Performance of Matched Filters and Correlators

&

Line Code

ID	الاسم
19016377	محمد سامي محمد عبدالعزيز
19015660	رشاد السيد ابراهيم احمد
19015336	تسنيم عبدالصمد رزق محمد

Part 1

```
clc; clear all; close all;
%% (1)__Simulation Parameters

numBits = 1e5;       %bits number
snrRange = 0:2:30;   %SNR(0,2,4,6,...,30)dB
m_user=input('Enter  the Number of samples that represents waveform : ');
if isempty(m_user)
    numSamples=20; %defualt number of samples
else
    numSamples=m_user;
end
samplingInstant = 20;

% Generate rectangular pulse s1(t)
Amp_s1 = input('Enter  an amplitude number for s1(t) : ');
if isempty(Amp_s1)
    s1 = ones(1,numSamples);
else
        s1 = Amp_s1 * ones(1,numSamples);
end
% Generate rectangular pulse s2(t)
Amp_s2 = input('Enter  an amplitude number for s2(t) : ');
if isempty(Amp_s2)
    s2 = zeros(1,numSamples); %defualt
else
        s2 = Amp_s2 * ones(1,numSamples);
end

receiverType = {'Matched Filter', 'Correlator'};

%% (2)__Generate random binary data vector

bits = randi([0 1], 1, numBits);     % Generate Random Binary Data Vector


%% (3)__Represent Bits with Proper Waveform (20 sample for each) & calculate power

waveform=repelem(bits,numSamples);
Powerr = 1/(numBits*numSamples) * (sum(waveform.^2))  % calculate power --------->

%% ..........................................................................
BER_MF=zeros(1, length(snrRange)); %array for BER valuse at every SNR

for SNR_i = 1:length(snrRange)
%% (4)__Apply noise to samples & calculate noise power based on SNR

SNR=snrRange(SNR_i);

ReSequence = awgn(bits, SNR, 'measured'); % recived waveform with noise

SNR_not_dB = 10^(SNR/10);  % SNR ratio    SNRdB=10*log_10(snr)
s1_power = Amp_s1^2;
noise_power(SNR_i) = s1_power/SNR_not_dB

%% (5)__Apply convolution process in the receiver

% matched
filter=fliplr(s1 - s2);          FL=length(filter);  %filter length
output_MF = zeros(1, (2*20-1)*numBits);
```

```matlab
%% (5)__Apply convolution process in the receiver

% matched
filter=fliplr(s1 - s2);          FL=length(filter);  %filter length
output_MF = zeros(1, (2*20-1)*numBits);

for i = 0:numBits-1
    Bit_20 = ReSequence((i*20)+1:(i+1)*20);          %take 20 samples
    conv_20 = conv(Bit_20,filter);  %length = (20+20-1)=  39
    output_MF( (FL+numSamples-1)*i+1:(FL+numSamples-1)*i+length(conv_20) ) = conv_20;
end          %  (20+20-1)=  39          (1:39 40:78 79:. .....)

% Correlator Receiver
    y_t = zeros(1, numBits);
 g=s1 - s2;
     for k=1:numBits
    y_t = sum(ReSequence(((k-1)*numSamples+1:k*numSamples).*g));
    end
% simple detector

for i = 1:1:length(bits)
 S_t(i)= ReSequence(i*numSamples);
end

%% (6)__Sample the output of the Matched filter

sampledOutput_MF = output_MF(numSamples:(samplingInstant+numSamples)-1:end);
% take sample num 20 every 39 samples

%% (7)__Decide whether the recived seq is 1 or 0        %   ()   ------------------>

%sample detector
Vth = (s1(10)+s2(10))/2;
rx_bits_s = S_t >= Vth;  % 1 or 0
% '''matched'''
Vth_m_c =mean(sampledOutput_MF);
detectedBits_MF = sampledOutput_MF >= Vth;  % 1 or 0

% '''CorrelatorMake''' decision based on threshold
rx_bits_c = y_t >= Vth;  % 1 or 0

%% (8,9)__calculate number of errors,Save the BER of each SNR in matrix

numErrorsMF = biterr(bits, detectedBits_MF);
BER=numErrorsMF/numBits;
BER_MF(SNR_i) =BER;
if BER==0
    fprintf('MF BER=0 at SNR= %d',SNR);
end

errors(SNR_i) = sum(xor(bits,rx_bits_c));
    BER_c(SNR_i) = errors(SNR_i)/numBits;

error(SNR_i) = sum(xor(bits,rx_bits_s));
    BER_S(SNR_i) = error(SNR_i)/numBits;

end

%BER_MF
%% (10)__Plot the BER curve against SNR
figure;
semilogy(snrRange, BER_S, 'r-x');
xlabel('SNR (dB)');
ylabel('BER');
title('Simple Detector SNR & BER relation');
```

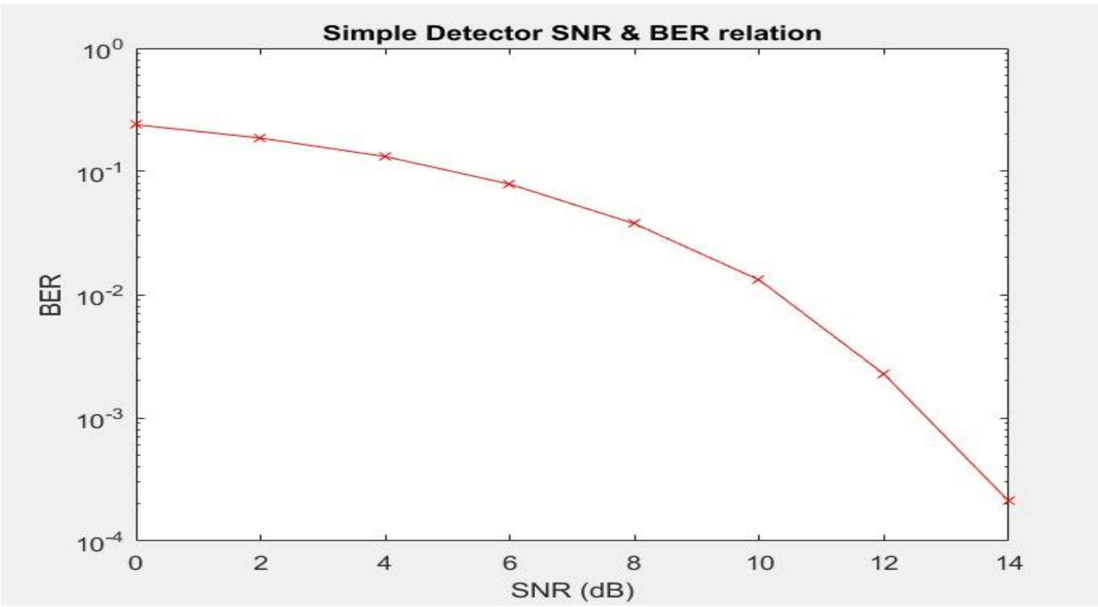
```
figure;
semilogy(snrRange, BER_MF, 'b-x');
xlabel('SNR (dB)');
ylabel('BER');
title('Matched Filter SNR & BER relation');

figure;
semilogy(snrRange, BER_c, 'g-o');
xlabel('SNR (dB)');
ylabel('BER');
title('Matched Filter SNR & BER relation');


figure;
semilogy(snrRange, BER_c, 'g-o');
hold on;
semilogy(snrRange, BER_MF, 'b-x');
hold on;
semilogy(snrRange, BER_S, 'r-x');
xlabel('SNR (dB)');
ylabel('BER');
title('Matched Filter & Correlated SNR & BER relation');
```

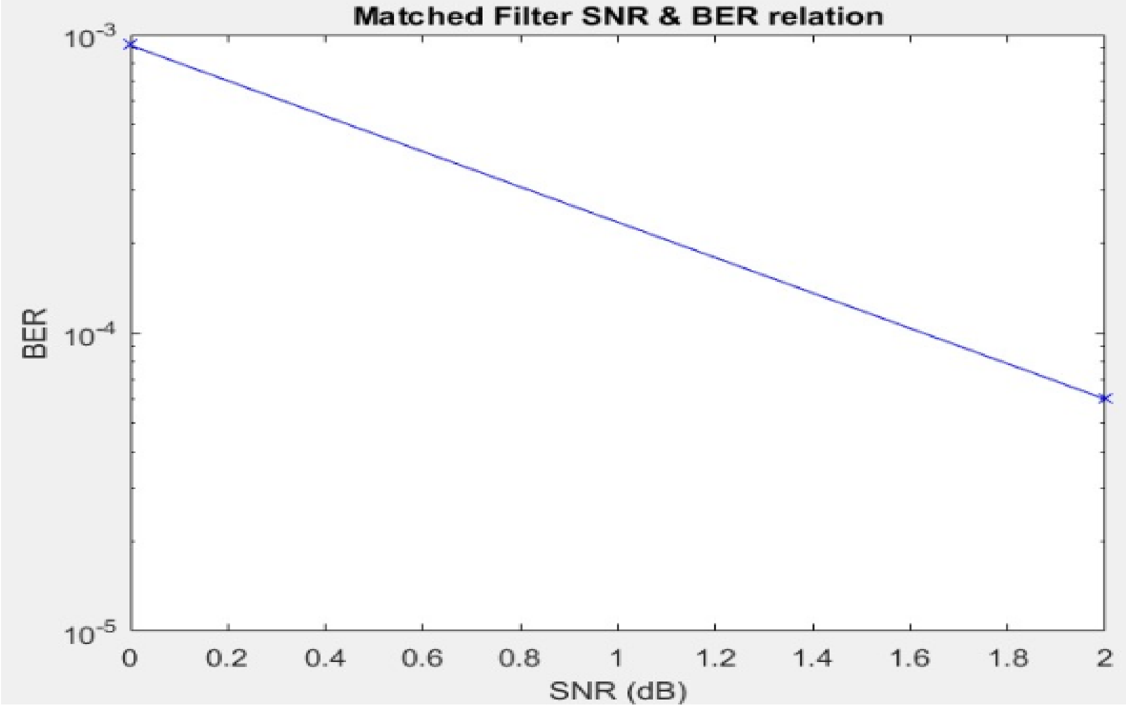
Plot

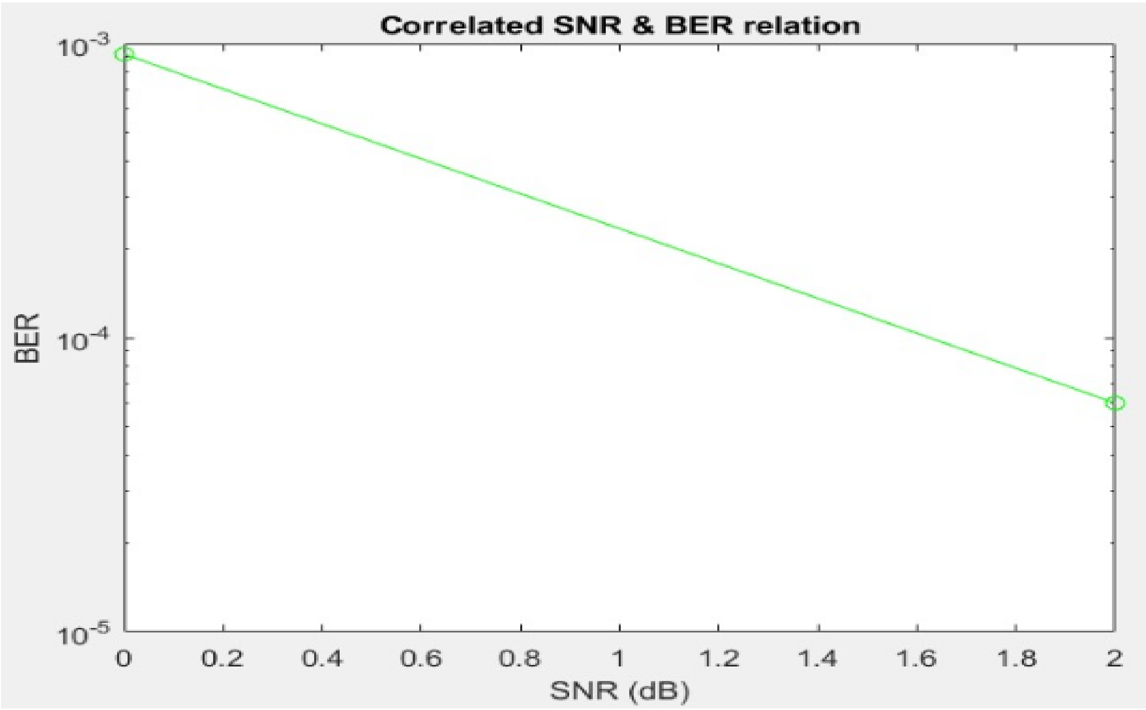
```
Command Window
Enter  the Number of samples that represents waveform : 20
Enter  an amplitude number for s1(t) : 1
Enter  an amplitude number for s2(t) : 0

Powerr =

    0.5027

MF BER=0 at SNR= 30
noise_power =

  Columns 1 through 13

    1.0000    0.6310    0.3981    0.2512    0.1585    0.1000    0.0631    0.0398    0.0251    0.0158    0.0100    0.0063    0.0040

  Columns 14 through 16

    0.0025    0.0016    0.0010
```

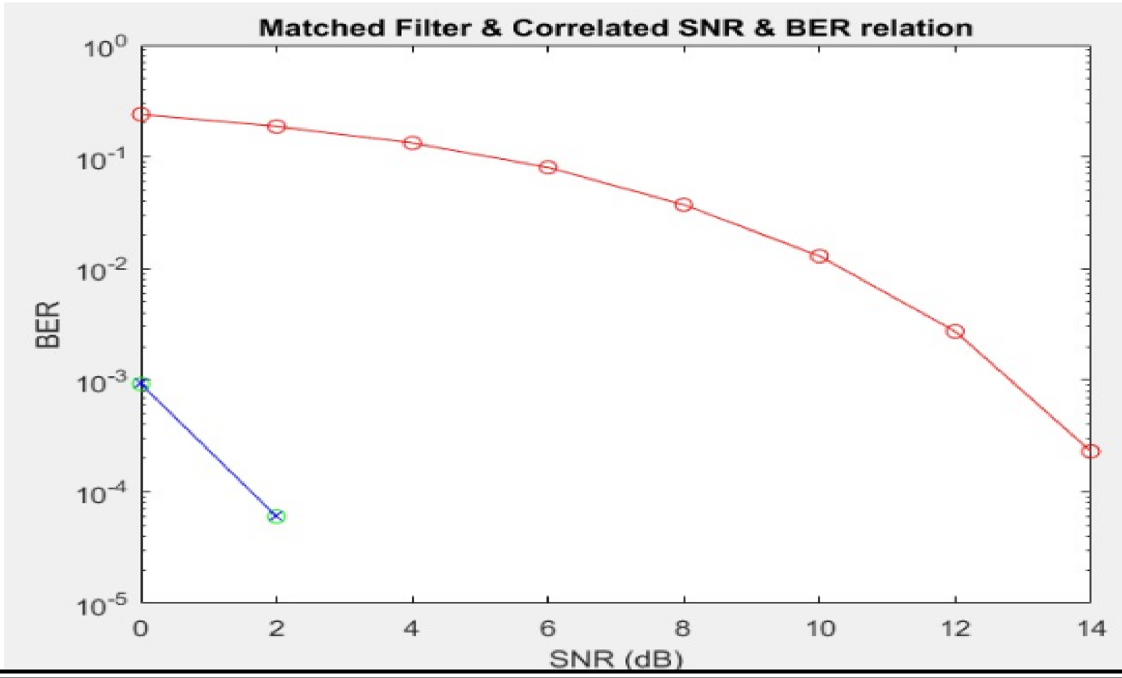
a matched filter will provide a lower bit error rate than a simple detector for the same level of noise.

transmitted power =0.5027




Matched Filter SNR & BER relation
BER
SNR (dB)
10^{-3}
10^{-4}
10^{-5}
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2


Correlated SNR & BER relation
BER
SNR (dB)
10^{-3}
10^{-4}
10^{-5}
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2


Matched Filter & Correlated SNR & BER relation
BER
SNR (dB)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0 2 4 6 8 10 12 14

Part 2

Line code

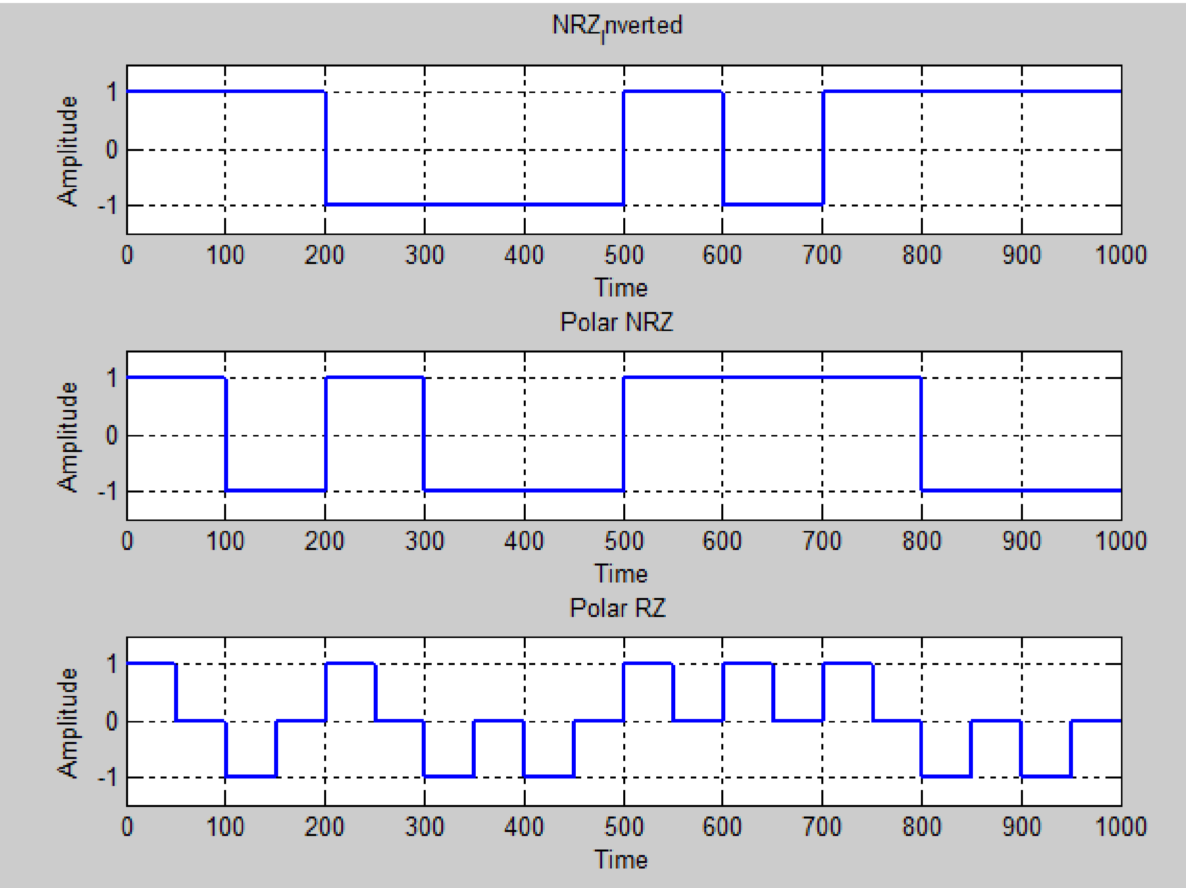
```
n = 10;
% Set pulse duration and sampling rate
T = 1;                 % Pulse duration in seconds
fs = 100;              % Sampling rate in Hz
t = 0:1/fs:T-1/fs;    % Time vector for one pulse

% Generate random bits
bits    = randi([0 1], 1, n)

%% --------------Non return zero inverted ---------------

NRZ_I  = zeros(1, length(bits)*length(t));
 x=-1;
 for i = 1:length(bits)
     if bits(i)==1
         NRZ_I((i-1)*length(t)+1:(i)*length(t)) = -x;
         x=-x;
     else
         NRZ_I((i-1)*length(t)+1:(i)*length(t)) = x;
     end

 end
figure
 plot(NRZ_I, 'LineWidth', 2);
 axis([0 length(NRZ_I) -1.5 1.5]);
 grid on;
 xlabel('Time');
 ylabel('Amplitude');
 title('NRZ_Inverted ');

%% ----------------------- Polar  ------------------------

%----------------------- NRZ   ------------------
Polar_NRZ = zeros(1, length(bits)*length(t));

for i = 1:length(bits)
    if bits(i) == 1
        Polar_NRZ ((i-1)*length(t)+1:i*length(t)) = 1;
    else
        Polar_NRZ ((i-1)*length(t)+1:i*length(t)) = -1;
    end
end

subplot(4,1,3);
plot(Polar_NRZ , 'LineWidth', 2);
axis([0 length(Polar_NRZ) -1.5 1.5]);
grid on;
xlabel('Time');
ylabel('Amplitude');
title('Polar NRZ ');


%----------------------- RZ   ------------------

Polar_RZ  = zeros(1, length(bits)*length(t));
s=length(Polar_RZ);
for i = 1:length(bits)
    if bits(i) == 1
           Polar_RZ((i-1)*length(t)+1:i*length(t)-50) =1;
    else
         Polar_RZ((i-1)*length(t)+1:i*length(t)-50)=-1;
    end

end
```

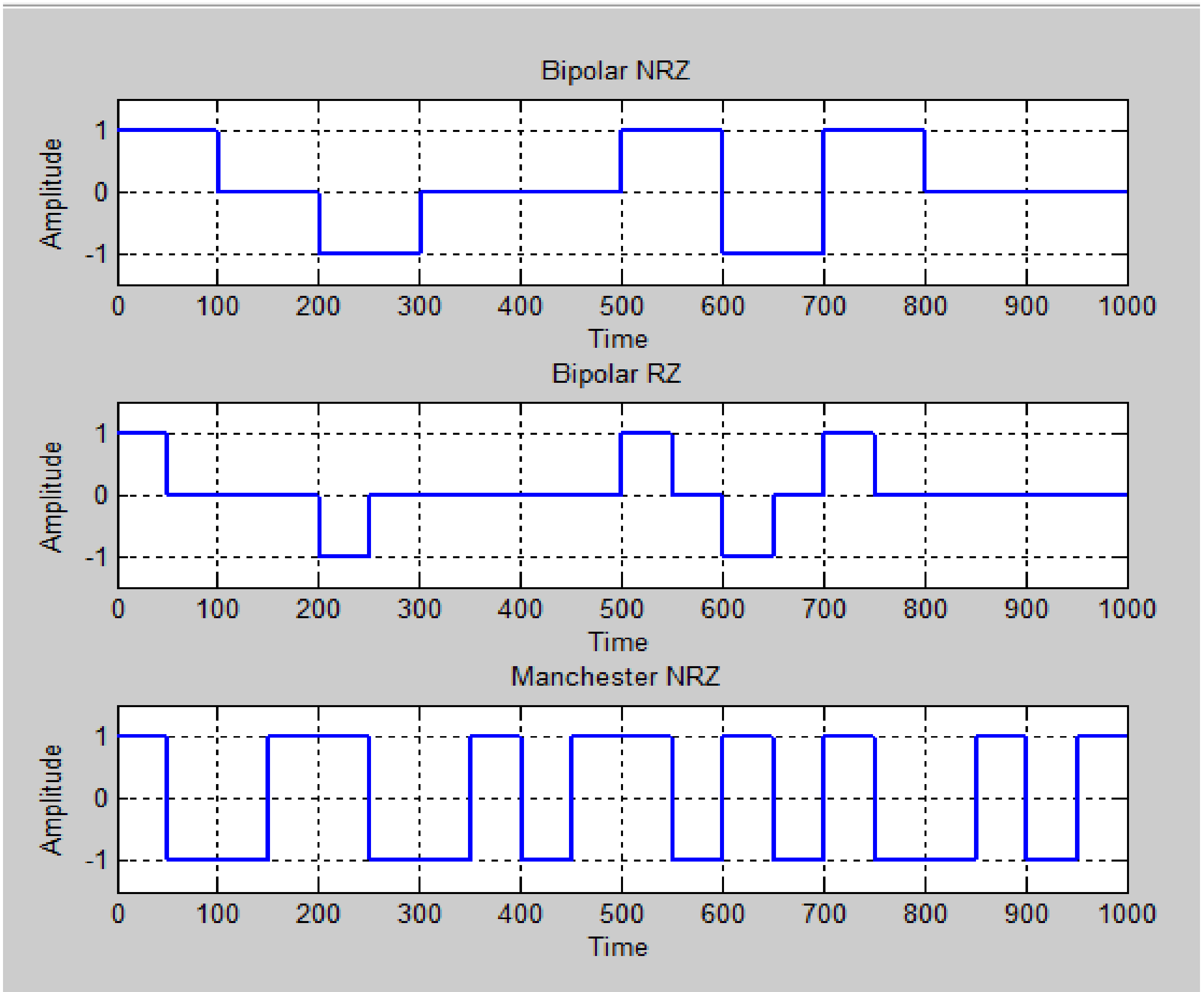
```
subplot(4,1,4);
plot(Polar_RZ, 'LineWidth', 2);
axis([0 s -1.5 1.5]);
grid on;
xlabel('Time');
ylabel('Amplitude');
title('Polar RZ ');

%%      Bipolar

%----------------------- NRZ   ------------------
Biolar_NRZ  = zeros(1, length(bits)*length(t));
x=1;
for i = 1:length(bits)
    if bits(i)  == 1
          Biolar_NRZ((i-1)*length(t)+1:(i)*length(t))  = x;
           x =-x;
    else
        Biolar_NRZ((i-1)*length(t)+1:i*length(t))  = 0;
    end
end
figure
subplot(3,1,1);
plot(Biolar_NRZ, 'LineWidth', 2);
axis([0 length(Biolar_NRZ) -1.5 1.5]);
grid on;
xlabel('Time');
ylabel('Amplitude');
title('Bipolar NRZ ');

%--------------------- RZ -----------------
Biolar_RZ = zeros(1, length(bits)*length(t));
s=length(Biolar_RZ);
c=1;
for i = 1:length(bits)
    if bits(i)  == 1
          Biolar_RZ((i-1)*length(t)+1:i*length(t)-50)  =c;
          c=-c;
    else
        Biolar_RZ((i-1)*length(t)+1:i*length(t)-50)=0;
    end
end
subplot(3,1,2);
plot(Biolar_RZ, 'LineWidth', 2);
axis([0 s -1.5 1.5]);
grid on;
xlabel('Time');
ylabel('Amplitude');
title('Bipolar RZ ');

%% ------------------------- Mancheseter ------------------
Mancheseter = zeros(1, length(bits)*length(t));
s=length(Mancheseter);
for i = 1:length(bits)
    if bits(i)  == 1
          Mancheseter((i-1)*length(t)+1:i*length(t)-50)  =1;
          Mancheseter(i*length(t)-50+1:i*length(t))  =-1;


    else
        Mancheseter((i-1)*length(t)+1:i*length(t)-50)  =-1;
          Mancheseter(i*length(t)-50+1:i*length(t))  =1;
    end

end

subplot(3,1,3);
plot(Mancheseter, 'LineWidth', 2);
axis([0 s -1.5 1.5]);
grid on;
xlabel('Time');
ylabel('Amplitude');
title('Manchester NRZ ');
```

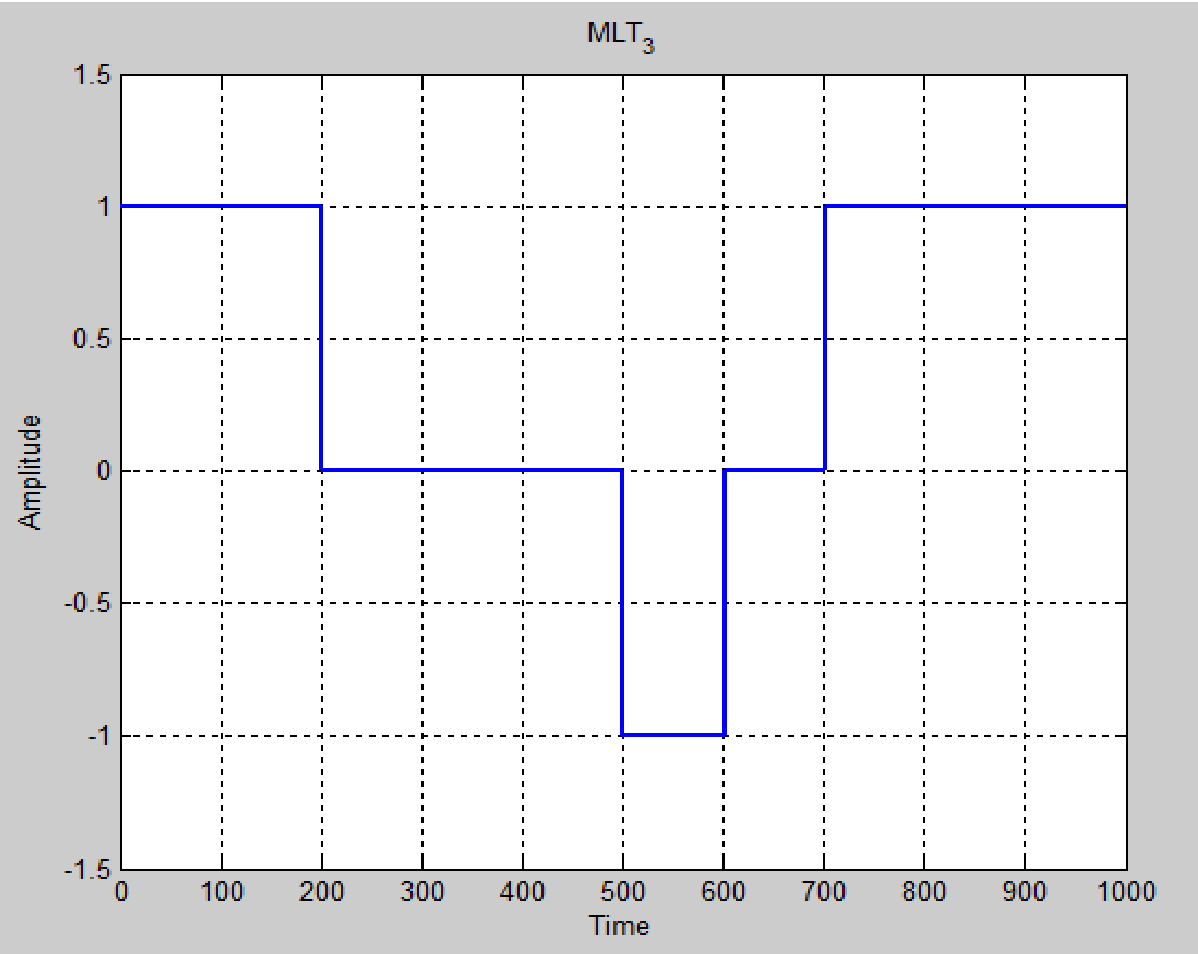
```matlab
%% -------------------MLT_3-----------------------------

MLT3 = zeros(1, length(bits)*length(t));
prev_value=[ 0 1 0 -1 ];
pulse=ones(1,length(t));
Count=1;
for i = 1:length(bits)

    if bits(i) == 0
        MLT3((i-1)*length(t)+1:i*length(t)) =prev_value(Count)*pulse ;
    else
        Count=Count+1;
            if Count>4 %because the size of prev_value is 4 and repeated continuous
            Count=1;
            end
        MLT3((i-1)*length(t)+1:i*length(t)) =prev_value(Count)*pulse;
    end

    end

% Plot the modulated signals
figure
plot( MLT3, 'LineWidth', 2);
axis([0 length(MLT3) -1.5 1.5]);
grid on;
xlabel('Time');
ylabel('Amplitude');
title('MLT_3');
```

Plots:

```
bits =

     1     0     1     0     0     1     1     1     0     0
```

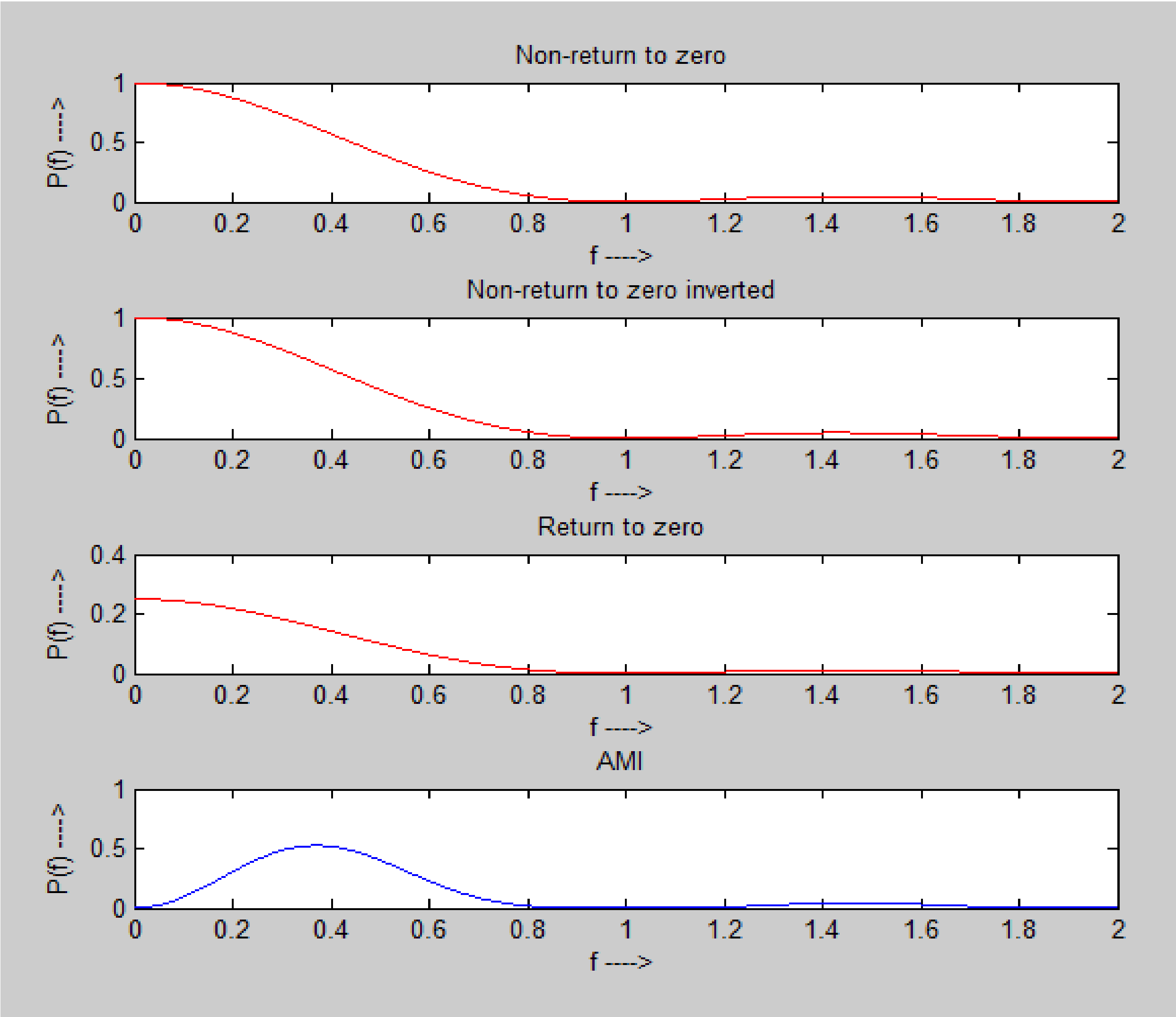


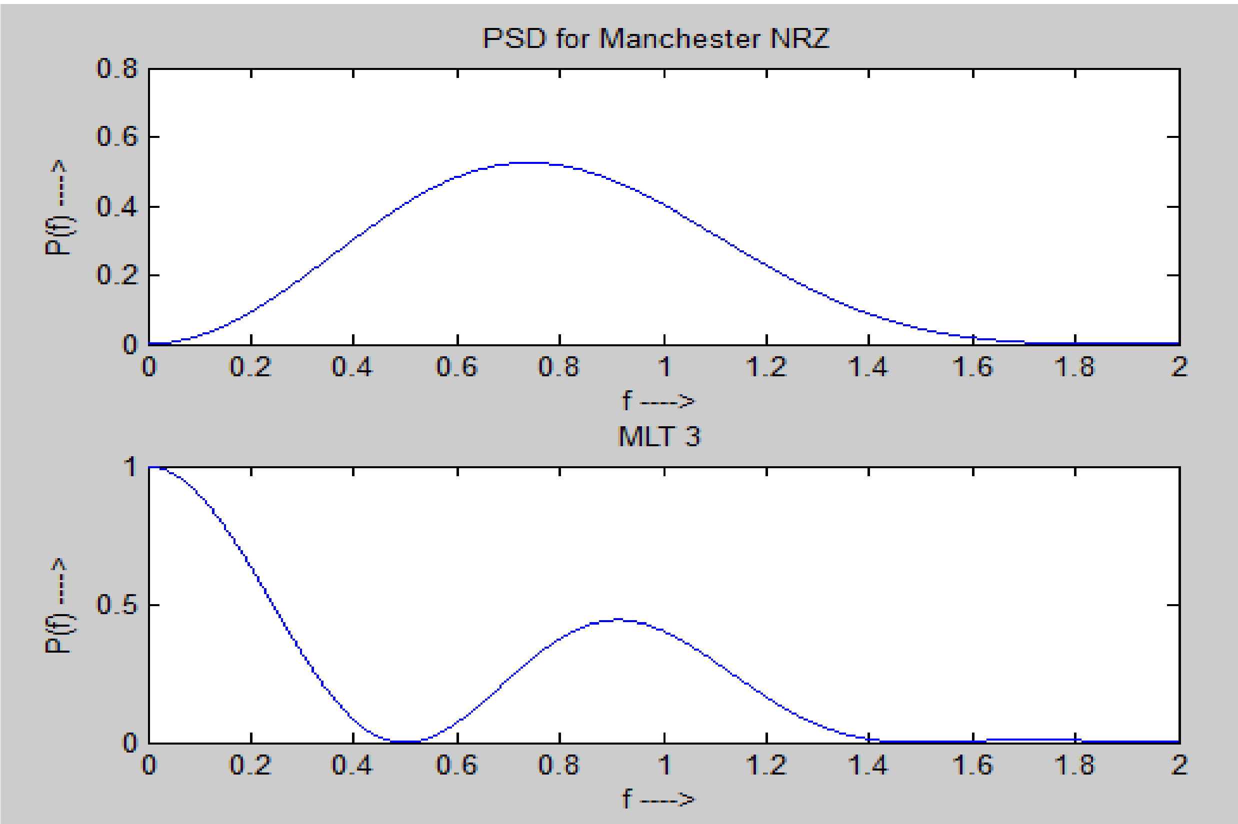

Bipolar NRZ
Amplitude
1
0
-1
0
100
200
300
400
500
600
700
800
900
1000
Time
Bipolar RZ
Amplitude
1
0
-1
0
100
200
300
400
500
600
700
800
900
1000
Time
Manchester NRZ
Amplitude
1
0
-1
0
100
200
300
400
500
600
700
800
900
1000
Time


MLT3
Amplitude
1.5
1
0.5
0
-0.5
-1
-1.5
0
100
200
300
400
500
600
700
800
900
1000
Time

PSD

```
% Set pulse duration and sampling rate
T = 1;              % Pulse duration in seconds
fs = 100;           % Sampling rate in Hz
t = 0:1/fs:T-1/fs;
f=0:0.001:2;
Tb = 1;
A=1;
x=f*Tb;
%%  Polar NRZ

PSD_Pol_NRZ=A.^2*Tb*(sinc(x).*sinc(x));
figure
subplot(4,1,1);
plot(f,PSD_Pol_NRZ,'r')
xlabel('f ---->')
ylabel('P(f)  ---->')
title('Non-return to zero ');

%%  Polar NRZi

PSD_Pol_NRZ=A.^2*Tb*(sinc(x).*sinc(x));
subplot(4,1,2);
plot(f,PSD_Pol_NRZ,'r')
xlabel('f ---->')
ylabel('P(f)  ---->')
title('Non-return to zero inverted ');
%%  Polar RZ

PSD_Pol_RZ=A.^2*0.25*Tb*(sinc(x).*sinc(x));

subplot(4,1,3);
plot(f,PSD_Pol_RZ,'r')
xlabel('f ---->')
ylabel('P(f)  ---->')
title('Return to zero ');
%%  Biolar NRZ
PSD_Bipol_NRZ=A.^2*Tb*(sinc(x)).^2.*(sin(pi*x)).^2;
subplot(4,1,4);
plot(f,PSD_Bipol_NRZ,'b');
xlabel('f ---->')
ylabel('P(f)  ---->')
title('AMI ');


%% Manchester NRZ
PSD_Manchester = A.^2*Tb*(sinc(x/2)).^2.*(sin(pi*x/2)).^2;
figure
subplot(2,1,1);
plot(f,PSD_Manchester,'bl')
xlabel('f ---->')
ylabel('P(f)  ---->')
title('PSD for Manchester NRZ ');

%%  MLT3
PSD_MLT3=A.^2*Tb*(sinc(x/2)).^2.*(cos(pi*x)).^2;
subplot(2,1,2);
plot(f,PSD_MLT3,'bl');
xlabel('f ---->')
ylabel('P(f)  ---->')
title('MLT 3');
```


Non-return to zero
Non-return to zero inverted
Return to zero
AMI
P(f) ---->
f ---->
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0.5


PSD for Manchester NRZ
MLT 3
P(f) ---->
f ---->
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0.5

Another soln PSD:

```
%% --- NRZ-Inverted
figure
subplot(3,1,1)
x=plot(periodogram(NRZ_I,[],'centered',512,100));
title('NRZ-Inverted')
```

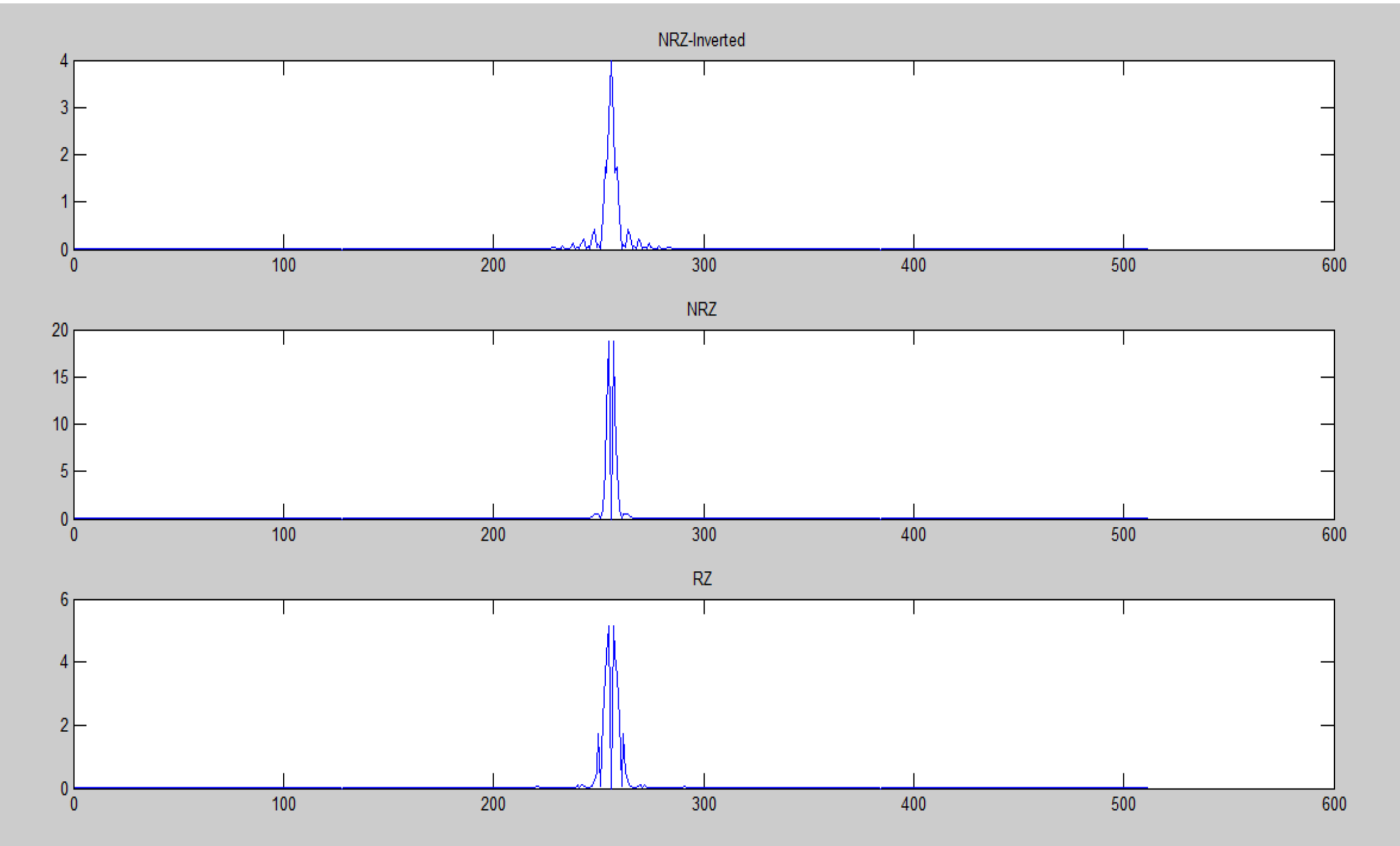
```
%% --- NRZ

subplot(3,1,2)
x=plot(  periodogram(Polar_NRZ,[],'centered',512,100));
title('NRZ')
```

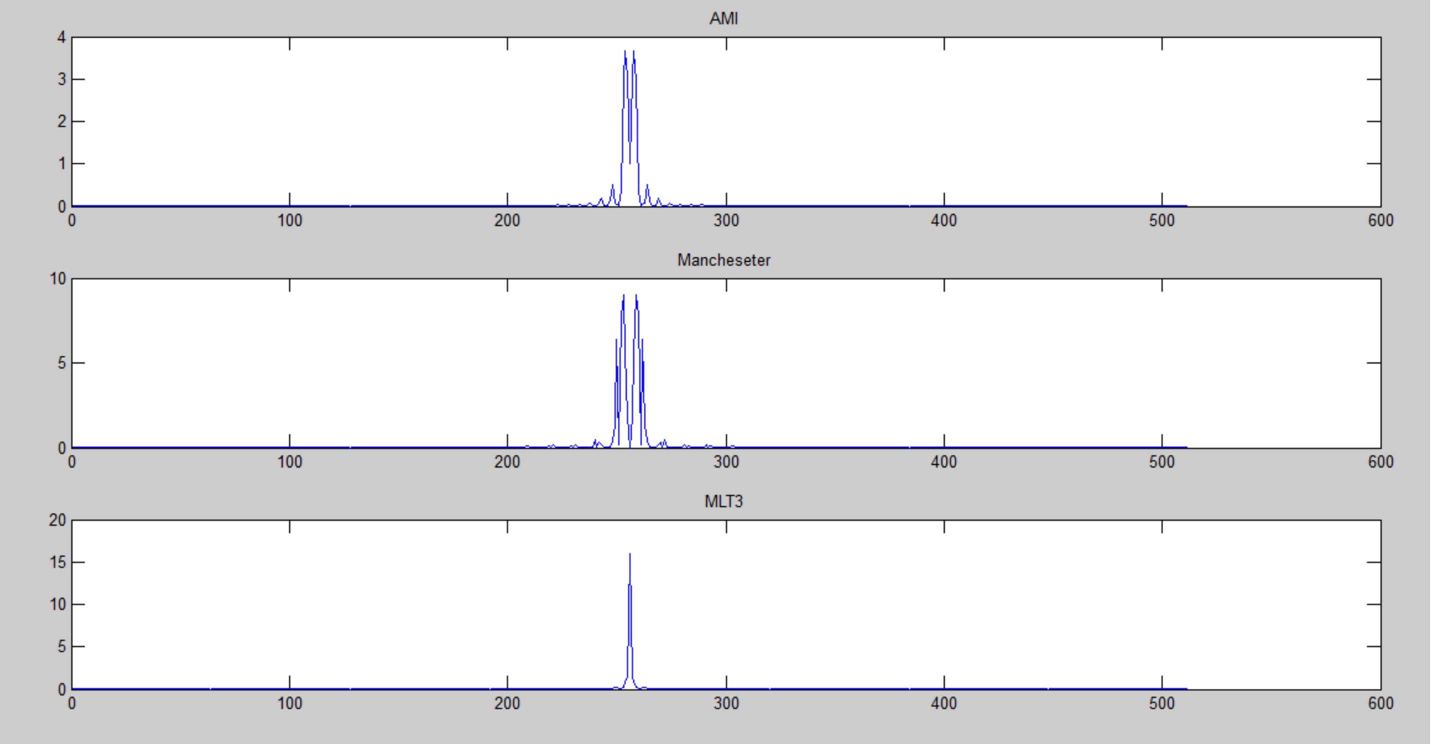
```
%% --- Rz

subplot(3,1,3)
x=plot(  periodogram(Polar_RZ,[],'centered',512,100));
title('RZ')
```

```
%% --- AMI
figure
subplot(3,1,1)
x=plot(  periodogram(Biolar_NRZ,[],'centered',512,100));
title('AMI')
```

```
%% --- Mancheseter

subplot(3,1,2)
x=plot(  periodogram(Mancheseter,[],'centered',512,100));
title('Mancheseter')
```

```
%% --- MLT3

subplot(3,1,3)
x=plot(  periodogram(MLT3,[],'centered',512,100));
title('MLT3')
```





Comments:

notes:

- In NRZ line coding, a binary 0 is represented by a constant "0" voltage level, while a binary "1" is represented by a constant non-zero voltage level.

- In RZ line coding, a binary "0" is represented by a zero voltage level, while a binary "1" is represented by a non-zero voltage level that returns to zero midway through the bit duration.

2- Polar:

Polar line codes are a type of line code that uses a positive signal represents a logical 1 and a negative signal represents a logical 0.

+V >> 1 -V >> 0

Advantages:

1- Better Bandwidth Efficiency: Polar line codes are more efficient than unipolar line codes, as they use less bandwidth to transmit the same amount of data.

2- No DC component: Polar NRZ has no DC component, which makes it suitable for long-distance transmission.

3-Simple: Polar NRZ is a simple line code to implement.

4-Easy to Implement: Polar RZ line codes are easy to implement and can be used in both hardware and software-based communication systems.

5- NRZ-I (NRZ-Inverted) helps in synchronization at the receiver due to use of transition to map binary '1'

6- Polar RZ solves synchronization problem observed in polar NRZ type.

Disadvantages:

1-Limited DC Balance: Polar RZ line codes do not have perfect DC balance, meaning that the number of positive and negative voltage transitions may not be equal.

2-Not self-clocking: Polar NRZ is not a self-clocking line code. This means that the receiver needs to have a separate clock signal to synchronize with the incoming data

-3 Polar RZ requires two signal changes to encode bit and hence it occupies more bandwidth.

3- Bipolar

Bipolar line codes are a type of line code that uses two voltage levels to represent logical 1 and logical 0 bits. The voltage level is positive or negative for logical 1 bits and zero voltage for logical 0 bits.

$\pm$ V >> 1

0 V >> 0

AMI (alternate mark inversion): AMI is a type of bipolar line code that uses a mark-space ratio of 1:1. This means that the number of mark (positive) symbols is equal to the number of space (negative) symbols.

Advantages:

1-No DC component: Bipolar line codes have no DC component, which makes them suitable for long-distance transmission.

2-Efficient: Bipolar line codes are efficient in terms of bandwidth usage.

3-DC Balanced: Bipolar line codes maintain a balance of positive and negative voltage levels.

4-Good Bandwidth Efficiency: Bipolar line codes are more bandwidth efficient than unipolar line codes because they use two voltage levels to represent two binary states.

5- Single error detection is possible using bipolar coding technique.

Disadvantages:

1- More complex: Bipolar line codes are more complex to implement than unipolar line codes

2- No clock signal is present for use.

3- Bipolar line codes require more power to transmit the signal since they use both positive and negative voltage levels.

4- Manchester

Manchester encoding is a type of line coding used in digital communication systems to represent binary data using transitions in the signal. In Manchester encoding, a binary 1 is represented by a transition from a positive voltage level to a negative voltage level, while a binary 0 is represented by a transition from a negative voltage level to a positive voltage level

advantage :
1- Bandwidth efficiency: Manchester encoding is bandwidth-efficient, which means that it uses less bandwidth than other line codes, such as unipolar NRZ. This is because the signal level changes at the middle of each bit time, which reduces the amount of time that the signal is at a constant level

2-Noise immunity: Manchester encoding is more noise-immune than other line codes,
3- Self-clocking: Manchester encoding is self-clocking, which means that the receiver can recover the clock signal from the signal itself.

Disadvantages:

1- Complexity: Manchester encoding is more complex to implement than other line codes, because the receiver needs to be able to detect the transitions in the signal.
2- Higher Bandwidth: Manchester encoding requires a higher bandwidth than other encoding techniques because it uses more transitions per bit.
3- Lower Spectral Efficiency: Manchester encoding has a lower spectral efficiency than other encoding techniques because it requires more bandwidth to transmit the same amount of data.

MLT-3 encoding:

MLT-3 encoding uses three levels (+V, 0, -V) and three transition rules to move between levels. It is similar to NRZ-I. Following rules are applied to encode bit pattern in the example below.
• If the next bit is zero ('0'), there is no transition.
• If the next bit is one ('1) and current level is not zero '0', the next level is 0.
• If the next bit is one ('1) and current level is zero ('0'), the next level is the opposite of the last nonzero level

Advantages:

1- It has signal rate which is $(1/4)^{th}$ of the bit rate
2- Due to its signal shape, it reduces required bandwidth.

Disadvantages:

1- It does not support self-synchronization for long string of zeros ('0').
2- It is more complex than NRZ-I due to use of three levels and complex transition rules.

Which type of line code has the highest bandwidth?
The type of line code with the highest bandwidth is Manchester encoding.

-Manchester encoding is a line code in which the signal level changes at the middle of each bit time. This ensures that there is always a transition in the signal, which makes it easy for the receiver to synchronize with the signal. Manchester encoding is also self-clocking, which means that the receiver can recover the clock signal from the signal itself. Manchester encoding has a bandwidth of twice the data rate.

This is because the signal level changes twice per bit time .

Mention 2 other used line codes and explain them mentioning the main advantages and disadvantages:

<u>Unipolar</u>:

In this type of line coding, the signal is represented by a single voltage level, usually positive.

Advantages:

1- It is simple encoding technique.

2- Unipolar NRZ : It requires less bandwidth for transmission.

3- Unipolar RZ : The spectral line present at the symbol rate can be used as clock signal.

Disadvantages:

1- Average amplitude of unipolar encoded signal is nonzero. This creates DC component which shifts zero level of the signal which can't travel through some medium such as microwave.

2-Unipolar encoding leads to synchronization issue at the receiver when long string of ones and zeros are present in the binary data. This is because such data do not produce any transitions which may create problems in error detection and recovery.

3- It does not have any error handling technique (i.e. Error correction)

4- It suffers from "Signal Droop" issue due to presence of low frequency components.

5- Unipolar RZ consumes twice bandwidth than unipolar NRZ.

<u>B8ZS (Bipolar with 8 Zero Substitution:</u>

- This code replaces any sequence of eight zeros with a special pattern consisting of alternating positive and negative pulses.

Advantages:

1- The coded signal does not contain D.C. component

2- It maintain synchronization

Disadvantages:

1- its complexity as well as its requirement for special hardware for implementation