Introduction to Digital Communication

EEC 382

Part 2

Name	ID
Hythem Ahmed Shaban Ahmed	19016856
Haitham Atia Elsayed	19016857
Ibrahim Abelwahab Mohamed	19015169

Alexandria University

Electronics and Communication Engineering

Introduction

Digital communication is the exchange of information using digital signals transmitted over communication channels such as wires, optical fibers, or wireless connections. To obtain digital signals from our analog world, we must convert analog signal to digital signals using ADC which uses pulse code modulation standard. PCM consists of 3 steps: sampling, quantizing, and encoding. Sampling is the process of converting continuous time analog signal to discrete time analog signal. Quantizing is the process of converting discrete time analog signal to discrete time digital signal. Coding is the process of giving each level of the quantized signal a code.

Objective

Compare the different types of line codes used in digital communications.

Theoretical Background

Line codes are essential to the process of encoding digital signals into a form that can be reliably transmitted and decoded at the receiver end. They are used to ensure accurate data transmission by minimizing errors caused by noise and other disturbances that can affect the communication channel.

There are various types of line codes that are used in digital communication, including unipolar, polar, bipolar, and Manchester codes, each with its unique advantages and limitations. The choice of the appropriate line code for a particular communication system depends on various factors such as the nature of the communication channel, the required data rate, and the level of error tolerance.

- Unipolar NRZ
  A High in data is represented by a positive pulse called as **Mark**, which has a duration T_0 equal to the symbol bit duration. A Low in data input has no pulse.

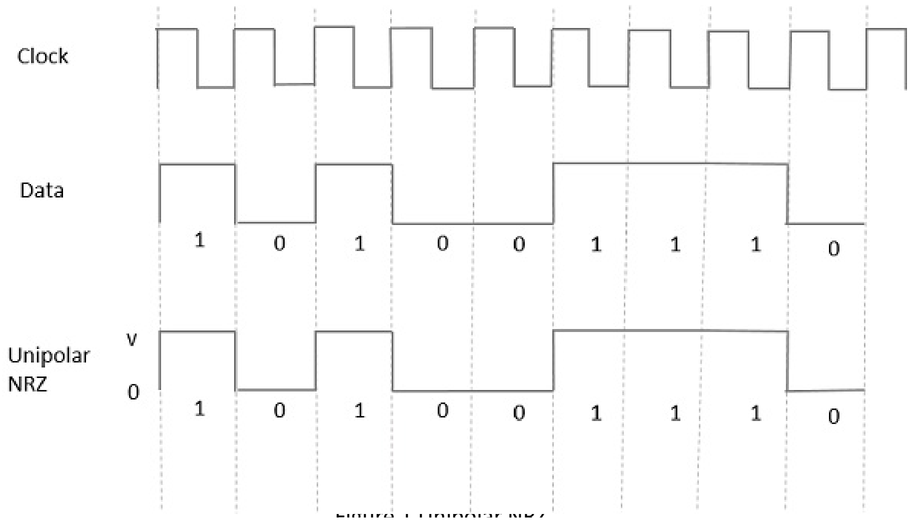


Figure 1 Unipolar NRZ

- Unipolar RZ
  A High in data, though represented by a **Mark pulse**, its duration T_0 is less than the symbol

bit duration. Half of the bit duration remains high but it immediately returns to zero and shows the absence of pulse during the remaining half of the bit duration.

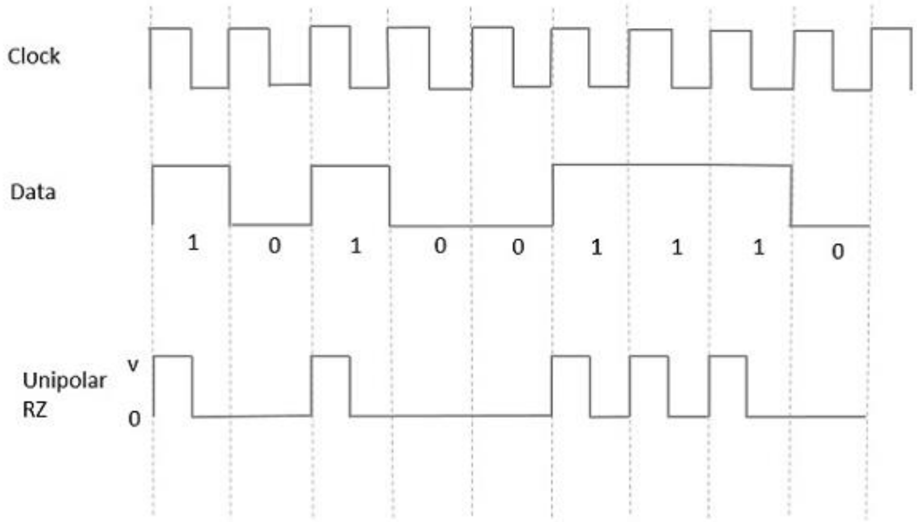


Figure 2 Unipolar RZ

- Polar NRZ (Alternative Mark Inversion)
  A High in data is represented by a positive pulse, while a Low in data is represented by a negative pulse.

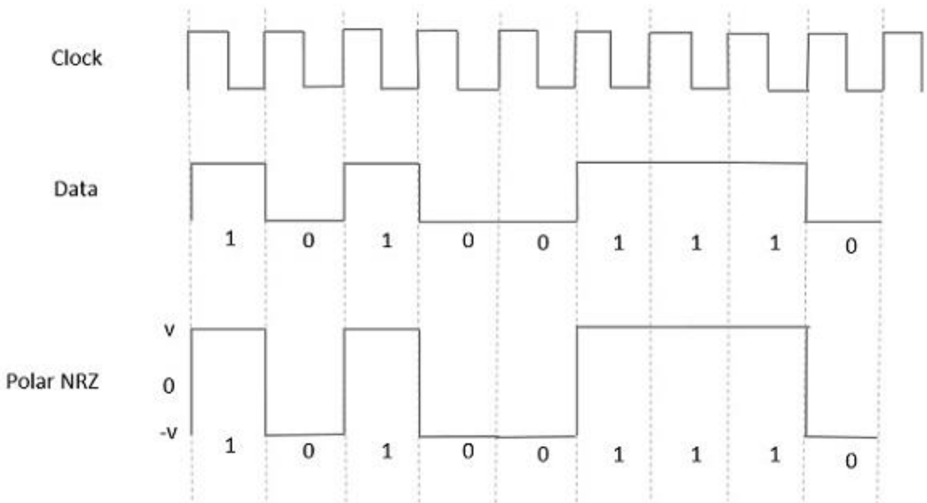


Figure 3 Polar NRZ

- Polar RZ
  a High in data, though represented by a **Mark pulse**, its duration T_0 is less than the symbol bit duration. Half of the bit duration remains high but it immediately returns to zero and shows the absence of pulse during the remaining half of the bit duration.
  However, for a Low input, a negative pulse represents the data, and the zero level remains same for the other half of the bit duration. The following figure depicts this clearly.
- Bipolar
  This is an encoding technique which has three voltage levels namely **+, -** and **0**. Such a signal is called as **duo-binary signal**.
  An example of this type is **Alternate Mark Inversion** AMI. For a **1**, the voltage level gets a

transition from + to – or from – to +, having alternate **1s** to be of equal polarity. A **0** will have a zero voltage level.

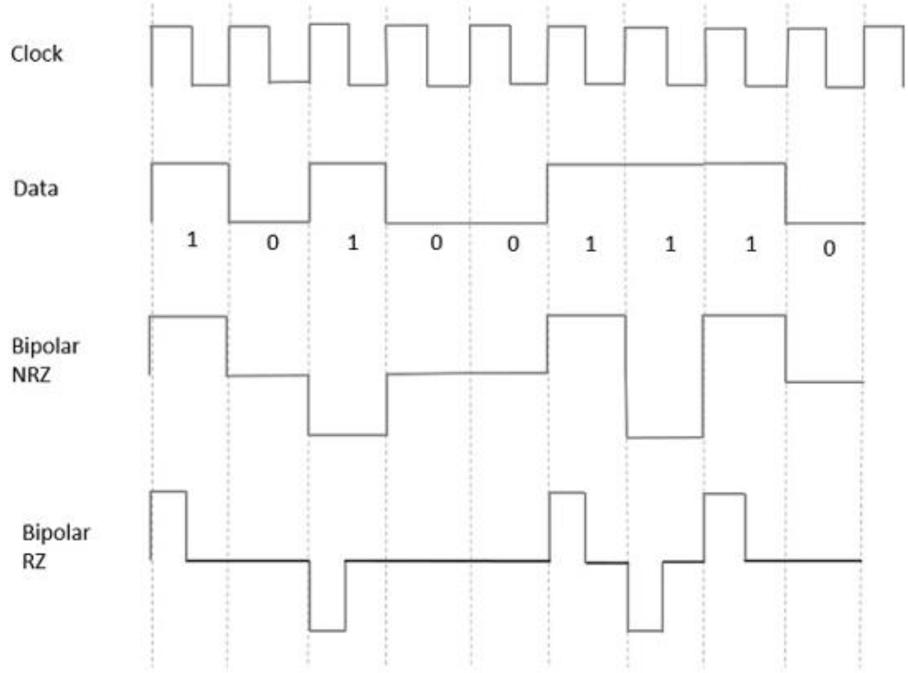


Figure 5 Bipolar

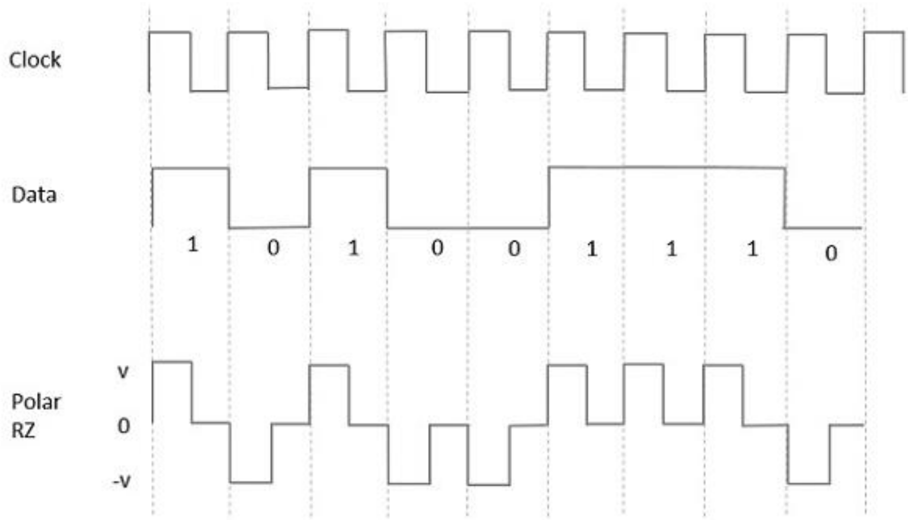


Figure 4 Polar RZ

- Manchester
  In this type of coding, the transition is done at the middle of the bit-interval. The transition for the resultant pulse is from High to Low in the middle of the interval, for the input bit 1. While the transition is from Low to High for the input bit **0**.

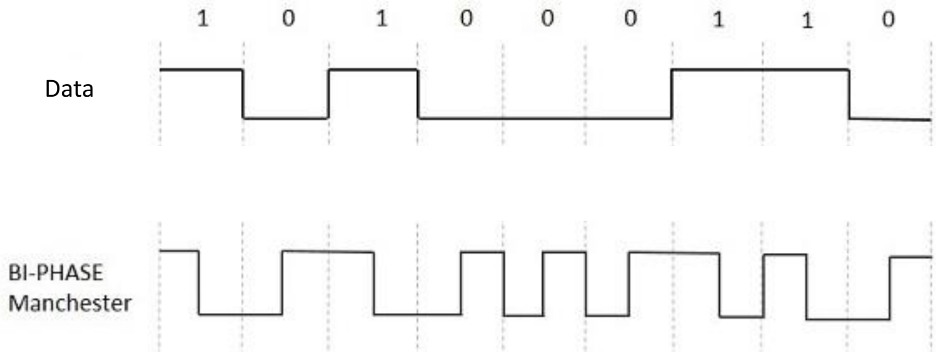


Figure 6 Manchester

- Multi-level Transmission 3
  MLT-3 cycles sequentially through the voltage levels −1, 0, +1, 0. It moves to the next state to transmit a 1 bit, and stays in the same state to transmit a 0 bit.

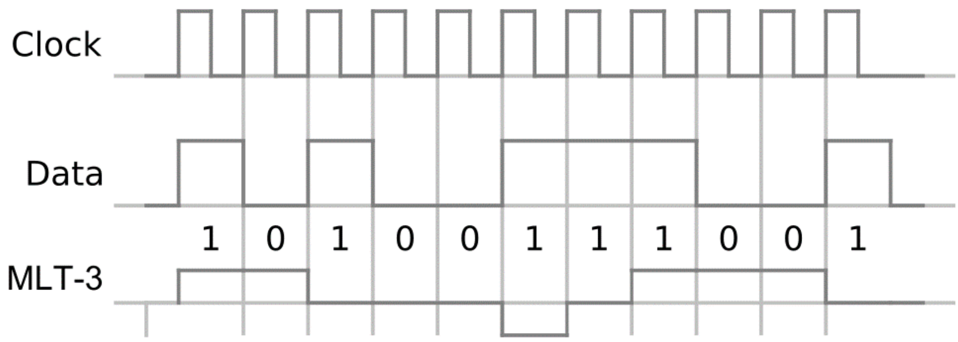


Figure 7 MLT-3

- Power Spectral Density
  It is the function which describes how the power of a signal got distributed at various frequencies, in the frequency domain.
  According to the Einstein-Wiener-Khintchine theorem, if the auto correlation function or power spectral density of a random process is known, the other can be found exactly.
  Hence, to derive the power spectral density, we shall use the time auto-correlation.

$$S(f) = |X(f)| \times \frac{1}{T_b}(\sum_{-\infty}^{\infty} R_n e^{-jn(2\pi f)T_b})$$

Procedures

1. Generate random bits of zeros and ones using randi function which takes the integer digits [0 1] and the number of digits to be generated

```
% Generate a random binary data signal
N = 10; % number of bits
bits = randi([0 1], 1, N);
```

2. Modulate this same vector using the different types of line codes.

```
% Define the line code parameters
Tb = 1; % bit period
fs = 1000; % sampling frequency (number of samples per bit)
t = 0:(1/fs):Tb*N-(1/fs); % time vector
```

a. Unipolar NRZ

```
UP_NRZ = zeros(1, fs*N);
% Generate the Unipolar NRZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        UP_NRZ((i-1)*fs+1:i*fs) = 0;
    else
        UP_NRZ((i-1)*fs+1:i*fs) = 1;
    end
end
```

b. Unipolar RZ

```
UP_RZ = zeros(1, fs*N);
% Generate the Unipolar RZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        UP_RZ((i-1)*fs+1:i*fs) = 0;
    else
        UP_RZ((i-1)*fs+1:(2*i-1)*fs/2) = 1;
        UP_RZ((2*i-1)*fs/2+1:i*fs) = 0;
    end
end
```

c. Polar NRZ

```
P_NRZ = zeros(1, fs*N);
% Generate the Polar NRZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        P_NRZ((i-1)*fs+1:i*fs) = -1;
    else
        P_NRZ((i-1)*fs+1:i*fs) = 1;
    end
end
```

d. Polar RZ

```
P_RZ = zeros(1, fs*N);
% Generate the Polar RZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        P_RZ((i-1)*fs+1:(2*i-1)*fs/2) = -1;
        P_RZ((2*i-1)*fs/2+1:i*fs) = 0;
    else
        P_RZ((i-1)*fs+1:(2*i-1)*fs/2) = 1;
        P_RZ((2*i-1)*fs/2+1:i*fs) = 0;
    end
end
```

e. Bipolar NRZ

```
BiP_NRZ = zeros(1, fs*N);
flag = 0; % it is used to memorize whether the last '1' bit is
modulated as +ve pulse or -ve pulse

% Generate the BiPolar NRZ (AMI) line code signal
for i = 1:length(bits)
    if bits(i) == 0
        BiP_NRZ((i-1)*fs+1:i*fs) = 0;
    else
        if flag == 0
            BiP_NRZ((i-1)*fs+1:i*fs) = 1;
            flag = 1;
        else
            BiP_NRZ((i-1)*fs+1:i*fs) = -1;
            flag = 0;
        end
    end
end
```

f. Manchester

```
Manchester = zeros(1, fs*N); % preallocate the code signal
% Generate the Manchester line code signal
for i = 1:length(bits)
    if bits(i) == 0
        Manchester((i-1)*fs+1:(2*i-1)*fs/2) = -1;
        Manchester((2*i-1)*fs/2+1:i*fs) = 1;
    else
        Manchester((i-1)*fs+1:(2*i-1)*fs/2) = 1;
        Manchester((2*i-1)*fs/2+1:i*fs) = -1;
    end
end
```

g. Multi-level Transmission 3

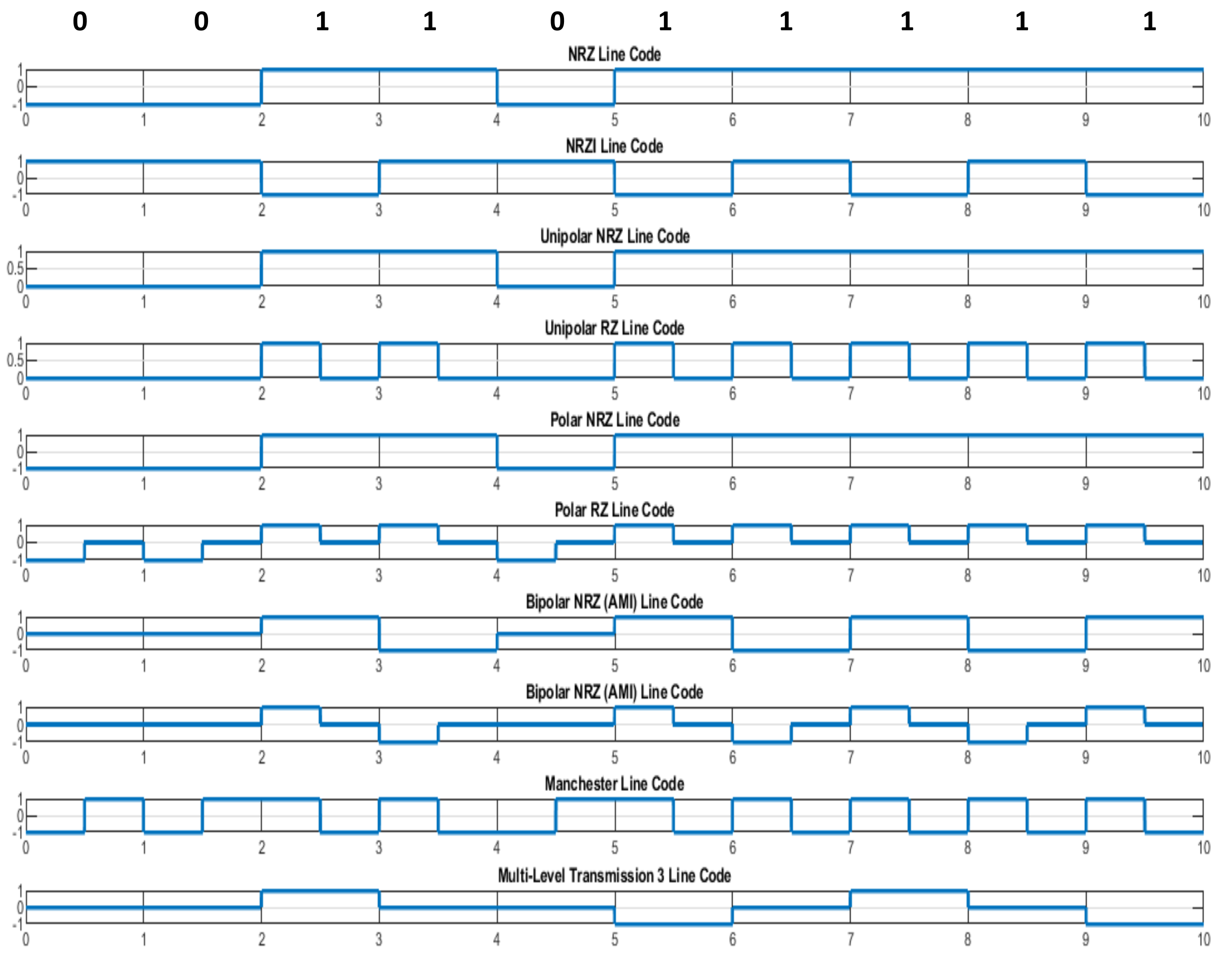
```
MultiLevel_Trans = zeros(1, fs*N);
prev_level = 0;
flag = 0;
% Generate the Multi-Level Transmission 3 line code signal
for i = 1:length(bits)
    if bits(i) == 0
        MultiLevel_Trans((i-1)*fs+1:i*fs)= prev_level;
        prev_level = MultiLevel_Trans(i*fs);
    else
        if prev_level == -1
            next_level = 0;
            flag = 0;
        elseif prev_level == 1
            next_level = 0;
            flag = 1;
        else
            if flag == 0
                next_level = 1;
                flag = 1;
            else
                next_level = -1;
                flag = 0;
            end
        end
        MultiLevel_Trans((i-1)*fs+1:i*fs)= next_level;
```

```
            prev_level = next_level;
        end
    end
```

3. Plot a sample of all previous line code modulation and plot them under each other on the same figure using subplot.
4. Find the power spectrum density of each code, and plot them in the same figure using subplot as previous. Pwelch function is used to estimate PSD of all line codes. And using PSD theoretical equations to plot all line codes.

Results

Plotting of different line codes

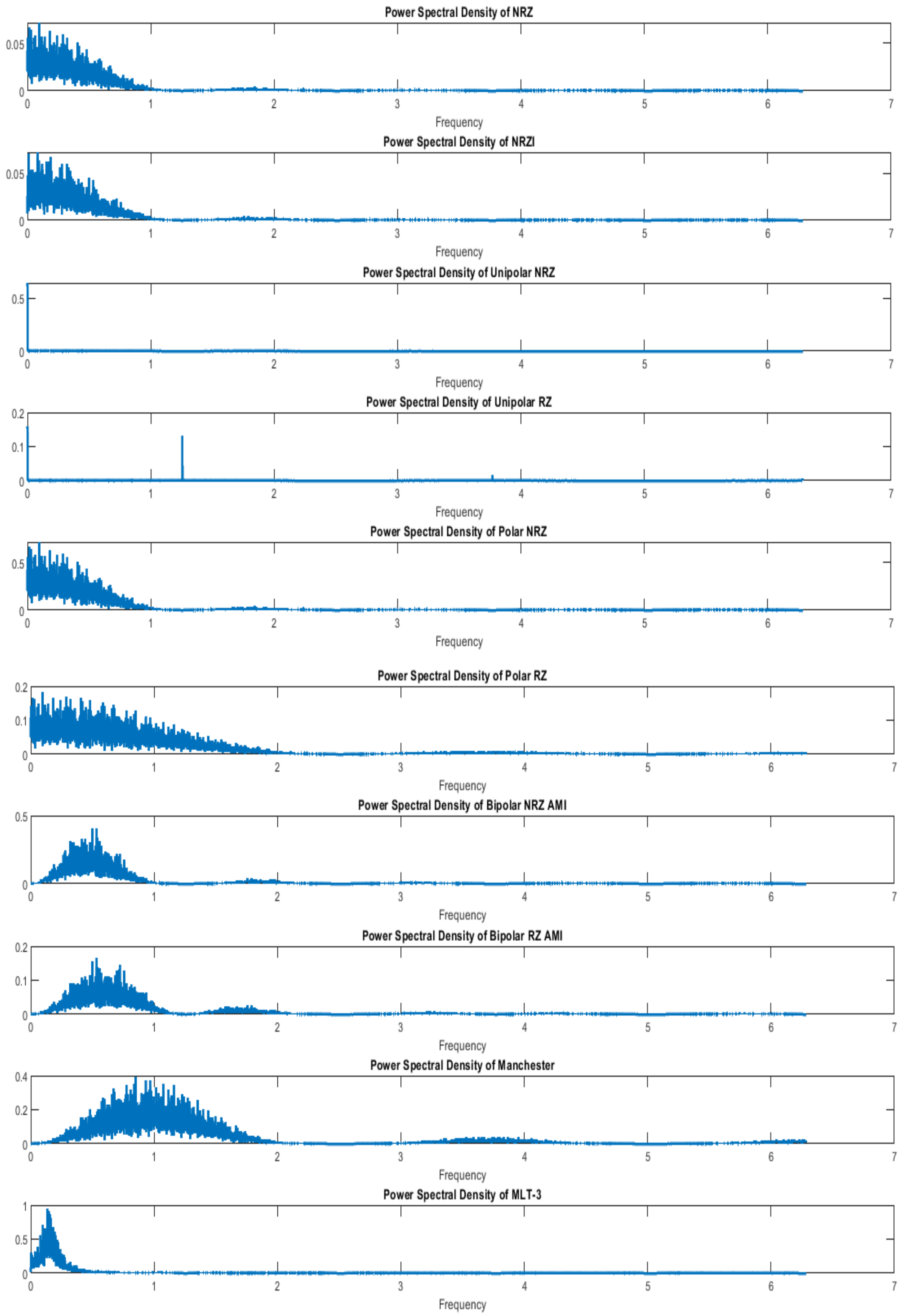


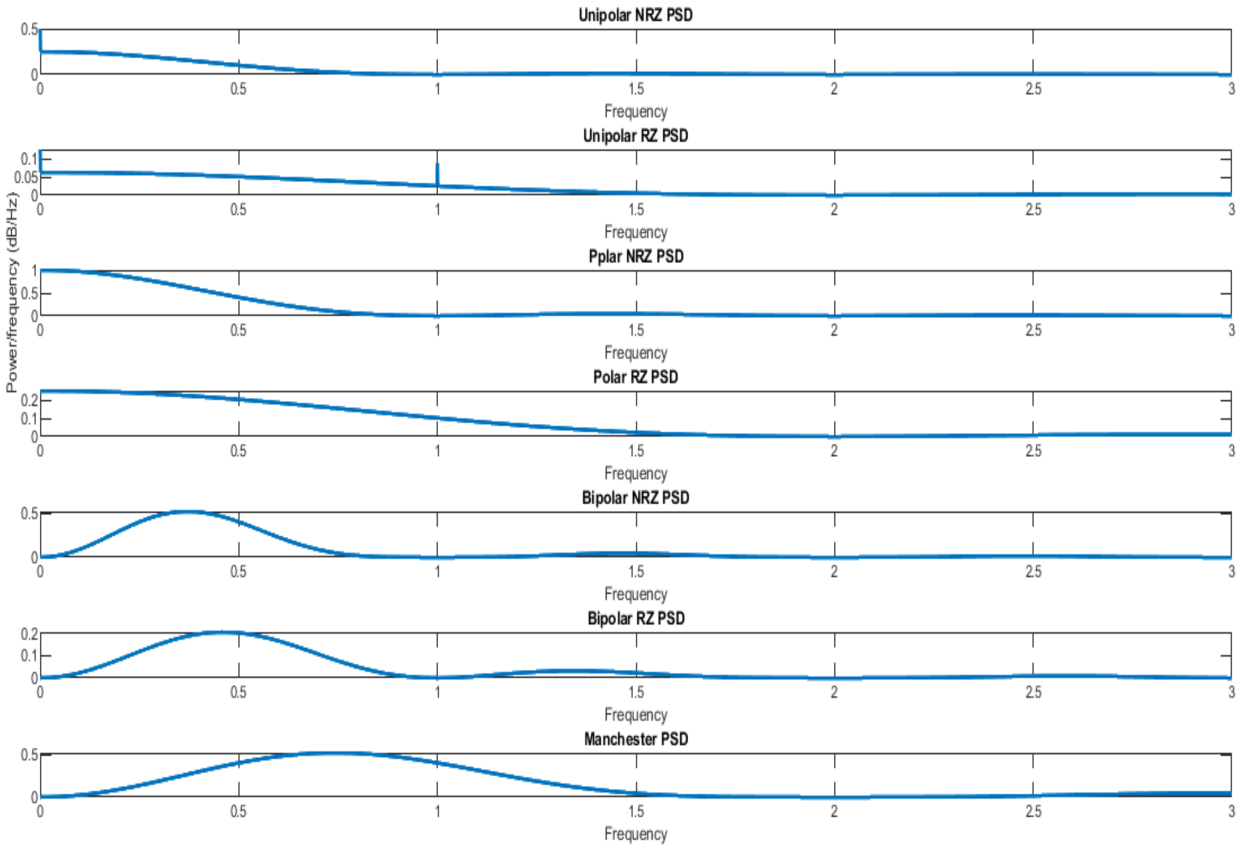
Line code of highest bandwidth

The line code of the highest bandwidth is Manchester encoding. Manchester encoding is a type of line coding that uses a transition in the middle of each bit period to represent a binary '1', and no transition to represent a binary '0'. This results in a signal that has twice the bandwidth of the original binary signal.

However, MLT-3 has a higher spectral efficiency because it can encode two bits per signal level change, which results in a more efficient use of bandwidth.

Plotting of Power Spectral Density





Comment:

1. NRZ (Non-Return-to-Zero): NRZ is a simple line coding scheme that uses a constant voltage level to represent a binary '1' and a zero voltage level to represent a binary '0'. The PSD of NRZ is a flat line at the baseband frequency with sidelobes that extend to higher frequencies. The bandwidth of NRZ is equal to the data rate.

2. RZ (Return-to-Zero): RZ is a line coding scheme that uses a positive or negative voltage pulse to represent a binary '1' and a zero voltage level to represent a binary '0'. The PSD of RZ has a peak at the baseband frequency and two sidelobes at higher frequencies. The bandwidth of RZ is twice the data rate.
3. Manchester: Manchester encoding uses a transition in the middle of each bit period to represent a binary '1' and no transition to represent a binary '0'. The PSD of Manchester has a peak at the baseband frequency and two additional peaks at frequencies that are one-half the data rate above and below the baseband frequency. The bandwidth of Manchester is twice the data rate.
4. AMI (Alternate Mark Inversion): AMI is a line coding scheme that uses a zero voltage level to represent a binary '0', and alternates between positive and negative voltage levels to represent successive binary '1's. The PSD of AMI has a peak at the baseband frequency and nulls at frequencies that are odd multiples of one-half the data rate. The bandwidth of AMI is equal to the data rate.
5. MLT-3 (Multi-Level Transmit): MLT-3 is a multi-level line coding scheme that uses three signal levels to encode binary data. The PSD of MLT-3 has a peak at the baseband frequency and

nulls at frequencies that are odd multiples of one-third the data rate. The bandwidth of MLT-3 is equal to the data rate.

Each line code has a different PSD that affects the system's bandwidth and noise immunity. The choice of line code depends on the specific requirements of the communication system, such as data rate, noise immunity, power consumption, and synchronization requirements.

Advantages and Disadvantages of different line codes

	Advantages	Disadvantages
NRZ	• Simple to implement and decode. • Good for short distances and low data rates. • No DC component in the signal.	• Poor noise immunity due to the lack of signal transitions. • Clock synchronization issues over long distances. • Susceptible to baseline wander and long runs of 0s or 1s
NRZI	• Provides signal transitions that improve noise immunity. • DC balance in the signal for better transmission over long distances. • Can be used for clock synchronization.	• More complex to implement and decode than NRZ. • Requires a means to distinguish between 0s and 1s when encoding. • Clock recovery can be difficult if transitions are missing or distorted.
Unipolar NRZ	• Simple to implement and decode. • No need for a bipolar signal, reducing complexity and cost. • Efficient use of bandwidth.	• Poor noise immunity due to the lack of signal transitions • Clock synchronization issues over long distances • Susceptible to baseline wander and long runs of 0s or 1s • DC drift can occur, leading to errors in decoding
Unipolar RZ	• Provides signal transitions that improve noise immunity. • DC balance in the signal for better transmission over long distances. • Can be used for clock synchronization.	• Requires twice the bandwidth of NRZ due to the need for signal transitions. • More complex to implement and decode than NRZ. • Clock recovery can be difficult if transitions are missing or distorted.
Polar NRZ	• Simple to implement and decode • No need for a bipolar signal, reducing complexity and cost • Efficient use of bandwidth • Provides good noise immunity due to the signal transitions	• Clock synchronization issues over long distances • Susceptible to baseline wander and long runs of 0s or 1s • DC drift can occur, leading to errors in decoding
Polar RZ	• Provides signal transitions that improve noise immunity • DC balance in the signal for better transmission over long distances • Can be used for clock synchronization	• Requires twice the bandwidth of NRZ due to the need for signal transitions • More complex to implement and decode than NRZ • Clock recovery can be difficult if transitions are missing or distorted
Bipolar NRZ	• Provides good noise immunity due to the signal transitions • DC balanced, which eliminates the problem of long runs of 0s or 1s • Can be used for clock synchronization	• More complex to implement and decode than unipolar NRZ • Requires a bipolar signal, which increases complexity and cost • Clock synchronization issues over long distances

	• Can carry both positive and negative data	• DC drift can occur, leading to errors in decoding
Bipolar RZ	• Provides signal transitions that improve noise immunity • DC balanced in the signal for better transmission over long distances • Can be used for clock synchronization • Can carry both positive and negative data	• Requires twice the bandwidth of bipolar NRZ due to the need for signal transitions • More complex to implement and decode than bipolar NRZ • Clock recovery can be difficult if transitions are missing or distorted
Manchester	• Provides a clock signal that can be extracted from the transitions. • Balanced DC levels for better noise immunity. • Easy to implement and decode.	• Wastes bandwidth due to the need for transitions in each bit period. • Requires twice the bandwidth of NRZ. • Clock recovery can be difficult if transitions are missing or distorted.
MLT-3	• Efficient use of bandwidth • Balanced DC levels for better noise immunity • Good for high data rates and long distances	• Requires more complex encoding and decoding circuitry. • Clock synchronization issues over long distances. • Susceptible to errors if there are long runs of 0s or 1s.

Other line codes

1. 4B/5B Encoding

4B/5B encoding is a line coding scheme used in digital communication to encode 4-bit binary data into 5-bit symbols. The encoding scheme uses a lookup table to convert 4-bit data into a 5-bit symbol, resulting in a 25% overhead in bandwidth.

Here is how 4B/5B encoding works:

- The input data is divided into 4-bit nibbles.
- Each nibble is looked up in a pre-defined table to find the corresponding 5-bit symbol.
- The 5-bit symbol is then transmitted on the communication channel.
- At the receiver end, the 5-bit symbols are decoded back into 4-bit data using the same lookup table.

2. 8B/6T Encoding

8B/6T encoding is a line coding scheme used in digital communication to convert 8-bit binary data into 6-bit symbols. The encoding scheme uses a lookup table to convert 8-bit data into a 6-bit symbol, resulting in a 25% reduction in the bandwidth.

Here is how 8B/6T encoding works:

- The input data is divided into 8-bit bytes.
- Each byte is looked up in a pre-defined table to find the corresponding 6-bit symbol.
- The 6-bit symbol is then transmitted on the communication channel.

- At the receiver end, the 6-bit symbols are decoded back into 8-bit data using the same lookup table.

	Advantages	Disadvantages
4B/5B Encoding	• DC balance. • Error detection. • Clock recovery.	• Bandwidth overhead. • Complexity.
8B/6T Encoding	• Bandwidth reduction. • DC balance. • Error detection.	• Complexity. • Limited symbol set

Overall, 4B/5B encoding is a useful line coding scheme for applications where DC balance, error detection, and clock recovery are important considerations, but the additional bandwidth overhead and complexity should be taken into account when considering its use.

However, 8B/6T encoding is a useful line coding scheme for applications where bandwidth reduction, DC balance, and error detection are important considerations. However, the additional complexity and limited symbol set should be taken into account when considering its use.

MATLAB Code

```
clear
clc
% Generate a random binary data signal
N = 10000; % number of bits
bits = randi([0 1], 1, N);
% Define the line code parameters
Tb = 1; % bit period
fs = 10; % sampling frequency (number of samples per bit)
t = 0:(1/fs):Tb*N-(1/fs); % time vector
%%
for i = 1:length(bits)
    clk((i-1)*fs+1:(2*i-1)*fs/2) = 1;
    clk((2*i-1)*fs/2+1:i*fs) = 0;
end
% Plot the input bits signal
figure
subplot(11,1,1)
plot(t, clk, 'LineWidth', 2);
grid on;

title('clk Signal');
%%
NRZ = zeros(1, fs*N);
% Generate the NRZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        NRZ((i-1)*fs+1:i*fs) = -1;
    else
        NRZ((i-1)*fs+1:i*fs) = 1;
    end
end

% Plot the NRZ line code signal
subplot(11,1,2)
```

```matlab
plot(t, NRZ, 'LineWidth', 2);
grid on;

title('NRZ Line Code');
%%
NRZI = zeros(1, fs*N);
current_state = 1;
% Generate the NRZI line code signal
for i = 1:length(bits)
    if bits(i) == 0
        NRZI((i-1)*fs+1:i*fs) = current_state;
    else
        NRZI((i-1)*fs+1:i*fs) = - current_state;
        current_state = - current_state;
    end
end

% Plot the NRZI line code signal
subplot(11,1,3)
plot(t, NRZI, 'LineWidth', 2);
grid on;

title('NRZI Line Code');
%%
UP_NRZ = zeros(1, fs*N);
% Generate the Unipolar NRZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        UP_NRZ((i-1)*fs+1:i*fs) = 0;
    else
        UP_NRZ((i-1)*fs+1:i*fs) = 1;
    end
end

% Plot the Unipolar NRZ line code signal
subplot(11,1,4)
plot(t, UP_NRZ, 'LineWidth', 2);
grid on;


title('Unipolar NRZ Line Code');

%%
UP_RZ = zeros(1, fs*N);
% Generate the Unipolar RZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        UP_RZ((i-1)*fs+1:i*fs) = 0;
    else
        UP_RZ((i-1)*fs+1:(2*i-1)*fs/2) = 1;
        UP_RZ((2*i-1)*fs/2+1:i*fs) = 0;
    end
end

% Plot the Unipolar RZ line code signal
subplot(11,1,5)
plot(t, UP_RZ, 'LineWidth', 2);
grid on;
```

```
title('Unipolar RZ Line Code');
%%
P_NRZ = zeros(1, fs*N);
% Generate the Polar NRZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        P_NRZ((i-1)*fs+1:i*fs) = -1;
    else
        P_NRZ((i-1)*fs+1:i*fs) = 1;
    end
end

% Plot the Polar NRZ line code signal
subplot(11,1,6)
plot(t, P_NRZ, 'LineWidth', 2);
grid on;


title('Polar NRZ Line Code');
%%
P_RZ = zeros(1, fs*N);
% Generate the Polar RZ line code signal
for i = 1:length(bits)
    if bits(i) == 0
        P_RZ((i-1)*fs+1:(2*i-1)*fs/2) = -1;
        P_RZ((2*i-1)*fs/2+1:i*fs) = 0;
    else
        P_RZ((i-1)*fs+1:(2*i-1)*fs/2) = 1;
        P_RZ((2*i-1)*fs/2+1:i*fs) = 0;
    end
end

% Plot the Polar RZ line code signal
subplot(11,1,7)
plot(t, P_RZ, 'LineWidth', 2);
grid on;


title('Polar RZ Line Code');
%%
BiP_NRZ = zeros(1, fs*N);
flag = 0;
% Generate the Bipolar NRZ (AMI) line code signal
for i = 1:length(bits)
    if bits(i) == 0
        BiP_NRZ((i-1)*fs+1:i*fs) = 0;
    else
        if flag == 0
            BiP_NRZ((i-1)*fs+1:i*fs) = 1;
            flag = 1;
        else
            BiP_NRZ((i-1)*fs+1:i*fs) = -1;
            flag = 0;
        end
    end
end

% Plot the Bipolar NRZ (AMI) line code signal
subplot(11,1,8)
plot(t, BiP_NRZ, 'LineWidth', 2);
```

```
grid on;


title('Bipolar NRZ (AMI) Line Code');
%%
BiP_RZ = zeros(1, fs*N);
flag = 0;
% Generate the Bipolar RZ (AMI) line code signal
for i = 1:length(bits)
    if bits(i) == 0
        BiP_RZ((i-1)*fs+1:i*fs) = 0;
    else
        if flag == 0
            BiP_RZ((i-1)*fs+1:(2*i-1)*fs/2) = 1;
            BiP_RZ((2*i-1)*fs/2+1:i*fs) = 0;
            flag = 1;
        else
            BiP_RZ((i-1)*fs+1:(2*i-1)*fs/2) = -1;
            BiP_RZ((2*i-1)*fs/2+1:i*fs) = 0;
            flag = 0;
        end
    end
end

% Plot the Bipolar RZ (AMI) line code signal
subplot(11,1,9)
plot(t, BiP_RZ, 'LineWidth', 2);
grid on;


title('Bipolar NRZ (AMI) Line Code');
%%
Manchester = zeros(1, fs*N); % preallocate the code signal
% Generate the Manchester line code signal
for i = 1:length(bits)
    if bits(i) == 0
        Manchester((i-1)*fs+1:(2*i-1)*fs/2) = -1;
        Manchester((2*i-1)*fs/2+1:i*fs) = 1;
    else
        Manchester((i-1)*fs+1:(2*i-1)*fs/2) = 1;
        Manchester((2*i-1)*fs/2+1:i*fs) = -1;
    end
end

% Plot the Manchester line code signal
subplot(11,1,10)
plot(t, Manchester, 'LineWidth', 2);
grid on;


title('Manchester Line Code');
%%
MultiLevel_Trans = zeros(1, fs*N);
prev_level = 0;
flag = 0;
% Generate the Multi-Level Transmission 3 line code signal
for i = 1:length(bits)
    if bits(i) == 0
        MultiLevel_Trans((i-1)*fs+1:i*fs)= prev_level;
        prev_level = MultiLevel_Trans(i*fs);
```

```matlab
    else
        if prev_level == -1
            next_level = 0;
            flag = 0;
        elseif prev_level == 1
            next_level = 0;
            flag = 1;
        else
            if flag == 0
                next_level = 1;
                flag = 1;
            else
                next_level = -1;
                flag = 0;
            end
        end
        MultiLevel_Trans((i-1)*fs+1:i*fs)= next_level;
        prev_level = next_level;
    end
end

% Plot the Multi-Level Transmission 3 line code signal
subplot(11,1,11)
plot(t, MultiLevel_Trans, 'LineWidth', 2);
grid on;

title('Multi-Level Transmission 3 Line Code');
%%
%PSD of NRZ

% Calculate the PSD using the Welch method
[NRZ_psd, f] = pwelch(NRZ);

% Plot the PSD
figure
subplot(5,1,1)
plot(2*f,(NRZ_psd)./100,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of NRZ');
%%
%PSD of NRZI

% Calculate the PSD using the Welch method
[NRZI_psd, f] = pwelch(NRZI);

% Plot the PSD
subplot(5,1,2)
plot(2*f,(NRZI_psd)./100,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of NRZI');
%%
%PSD of Unipolar NRZ

% Calculate the PSD using the Welch method
[UP_NRZ_psd, f] = pwelch(UP_NRZ);

% Plot the PSD
subplot(5,1,3)
plot(2*f,(UP_NRZ_psd)./1000,'LineWidth', 2);
xlabel('Frequency');
```

```
title('Power Spectral Density of Unipolar NRZ');
%%
%PSD of Unipolar RZ

% Calculate the PSD using the Welch method
[UP_RZ_psd, f] = pwelch(UP_RZ);

% Plot the PSD
subplot(5,1,4)
plot(2*f, (UP_RZ_psd)./1000,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of Unipolar RZ');
%%
%PSD of Polar NRZ

% Calculate the PSD using the Welch method
[P_NRZ_psd, f] = pwelch(P_NRZ);

% Plot the PSD
subplot(5,1,5)
plot(2*f,(P_NRZ_psd)./10,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of Polar NRZ');
%%
%PSD of Polar RZ

% Calculate the PSD using the Welch method
[P_RZ_psd, f] = pwelch(P_RZ);

% Plot the PSD
figure
subplot(5,1,1)
plot(2*f,(P_RZ_psd)./10,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of Polar RZ');
%%
%PSD of Bipolar NRZ AMI

% Calculate the PSD using the Welch method
[BiP_NRZ_psd, f] = pwelch(BiP_NRZ);

% Plot the PSD
subplot(5,1,2)
plot(2*f,(BiP_NRZ_psd)./10,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of Bipolar NRZ AMI');
%%
%PSD of Bipolar RZ AMI

% Calculate the PSD using the Welch method
[BiP_RZ_psd, f] = pwelch(BiP_RZ);

% Plot the PSD
subplot(5,1,3)
plot(2*f,(BiP_RZ_psd)./10,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of Bipolar RZ AMI');
%%
%PSD of Manchester
```

```
% Calculate the PSD using the Welch method
[Manchester_psd, f] = pwelch(Manchester);
% Plot the PSD
subplot(5,1,4)
plot(2*f, (Manchester_psd)./10,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of Manchester');
%%
%PSD of MLT-3

% Calculate the PSD using the Welch method
[MultiLevel_Trans_psd, f] = pwelch(MultiLevel_Trans);

% Plot the PSD
subplot(5,1,5)
plot(2*f,(MultiLevel_Trans_psd)./10,'LineWidth', 2);
xlabel('Frequency');
title('Power Spectral Density of MLT-3');
```

Theoretical PSD:

```
clear
clc

Tb = 1;
f = 0:0.001:3;
A = 1;

UP_NRZ_PSD = (A^2 * Tb/4) .* sinc(f * Tb).^2 + ((A^2) / 4) * (f==0);
UP_RZ_PSD = (A^2 * Tb/16) .* (sinc(f * Tb/2)).^2 + (A^2 / 16)*(f==0) + (A^2
/ 16)*(f==Tb);
P_NRZ_PSD = (A^2 * Tb) .* (sinc(f * Tb)).^2 ;
P_RZ_PSD = (A^2 * Tb/4) .* (sinc(f * Tb/2)).^2 ;
BP_NRZ_PSD = (A^2 * Tb) .* sinc(f * Tb).^2 .* sin(pi * f * Tb).^2;
BP_RZ_PSD = (A^2 * Tb/4) .* sinc(f * Tb/2).^2 .* sin(pi * f * Tb).^2;
Manchester_PSD = (A^2 * Tb) .* sinc(f * Tb/2).^2 .* sin(pi * f * Tb/2).^2;


% Plot the PSD
figure
subplot(7,1,1)
plot(f, (UP_NRZ_PSD),'LineWidth', 2);
xlabel('Frequency');
title('Unipolar NRZ PSD');

subplot(7,1,2)
plot(f, (UP_RZ_PSD),'LineWidth', 2);
xlabel('Frequency');
title('Unipolar RZ PSD');

subplot(7,1,3)
plot(f, (P_NRZ_PSD),'LineWidth', 2);
xlabel('Frequency');
ylabel('Power/frequency (dB/Hz)');
title('Pplar NRZ PSD');

subplot(7,1,4)
plot(f, (P_RZ_PSD),'LineWidth', 2);
```

```
xlabel('Frequency');
title('Polar RZ PSD');

subplot(7,1,5)
plot(f, (BP_NRZ_PSD),'LineWidth', 2);
xlabel('Frequency');
title('Bipolar NRZ PSD');

subplot(7,1,6)
plot(f, (BP_RZ_PSD),'LineWidth', 2);
xlabel('Frequency');
title('Bipolar RZ PSD');

subplot(7,1,7)
plot(f, (Manchester_PSD),'LineWidth', 2);
xlabel('Frequency');
title('Manchester PSD');
```