Digital communications

Project 1

Team number: 39

Name	Section	BN
عمرو عبد المتجلي احمد متولي	3	16
علياء عصام الدين نجيب محمد	3	6
محمد تامر محمود محمد	3	44
كريم بهجت عبد العظيم ابراهيم	3	27
محمد علي علي النجار	4	2

Table of Contents

1)Role of each member:

Name	Role
عمرو عبد المتجلي احمد متولي	• Contributed to writing report (inserting SSs from the code). • Calculating & plotting Statistical mean codes. • Contributed to writing Calculating statistical autocorrelation code. • Mapping polar RZ code. • Revised some of the code.
علياء عصام الدين نجيب محمد	• Calculating and plotting time autocorrelation and PSD. • Contributed in writing statistical autocorrelation code. • Write the code for the delay in the waveforms. • Contributed in writing report.
محمد تامر محمود محمد	• Calculated and plotted time mean • Generated polar NRZ waveforms • Generated unipolar waveforms • Checked the code • Contributed in writing report
كريم بهجت عبد العظيم ابراهيم	• Writing the Report. • Contributed to writing Mapping polar RZ code. • Revised some of the code
محمد علي علي النجار	• Contributed in writing statistical autocorrelation code. • Contributed in calculating and plotting time autocorrelation and PSD. • Contributed in writing report

2)Problem description

Implementation of software radio techniques. To convert the traditional radio hardware issues into software problems.

3)Introduction

Unlike conventional radio systems, which use lots of hardware like analog circuits and mixtures of analog and digital parts, software radio uses software to decide how to send signals and understand incoming ones. This makes it much more flexible and easier to control things like how the signals are changed and encoded.

4)Control flags

- A: Constant representing the amplitude of the waveforms.
- Number_of_waveforms: Number of waveforms to generate.
- Number_of_bits: Number of bits in each waveform.
- polar_NRZ_Ensemble, polar_RZ_Ensemble, unipolar_Ensemble: Matrices to store the generated waveforms for polar NRZ, polar RZ, and unipolar encoding, respectively.
- polar_NRZ_delays, polar_RZ_delays, unipolar_delays: Arrays storing random delays for each waveform for polar NRZ, polar RZ, and unipolar encoding, respectively.
- polar_NRZ_start_indices, polar_RZ_start_indices, unipolar_start_indices: Arrays storing random start indices for each waveform for polar NRZ, polar RZ, and unipolar encoding, respectively.
- polar_NRZ_Waveform, polar_RZ_Waveform, unipolar_Waveform: Arrays representing randomly generated bit sequences for polar NRZ, polar RZ, and unipolar encoding, respectively.
- polar_NRZ_Tx1, polar_RZ_Tx1, unipolar_Tx1: Arrays representing the initial transmitted signals for polar NRZ, polar RZ, and unipolar encoding, respectively.
- polar_NRZ_Tx, polar_RZ_Tx, unipolar_Tx: Arrays representing the repeated transmitted signals for polar NRZ, polar RZ, and unipolar encoding, respectively.

- polar_RZ_plot_sequence, polar_NRZ_plot_sequence, unipolar_plot_sequence: Arrays representing the plot sequences for polar RZ, polar NRZ, and unipolar encoding, respectively.
- polar_RZ_statistical_mean, polar_NRZ_statistical_mean, unipolar_statistical_mean: Arrays representing the statistical mean of the waveforms for polar RZ, polar NRZ, and unipolar encoding, respectively.
- polar_RZ_time_mean, polar_NRZ_time_mean, unipolar_time_mean: Arrays representing the time mean of the waveforms for polar RZ, polar NRZ, and unipolar encoding, respectively.
- polar_RZ_stat_autocorr, polar_NRZ_stat_autocorr, unipolar_stat_autocorr: Arrays representing the statistical autocorrelation of the waveforms for polar RZ, polar NRZ, and unipolar encoding, respectively.
- polar_RZ_avg_autocorr, polar_NRZ_avg_autocorr, unipolar_avg_autocorr: Arrays representing the average autocorrelation across all realizations for polar RZ, polar NRZ, and unipolar encoding, respectively.
- polar_RZ_PSD, polar_NRZ_PSD, unipolar_PSD: Arrays representing the power spectral density of the waveforms for polar RZ, polar NRZ, and unipolar encoding, respectively.
- polar_RZ_autocorrelation, polar_NRZ_autocorrelation, unipolar_autocorrelation: Arrays representing the autocorrelation of the waveforms for polar RZ, polar NRZ, and unipolar encoding, respectively.
- polar_RZ_autocorr_values, polar_NRZ_autocorr_values, unipolar_autocorr_values: Arrays representing the autocorrelation values for polar RZ, polar NRZ, and unipolar encoding, respectively.

These variables are used to generate, analyze, and visualize various properties of polar RZ, polar NRZ, and unipolar waveforms in a communication system

5)Generation of data

```
% Initialize a [Number_of_waveforms X Number_of_bits] empty matrix
polar_NRZ_Ensemble = zeros(Number_of_waveforms,(Number_of_bits+1)*7);
polar_RZ_Ensemble = zeros(Number_of_waveforms,(Number_of_bits+1)*7);
unipolar_Ensemble = zeros(Number_of_waveforms,(Number_of_bits+1)*7);
```

```
for i= 1:Number_of_waveforms

    polar_NRZ_start_index = polar_NRZ_start_indices(i);
    polar_RZ_start_index = polar_RZ_start_indices(i);
    unipolar_start_index = unipolar_start_indices(i);
    polar_NRZ_delay = polar_NRZ_delays(i);
    polar_RZ_delay = polar_RZ_delays(i);
    unipolar_delay = unipolar_delays(i);
    polar_NRZ_Waveform = randi([0, 1], 1, Number_of_bits+1); % Generate a random vector of 100 elements where each element is either 0 or 1
    polar_RZ_Waveform = randi([0, 1], 1, Number_of_bits+1);
    unipolar_Waveform = randi([0, 1], 1, Number_of_bits+1);
    polar_NRZ_Tx1=((2*polar_NRZ_Waveform)-1)*A; % maping for 0 to be -A, 1 to be A
    polar_NRZ_Tx=repelem(polar_NRZ_Tx1,7);
    unipolar_Tx1=unipolar_Waveform*A; % maping for 0 to be 0, 1 to be A
    unipolar_Tx=repelem(unipolar_Tx1,7);
    polar_RZ_Tx1=((2*polar_RZ_Waveform)-1)*A; % maping for 0 to be -A, 1 to be A
    polar_RZ_Tx=repelem(polar_RZ_Tx1,7);
    for k=4:7:(Number_of_bits+1)*7
        polar_RZ_Tx(k+1:k+3) = 0;
    end
```

Code generating data.

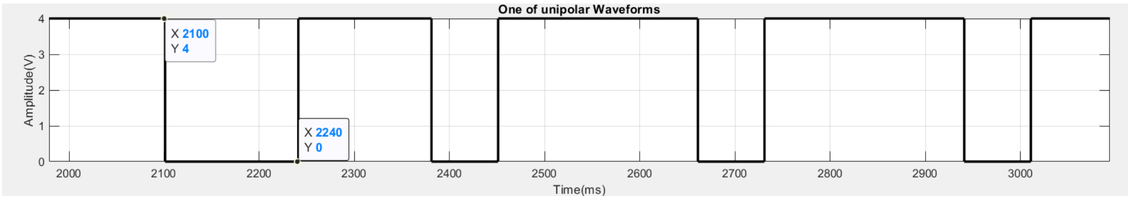
6)Creation of unipolar ensemble

```
19  for i= 1:Number_of_waveforms
48      % the above array contains 100 elements, corresponding to the levels that will be transmitted in every pulse
49      polar_NRZ_Ensemble(i, :) = polar_NRZ_shifted_bits; % every element represents 10 ms in all 3 ensembles
50      polar_RZ_Ensemble(i, :) = polar_RZ_shifted_bits;
51      unipolar_Ensemble(i, :) = unipolar_shifted_bits;
52  end
```

Line 51 ... also note that unipolar_shifted_bits is created in section 9

```
53  figure(1);
54  polar_RZ_plot_sequence = repelem(polar_RZ_Tx,1,10); %the vector plot sequence is a vector where each bit is represented as 70ms
55  polar_NRZ_plot_sequence = repelem(polar_NRZ_Tx,1,10);
56  unipolar_plot_sequence = repelem(unipolar_Tx,1,10);
57  subplot(3,1,1);
58  stairs(polar_RZ_plot_sequence, 'k', 'LineWidth', 2);
59  xlim([0, (Number_of_bits+1)*70]);
60  title('One of polar RZ Waveforms');
61  ylabel('Amplitude(V)');
62  xlabel('Time(ms)');
63  grid on;
64  subplot(3,1,2);
65  stairs(polar_NRZ_plot_sequence, 'k', 'LineWidth', 2);
66  xlim([0, (Number_of_bits+1)*70]);
67  title('One of polar NRZ Waveforms');
68  ylabel('Amplitude(V)');
69  xlabel('Time(ms)');
70  grid on;
71  subplot(3,1,3);
72  stairs(unipolar_plot_sequence, 'k', 'LineWidth', 2);
73  xlim([0, (Number_of_bits+1)*70]);
74  title('One of unipolar Waveforms');
75  ylabel('Amplitude(V)');
76  xlabel('Time(ms)');
77  grid on;
```

Code for plotting one of unipolar waveforms without delay.



One of unipolar waveforms(zoomed)

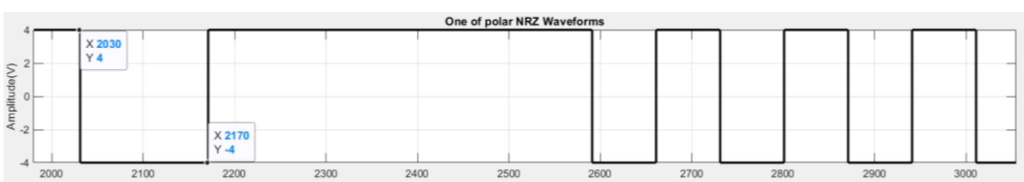
7)Creation of polar NRZ ensemble

```
19 - for i= 1:Number_of_waveforms
48        % the above array contains 100 elements, corresponding to the levels that will be transmitted in every pulse
49 -      polar_NRZ_Ensemble(i, :) = polar_NRZ_shifted_bits; % every element represents 10 ms in all 3 ensembles
50 -      polar_RZ_Ensemble(i, :) = polar_RZ_shifted_bits;
51 -      unipolar_Ensemble(i, :) = unipolar_shifted_bits;
52 -  end
```

Line 49 ... also note that polar_NRZ_shifted_bits is created in section 9

```
figure(1);
polar_RZ_plot_sequence = repelem(polar_RZ_Tx,1,10); %the vector plot sequence is a vector where each bit is represented as 70ms
polar_NRZ_plot_sequence = repelem(polar_NRZ_Tx,1,10);
unipolar_plot_sequence = repelem(unipolar_Tx,1,10);
subplot(3,1,1);
stairs(polar_RZ_plot_sequence, 'k', 'LineWidth', 2);
xlim([0, (Number_of_bits+1)*70]);
title('One of polar RZ Waveforms');
ylabel('Amplitude(V)');
xlabel('Time(ms)');
grid on;
subplot(3,1,2);
stairs(polar_NRZ_plot_sequence, 'k', 'LineWidth', 2);
xlim([0, (Number_of_bits+1)*70]);
title('One of polar NRZ Waveforms');
ylabel('Amplitude(V)');
xlabel('Time(ms)');
grid on;
```

Code for plotting one of polar NRZ waveforms without delay.



One of polar NRZ waveforms(zoomed)

8)Creation of polar RZ ensemble

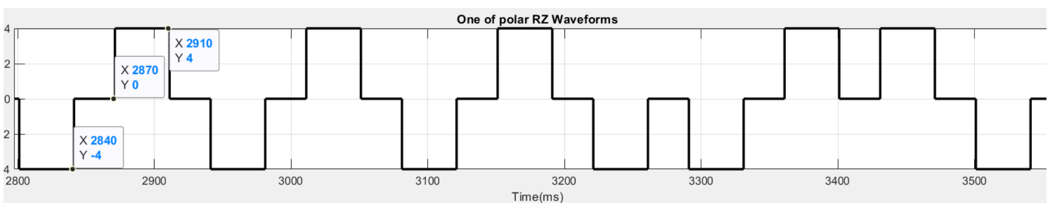
```
19 - for i= 1:Number_of_waveforms
```

```
48      % the above array contains 100 elements, corresponding to the levels that will be transmitted in every pulse
49 -    polar_NRZ_Ensemble(i, :) = polar_NRZ_shifted_bits; % every element represents 10 ms in all 3 ensembles
50 -    polar_RZ_Ensemble(i, :) = polar_RZ_shifted_bits;
51 -    unipolar_Ensemble(i, :) = unipolar_shifted_bits;
52 - end
```

Line 50 ... also note that polar_RZ_shifted_bits is created in section 9

```
53 - figure(1);
54 - polar_RZ_plot_sequence = repelem(polar_RZ_Tx,1,10); %the vector plot sequence is a vector where each bit is represented as 70ms
55 - polar_NRZ_plot_sequence = repelem(polar_NRZ_Tx,1,10);
56 - unipolar_plot_sequence = repelem(unipolar_Tx,1,10);
57 - subplot(3,1,1);
58 - stairs(polar_RZ_plot_sequence, 'k', 'LineWidth', 2);
59 - xlim([0, (Number_of_bits+1)*70]);
60 - title('One of polar RZ Waveforms');
61 - ylabel('Amplitude(V)');
62 - xlabel('Time(ms)');
63 - grid on;
```

Code for plotting one of polar RZ waveforms without delay.



One of polar RZ waveforms(zoomed)

9) Applying random initial time shifts for each waveform

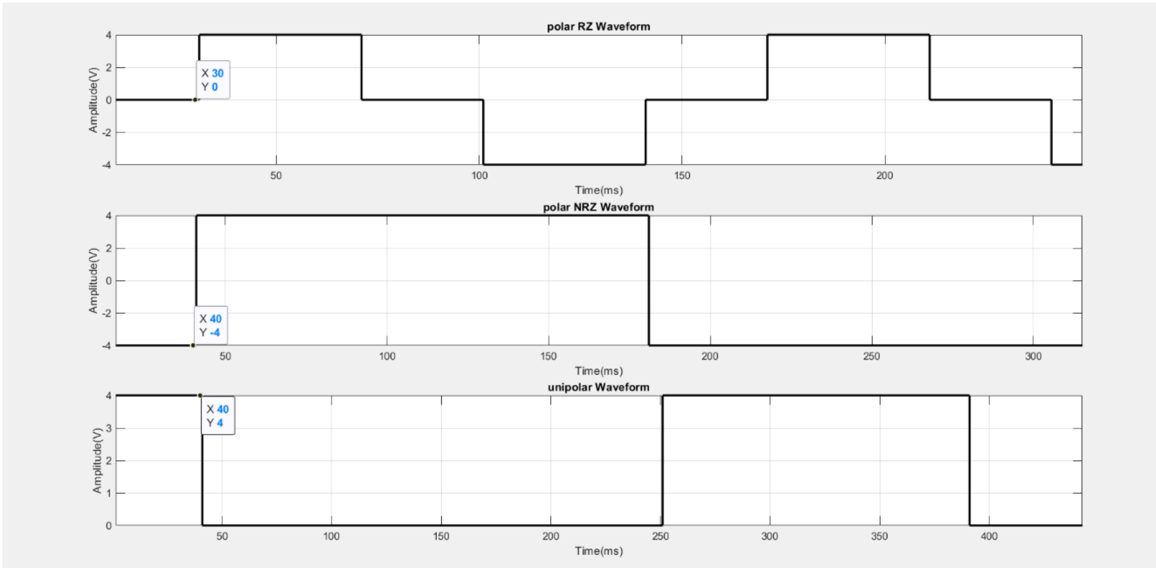
```
% Generate random delays for each sample
polar_NRZ_delays = randi([0, 6], 1, Number_of_waveforms);
polar_RZ_delays = randi([0, 6], 1, Number_of_waveforms);
unipolar_delays = randi([0, 6], 1, Number_of_waveforms);
% Generate random start indices based on delays
polar_NRZ_start_indices = randi([0, (Number_of_bits - 1)], 1, Number_of_waveforms);
polar_RZ_start_indices = randi([0, (Number_of_bits - 1)], 1, Number_of_waveforms);
unipolar_start_indices = randi([0, (Number_of_bits - 1)], 1, Number_of_waveforms);
for i= 1:Number_of_waveforms

    polar_NRZ_start_index = polar_NRZ_start_indices(i);
    polar_RZ_start_index = polar_RZ_start_indices(i);
    unipolar_start_index = unipolar_start_indices(i);
    polar_NRZ_delay = polar_NRZ_delays(i);
    polar_RZ_delay = polar_RZ_delays(i);
    unipolar_delay = unipolar_delays(i);
```

```
    % Shift the bits based on start index and delay
    polar_NRZ_shifted_bits = circshift(polar_NRZ_Tx, [0, -polar_NRZ_start_index]);
    polar_RZ_shifted_bits = circshift(polar_RZ_Tx, [0, -polar_RZ_start_index]);
    unipolar_shifted_bits = circshift(unipolar_Tx, [0, -unipolar_start_index]);
    % Discard the additional one bit
    polar_NRZ_sample_bits = polar_NRZ_shifted_bits(1:end-7);
    polar_RZ_sample_bits = polar_RZ_shifted_bits(1:end-7);
    unipolar_sample_bits = unipolar_shifted_bits(1:end-7);
```

Code for applying random initial time shifts for each waveform.

```
    unipolar_Ensemble(i, :) = unipolar_shifted_bits;
end
disp(polar_RZ_Ensemble);
figure(1);
polar_RZ_plot_sequence = repelem(polar_RZ_shifted_bits,1,10); %the vector plot sequence is a vector where each bit is represented as 70ms
polar_NRZ_plot_sequence = repelem(polar_NRZ_shifted_bits,1,10);
unipolar_plot_sequence = repelem(unipolar_shifted_bits,1,10);
subplot(3,1,1);
stairs(polar_RZ_plot_sequence, 'k', 'LineWidth', 2);
xlim([0, (Number_of_bits+1)*70]);
title('polar RZ Waveform');
ylabel('Amplitude(V)');
xlabel('Time(ms)');
grid on;
subplot(3,1,2);
stairs(polar_NRZ_plot_sequence, 'k', 'LineWidth', 2);
xlim([0, (Number_of_bits+1)*70]);
title('polar NRZ Waveform');
ylabel('Amplitude(V)');
xlabel('Time(ms)');
grid on;
subplot(3,1,3);
stairs(unipolar_plot_sequence, 'k', 'LineWidth', 2);
xlim([0, (Number_of_bits+1)*70]);
title('unipolar Waveform');
ylabel('Amplitude(V)');
xlabel('Time(ms)');
grid on;
```

Code for plotting waveforms with delay.



Showing delay

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

```
%% Statistical mean
polar_RZ_column_sum=zeros(1,700);
polar_NRZ_column_sum=zeros(1,700);
unipolar_column_sum=zeros(1,700);
polar_RZ_column_sum = sum(polar_RZ_Ensemble(:, 8));
polar_NRZ_column_sum = sum(polar_NRZ_Ensemble(:, 8));
unipolar_column_sum = sum(unipolar_Ensemble(:, 8));
for j= 9:(Number_of_bits+1)*7
    polar_RZ_column_sum = [polar_RZ_column_sum, sum(polar_RZ_Ensemble(:, j))];
    polar_NRZ_column_sum = [polar_NRZ_column_sum, sum(polar_NRZ_Ensemble(:, j))];
    unipolar_column_sum = [unipolar_column_sum, sum(unipolar_Ensemble(:, j))];

end
```

11) Calculating the statistical mean

```
%% Statistical mean
polar_RZ_column_sum=zeros(1,700);
polar_NRZ_column_sum=zeros(1,700);
unipolar_column_sum=zeros(1,700);
polar_RZ_column_sum = sum(polar_RZ_Ensemble(:, 8));
polar_NRZ_column_sum = sum(polar_NRZ_Ensemble(:, 8));
unipolar_column_sum = sum(unipolar_Ensemble(:, 8));
for j= 9:(Number_of_bits+1)*7
    polar_RZ_column_sum = [polar_RZ_column_sum, sum(polar_RZ_Ensemble(:, j))];
    polar_NRZ_column_sum = [polar_NRZ_column_sum, sum(polar_NRZ_Ensemble(:, j))];
    unipolar_column_sum = [unipolar_column_sum, sum(unipolar_Ensemble(:, j))];

end
polar_RZ_statistical_mean = polar_RZ_column_sum/Number_of_waveforms;
polar_NRZ_statistical_mean = polar_NRZ_column_sum/Number_of_waveforms;
unipolar_statistical_mean = unipolar_column_sum/Number_of_waveforms;
```

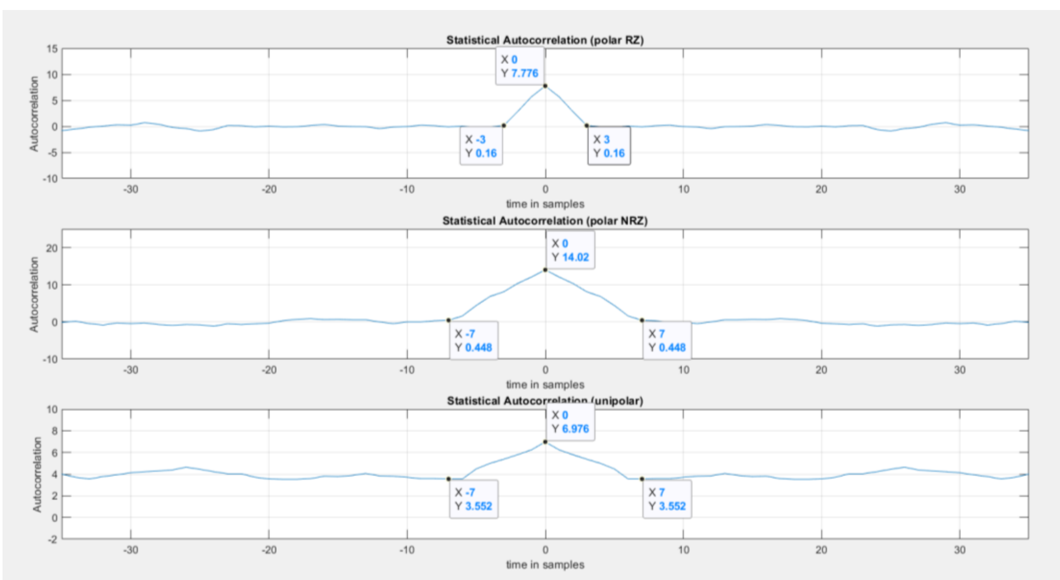
Code for calculating statistical mean.

Line code	Unipolar	Polar RZ	Polar NRZ
Analytical results	2	0	0
Simulated results	$\approx 2\ (as\ shown\ in\ graph\ below)$	$\approx 0\ (as\ shown\ in\ graph\ below)$	$\approx 0\ (as\ shown\ in\ graph\ below)$

12)Plotting the statistical mean

```
time = 1:Number_of_bits*7; % X-axis for plotting
figure(2);
subplot(3,1,1);
plot(time,polar_RZ_statistical_mean);
ylim([min(-4), max(4)]);
title('Polar RZ Statistical mean');
xlabel('Time (ms) (each sample corresponding to 10 ms)');
ylabel('Amplitude');
grid on;
subplot(3,1,2);
plot(time,polar_NRZ_statistical_mean);
ylim([min(-4), max(4)]);
title('Polar NRZ Statistical mean');
xlabel('Time (ms) (each sample corresponding to 10 ms)');
ylabel('Amplitude');
grid on;
subplot(3,1,3);
plot(time,unipolar_statistical_mean);
ylim([min(-4), max(4)]);
title('unipolar Statistical mean');
xlabel('Time (ms) (each sample corresponding to 10 ms)');
ylabel('Amplitude');
grid on;
```

Code for plotting the statistical mean



Plot for statistical mean

13) Calculating the statistical autocorrelation

```
%% Statistical Autocorrelation
polar_RZ_stat_autocorr = zeros(1,Number_of_bits*7);
polar_NRZ_stat_autocorr = zeros(1,Number_of_bits*7);
unipolar_stat_autocorr = zeros(1,Number_of_bits*7);

% Calculate autocorrelation for each realization
polar_RZ_Ensemble_without_delay_bit = polar_RZ_Ensemble(:, 8:end);
polar_NRZ_Ensemble_without_delay_bit = polar_NRZ_Ensemble(:, 8:end);
unipolar_Ensemble_without_delay_bit = unipolar_Ensemble(:, 8:end);
dimensions = size(polar_RZ_Ensemble_without_delay_bit);
center_index = (dimensions(2)/2)+1;

for lag = (1-center_index):(center_index-2)

    for i = 1:Number_of_waveforms
        polar_RZ_sample = polar_RZ_Ensemble_without_delay_bit(i, :);
        polar_NRZ_sample = polar_NRZ_Ensemble_without_delay_bit(i, :);
        unipolar_sample = unipolar_Ensemble_without_delay_bit(i, :);
        polar_RZ_stat_autocorr(lag+center_index) = polar_RZ_stat_autocorr(lag+center_index) + (polar_RZ_sample(center_index)*polar_RZ_sample(lag+center_index));
        polar_NRZ_stat_autocorr(lag+center_index) = polar_NRZ_stat_autocorr(lag+center_index) + (polar_NRZ_sample(center_index)*polar_NRZ_sample(lag+center_index));
        unipolar_stat_autocorr(lag+center_index) = unipolar_stat_autocorr(lag+center_index) + (unipolar_sample(center_index)*unipolar_sample(lag+center_index));

    end

end

% Calculate average autocorrelation across all realizations
polar_RZ_avg_autocorr = polar_RZ_stat_autocorr/Number_of_waveforms;
polar_NRZ_avg_autocorr = polar_NRZ_stat_autocorr/Number_of_waveforms;
```

Code for calculating statistical autocorrelation.

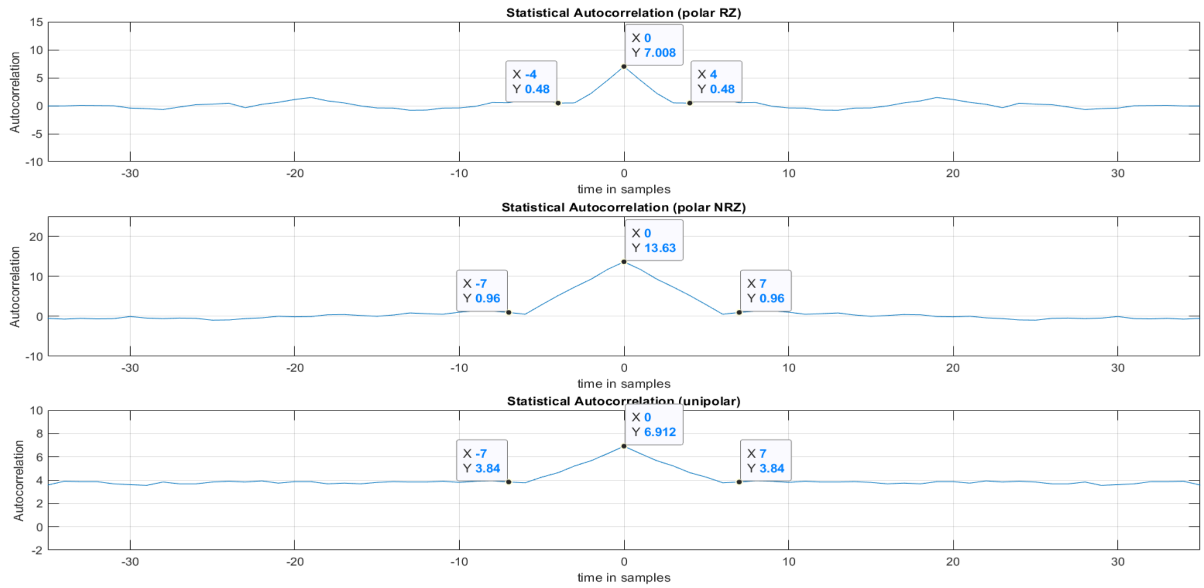
Analytical results: (nearly same as plots below), τ *is the* time shift, A is the amplitude (=4),T is the bit duration (=70 ms).

	Unipolar NRZ	Polar NRZ	Polar RZ
Autocorrelation	$R_x(\tau) = \frac{A^2}{2}\left(1 - \frac{\tau}{2T}\right), \lvert\tau\rvert < T$ $R_x(\tau) = \frac{A^2}{4}$, $\lvert\tau\rvert > T$	$R_x(\tau) = A^2(1 - \frac{\tau}{T}), \lvert\tau\rvert < T$ $R_x(\tau) = 0$, O.T	$R_x(\tau) = \frac{A^2}{2}(1 - \frac{2\tau}{T}), \lvert\tau\rvert < \frac{T}{2}$ $R_x(\tau) = 0$, O.T
Total Power	R (0) = 8	R (0) = 16	R (0) = 8
DC Power	R (∞) = 4	R (∞) = 0	R (∞) = 0

14)Plotting the statistical autocorrelation

```
%getting the zero lag at index zero
polar_RZ_avg_autocorr = circshift(polar_RZ_avg_autocorr, center_index-2);
polar_NRZ_avg_autocorr = circshift(polar_NRZ_avg_autocorr, center_index-2);
unipolar_avg_autocorr = circshift(unipolar_avg_autocorr, center_index-2);
%Flipping R_tau to the -ve quad
polar_RZ_autocorrelation = [fliplr(polar_RZ_avg_autocorr(2:end)) polar_RZ_avg_autocorr];
polar_NRZ_autocorrelation = [fliplr(polar_NRZ_avg_autocorr(2:end)) polar_NRZ_avg_autocorr];
unipolar_autocorrelation = [fliplr(unipolar_avg_autocorr(2:end)) unipolar_avg_autocorr];
figure(4)
time = -((Number_of_bits*7)-1):(Number_of_bits*7)-1;
subplot(3,1,1);
plot (time, polar_RZ_autocorrelation);
xlim([-35 35]);
ylim([min(-10), max(15)]);
xlabel("time in samples");
ylabel("Autocorrelation");
grid on;
title ("Statistical Autocorrelation (polar RZ)");
subplot(3,1,2);
plot (time, polar_NRZ_autocorrelation);
xlim([-35 35]);
ylim([min(-10), max(25)]);
xlabel("time in samples");
ylabel("Autocorrelation");
grid on;
title ("Statistical Autocorrelation (polar NRZ)");
subplot(3,1,3);
plot (time, unipolar_autocorrelation);
xlim([-35 35]);
ylim([min(-2), max(10)]);
xlabel("time in samples");
ylabel("Autocorrelation");
grid on;
title ("Statistical Autocorrelation (unipolar)");
```

Code for plotting statistical autocorrelation



Statistical Autocorrelation resulting graphs for each line code (note that X-axis is the time shift & each sample represent 10 ms)

15) Is the process stationary?

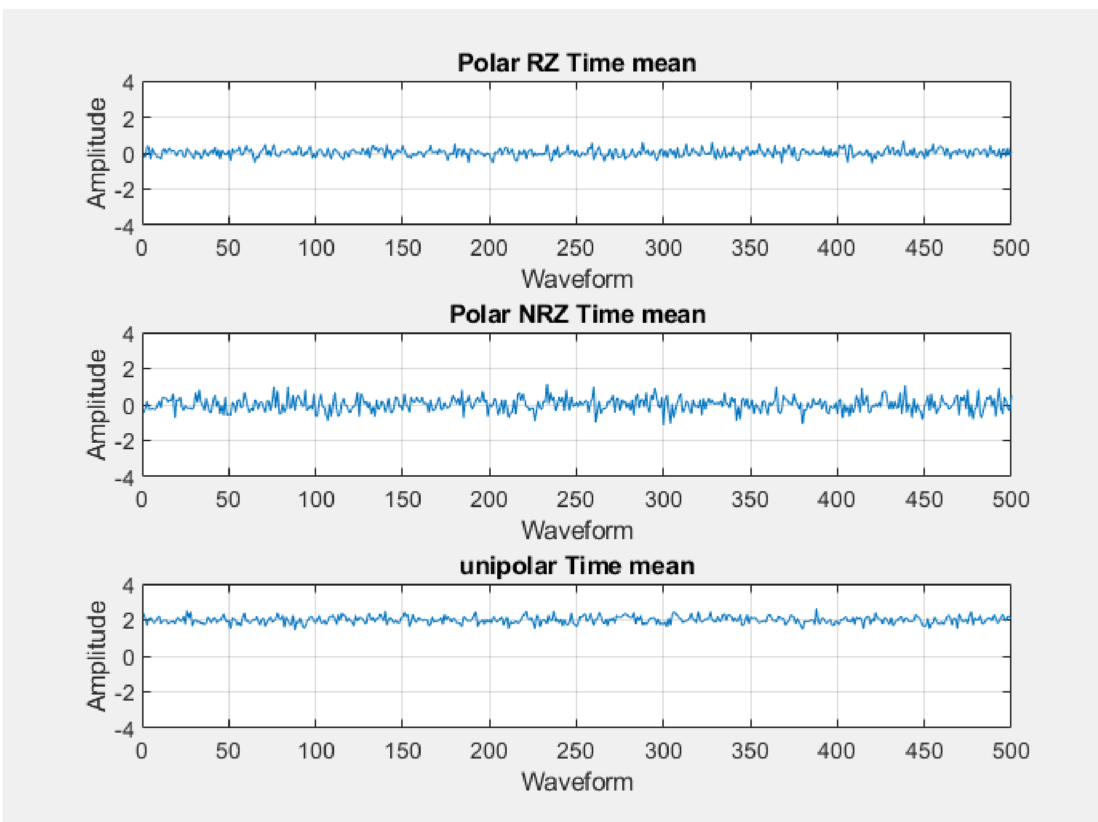
Answer:

Yes, the process can be considered a wide sense stationary process ;as the statistical mean is nearly constant as shown in the graphs, also it is obvious from the statistical autocorrelation calculation that it is function of time shift.

16)Computing the time mean

```
%% Time mean
%preallocating time mean matrices
polar_RZ_time_mean=zeros(Number_of_waveforms,1);
polar_NRZ_time_mean=zeros(Number_of_waveforms,1);
unipolar_time_mean=zeros(Number_of_waveforms,1);
polar_RZ_time_sum=zeros(Number_of_waveforms,1);
polar_NRZ_time_sum=zeros(Number_of_waveforms,1);
unipolar_time_sum=zeros(Number_of_waveforms,1);
%calculating mean of every realization
for i=1:Number_of_waveforms
    for j=1:Number_of_bits*7
    polar_RZ_time_sum(i,1)=polar_RZ_time_sum(i,1) + polar_RZ_Ensemble(i,j);
    polar_NRZ_time_sum(i,1)=polar_NRZ_time_sum(i,1) + polar_NRZ_Ensemble(i,j);
    unipolar_time_sum(i,1)=unipolar_time_sum(i,1) + unipolar_Ensemble(i,j);
    end
    polar_RZ_time_mean(i,1)=polar_RZ_time_sum(i,1)/(Number_of_bits*7);
    polar_NRZ_time_mean(i,1)=polar_NRZ_time_sum(i,1)/(Number_of_bits*7);
    unipolar_time_mean(i,1)=unipolar_time_sum(i,1)/(Number_of_bits*7);

end
```

Code for calculating time mean



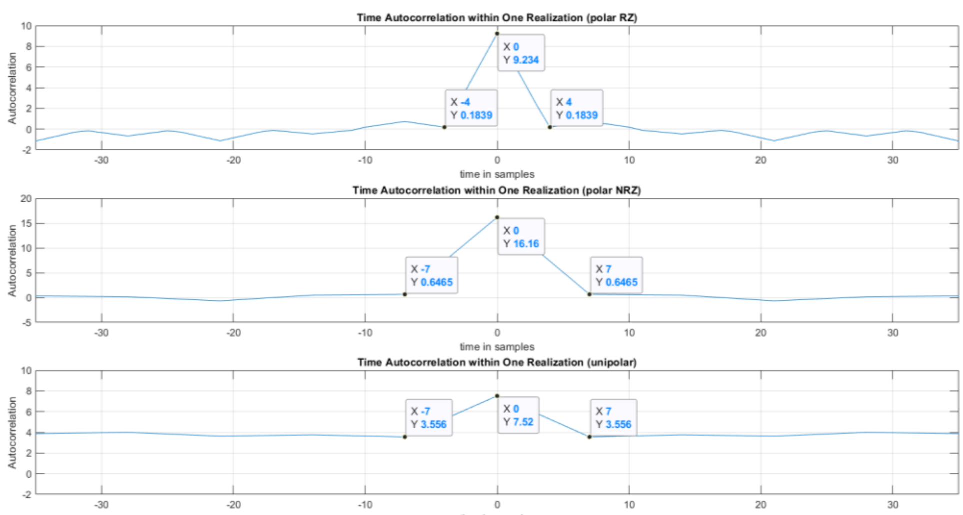
Time mean resulting graphs for each line code

Line code type	Polar RZ	Polar NRZ	unipolar
Ideally if infinite ensemble	$\frac{1}{T}(\frac{AT}{4}-\frac{AT}{4})=0$	$\frac{1}{T}(\frac{AT}{2}-\frac{AT}{2})=0$	$\frac{1}{T}(\frac{AT}{2}+0*\frac{AT}{2})=\frac{A}{2}$

17)Computing time autocorrelation of one waveform

```
%% Time Autocorrelation
% Preallocate array to store autocorrelation values
polar_RZ_autocorr_values = zeros(Number_of_bits*7, 1);
polar_NRZ_autocorr_values = zeros(Number_of_bits*7, 1);
unipolar_autocorr_values = zeros(Number_of_bits*7, 1);
% Calculate autocorrelation for the realization
time_axis = 0:Number_of_bits*7-1;
polar_RZ_centered_realization = polar_RZ_Ensemble(1, :);
polar_NRZ_centered_realization = polar_NRZ_Ensemble(1, :);
unipolar_centered_realization = unipolar_Ensemble(1, :);
for lag = time_axis
    polar_RZ_autocorr_values(lag+1) = sum(polar_RZ_centered_realization(1:end-lag) .* polar_RZ_centered_realization(1+lag:end)) / (Number_of_bits*7 - lag);
    polar_NRZ_autocorr_values(lag+1) = sum(polar_NRZ_centered_realization(1:end-lag) .* polar_NRZ_centered_realization(1+lag:end)) / (Number_of_bits*7 - lag);
    unipolar_autocorr_values(lag+1) = sum(unipolar_centered_realization(1:end-lag) .* unipolar_centered_realization(1+lag:end)) / (Number_of_bits*7 - lag);
end
```

Code computing time autocorrelation of one waveform



Time Autocorrelation resulting graphs for each line code.

$$R_X(\tau) =< x(t_1)x(t_1+\tau) > = \frac{1}{N}\sum_n(x(n)x(n+m))$$

(Of a certain waveform)

	Unipolar NRZ	Polar NRZ	Polar RZ
Total Power	R (0) = 8	R (0) = 16	R (0) = 8
DC Power	R (∞) = 4	R (∞) = 0	R (∞) = 0

18) Is the random process ergodic?

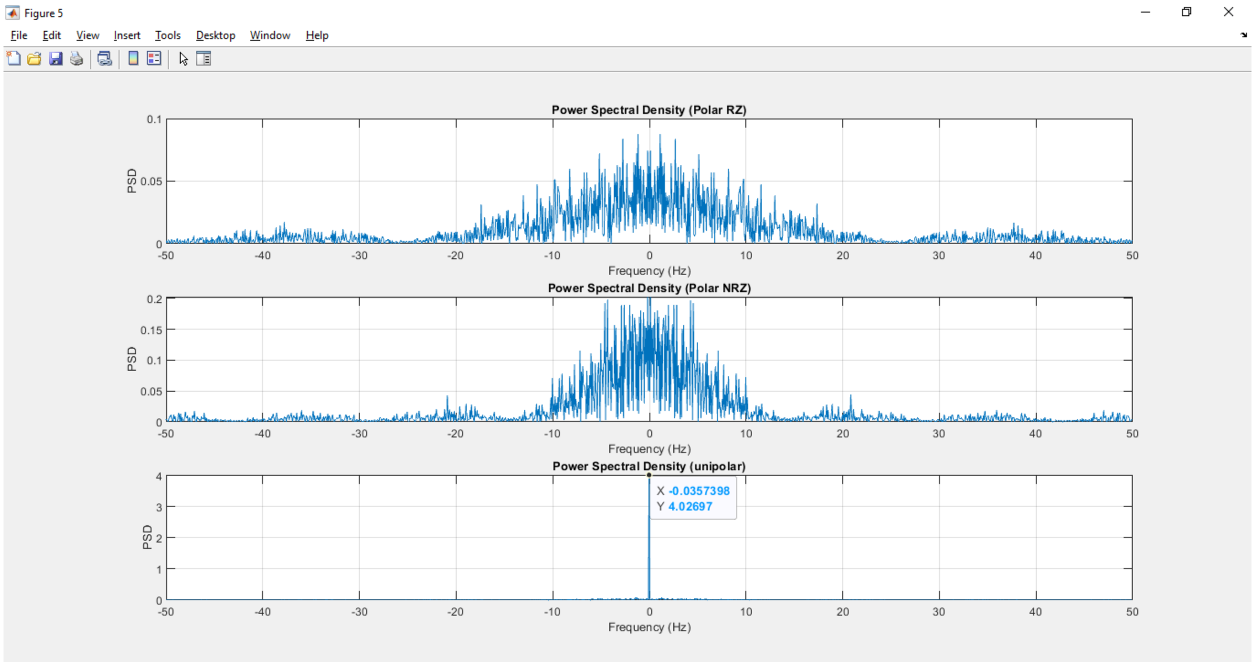
Answer:

Yes, the random process can be considered as an ergodic process; as it is shown in the graphs that the time mean is nearly equal to the statistical mean & the time autocorrelation is nearly equal to the statistical autocorrelation.

19)Plotting the PSD of the ensemble

```
%% PSD
figure(5)
polar_RZ_PSD = abs(fftshift(fft(polar_RZ_autocorrelation))) / 1399;
polar_RZ_freq_resolution = 100 / length(polar_RZ_PSD); % Frequency resolution where sampling freq is 100
polar_RZ_freq_axis = (-50 + (0:length(polar_RZ_PSD)-1) * polar_RZ_freq_resolution); % Create frequency axis from -50 to 50
subplot(3,1,1);
plot(polar_RZ_freq_axis, polar_RZ_PSD);
xlabel("Frequency (Hz)");
ylabel("PSD");
grid on;
title ("Power Spectral Density (Polar RZ)");
polar_NRZ_PSD = abs(fftshift(fft(polar_NRZ_autocorrelation))) / 1399;
polar_NRZ_freq_resolution = 100 / length(polar_NRZ_PSD); % Frequency resolution where sampling freq is 100
polar_NRZ_freq_axis = (-50 + (0:length(polar_NRZ_PSD)-1) * polar_NRZ_freq_resolution); % Create frequency axis from -50 to 50
subplot(3,1,2);
plot(polar_NRZ_freq_axis, polar_NRZ_PSD);
xlabel("Frequency (Hz)");
ylabel("PSD");
grid on;
title ("Power Spectral Density (Polar NRZ)");
unipolar_PSD = abs(fftshift(fft(unipolar_autocorrelation))) / 1399;
unipolar_freq_resolution = 100 / length(unipolar_PSD); % Frequency resolution where sampling freq is 100
unipolar_freq_axis = (-50 + (0:length(unipolar_PSD)-1) * unipolar_freq_resolution); % Create frequency axis from -50 to 50
subplot(3,1,3);
plot(unipolar_freq_axis, unipolar_PSD);
xlabel("Frequency (Hz)");
ylabel("PSD");
grid on;
title ("Power Spectral Density (unipolar)");
```

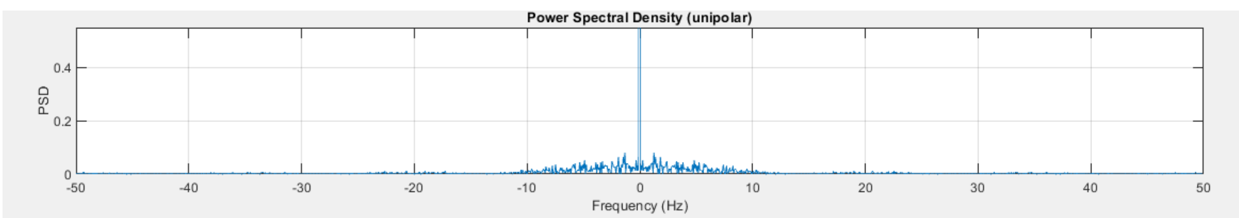
Code for PSD plotting



PSD plot

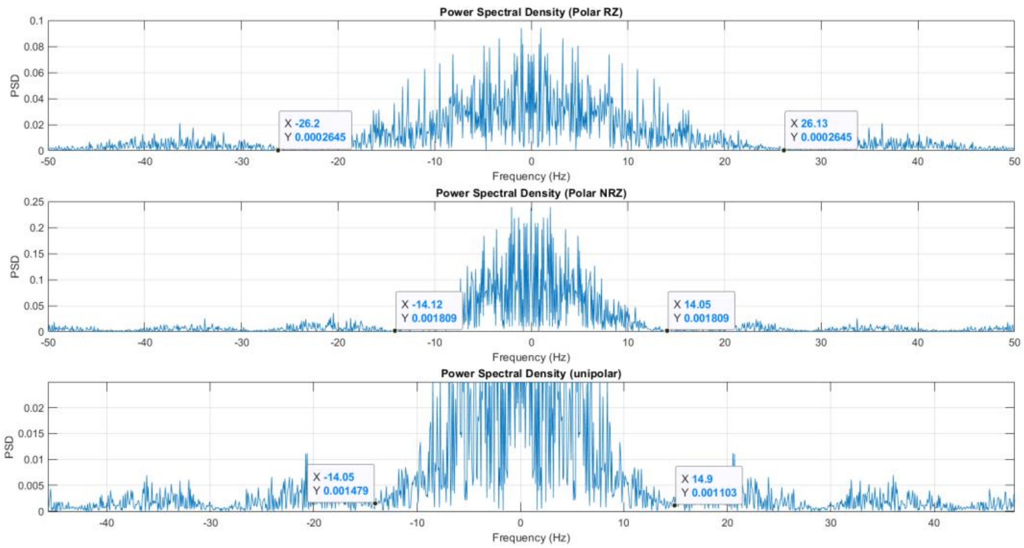
	Unipolar NRZ	Polar NRZ	Polar RZ
PSD analytical results $S_X(f) = F.T\,(R_X(\tau))$	$\frac{A^{\wedge}2}{4}(\mathrm{T}\ \mathrm{sinc}^2(\frac{\omega T}{2})+\delta(t))$	$(A^{\wedge}2)\ \mathrm{T}\ \mathrm{sinc}^2(\frac{\omega T}{2})$	$\frac{A^{\wedge}2}{4}\mathrm{T}\ \mathrm{sinc}^2(\frac{\omega T}{4})$

The graph is nearly sinc^2 (FT of triangle in polar RZ & polar NRZ) also in unipolar NRZ the presence of the impulse at 0 Hz (DC) is obvious also the graph is sinc^2 (can be seen if zoomed in)



Zoomed in PSD graph of unipolar NRZ line code.

20)Computing bandwidth of the transmitted signal



In case of normalized PSD

Line code type	Polar RZ	Polar NRZ	Unipolar
Bandwidth	Bitrate = 26.13 Hz	Bitrate = 14.05 Hz	Bitrate = 14.9 Hz

- We notice that the PSD has a $sinc^2$ (FT of triangle in polar RZ & polar NRZ which is a triangle) and the BW is taken from the max value to the first zero.
- The Bandwidth of Polar NRZ is approximately equal to that of Unipolar NRZ = Bitrate (R) = 14 HZ.
- The Bandwidth of Polar RZ is double that of Polar NRZ and Unipolar NRZ = 2*Bitrate (R).

21) Full MATLAB code

```
close all;
clear all;
clc;
A=4;
Number_of_waveforms = 500;
Number_of_bits = 100;
```

```
% Initialize a [Number_of_waveforms X Number_of_bits] empty
matrix
polar_NRZ_Ensemble =
zeros(Number_of_waveforms,(Number_of_bits+1)*7);
polar_RZ_Ensemble =
zeros(Number_of_waveforms,(Number_of_bits+1)*7);
unipolar_Ensemble =
zeros(Number_of_waveforms,(Number_of_bits+1)*7);
% Generate random start indices based on delays
polar_NRZ_start_indices = randi([0, (Number_of_bits - 1)],
1, Number_of_waveforms);
polar_RZ_start_indices = randi([0, (Number_of_bits - 1)],
1, Number_of_waveforms);
unipolar_start_indices = randi([0, (Number_of_bits - 1)],
1, Number_of_waveforms);
for i= 1:Number_of_waveforms

    polar_NRZ_start_index = polar_NRZ_start_indices(i);
    polar_RZ_start_index = polar_RZ_start_indices(i);
    unipolar_start_index = unipolar_start_indices(i);
    polar_NRZ_Waveform = randi([0, 1], 1,
Number_of_bits+1); % Generate a random vector of 100
elements where each element is either 0 or 1
    polar_RZ_Waveform = randi([0, 1], 1, Number_of_bits+1);
    unipolar_Waveform = randi([0, 1], 1, Number_of_bits+1);
    polar_NRZ_Tx1=((2*polar_NRZ_Waveform)-1)*A; % maping
for 0 to be -A, 1 to be A
    polar_NRZ_Tx=repelem(polar_NRZ_Tx1,7);
    unipolar_Tx1=unipolar_Waveform*A; % maping for 0 to be
0, 1 to be A
    unipolar_Tx=repelem(unipolar_Tx1,7);
    polar_RZ_Tx1=((2*polar_RZ_Waveform)-1)*A; % maping for
0 to be -A, 1 to be A
    polar_RZ_Tx=repelem(polar_RZ_Tx1,7);
    for k=4:7:(Number_of_bits+1)*7
        polar_RZ_Tx(k+1:k+3) = 0;
    end

    % Shift the bits based on start index and delay
    polar_NRZ_shifted_bits = circshift(polar_NRZ_Tx, [0, -
polar_NRZ_start_index]);
    polar_RZ_shifted_bits = circshift(polar_RZ_Tx, [0, -
polar_RZ_start_index]);
```

```
    unipolar_shifted_bits = circshift(unipolar_Tx, [0, -
unipolar_start_index]);
    % Discard the additional one bit
    polar_NRZ_sample_bits = polar_NRZ_shifted_bits(1:end-
7);
    polar_RZ_sample_bits = polar_RZ_shifted_bits(1:end-7);
    unipolar_sample_bits = unipolar_shifted_bits(1:end-7);
    % the above array contains 100 elements, corresponding
to the levels that will be transmitted in every pulse
    polar_NRZ_Ensemble(i, :) = polar_NRZ_shifted_bits; %
every element represents 10 ms
    polar_RZ_Ensemble(i, :) = polar_RZ_shifted_bits;
    unipolar_Ensemble(i, :) = unipolar_shifted_bits;
end
figure(1);
polar_RZ_plot_sequence =
repelem(polar_RZ_shifted_bits,1,10); %the vector plot
sequence is a vector where each bit is represented as 70ms
polar_NRZ_plot_sequence =
repelem(polar_NRZ_shifted_bits,1,10);
unipolar_plot_sequence =
repelem(unipolar_shifted_bits,1,10);
subplot(3,1,1);
stairs(polar_RZ_plot_sequence, 'k', 'LineWidth', 2);
xlim([0, (Number_of_bits+1)*70]);
title('polar RZ Waveform');
ylabel('Amplitude(V)');
xlabel('Time(ms)');
grid on;
subplot(3,1,2);
stairs(polar_NRZ_plot_sequence, 'k', 'LineWidth', 2);
xlim([0, (Number_of_bits+1)*70]);
title('polar NRZ Waveform');
ylabel('Amplitude(V)');
xlabel('Time(ms)');
grid on;
subplot(3,1,3);
stairs(unipolar_plot_sequence, 'k', 'LineWidth', 2);
xlim([0, (Number_of_bits+1)*70]);
title('unipolar Waveform');
ylabel('Amplitude(V)');
xlabel('Time(ms)');
grid on;
%% Statistical mean
```

```
polar_RZ_column_sum=zeros(1,700);
polar_NRZ_column_sum=zeros(1,700);
unipolar_column_sum=zeros(1,700);
polar_RZ_column_sum = sum(polar_RZ_Ensemble(:, 8));
polar_NRZ_column_sum = sum(polar_NRZ_Ensemble(:, 8));
unipolar_column_sum = sum(unipolar_Ensemble(:, 8));
for j= 9:(Number_of_bits+1)*7
    polar_RZ_column_sum = [polar_RZ_column_sum,
sum(polar_RZ_Ensemble(:, j))];
    polar_NRZ_column_sum = [polar_NRZ_column_sum,
sum(polar_NRZ_Ensemble(:, j))];
    unipolar_column_sum = [unipolar_column_sum,
sum(unipolar_Ensemble(:, j))];

end
polar_RZ_statistical_mean =
polar_RZ_column_sum/Number_of_waveforms;
polar_NRZ_statistical_mean =
polar_NRZ_column_sum/Number_of_waveforms;
unipolar_statistical_mean =
unipolar_column_sum/Number_of_waveforms;
time = 1:Number_of_bits*7; % X-axis for plotting
figure(2);
subplot(3,1,1);
plot(time,polar_RZ_statistical_mean);
ylim([min(-4), max(4)]);
title('Polar RZ Statistical mean');
xlabel('Time (ms) (each sample corresponding to 10 ms)');
ylabel('Amplitude');
grid on;
subplot(3,1,2);
plot(time,polar_NRZ_statistical_mean);
ylim([min(-4), max(4)]);
title('Polar NRZ Statistical mean');
xlabel('Time (ms) (each sample corresponding to 10 ms)');
ylabel('Amplitude');
grid on;
subplot(3,1,3);
plot(time,unipolar_statistical_mean);
ylim([min(-4), max(4)]);
title('unipolar Statistical mean');
xlabel('Time (ms) (each sample corresponding to 10 ms)');
ylabel('Amplitude');
grid on;
```

```
%% Time mean
%preallocating time mean matrices
polar_RZ_time_mean=zeros(Number_of_waveforms,1);
polar_NRZ_time_mean=zeros(Number_of_waveforms,1);
unipolar_time_mean=zeros(Number_of_waveforms,1);
polar_RZ_time_sum=zeros(Number_of_waveforms,1);
polar_NRZ_time_sum=zeros(Number_of_waveforms,1);
unipolar_time_sum=zeros(Number_of_waveforms,1);
%calculating mean of every realization
for i=1:Number_of_waveforms
    for j=1:Number_of_bits*7
    polar_RZ_time_sum(i,1)=polar_RZ_time_sum(i,1) +
polar_RZ_Ensemble(i,j);
    polar_NRZ_time_sum(i,1)=polar_NRZ_time_sum(i,1) +
polar_NRZ_Ensemble(i,j);
    unipolar_time_sum(i,1)=unipolar_time_sum(i,1) +
unipolar_Ensemble(i,j);
    end

polar_RZ_time_mean(i,1)=polar_RZ_time_sum(i,1)/(Number_of_b
its*7);

polar_NRZ_time_mean(i,1)=polar_NRZ_time_sum(i,1)/(Number_of
_bits*7);

unipolar_time_mean(i,1)=unipolar_time_sum(i,1)/(Number_of_b
its*7);

end
waveform = 1:Number_of_waveforms;
figure(3)
subplot(3,1,1);
plot(waveform,polar_RZ_time_mean);
ylim([min(-4), max(4)]);
title('Polar RZ Time mean');
xlabel('Waveform');
ylabel('Amplitude');
grid on;

subplot(3,1,2);
plot(waveform,polar_NRZ_time_mean);
ylim([min(-4), max(4)]);
title('Polar NRZ Time mean');
xlabel('Waveform');
```

```
ylabel('Amplitude');
grid on;

subplot(3,1,3);
plot(waveform,unipolar_time_mean);
ylim([min(-4), max(4)]);
title('unipolar Time mean');
xlabel('Waveform');
ylabel('Amplitude');
grid on;
%% Statistical Autocorrelation
polar_RZ_stat_autocorr = zeros(1,Number_of_bits*7);
polar_NRZ_stat_autocorr = zeros(1,Number_of_bits*7);
unipolar_stat_autocorr = zeros(1,Number_of_bits*7);

% Calculate autocorrelation for each realization
polar_RZ_Ensemble_without_delay_bit = polar_RZ_Ensemble(:,
8:end);
polar_NRZ_Ensemble_without_delay_bit =
polar_NRZ_Ensemble(:, 8:end);
unipolar_Ensemble_without_delay_bit = unipolar_Ensemble(:,
8:end);
dimensions = size(polar_RZ_Ensemble_without_delay_bit);
center_index = (dimensions(2)/2)+1;

for lag = (1-center_index):(center_index-2)

    for i = 1:Number_of_waveforms
        polar_RZ_sample =
polar_RZ_Ensemble_without_delay_bit(i, :);
        polar_NRZ_sample =
polar_NRZ_Ensemble_without_delay_bit(i, :);
        unipolar_sample =
unipolar_Ensemble_without_delay_bit(i, :);
        polar_RZ_stat_autocorr(lag+center_index) =
polar_RZ_stat_autocorr(lag+center_index) +
(polar_RZ_sample(center_index)*polar_RZ_sample(lag+center_i
ndex));
        polar_NRZ_stat_autocorr(lag+center_index) =
polar_NRZ_stat_autocorr(lag+center_index) +
(polar_NRZ_sample(center_index)*polar_NRZ_sample(lag+center
_index));
        unipolar_stat_autocorr(lag+center_index) =
unipolar_stat_autocorr(lag+center_index) +
```

```
(unipolar_sample(center_index)*unipolar_sample(lag+center_i
ndex));

    end

end

% Calculate average autocorrelation across all realizations
polar_RZ_avg_autocorr =
polar_RZ_stat_autocorr/Number_of_waveforms;
polar_NRZ_avg_autocorr =
polar_NRZ_stat_autocorr/Number_of_waveforms;
unipolar_avg_autocorr =
unipolar_stat_autocorr/Number_of_waveforms;
%getting the zero lag at index zero
polar_RZ_avg_autocorr = circshift(polar_RZ_avg_autocorr,
center_index-2);
polar_NRZ_avg_autocorr = circshift(polar_NRZ_avg_autocorr,
center_index-2);
unipolar_avg_autocorr = circshift(unipolar_avg_autocorr,
center_index-2);
%Flipping R_tau to the -ve quad
polar_RZ_autocorrelation =
[fliplr(polar_RZ_avg_autocorr(2:end))
polar_RZ_avg_autocorr];
polar_NRZ_autocorrelation =
[fliplr(polar_NRZ_avg_autocorr(2:end))
polar_NRZ_avg_autocorr];
unipolar_autocorrelation =
[fliplr(unipolar_avg_autocorr(2:end))
unipolar_avg_autocorr];
figure(4)
time = -((Number_of_bits*7)-1):(Number_of_bits*7)-1;
subplot(3,1,1);
plot (time, polar_RZ_autocorrelation);
xlim([-35 35]);
ylim([min(-10), max(15)]);
xlabel("time in samples");
ylabel("Autocorrelation");
grid on;
title ("Statistical Autocorrelation (polar RZ)");
subplot(3,1,2);
plot (time, polar_NRZ_autocorrelation);
xlim([-35 35]);
```

```
ylim([min(-10), max(25)]);
xlabel("time in samples");
ylabel("Autocorrelation");
grid on;
title ("Statistical Autocorrelation (polar NRZ)");
subplot(3,1,3);
plot (time, unipolar_autocorrelation);
xlim([-35 35]);
ylim([min(-2), max(10)]);
xlabel("time in samples");
ylabel("Autocorrelation");
grid on;
title ("Statistical Autocorrelation (unipolar)");
%% PSD
figure(5)
polar_RZ_PSD = abs(fftshift(fft(polar_RZ_autocorrelation)))
/ 1399;
polar_RZ_freq_resolution = 100 / length(polar_RZ_PSD); %
Frequency resolution where sampling freq is 100
polar_RZ_freq_axis = (-50 + (0:length(polar_RZ_PSD)-1) *
polar_RZ_freq_resolution); % Create frequency axis from -50
to 50
subplot(3,1,1);
plot(polar_RZ_freq_axis, polar_RZ_PSD);
xlabel("Frequency (Hz)");
ylabel("PSD");
grid on;
title ("Power Spectral Density (Polar RZ)");
polar_NRZ_PSD =
abs(fftshift(fft(polar_NRZ_autocorrelation))) / 1399;
polar_NRZ_freq_resolution = 100 / length(polar_NRZ_PSD); %
Frequency resolution where sampling freq is 100
polar_NRZ_freq_axis = (-50 + (0:length(polar_NRZ_PSD)-1) *
polar_NRZ_freq_resolution); % Create frequency axis from -
50 to 50
subplot(3,1,2);
plot(polar_NRZ_freq_axis, polar_NRZ_PSD);
xlabel("Frequency (Hz)");
ylabel("PSD");
grid on;
title ("Power Spectral Density (Polar NRZ)");
unipolar_PSD = abs(fftshift(fft(unipolar_autocorrelation)))
/ 1399;
```

```
unipolar_freq_resolution = 100 / length(unipolar_PSD); %
Frequency resolution where sampling freq is 100
unipolar_freq_axis = (-50 + (0:length(unipolar_PSD)-1) *
unipolar_freq_resolution); % Create frequency axis from -50
to 50
subplot(3,1,3);
plot(unipolar_freq_axis, unipolar_PSD);
xlabel("Frequency (Hz)");
ylabel("PSD");
grid on;
title ("Power Spectral Density (unipolar)");
%% Time Autocorrelation

% Preallocate array to store autocorrelation values
polar_RZ_autocorr_values = zeros(Number_of_bits*7, 1);
polar_NRZ_autocorr_values = zeros(Number_of_bits*7, 1);
unipolar_autocorr_values = zeros(Number_of_bits*7, 1);
% Calculate autocorrelation for the realization
time_axis = 0:Number_of_bits*7-1;
polar_RZ_centered_realization = polar_RZ_Ensemble(1, :);
polar_NRZ_centered_realization = polar_NRZ_Ensemble(1, :);
unipolar_centered_realization = unipolar_Ensemble(1, :);
for lag = time_axis
    polar_RZ_autocorr_values(lag+1) =
sum(polar_RZ_centered_realization(1:end-lag) .*
polar_RZ_centered_realization(1+lag:end)) /
(Number_of_bits*7 - lag);
    polar_NRZ_autocorr_values(lag+1) =
sum(polar_NRZ_centered_realization(1:end-lag) .*
polar_NRZ_centered_realization(1+lag:end)) /
(Number_of_bits*7 - lag);
    unipolar_autocorr_values(lag+1) =
sum(unipolar_centered_realization(1:end-lag) .*
unipolar_centered_realization(1+lag:end)) /
(Number_of_bits*7 - lag);
end
% Plot time autocorrelation within one realization
figure(6)
time_axis = [-fliplr(time_axis(2:end)), time_axis];
polar_RZ_time_Autocorrelation =
[flipud(polar_RZ_autocorr_values(2:end));
polar_RZ_autocorr_values];
```

```
polar_NRZ_time_Autocorrelation =
[flipud(polar_NRZ_autocorr_values(2:end));
polar_NRZ_autocorr_values];
unipolar_time_Autocorrelation =
[flipud(unipolar_autocorr_values(2:end));
unipolar_autocorr_values];
subplot(3,1,1);
plot(time_axis,polar_RZ_time_Autocorrelation);
xlim([-35 35]);
ylim([min(-2), max(10)]);
xlabel('time in samples');
ylabel('Autocorrelation');
grid on;
title('Time Autocorrelation within One Realization (polar
RZ)');
subplot(3,1,2);
plot(time_axis,polar_NRZ_time_Autocorrelation);
xlim([-35 35]);
ylim([min(-5), max(20)]);
xlabel('time in samples');
ylabel('Autocorrelation');
grid on;
title('Time Autocorrelation within One Realization (polar
NRZ)');
subplot(3,1,3);
plot(time_axis,unipolar_time_Autocorrelation);
xlim([-35 35]);
ylim([min(-2), max(10)]);
xlabel('time in samples');
ylabel('Autocorrelation');
grid on;
title('Time Autocorrelation within One Realization
(unipolar)');
```