); grid on;

subplot(3,2,6);

plot(t\_ms, PolarRZ\_Mean, 'b', 'LineWidth', 2); hold on;

yline(PolarRZ\_Theoretical, '--k', 'LineWidth', 1.5);

xlabel('Time (ms)'); ylabel('Amplitude');

title('Polar RZ Mean Waveform');

legend('Simulated', 'Theoretical'); grid on;

ylim([-A, A]); xlim([min(t\_ms), max(t\_ms)]);

sgtitle('Statistical vs Mean Waveform (Unipolar, Polar NRZ, Polar RZ)');

end

function [R\_avg] = plot\_two\_realizations(R\_linecode1, R\_linecode2, tau\_vec, max\_lag)

% Plots the time autocorrelation for two realizations of a single line code,

% and their average.

%

% Inputs:

% R\_linecode1, R\_linecode2 - Vectors of time autocorrelation (one per realization)

% tau\_vec - Vector of lags (non-negative only)

% max\_lag - Maximum lag value (samples) to define axis limits

x\_limit = max\_lag / 10;

% Ensure inputs are row vectors

R\_linecode1 = R\_linecode1(:).'; % force to row vector

R\_linecode2 = R\_linecode2(:).';

% Full tau range including negative lags

tau\_full = [-flip(tau\_vec(2:end)), tau\_vec]; % symmetric lags

% Construct full autocorrelation by mirroring

R\_linecode1\_full = [fliplr(R\_linecode1(2:end)), R\_linecode1];

R\_linecode2\_full = [fliplr(R\_linecode2(2:end)), R\_linecode2];

% Compute average autocorrelation

R\_avg = 0.5 \* (R\_linecode1\_full + R\_linecode2\_full);

% Plotting

figure('Name', 'Time Autocorrelation for Two Realizations + Average');

% --- First subplot: the two realizations

subplot(2,1,1);

plot(tau\_full, R\_linecode1\_full, 'b', 'LineWidth', 1.5); hold on;

plot(tau\_full, R\_linecode2\_full, 'r--', 'LineWidth', 1.5);

grid on;

xlim([-x\_limit x\_limit]);

xlabel('Time Lag (samples)');

ylabel('Magnitude');

title('Time Autocorrelation of Line Code - Two Realizations');

legend('Realization 1 (t1)', 'Realization 2 (t2)');

% --- Second subplot: average autocorrelation

subplot(2,1,2);

plot(tau\_full, R\_avg, 'k', 'LineWidth', 2);

grid on;

xlim([-x\_limit x\_limit]);

xlabel('Time Lag (samples)');

ylabel('Magnitude');

title('Average Time Autocorrelation');

legend('Average of t1 and t2');

end

function [R\_avg\_unipolar, R\_avg\_polar\_nrz, R\_avg\_polar\_rz, tau\_full] = ...

get\_Ravg(Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted, t1, t2)

% Computes the average time autocorrelation (R\_avg) of each line code over two time instances

%

% Inputs:

% Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted : matrices with realizations over time

% t1, t2 : time indices to extract 2 realizations

%

% Outputs:

% R\_avg\_unipolar, R\_avg\_polar\_nrz, R\_avg\_polar\_rz : averaged autocorrelations

% tau\_full : vector of symmetric lag values

% Compute autocorrelation for two time slices

[R1\_unipolar, R1\_polar\_nrz, R1\_polar\_rz, tau\_vec] = ...

compute\_time\_autocorr(Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted, t1);

[R2\_unipolar, R2\_polar\_nrz, R2\_polar\_rz, ~] = ...

compute\_time\_autocorr(Unipolar\_Shifted, PolarNRZ\_Shifted, PolarRZ\_Shifted, t2);

% Symmetric tau range (include negative lags)

tau\_full = [-flip(tau\_vec(2:end)), tau\_vec];

% Symmetric autocorrelations

R1\_unipolar\_full = [fliplr(R1\_unipolar(2:end)), R1\_unipolar];

R2\_unipolar\_full = [fliplr(R2\_unipolar(2:end)), R2\_unipolar];

R1\_polar\_nrz\_full = [fliplr(R1\_polar\_nrz(2:end)), R1\_polar\_nrz];

R2\_polar\_nrz\_full = [fliplr(R2\_polar\_nrz(2:end)), R2\_polar\_nrz];

R1\_polar\_rz\_full = [fliplr(R1\_polar\_rz(2:end)), R1\_polar\_rz];

R2\_polar\_rz\_full = [fliplr(R2\_polar\_rz(2:end)), R2\_polar\_rz];

% Average of the two realizations

R\_avg\_unipolar = 0.5 \* (R1\_unipolar\_full + R2\_unipolar\_full);

R\_avg\_polar\_nrz = 0.5 \* (R1\_polar\_nrz\_full + R2\_polar\_nrz\_full);

R\_avg\_polar\_rz = 0.5 \* (R1\_polar\_rz\_full + R2\_polar\_rz\_full);

end

function plot\_combined\_autocorrelation(R\_unipolar, R\_polarNRZ, R\_polarRZ, ...

tau\_vec, max\_lag, Tb, A, ...

Unipolar\_AutoCorr, PolarNRZ\_AutoCorr, PolarRZ\_AutoCorr)

% Combined plot of time-domain and statistical autocorrelations

%

% Inputs:

% R\_unipolar, R\_polarNRZ, R\_polarRZ - Time autocorrelation matrices (1 realization)

% tau\_vec - Non-negative tau vector

% max\_lag - Max lag for axis limits

% Tb - Bit duration

% A - Amplitude

% \*\_AutoCorr - Statistical autocorrelation vectors

x\_limit = max\_lag / 10;

% Full symmetric tau vector

tau\_full = [-flip(tau\_vec(2:end)), tau\_vec];

tau\_sec = tau\_full \* (Tb / 7); % seconds

% Reconstruct full symmetric time-domain autocorrelation

R\_unipolar\_full = [fliplr(R\_unipolar(1,2:end)), R\_unipolar(1,:)];

R\_polarNRZ\_full = [fliplr(R\_polarNRZ(1,2:end)), R\_polarNRZ(1,:)];

R\_polarRZ\_full = [fliplr(R\_polarRZ(1,2:end)), R\_polarRZ(1,:)];

% Theoretical autocorrelation

Rx\_Unipolar = (abs(tau\_sec) < Tb) .\* ((A^2 / 2) .\* (1 - (abs(tau\_sec) / (2\*Tb)))) + ...

(abs(tau\_sec) >= Tb) .\* (A^2 / 4);

Rx\_PolarNRZ = (abs(tau\_sec) < Tb) .\* (A^2 .\* (1 - abs(tau\_sec) / Tb));

Rx\_PolarRZ = (abs(tau\_sec) < Tb/2) .\* ((4/7) \* A^2 .\* (1 - (8 \* abs(tau\_sec) ./ (7 \* Tb))));

% Statistical lag axis

t = -max\_lag:max\_lag;

tau = abs(t);

% Theoretical for statistical view

Rx\_Unipolar\_stat = (tau < Tb) .\* ((A^2 / 2) .\* (1 - ( tau / (2\*Tb)))) + ...

(tau >= Tb) .\* (A^2 / 4);

Rx\_PolarNRZ\_stat = (tau < Tb) .\* (A^2 .\* (1 - (tau / Tb)));

Rx\_PolarRZ\_stat = (tau < Tb/2) .\* ((4/7) \* A^2 .\* (1 - (8 \* tau ./ (7 \* Tb))));

figure('Name', 'Autocorrelation: Time-Domain & Statistical');

% ---- Comparison: Time vs Statistical Autocorrelation ----

figure('Name', 'Comparison: Time vs Statistical Autocorrelations');

% Unipolar NRZ

subplot(3, 1, 1);

plot(tau\_full, R\_unipolar\_full, 'b', 'LineWidth', 1.5); hold on;

plot(t, Unipolar\_AutoCorr, '--r', 'LineWidth', 1.5);

xlim([-x\_limit, x\_limit]);

title('Unipolar NRZ: Time vs Statistical');

xlabel('Lag'); ylabel('Autocorrelation');

legend('Time-domain', 'Statistical'); grid on;

% Match the overlapping segment

min\_len = min(length(R\_unipolar\_full), length(Unipolar\_AutoCorr));

R\_trimmed = R\_unipolar\_full(1:min\_len);

AutoCorr\_trimmed = Unipolar\_AutoCorr(1:min\_len);

% Now compare

disp("The Statical and Time differnece is");

disp(norm(R\_trimmed - AutoCorr\_trimmed));

% Polar NRZ

subplot(3, 1, 2);

plot(tau\_full, R\_polarNRZ\_full, 'b', 'LineWidth', 1.5); hold on;

plot(t, PolarNRZ\_AutoCorr, '--r', 'LineWidth', 1.5);

xlim([-x\_limit, x\_limit]);

title('Polar NRZ: Time vs Statistical');

xlabel('Lag'); ylabel('Autocorrelation');

legend('Time-domain', 'Statistical'); grid on;

% Polar RZ

subplot(3, 1, 3);

plot(tau\_full, R\_polarRZ\_full, 'b', 'LineWidth', 1.5); hold on;

plot(t, PolarRZ\_AutoCorr, '--r', 'LineWidth', 1.5);

xlim([-x\_limit, x\_limit]);

title('Polar RZ: Time vs Statistical');

xlabel('Lag'); ylabel('Autocorrelation');

legend('Time-domain', 'Statistical'); grid on;

end

function [PSD\_unipolar ,PSD\_polarNRZ ,PSD\_polarRZ] =...

plot\_linecode\_psd(R\_Unipolar, R\_PolarNRZ, R\_PolarRZ, fs, A, Tb)

% Function to plot PSDs for Unipolar NRZ, Polar NRZ, and Polar RZ line codes

% using FFT of average autocorrelation sequences and compare with theoretical PSDs.

%

% Inputs:

% R\_Unipolar - Average autocorrelation of Unipolar NRZ

% R\_PolarNRZ - Average autocorrelation of Polar NRZ

% R\_PolarRZ - Average autocorrelation of Polar RZ

% fs - Sampling frequency in Hz

% A - Amplitude of the signal

% Tb - Bit period (duration of one bit)

% Ensure all inputs are column vectors

R\_Unipolar = R\_Unipolar(:);

R\_PolarNRZ = R\_PolarNRZ(:);

R\_PolarRZ = R\_PolarRZ(:);

% Number of samples

n = length(R\_Unipolar);

% Index of center (DC component after fftshift)

center\_idx = ceil(n / 2);

% Remove DC Pulse

mu\_uni = mean(R\_Unipolar(end-10:end)); % Use tail values

R\_Unipolar = R\_Unipolar - mu\_uni; % Remove DC

% Compute FFTs of autocorrelations

fft\_unipolar = fft(R\_Unipolar) / n;

fft\_polarNRZ = fft(R\_PolarNRZ) / n;

fft\_polarRZ = fft(R\_PolarRZ) / n;

% Compute PSD magnitudes

PSD\_unipolar = abs(fft\_unipolar);

PSD\_polarNRZ = abs(fft\_polarNRZ);

PSD\_polarRZ = abs(fft\_polarRZ);

% Frequency axis centered around 0

freq\_axis = (-n/2 : n/2 - 1) \* (fs / n);

% Center the FFTs for proper plotting

PSD\_unipolar = A\*fftshift(PSD\_unipolar);

PSD\_polarNRZ = A\*fftshift(PSD\_polarNRZ);

PSD\_polarRZ = A\*fftshift(PSD\_polarRZ);

% Compute theoretical PSDs

S\_unipolar\_nrz = (A^2 \* Tb / 4) \* (sin(pi \* freq\_axis \* Tb) ./ (pi \* freq\_axis \* Tb)).^2;

S\_polar\_nrz = (A^2 \* Tb / 2) \* (sin(pi \* freq\_axis \* Tb / 2) ./ (pi \* freq\_axis \* Tb / 2)).^2;

S\_polar\_rz = (A^2 \* Tb / 4) \* (sin(pi \* freq\_axis \* Tb / 4) ./ (pi \* freq\_axis \* Tb / 4)).^2;

% Handle zero frequency case (Dirac delta at f = 0)

S\_unipolar\_nrz(freq\_axis == 0) = A^2 / 4;

S\_polar\_nrz(freq\_axis == 0) = 0; % No delta for Polar NRZ

S\_polar\_rz(freq\_axis == 0) = 0; % No delta for Polar RZ

% Plot PSDs

figure('Name', 'Power Spectral Density via Average Autocorrelation');

% Plot Unipolar NRZ

subplot(3,1,1);

plot(freq\_axis, PSD\_unipolar, 'b', 'LineWidth', 1.5);

hold on;

plot(freq\_axis, S\_unipolar\_nrz, 'r--', 'LineWidth', 1.5); % Theoretical PSD

hold off;

grid on;

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('PSD of Unipolar NRZ');

xlim([-fs/2, fs/2]);

ylim([0, max(PSD\_unipolar)\*1.1]);

legend('Empirical', 'Theoretical');

% Plot Polar NRZ

subplot(3,1,2);

plot(freq\_axis, PSD\_polarNRZ, 'r', 'LineWidth', 1.5);

hold on;

plot(freq\_axis, S\_polar\_nrz, 'g--', 'LineWidth', 1.5); % Theoretical PSD

hold off;

grid on;

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('PSD of Polar NRZ');

xlim([-fs/2, fs/2]);

ylim([0, max(PSD\_polarNRZ)\*1.1]);

legend('Empirical', 'Theoretical');

% Plot Polar RZ

subplot(3,1,3);

plot(freq\_axis, PSD\_polarRZ, 'g', 'LineWidth', 1.5);

hold on;

plot(freq\_axis, S\_polar\_rz, 'b--', 'LineWidth', 1.5); % Theoretical PSD

hold off;

grid on;

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('PSD of Polar RZ');

xlim([-fs/2, fs/2]);

ylim([0, max(PSD\_polarRZ)\*1.1]);

legend('Empirical', 'Theoretical');

end