Faculty of Engineering- Cairo University

Electronics and Electrical Communications Department

Digital Communications

Project # 1

Team 31

Name	ID	Sec.	B.N.
عبدالله ابراهيم السيد عبدالهادي	9210602	2	48
عمر احمد عبد العليم مرزوق	9210704	3	7
فتحى مصطفى فتحى عبدالحميد	9210808	3	19
محمود حمدى جابر عبد التواب	9211083	4	10

Contents

Table of Figures

0. Role of each member

Members	Role
Abdallah Ibrahim	Line Code Generation
Omar Marzouk	Statistical and Time mean
Fathy Mostafa	Statistical and Time autocorrelation
Mahmoud Hamdy	Power Spectral Density
All of the members	Report Organization and check of code reliability

I. Introduction

Software-defined radio (SDR) revolutionizes the way radio systems are conceptualized and implemented. By bringing the processing tasks closer to the antenna, SDR converts traditional hardware problems into software challenges. The core principle of SDR lies in its ability to define transmitted waveforms and demodulate received signals entirely through software, a stark departure from conventional radios reliant on analog circuitry or hybrid digital-analog systems.

In the realm of SDR, the transmitter's code defines modulation techniques, coding schemes, and other crucial parameters. Beyond its traditional application in wireless communication, SDR extends its utility to wired data transmission, replacing antennas with cable connections and employing various line codes. These codes facilitate framing, coding, and line modulation, enabling seamless data transmission through wired channels.

II. Problem description

Consider a simplified example to illustrate the application of SDR: transmitting binary data using polar Non-Return-to-Zero (NRZ) signaling. In this scenario, voltage levels are used corresponding to the binary data. For instance, transmitting '1' corresponds to a positive voltage level, while '0' corresponds to a negative voltage level.

The output of the transmitter code generates a sequence of voltage levels, mimicking the waveform sent to the digital-to-analog converter (DAC). As this process involves random bits and precise timing, the DAC's output can be regarded as a random process.

We aim to generate an ensemble of 500 waveforms, each containing 100 bits, for three distinct line codes: Unipolar Signaling, Polar NRZ, and Return to Zero (RZ). Several computations are required for each line code, including statistical mean calculation, stationarity assessment, ensemble autocorrelation function determination, time mean, autocorrelation function analysis for a single waveform, ergodicity evaluation, and bandwidth estimation of the transmitted signal.

This investigation sheds light on the characteristics and performance metrics of different line codes within the SDR framework, contributing to the understanding and optimization of software-defined radio systems.

III. Control flags

Control Flags	Values
Number of bits	100
Number of realizations	500
Number of samples per bit	7 samples/bit
Time shift	From 0 to 6

IV. Generation of data

We will generate 500 realizations of random data and the length of each realization is 100 bits. Each bit will have duration of 70 msec. The ADC time per sample is 10 msec, which means each bit will have 7 samples. After generating line codes, we generate a random delay where the delay is translated to random number of samples from 0 to 6.

```
global num_bits;
global num_realizations;
% Define scaling factor A, number of bits, and number of realizations
A = 4;
num_bits = 100;
num_realizations = 500;
% Generate random binary data and transmission delays
Data = randi([0, 1], num_realizations, num_bits + 1);
td = randi([0, 6], num_realizations, 1);
```

V. Creating Uni-polar Ensemble

For generating unipolar we need to get A for 1 and 0 for 0 so we will multiply the generated data bits (1 or 0) by A

```
% Unipolar Encoding
unipolar_Tx = Data * A;
unipolar_Tx_out = repelem(unipolar_Tx, 1, 7);  % Repeat each element 7 times
```

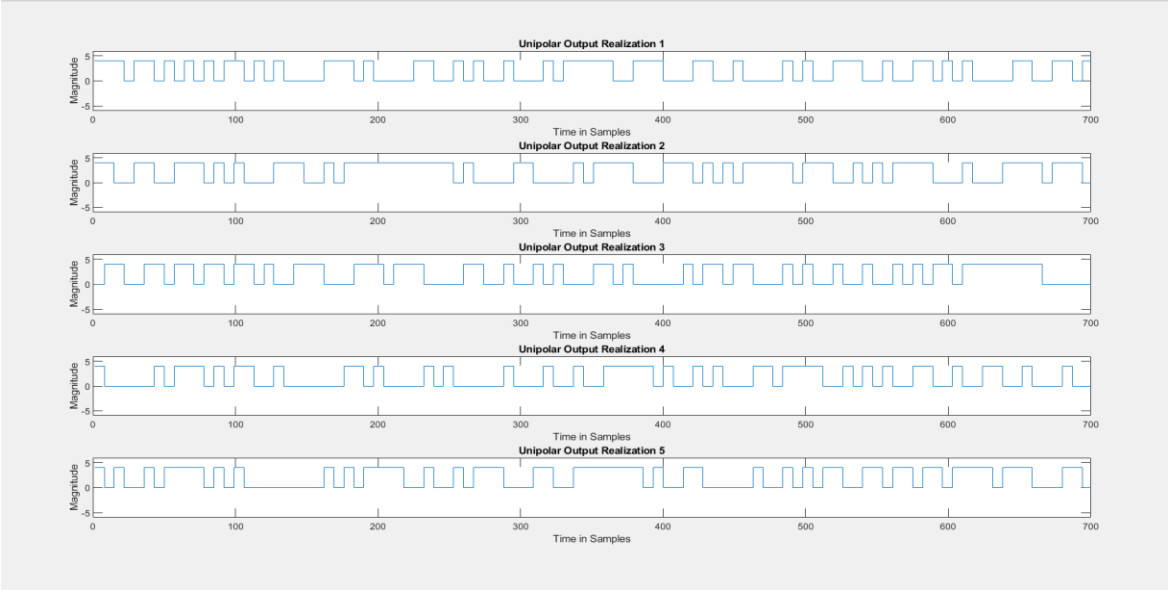

Figure 1: Five realizations from Uni-polar line code

VI. Creating Polar NRZ Ensemble

For generating polar NRZ we need to get the bit in form of –A for 0 and A for 1 so we multiply the data by 2 and subtract 1 from the result then multiply it by A

```
% Polar NRZ Encoding
polar_NRZ_Tx = ((2 * Data) - 1) * A;
polar_NRZ_Tx_out = repelem(polar_NRZ_Tx, 1, 7); % Repeat each element 7 times
```

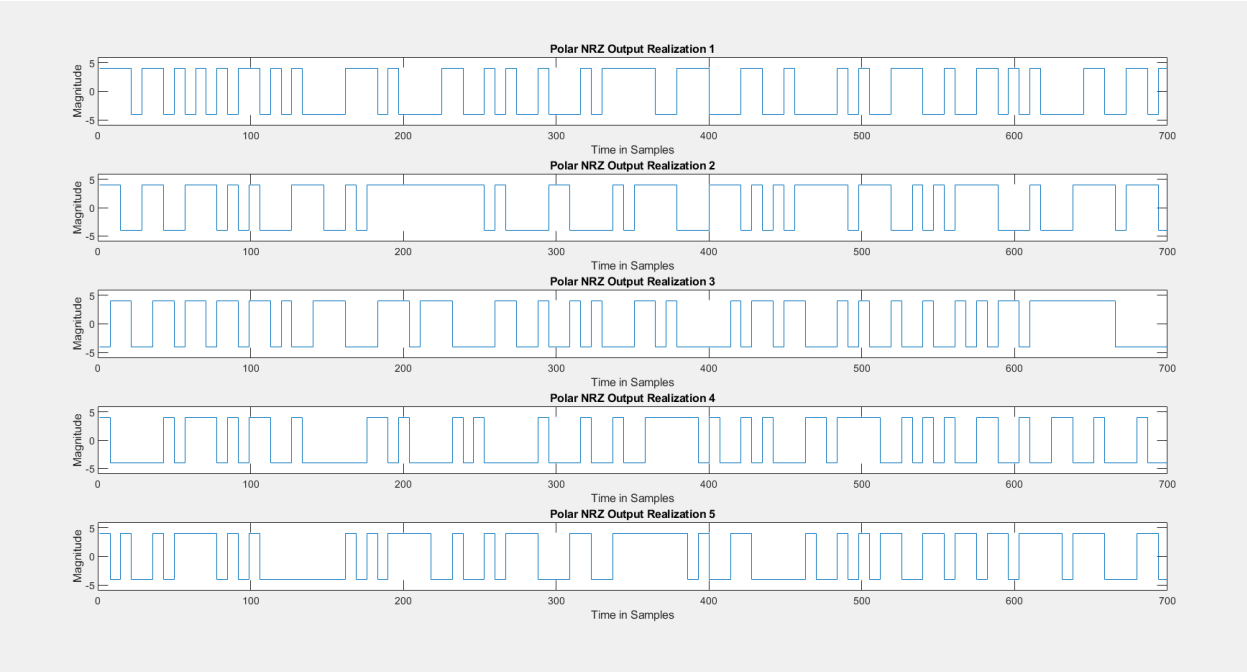

Figure 2: Five realizations from Polar NRZ Line Code

VII. Creating Polar RZ Ensemble

For generating polar RZ we need to get A for 1 and –A for 0 but only for half the time period ,since the DAC sample the bits every 10 msec then the bit will be sampled 7 times so we will generate 4 of the sampled bits and the rest will be zero

```
% Polar RZ Encoding
polar_RZ_Tx_out = repelem(polar_NRZ_Tx, 1, 4);  % Repeat each element 4 times
% Add three zeros after every set of four elements
polar_RZ_Tx_out_new = zeros(num_realizations, num_bits * 7 + 7);
for i = 1:num_realizations
    n = 0;
    for j = 1:4:num_bits*4 +4
        % Insert 3 zeros after every set of four elements
        polar_RZ_Tx_out_new(i, j + 3 * n:j + 3 + 3 * n) = polar_RZ_Tx_out(i, j:j + 3);
        n = n + 1;
    end
end
```

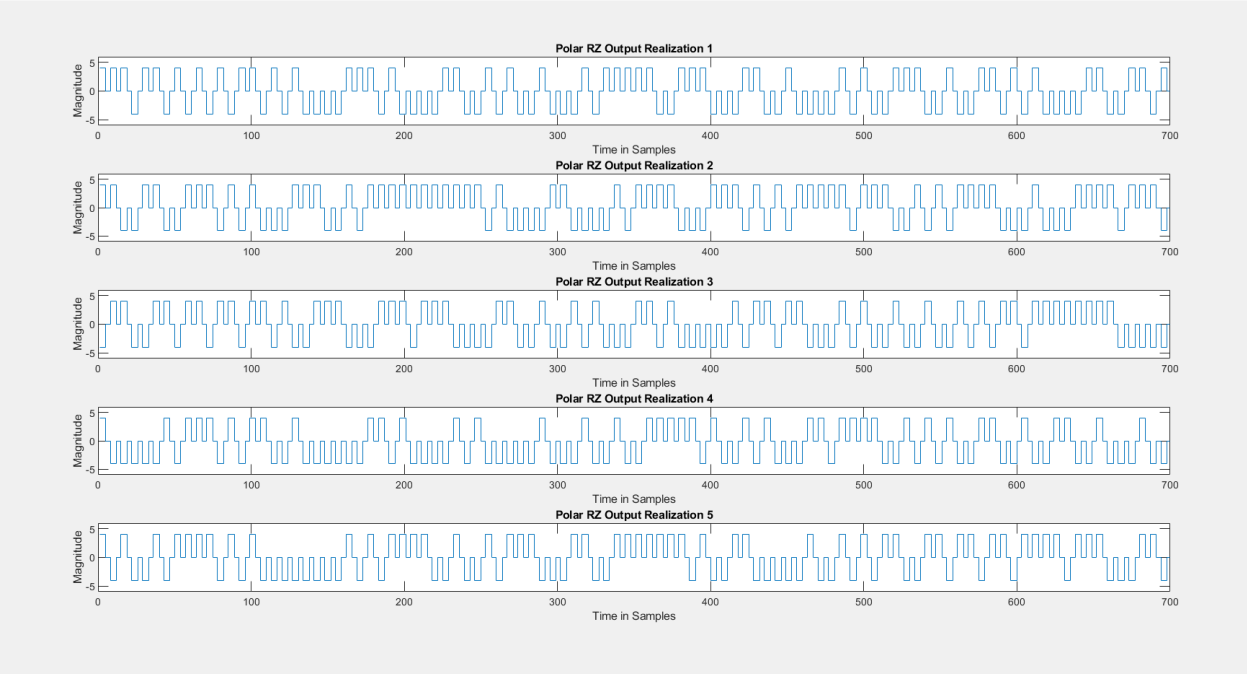

Figure 3: Five realizations from Polar RZ Line Code

VIII. Applying random initial time shift for each waveform

To apply the random initial time shift we added an additional bit when generating the data random bits

```
% Generate random binary data and transmission delays
Data = randi([0, 1], num_realizations, num_bits + 1);
```

IX. Getting the cell array reading to calculate the statistical mean and autocorrelation

To get the cell array ready for mean and autocorrelation calculation we take the 700 elements after the time delay td for each realization

```
for i = 1:num_realizations
    % Extract the relevant portion based on transmission delays
    unipolar_Tx_out_new(i, :) = unipolar_Tx_out(i, td(i) + 1:td(i) + num_bits * 7);
end
for i = 1:num_realizations
    % Extract the relevant portion based on transmission delays
    polar_NRZ_Tx_out_new(i, :) = polar_NRZ_Tx_out(i, td(i) + 1:td(i) + num_bits * 7);
end
for i = 1:num_realizations
    % Extract the relevant portion based on transmission delays
    polar_RZ_Tx_out_final(i, :) = polar_RZ_Tx_out_new(i, td(i) + 1:td(i) + num_bits *
7);
end
```

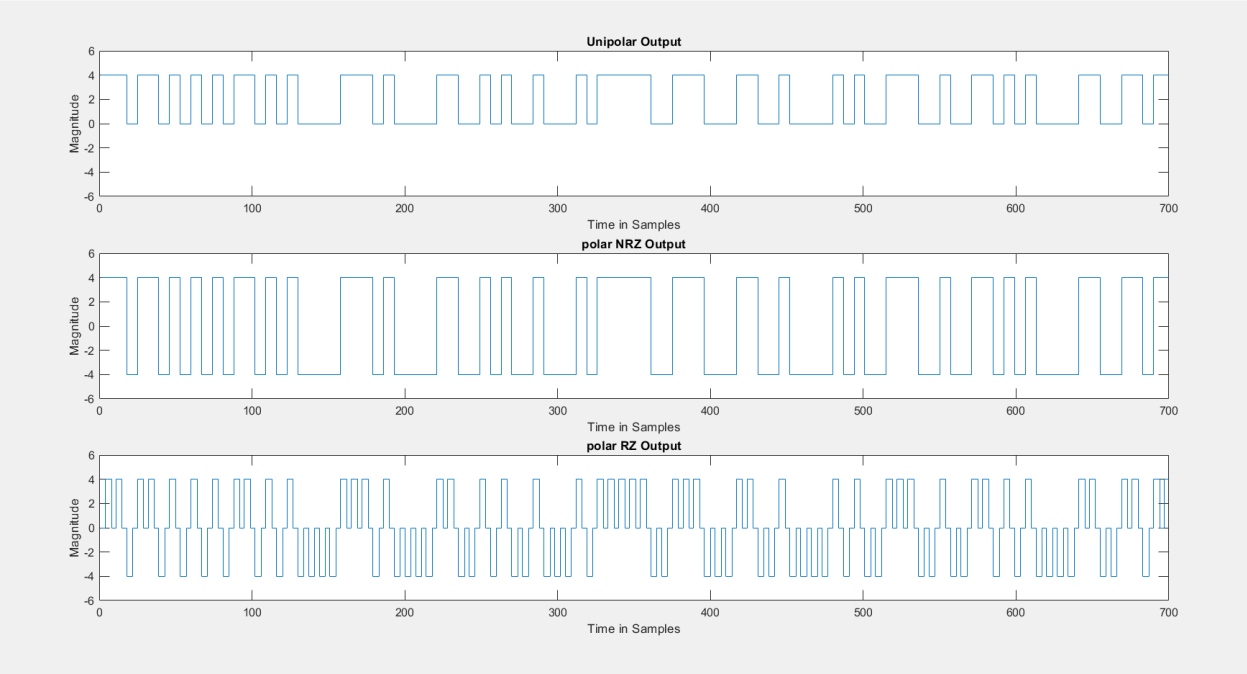

Figure 4: line codes after cutting out the time delay

X. Q1: Calculating the Statistical mean

We can calculate the Statistical mean using this formula

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

Unipolar:

$$\overline{\boldsymbol{x(t)}} = \boldsymbol{P(A) * A + P(0) * 0} = \frac{\boldsymbol{A}}{\boldsymbol{2}}$$

Polar NRZ:

$$\overline{\boldsymbol{x(t)}} = \boldsymbol{P(A) * A + P(-A) * -A = 0}$$

Polar RZ:

$$\overline{\boldsymbol{x(t)}} = \boldsymbol{P(A) * A + P(-A) * -A = 0}$$

XI. Plotting the statistical mean

```
% Calculate The statistical mean for each line code
stat_unipolar_mean = zeros(1, 700);
stat_polar_NRZ_mean = zeros(1, 700);
stat_polar_RZ_mean = zeros(1, 700);
for i = 1:700
    stat_unipolar_mean(i) = sum(unipolar_Tx_out_new(:, i)) / 500;
    stat_polar_NRZ_mean(i) = sum(polar_NRZ_Tx_out_new(:, i)) / 500;
    stat_polar_RZ_mean(i) = sum(polar_RZ_Tx_out_final(:, i)) / 500;
end
```

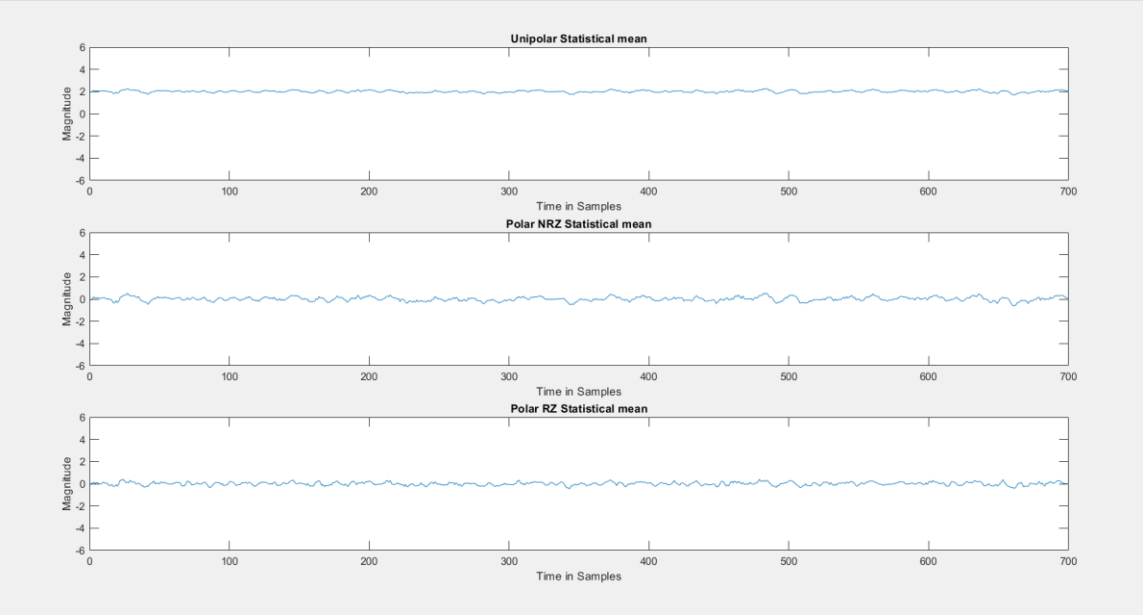

Figure 5: Line Codes Statistical mean

As shown in figure [5] the statistical mean is compatible with the results of hand analysis equations where we set A=4, the unipolar NRZ mean is 2, and Polar NRZ, RZ mean is zero.

XII. Q3: Calculating Statistical Autocorrelation

Hand analysis Unipolar NRZ:

$$R_X(\tau) = E\big(X(t) * X(t+\tau)\big)$$

$$E\big(X(t) * X(t+\tau)\big) = A^2 * P(A,A) + 0 * A * P(0,A) + A * 0 * P(A,0) + 0 * 0 * P(0,0)$$

$$= A^2 * P(A,A)$$

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

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

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

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

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

For $\underline{\tau < T}$

$$R_X(\tau) = \frac{A^2}{2} * (1 - \frac{\tau}{2T})$$

For $\underline{\tau > T}$

$$R_X(\tau) = A^2 * p(A,A) = A^2 * \frac{1}{2} * \frac{1}{2} = \frac{A^2}{4}$$

So $R_X(\tau) = \frac{A^2}{2} * \left(1 - \frac{\tau}{2T}\right) \quad |\tau| < T$

$= \frac{A^2}{4} \qquad |\tau| > T$

$R_X(0) = 8 \quad R_X(\infty) = 4$

Hand analysis Polar NRZ:

$R_X(\tau) = E\big(X(t) * X(t+\tau)\big)$

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

$= A^2 * P(A,A) + A^2 * P(-A,-A) - A^2 * P(A,-A) - A^2 * P(-A,A).$

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

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

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

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

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

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

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

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

For $\underline{\tau < T}$

$$R_X(\tau) = 2 * A^2 * \left(1 - \frac{\tau}{2T}\right) * \frac{1}{2} - 2A^2 * \frac{\tau}{4T} = A^2 * \left(1 - \frac{\tau}{T}\right).$$

For $\underline{\tau > T}$

$P(x(t)), P(x(t+\tau))$ **are independent**

$$R_X(\tau) = A^2 * \frac{1}{4} + A^2 * \frac{1}{4} - A^2 * \frac{1}{4} - A^2 * \frac{1}{4} = zero.$$

So $R_X(\tau) = A^2 * \left(1 - \frac{\tau}{T}\right) \quad |\tau| < T$

$= zero \quad |\tau| > T$

$R_X(0) = 16 \quad R_X(\infty) = 0$

Hand analysis Polar RZ:

Hand analysis here is the same as polar RZ but with multiply by $^4/_7$

So $R_X(\tau) = \frac{4}{7} * A^2 * \left(1 - \frac{7*\tau}{4*T}\right) \quad |\tau| < T$

$= zero \quad |\tau| > T$

$R_X(0) = 16 * \frac{4}{7} = 9.1428 \quad R_X(\infty) = 0$

Therefore, the autocorrelation should peak when the time difference (τ) is zero because the signal correlates most strongly with itself at that point compared to any other time. As τ increases, the autocorrelation diminishes until it reaches a near-constant level.

XIII. Plotting Statistical Autocorrelation

```
% Function to calculate statistical autocorrelation
% Rx(τ) = E(X[n]*X[i])  where τ = n - i
function result = calc_stat_autocorr(x)
    global num_realizations;
    global num_bits;
    stat_autocorr = zeros(1, num_bits * 7);
    for i = 1:num_bits * 7
        stat_autocorr(i) = (1/num_realizations) * sum(x(:, 1) .* x(:, i));
    end
    result = [fliplr(stat_autocorr(2:end)) stat_autocorr];
end
```

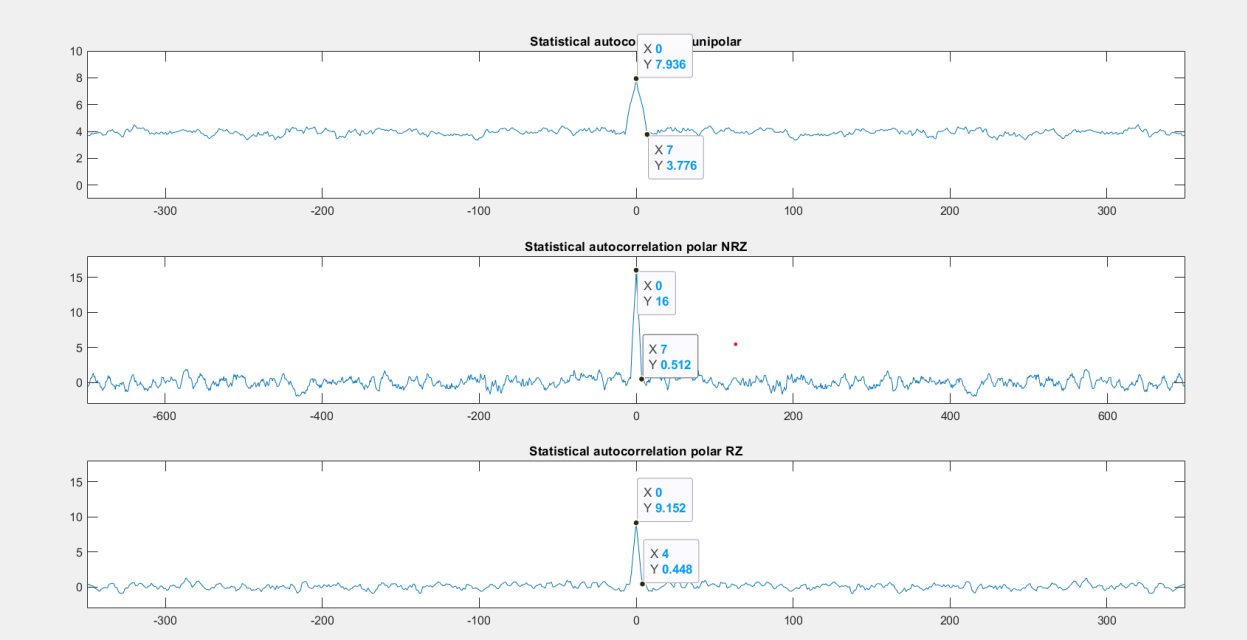

Figure 6: Statistical Autocorrelation.

As shown in graph the autocorrelation must be maximum at time difference (τ) equals to zero because the correlation of the signal and itself at any time t must be the highest comparing with the signal at time t and the signal at any other time instance, and by increasing the time difference (τ), the autocorrelation decreases till become approximately constant.

- for unipolar NRZ starts to be constant at $\tau = 7\ it\ means\ after\ 7\ samples\ which\ is\ the\ bit$ duration and the constant = 4 and have Maximum at $\tau = 0\ R_X(0) \cong 8.$

- For Polar NRZ starts to be constant at $\tau = 7\ it\ means\ after\ 7\ samples\ which\ is\ the\ bit\ duration$ and the constant = zero and have Maximum at $\tau = 0\ R_X(0) \cong 16.$

- For Polar RZ starts to be constant at $\tau = 4\ it\ means\ after\ 4\ samples\ which$ is bit duration $\frac{4}{7} * T$ and the constant = zero and have Maximum at $\tau = 0\ R_X(0) \cong 9.1428.$

XIV. Q2:Is the process Stationary?

From the previous hand analysis and graph we notice that:

•The mean is almost constant over the time.

	Unipolar	Polar NRZ	Polar RZ
Statistical mean	$\frac{A}{2}$	**Zero**	**Zero**

•Statistical autocorrelation is only function in the time difference.

	Unipolar	Polar NRZ	Polar RZ
$\lvert\tau\rvert < T$	$\frac{A^2}{2} * (1 - \frac{\tau}{2T})$	$A^2 * \left(1 - \frac{\tau}{T}\right)$	$\frac{4}{7} * A^2 * \left(1 - \frac{7 * \tau}{4 * T}\right)$
$\lvert\tau\rvert > T$	$\frac{A^2}{4}$	**Zero**	**Zero**

so conditions of WSS are valid so we can say the Unipolar NRZ, Polar NRZ, Polar RZ are WSS.

XV. Q4:Computing the Time mean and Autocorrelation of one waveform

Time mean

```
time_unipolar_mean = zeros(1, 500);
time_polar_NRZ_mean = zeros(1, 500);
time_polar_RZ_mean = zeros(1, 500);
for i = 1:500
    time_unipolar_mean(i) = sum(unipolar_Tx_out_new(i, :)) / 700;
    time_polar_NRZ_mean(i) = sum(polar_NRZ_Tx_out_new(i, :)) / 700;
    time_polar_RZ_mean(i) = sum(polar_RZ_Tx_out_final(i, :)) / 700;
end
```

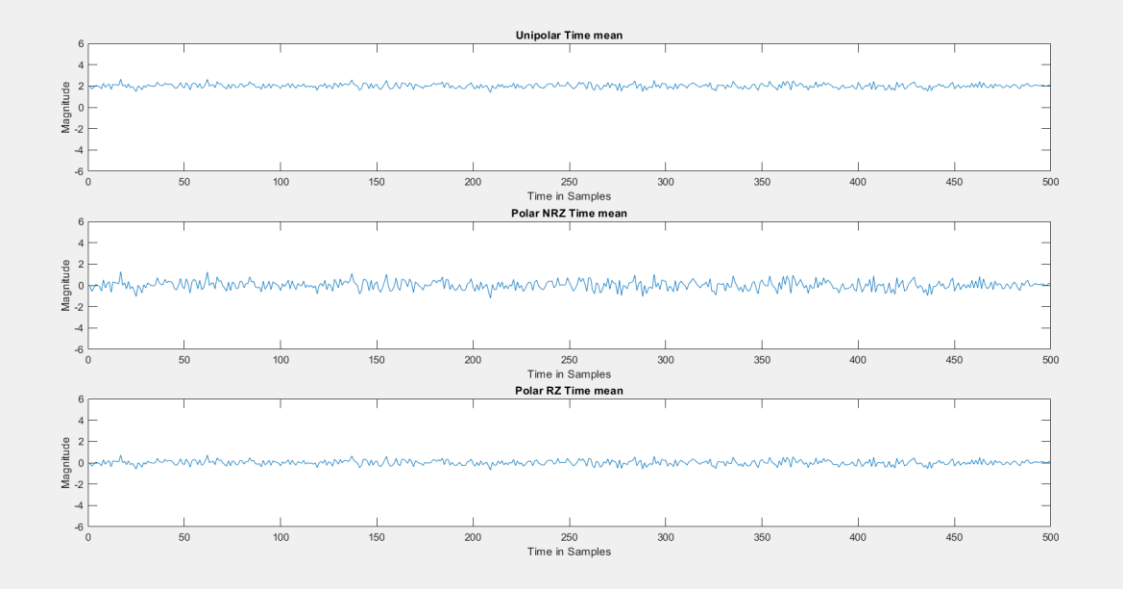

Figure 7: Line Code Time mean

We can calculate the Time mean using this formula

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

Unipolar:

$$\widetilde{x(t)} = \frac{1}{T}\left[\frac{T}{2} * A + \frac{T}{2} * 0\right] = \frac{A}{2}$$

Polar NRZ:

$$\widetilde{x(t)} = \frac{1}{T}\left[\frac{T}{2} * A + \frac{T}{2} * -A\right] = 0$$

Polar RZ:

$$\widetilde{x(t)} = \frac{1}{T}\left[\frac{T}{4} * A + \frac{T}{4} * -A\right] = 0$$

Time Autocorrelation

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

```
% Function to calculate time-domain autocorrelation
% mean across time for each τ value
function result = calc_time_autocorr(x)
    global num_bits;
    time_autocorr = zeros(1, num_bits * 7);
    for tau = 0:num_bits * 7 - 1
        sum_time_autocorr = 0;
        for counter = 1:(num_bits * 7 - tau)
            sum_time_autocorr = sum_time_autocorr + (x(1, counter) * x(1, (counter
+ tau)));
        end
        % Divide by the number of samples
        time_autocorr((tau+1)) = (1/(num_bits * 7 - tau)) * sum_time_autocorr;
    end
     result = [fliplr(time_autocorr(2:end)) time_autocorr];
```

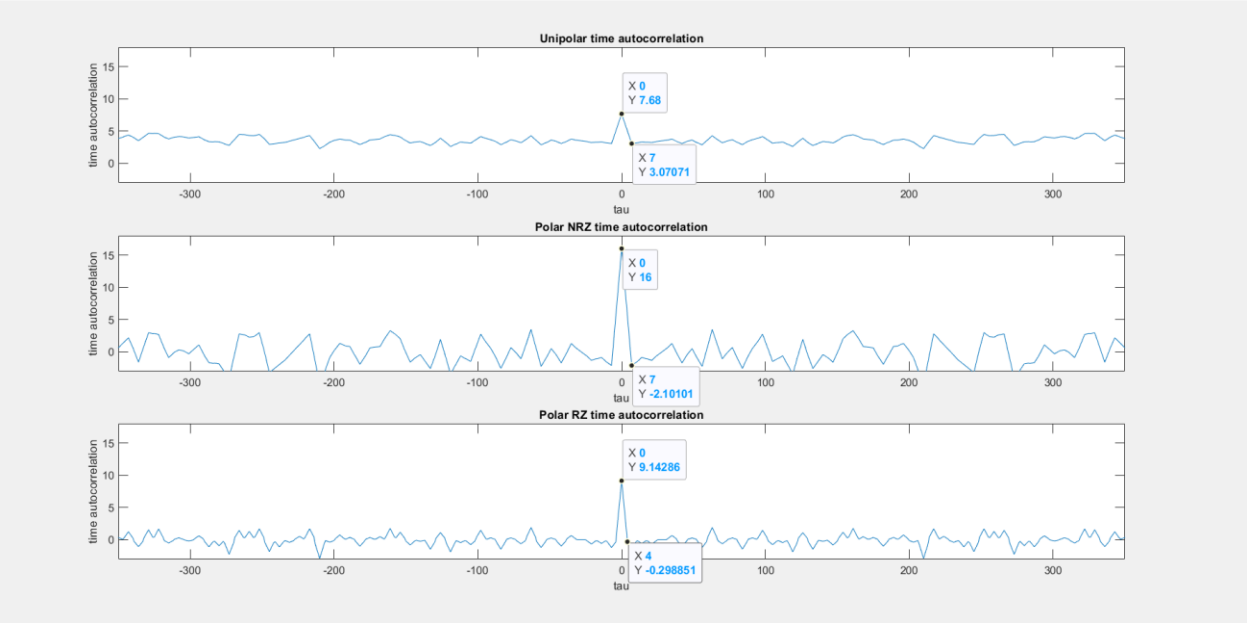

Figure 8: Time Autocorrelation.

- As shown on the graph the time autocorrelation is very similar to ensemble autocorrelation.

XVI. Q5:Is the random process ergodic?

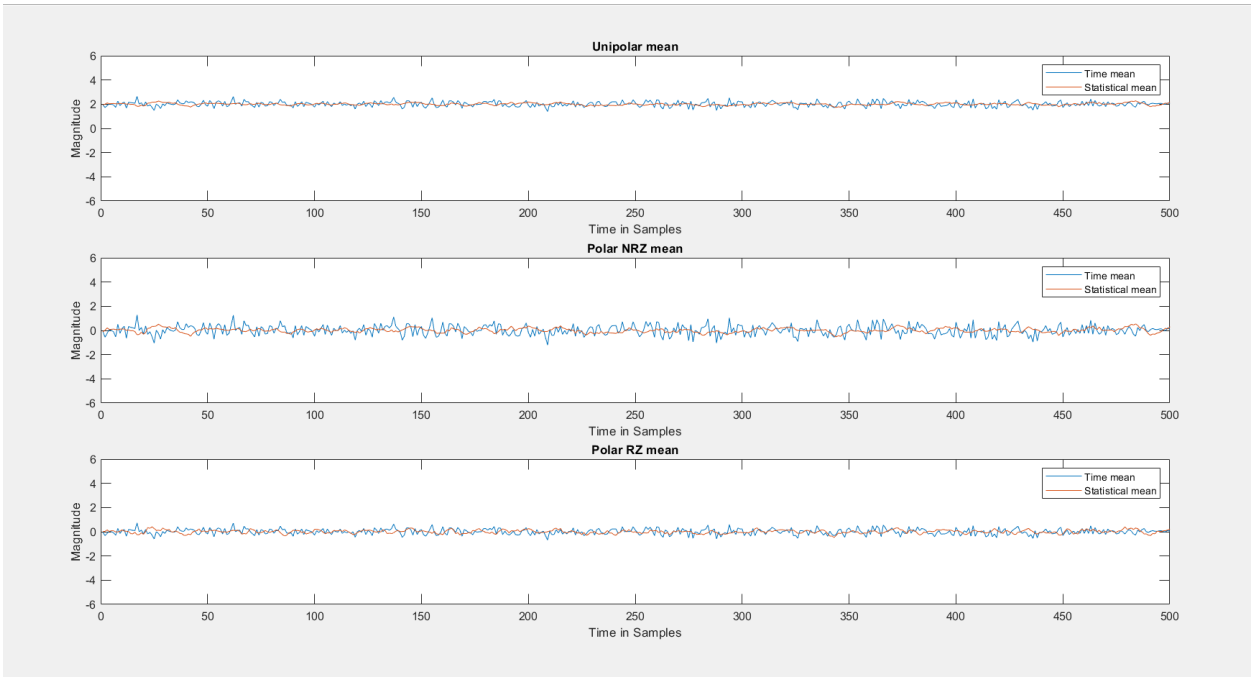

Figure 9: Time and Statistical mean overlapped

- The time mean is similar to statistical mean and both are approximately constant.

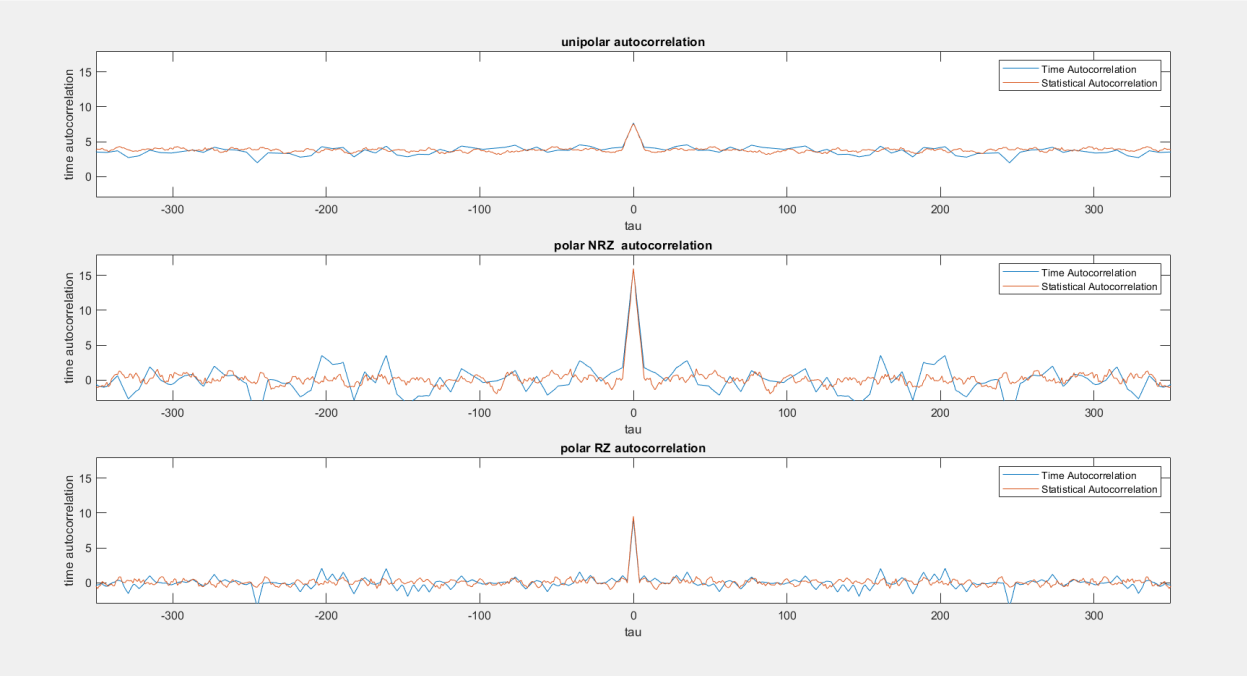

Figure 10: Time and Statistical Autocorrelation

•The time autocorrelation is similar to ensemble autocorrelation.

So the conditions of ergodic are valid so we can say the Unipolar NRZ, Polar NRZ, and Polar RZ are Ergodic processes. Note that the small change between time and ensemble mean or autocorrelation will disappear when the number of bits is equal to the number of realizations.

XVII. Plotting the PSD of the ensemble

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

- for Unipolar NRZ $S_{XX}(w) = \frac{A^2}{4}\left(Tsinc^2\left(\frac{wT}{2}\right) + \delta(t)\right)$.
- for Polar NRZ $S_{XX}(w) = A^2 Tsinc^2\left(\frac{wT}{2}\right)$
- for Polar RZ $S_{XX}(w) = A^2 * \frac{16}{49} Tsinc^2\left(\frac{wT}{2} * \frac{4}{7}\right)$

The simulated results are found in Figure [11].

Line code	Calculated at A f=0	Simulated A at f=0
Unipolar NRZ	$\boldsymbol{A = \frac{A^2}{4} * T = 0.28}$	**0.298**
Polar NRZ	$\boldsymbol{A = A^2 T = 1.12}$	**1.107**
Polar RZ	$\boldsymbol{A = A^2 * \frac{16}{49} T = 0.365}$	**0.27**

After observing the output of FFT function the amplitudes of PSD was very large compared to the expected values so we normalized the amplitude of PSD by dividing by sampling frequency (Fs) to get the correct amplitude values.

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

fft_polar_NRZ = abs(fft(polar_NRZ_autocorr)/fs);
fft_unipolar = abs(fft(unipolar_autocorr)/fs);
fft_polar_RZ = abs(fft(polar_RZ_autocorr)/fs);

N = length(polar_NRZ_autocorr); % Number of samples
f = (-N/2:N/2-1) * fs/N; % Frequency axis

% Plot Power Spectral Density
figure;
subplot(3, 1, 1);
plot(f, fftshift(fft_unipolar));
title('FFT of unipolar autocorrelation');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```

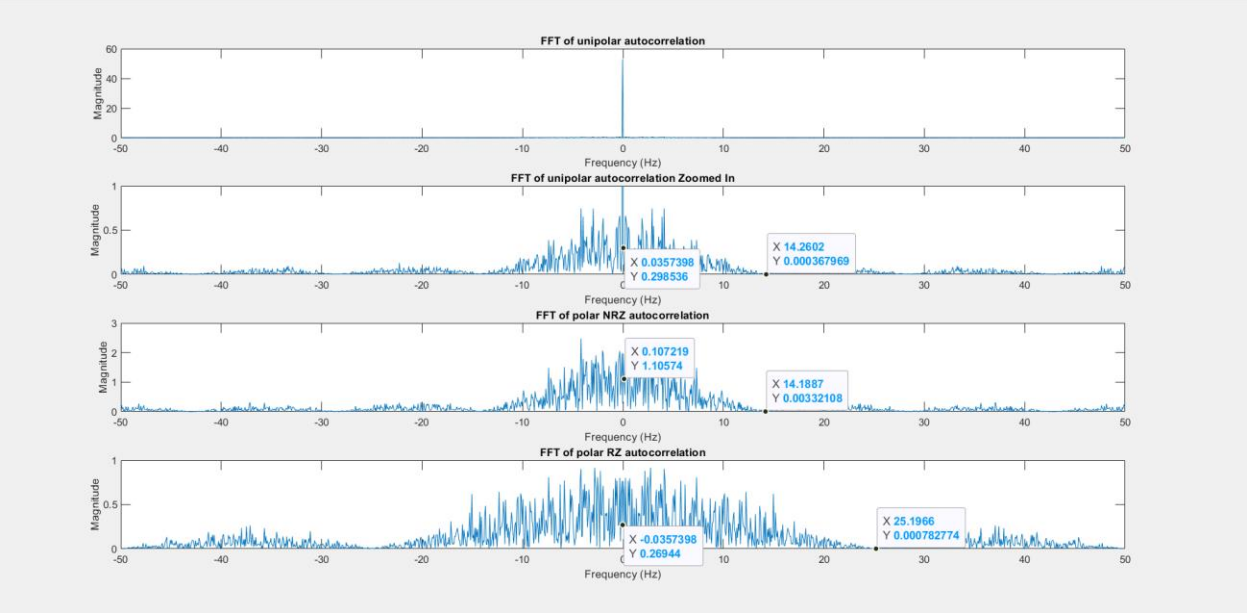

Figure 11: PSD Plot

The results of amplitudes and bandwidth was acceptable compared to exact values but we can improve this results by increasing number of realizations as shown in the last section.

XVIII. Q6: What is the bandwidth of the transmitted signal?

Line code	Calculated	Simulated
Unipolar NRZ	$BW = bit\ rate = \frac{1}{T} = 14.285HZ.$	14.2602Hz
Polar NRZ	$BW = bit\ rate = \frac{1}{T} = 14.285HZ.$	14.2602Hz
Polar RZ	$BW = bit\ rate = \frac{1}{T} * \frac{7}{4} = 25HZ.$	25.0536Hz

XIX. Test Case: Increasing number of bits and Realizations

Note: by increasing the number of samples, the accuracy will increase, and the time statistics will be closer to the statistics of the whole ensemble. Also, by increasing the number of realizations, the statistics will be more accurate, so the time statistics and the ensemble statistics will be the same. so we will test with Number of bits=1000 and realizations=5000.

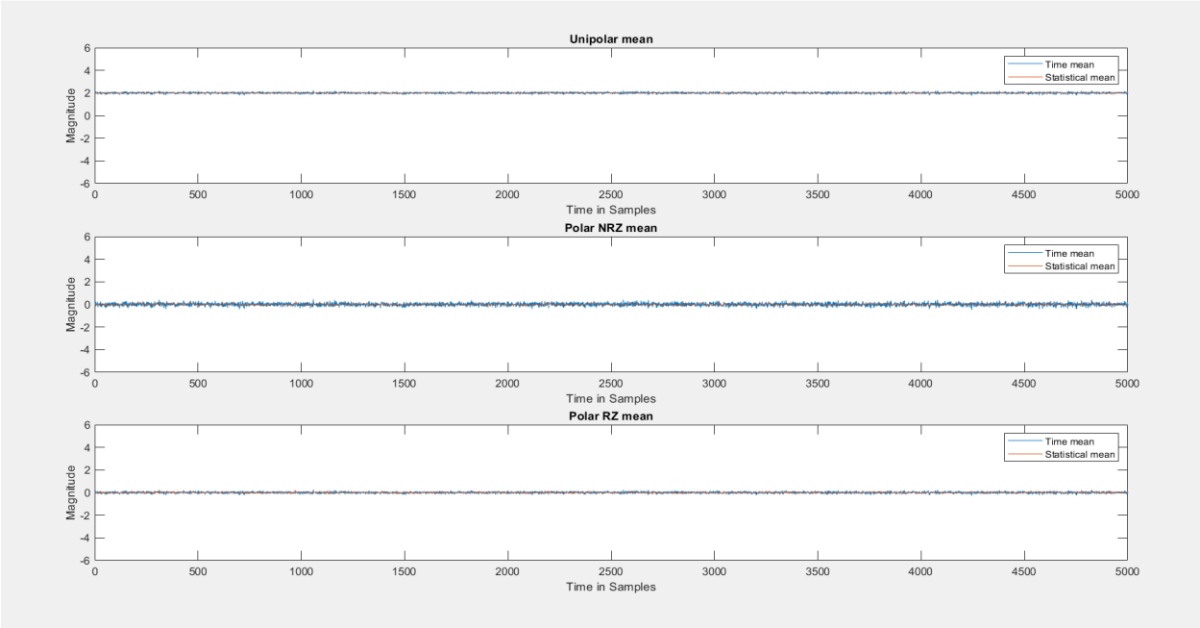

Figure 12: Time and Statistical mean with large number of realizations.

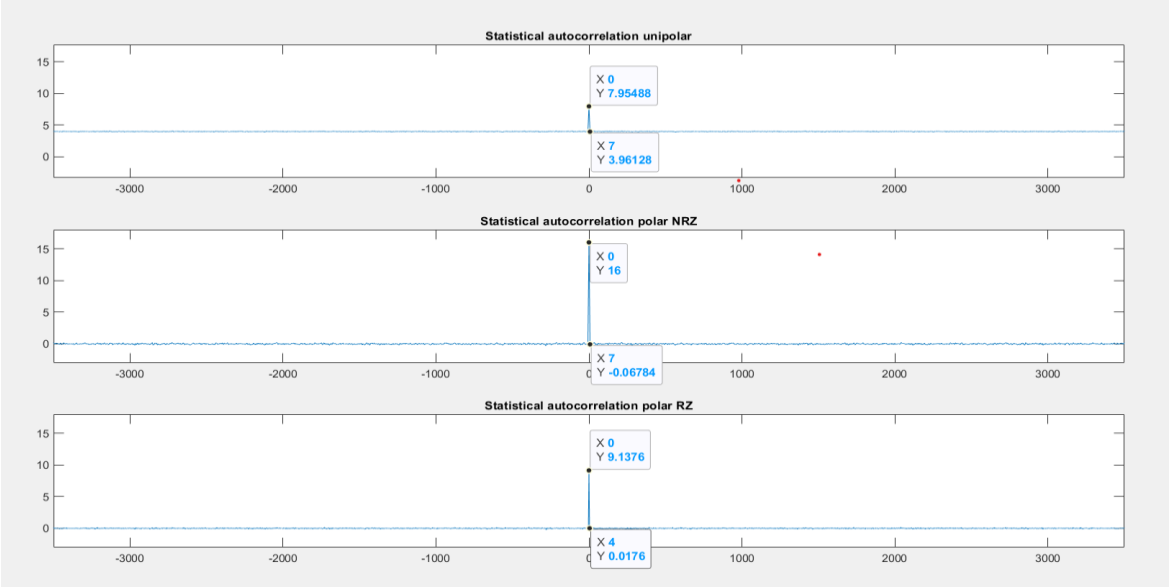

Figure 13: Statistical autocorrelation with large number of realizations

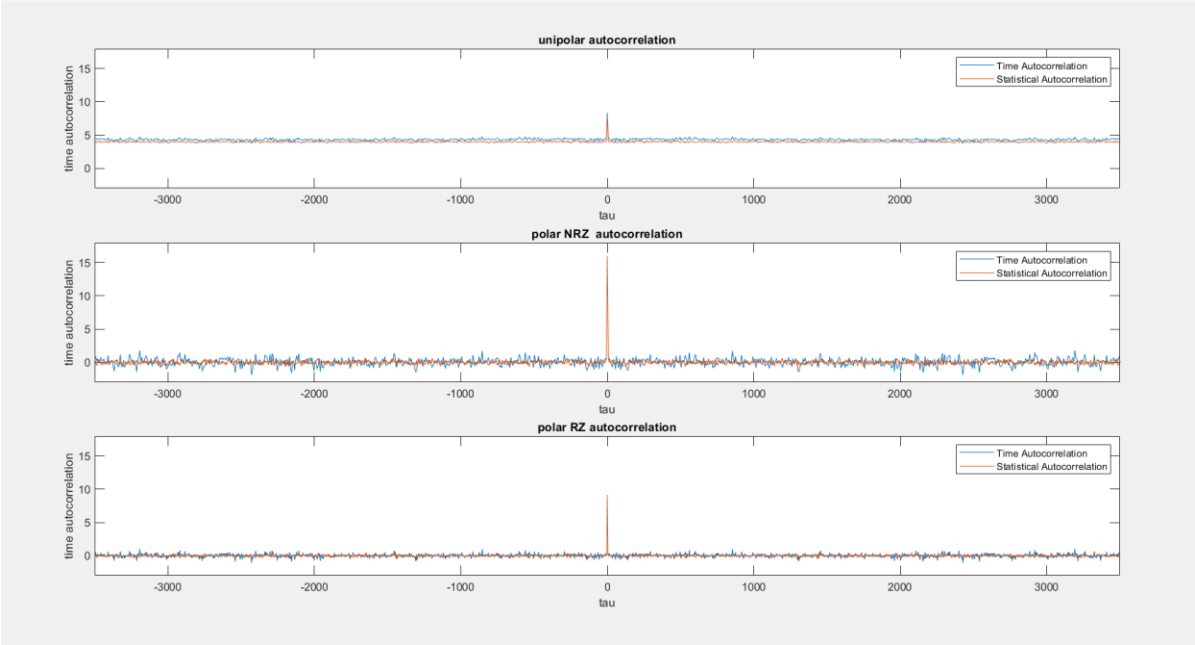

Figure 14: Time and Statistical autocorrelation with large number of realizations

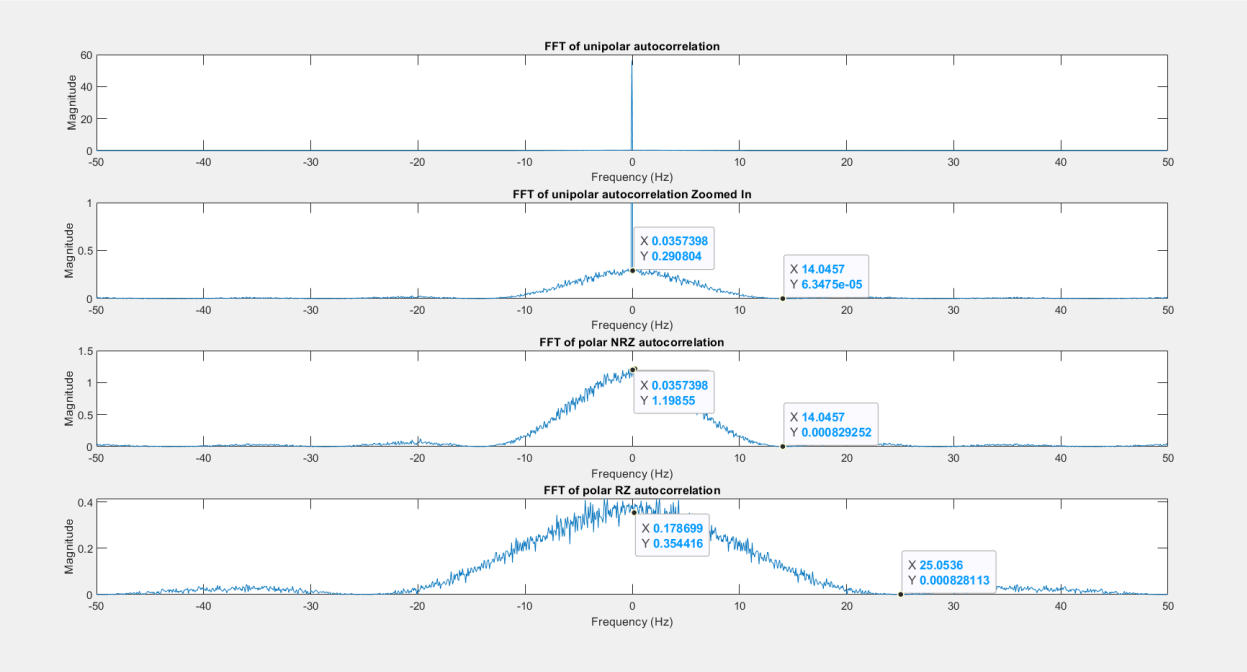

Figure 15: PSD plot with large number of realizations

- The time mean and statical mean are approximately the same and constant. Figure [12]

- Statistical autocorrelation for unipolar is start at $\tau = 0$ with the maximum and decrease until be constant=4, the time autocorrelation is the same statical approximately and varying around 4. Figure [13],[14]

- Statistical autocorrelation for polar NRZ is start at $\tau = 0$ with the maximum and decrease until be constant=zero, the time autocorrelation is the same Statistical approximately and varying around zero. Figure [13],[14]

- Statistical autocorrelation for polar RZ is start at $\tau = 0$ with the maximum and decrease until be constant=zero, the time autocorrelation is the same statical approximately and varying around zero. Figure [13],[14]

- The graph shows that the Power Spectral Density (PSD) on it matches what we calculated in theory, and it has the same bandwidth as our calculations predicted. We can also reach the same result by Averaging the autocorrelation with every possible t1. Figure [15]

Amplitudes at f=0 in case of 5000 realizations became more closer to the exact amplitudes.

Line code	Calculated A at f=0	Simulated A at f=0
Unipolar NRZ	$\boldsymbol{A = \frac{A^2}{4} * T = 0.28}$	**0.29**
Polar NRZ	$\boldsymbol{A = A^2 T = 1.12}$	**1.198**
Polar RZ	$\boldsymbol{A = A^2 * \frac{16}{49} T = 0.365}$	**0.354**

XX. Full matlab Code

```
close all;
clear;
clc;
global num_bits;
global num_realizations;

% Define scaling factor A, number of bits, and number of realizations
A = 4;
num_bits = 100;
num_realizations = 500;

% Generate random binary data and transmission delays
Data = randi([0, 1], num_realizations, num_bits + 1);
td = randi([0, 6], num_realizations, 1);

% Unipolar Encoding
unipolar_Tx = Data * A;
unipolar_Tx_out = repelem(unipolar_Tx, 1, 7);  % Repeat each element 7 times
unipolar_Tx_out_new = zeros(num_realizations, num_bits * 7);
for i = 1:num_realizations
    % Extract the relevant portion based on transmission delays
    unipolar_Tx_out_new(i, :) = unipolar_Tx_out(i, td(i) + 1:td(i) + num_bits * 7);
end

% Polar NRZ Encoding
polar_NRZ_Tx = ((2 * Data) - 1) * A;
polar_NRZ_Tx_out = repelem(polar_NRZ_Tx, 1, 7); % Repeat each element 7 times
polar_NRZ_Tx_out_new = zeros(num_realizations, num_bits * 7);
for i = 1:num_realizations
    % Extract the relevant portion based on transmission delays
    polar_NRZ_Tx_out_new(i, :) = polar_NRZ_Tx_out(i, td(i) + 1:td(i) + num_bits * 7);
end

% Polar RZ Encoding
polar_RZ_Tx_out = repelem(polar_NRZ_Tx, 1, 4);  % Repeat each element 4 times
% Add three zeros after every set of four elements
polar_RZ_Tx_out_new = zeros(num_realizations, num_bits * 7 + 7);
polar_RZ_Tx_out_final = zeros(num_realizations, num_bits * 7);
for i = 1:num_realizations
    n = 0;
    for j = 1:4:num_bits*4 +4
        % Insert 3 zeros after every set of four elements
        polar_RZ_Tx_out_new(i, j + 3 * n:j + 3 + 3 * n) = polar_RZ_Tx_out(i, j:j + 3);
        n = n + 1;
    end
end
for i = 1:num_realizations
    % Extract the relevant portion based on transmission delays
    polar_RZ_Tx_out_final(i, :) = polar_RZ_Tx_out_new(i, td(i) + 1:td(i) + num_bits *
7);
end
figure
for i=1:5
subplot(5,1,i)
stairs(unipolar_Tx_out(i,1:num_bits * 7))
title(sprintf("Unipolar Output Realization %i",i));
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);
end
figure
```

```
for i=1:5
subplot(5,1,i)
stairs(polar_RZ_Tx_out_new(i,1:num_bits * 7))
title(sprintf("Polar RZ Output Realization %i",i));
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);
end
figure
for i=1:5
subplot(5,1,i)
stairs(polar_NRZ_Tx_out(i,1:num_bits * 7))
title(sprintf("Polar NRZ Output Realization %i",i));
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);
end
% Visualize encoded signals
figure;
subplot(3,1,1)
stairs(unipolar_Tx_out_new(1,1:num_bits * 7))
title("Unipolar Output");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);
subplot(3,1,2)
stairs(polar_NRZ_Tx_out_new(1,1:num_bits * 7))
title("polar NRZ Output");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);
subplot(3,1,3)
stairs(polar_RZ_Tx_out_final(1,1:num_bits * 7))
title("polar RZ Output");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);
% Calculate The statistical mean for each line code
stat_unipolar_mean = zeros(1, num_bits * 7);
stat_polar_NRZ_mean = zeros(1, num_bits * 7);
stat_polar_RZ_mean = zeros(1, num_bits * 7);
for i = 1:num_bits * 7
    stat_unipolar_mean(i) = sum(unipolar_Tx_out_new(:, i)) / num_realizations;
    stat_polar_NRZ_mean(i) = sum(polar_NRZ_Tx_out_new(:, i)) / num_realizations;
    stat_polar_RZ_mean(i) = sum(polar_RZ_Tx_out_final(:, i)) / num_realizations;
end

% Calculate mean across time for each realization
time_unipolar_mean = zeros(1, num_realizations);
time_polar_NRZ_mean = zeros(1, num_realizations);
time_polar_RZ_mean = zeros(1, num_realizations);
for i = 1:num_realizations
    time_unipolar_mean(i) = sum(unipolar_Tx_out_new(i, :)) / (num_bits * 7);
    time_polar_NRZ_mean(i) = sum(polar_NRZ_Tx_out_new(i, :)) / (num_bits * 7);
    time_polar_RZ_mean(i) = sum(polar_RZ_Tx_out_final(i, :)) / (num_bits * 7);
end

% Plot the statistical mean signal
figure;
subplot(3,1,1)
plot(stat_unipolar_mean)
title("Unipolar Statistical mean");
xlabel("Time in Samples");
```

```
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);
subplot(3,1,2)
plot(stat_polar_NRZ_mean)
title("Polar NRZ Statistical mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);
subplot(3,1,3)
plot(stat_polar_RZ_mean)
title("Polar RZ Statistical mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_bits * 7 -6 6]);

% Plot mean signal strength over time
figure;
subplot(3,1,1)
plot(time_unipolar_mean)
title("Unipolar Time mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_realizations -6 6]);
subplot(3,1,2)
plot(time_polar_NRZ_mean)
title("Polar NRZ Time mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_realizations -6 6]);
subplot(3,1,3)
plot(time_polar_RZ_mean)
title("Polar RZ Time mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_realizations -6 6]);

%plotting time and statistical mean overlapped
figure;
subplot(3,1,1)
plot(time_unipolar_mean)
hold on;
plot(stat_unipolar_mean)
legend ("Time mean","Statistical mean")
title("Unipolar mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_realizations -6 6]);
subplot(3,1,2)
plot(time_polar_NRZ_mean)
hold on;
plot(stat_polar_NRZ_mean)
legend ("Time mean","Statistical mean")
title("Polar NRZ mean");
xlabel("Time in Samples");
ylabel("Magnitude");
axis([0 num_realizations -6 6]);
subplot(3,1,3)
plot(time_polar_RZ_mean)
hold on;
plot(stat_polar_RZ_mean)
legend ("Time mean","Statistical mean")
title("Polar RZ mean");
xlabel("Time in Samples");
```

```
ylabel("Magnitude");
axis([0 num_realizations -6 6]);

% Calculate statistical autocorrelation
x_axis = [1:1:(num_bits * 7) * 2 - 1];
unipolar_autocorr = calc_stat_autocorr(unipolar_Tx_out_new);
polar_NRZ_autocorr = calc_stat_autocorr(polar_NRZ_Tx_out_new);
polar_RZ_autocorr = calc_stat_autocorr(polar_RZ_Tx_out_final);

% Calculate autocorrelation across time
unipolar_time_autocorr = calc_time_autocorr(unipolar_Tx_out_new);
polar_NRZ_time_autocorr = calc_time_autocorr(polar_NRZ_Tx_out_new);
polar_RZ_time_autocorr = calc_time_autocorr(polar_RZ_Tx_out_final);

% Plot statistical autocorrelation
figure;
subplot(3, 1, 1)
plot(x_axis - num_bits * 7, unipolar_autocorr(x_axis))
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('Statistical autocorrelation unipolar');
subplot(3, 1, 2)
plot(x_axis - num_bits * 7, polar_NRZ_autocorr(x_axis))
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('Statistical autocorrelation polar NRZ');
subplot(3, 1, 3)
plot(x_axis - num_bits * 7, polar_RZ_autocorr(x_axis))
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('Statistical autocorrelation polar RZ');

% Plot time-domain autocorrelation
figure;
subplot(3, 1, 1)
plot(x_axis - num_bits * 7, unipolar_time_autocorr(x_axis))
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('Unipolar time autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');
subplot(3, 1, 2)
plot(x_axis - num_bits * 7, polar_NRZ_time_autocorr(x_axis))
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('Polar NRZ time autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');
subplot(3, 1, 3)
plot(x_axis - num_bits * 7, polar_RZ_time_autocorr(x_axis))
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('Polar RZ time autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');

% Plot time-domain and statistical autocorrelation overlapped
figure;
subplot(3,1,1)
plot(x_axis - num_bits * 7,unipolar_time_autocorr(x_axis))
hold on;
plot(x_axis - num_bits * 7,unipolar_autocorr(x_axis))
legend ("Time Autocorrelation","Statistical Autocorrelation")
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('unipolar autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');
subplot(3,1,2)
plot(x_axis - num_bits * 7,polar_NRZ_time_autocorr(x_axis))
```

```
hold on;
plot(x_axis - num_bits * 7,polar_NRZ_autocorr(x_axis))
legend ("Time Autocorrelation","Statistical Autocorrelation")
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('polar NRZ  autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');
subplot(3,1,3)
plot(x_axis - num_bits * 7,polar_RZ_time_autocorr(x_axis))
hold on;
plot(x_axis - num_bits * 7,polar_RZ_autocorr(x_axis))
legend ("Time Autocorrelation","Statistical Autocorrelation")
axis([-num_bits * 7/2 num_bits * 7/2 -3 18])
title('polar RZ autocorrelation');
xlabel('tau');
ylabel('time autocorrelation');

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

fft_polar_NRZ = abs(fft(polar_NRZ_autocorr));
fft_unipolar = abs(fft(unipolar_autocorr));
fft_polar_RZ = abs(fft(polar_RZ_autocorr));

N = length(polar_NRZ_autocorr); % Number of samples
f = (-N/2:N/2-1) * fs/N; % Frequency axis

% Plot Power Spectral Density
figure;
subplot(4, 1, 1);
plot(f, fftshift(fft_unipolar)/fs);
title('FFT of unipolar autocorrelation');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
subplot(4, 1, 2);
plot(f, fftshift(fft_unipolar)/fs);
title('FFT of unipolar autocorrelation Zoomed In');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
ylim([0, 1]);
subplot(4, 1, 3);
plot(f, fftshift(fft_polar_NRZ)/fs);
title('FFT of polar NRZ autocorrelation');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
subplot(4, 1, 4);
plot(f, fftshift(fft_polar_RZ)/fs);
title('FFT of polar RZ autocorrelation');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Function to calculate statistical autocorrelation
% Rx(?) = mean(X[n]*X[i])  where ? = n - i
function result = calc_stat_autocorr(x)
    global num_realizations;
    global num_bits;
    stat_autocorr = zeros(1, num_bits * 7);
    for i = 1:num_bits * 7
        stat_autocorr(i) = (1/num_realizations) * sum(x(:, 1) .* x(:, i));
    end
    result = [fliplr(stat_autocorr(2:end)) stat_autocorr];
end
```

```
% Function to calculate time-domain autocorrelation
% mean across time for each ? value
function result = calc_time_autocorr(x)
    global num_bits;
    time_autocorr = zeros(1, num_bits * 7);
    for tau = 0:num_bits * 7 - 1
        sum_time_autocorr = 0;
        for counter = 1:(num_bits * 7 - tau)
            sum_time_autocorr = sum_time_autocorr + (x(1, counter) * x(1, (counter +
tau)));
        end
        % Divide by the number of samples
        time_autocorr((tau+1)) = (1/(num_bits * 7 - tau)) * sum_time_autocorr;
    end
    result = [fliplr(time_autocorr(2:end)) time_autocorr];
end
```