7. SIMULATION OF PULSE CODE MODULATION AND DEMODULATION

AIM:
Simulate generation of a PCM signal and its demodulation to retrieve the information.

Pre Lab

1. What is meant by uniform and non-uniform quantization?
2. What are the quantization standards followed in different parts of the world?
3. What is the role of a CODEC in association with PCM and being used in computers?

THEORY:

Pulse code modulation is a process of converting an analog signal into digital. The voice or any data input is first sampled using a sampler (which is a simple switch) and then quantized. Quantization is the process of converting a given signal amplitude to an equivalent binary number with fixed number of bits. This quantization can be either mid-tread or mid- rise and it can be uniform or non-uniform based on the requirements. For example in speech signals, the higher amplitudes will be less frequent than the low amplitudes. So higher amplitudes are given less step size than the lower amplitudes and thus quantization is performed non-uniformly. After quantization the signal is digital and the bits are passed through a parallel to serial converter and then launched into the channel serially.

At the demodulator the received bits are first converted into parallel frames and each frame is de-quantized to an equivalent analog value. This analog value is thus equivalent to a sampler output. This is the demodulated signal.

Uniform PCM

In uniform PCM the interval [-Xmax, Xmax] of length 2 Xmax is divided into N equal subintervals, each of length Δ= (2Xmax) / N. If N is a power of 2 or N=2v, then v bits are required for representation of each level.

After quantization, the quantized levels are encoded using v bits for each quantized level. The encoding scheme that is usually employed is natural binary coding (NBC), meaning that the lowest level is mapped into a sequence of all 0's and the highest level is mapped into a sequence of all 1's. All the other levels are mapped in increasing order of the quantized value.
Figure 2.1 shows an analog signal which amplitude lies in the range of [-Xmax, Xmax]. The number of levels to which the signal is quantized is N= 8. In this case, each sample requires v=3 bits to represent digitally.

The message is periodically sampled and digitally encoded in PCM. Since the signal is digitally encoded, only certain discrete voltage levels can be represented and there will be some error in encoding a random analog signal. This error is known as the 'quantization error'. The ratio of the signal power to quantization error power is generally termed as SQNR (Signal to Quantization Noise Ratio).

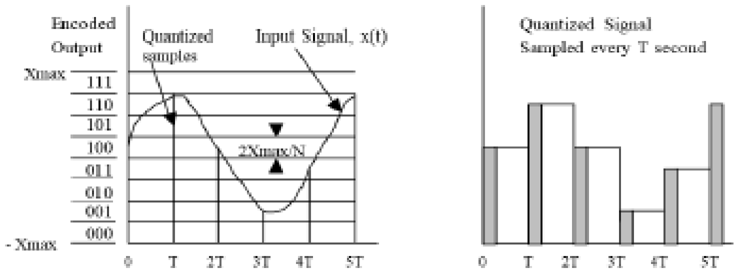


Figure 2.1 Quantization of a sampled analog signal

Simulation

In this simulation, we will quantize a sinewave signal (message) and encode it to binary bits. We will reconstruct the sinewave from binary bits. We will also calculate the SQNR for the 8-level and 16-level quantization. For the simulation, generate a sinusoidal signal with amplitude 1, and w=1. Using a uniform PCM scheme, quantize it once to 8 levels and once to 16 levels. Plot the original signal and the quantized signals on the same axis. Compare the resulting SQNRs in the two cases.

We arbitrarily choose the duration of the signal to be 10s. The resulting SQNRs are 18.90 dB for the 8-level PCM and 25.13 dB for 16-level PCM, compare these values with the results that you get from your simulation. The plots are shown in figure 2.2.

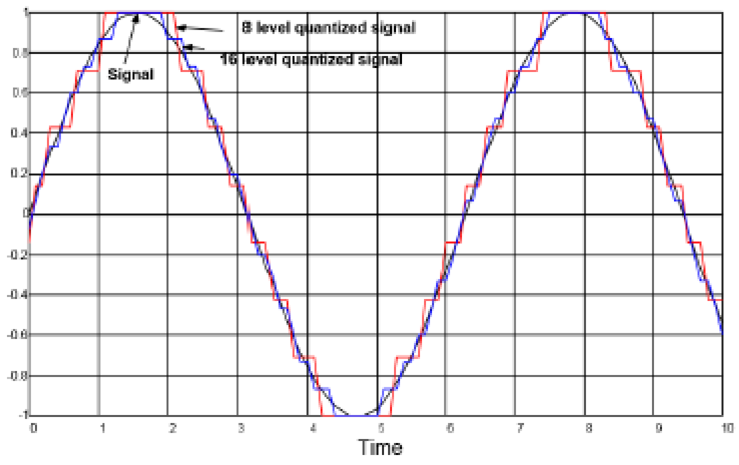


Figure 2.2 Sinusoidal signal before and after quantization.

Simulation Procedure

The following procedure only gives you an idea of how the PCM modulation and de-modulation are performed (actual implementation may vary).

1. Construct a time array from 0 to 10 sec with 0.1 intervals. Name it ‘t’.
2. Find the size of the array t. Use ‘length’ function. Name the size ‘m’.
3. Construct signal array ‘a’. Signal is a sinewave with unity amplitude and angular frequency.
4. Assign the number of quantization level n equal to 8.
5. Calculate, amax = max (abs (a)). See figure 2.3(a).
6. Calculate the following as shown in figure

   b = a + amax
   c = (n-1) x (b/(2amax))
   d = round(c)
   a_quan = 2 *amax * d/(n-1) – amax
   a_error = a-a_quan

7. Calculate $S = \sum_{i=1}^{m} a(i)\wedge 2$

$$N = \sum_{i=1}^{m} a\_error(i)\wedge 2$$

8. Calculate Signal to noise ratio, SQNR=10 log10(S/N)
9. Calculate binary coding to corresponding quantized value using dec2 bin function.
10. Repeat 4-9 with quantization level n=16.
11. Plot the input signal and quantized signal as in Figure 2.2. Use black solid line for signal, red solid line for 8 levels quantized and blue solid line for 16 level quantized signals. Use ‘figure’ function for multiple figures (you have to plot another figure in step 12.

12. Plot your variables a, b, c, d, a_quant and a_error for quantization level 8. Show your plots exactly as shown in the Figure 2.3 in a single template. Use ‘subplot’ function for that.
13. Compare the two quantized signals. Which is more close to input signal?
14. Tabulate few sample binary codes for both cases.
15. Reconstruct the original sinewave from the binary codes (both) using the Matlab function ‘bin2dec’. How is it different from the original signal?

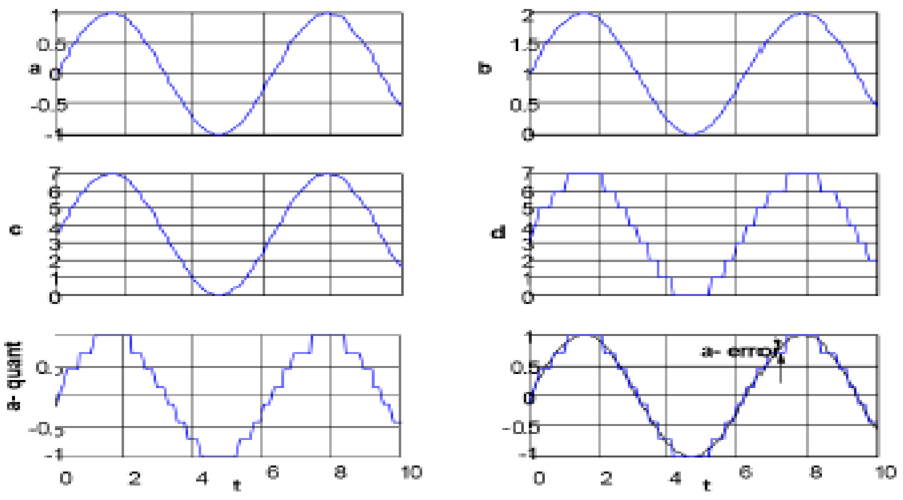


Figure 2.3 Steps of quantization in simulation

Model Code:

```
clc;
close all;
clear all;
n=input('Enter n value for n-bit PCM system : ');
n1=input('Enter number of samples in a period : ');
L=2^n;
% Sampling
x=0:2*pi/n1:4*pi; % n1 number of samples
s=8*sin(x);
subplot(3,1,1);
plot(s);
title('Analog Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(3,1,2);
stem(s);grid on; title('Sampled Sinal'); ylabel('Amplitude--->'); xlabel('Time--->');
% Quantization Process
vmax=8;
vmin=-vmax;
del=(vmax-vmin)/L;
part=vmin:del:vmax; % level are between vmin and vmax with difference of del
code=vmin-(del/2):del:vmax+(del/2); % Contaion Quantized valuses
[ind,q]=quantiz(s,part,code); % Quantization process
% ind contain index number and q contain quantized values
l1=length(ind);
l2=length(q);
for i=1:l1
if(ind(i)~=0) % To make index as binary decimal so started from 0 to N
   ind(i)=ind(i)-1;
```

```
end
i=i+1;
end
for i=1:l2
if(q(i)==vmin-(del/2)) % To make quantize value inbetween the levels
    q(i)=vmin+(del/2);
end
end
subplot(3,1,3);
stem(q);grid on; % Display the Quantize values
title('Quantized Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
% Encoding Process
figure
code=de2bi(ind,'left-msb'); % Convert the decimal to binary
k=1;
for i=1:l1
for j=1:n
    coded(k)=code(i,j); % convert code matrix to a coded row vector
    j=j+1;
    k=k+1;
end
i=i+1;
end
subplot(2,1,1); grid on;
stairs(coded); % Display the encoded signal
axis([0 100 -2 3]); title('Encoded Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
% Demodulation Of PCM signal
qunt=reshape(coded,n,length(coded)/n);
index=bi2de(qunt','left-msb'); % Getback the index in decimal form
q=del*index+vmin+(del/2); % getback Quantized values
subplot(2,1,2); grid on;
plot(q); % Plot Demodulated signal
title('Demodulated Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
```

Hardware Implementation ideas (Additional):

Synchronization

When the encoded bits are transmitted, how does the receiver know when the bit sequence starts and when it ends? For this type of encoding it is possible that some synchronization bits are transmitted periodically. The receiver can synchronize itself using these synchronizing bits. Additionally, some guard bits are needed to separate adjacent codes. The total number of bits necessary to transmit to be able to decode a signal is therefore much larger than the number of bits containing information. The pre- built board 'Module 296f' can be used for this experiment. This board implements PCM with two "0"s as the guard bits, the four information bits, and two more guard "0"s as followed by eight "1" as synchronization bits (Figure 2.8).

The Figure 2.4 shows the complete PCM system. In the transmitter the input message signal f(t) is sampled, quantized, and encoded to binary bits. The serial bit stream represents the PCM output. At the receiver the PCM bit stream is decoded. After D/A conversion the message signal is reconstructed. Reconstructed signal represents the original message signal with some quantization error.

The module 296f encodes a signal by comparing the input signal level to an 8 or 16-step ramp. Each step ramp represents the current state of the binary counter. To encode PCM, the ramp must repeat itself at least at Nyquist frequency, which is higher than the frequency of the message. The message signal and ramp are inputs to a comparator, which has two output states high, and low. When the ramp signal exceeds the message signal, the comparator output swings low and the counter value is latched and is held until the ramp repeats.

Decoding

When the data bits are separated from the bit streams (removing the synchronization and guard bits), they are sent to a D/A converter, where each bit is given the appropriate weight in voltage, and the voltage due to each bit is summed.

Modulator

As shown in the Figure 2.5, the clock frequency f1 feeds to the “clk” line of the 4-bit counter and also the shift-left control line of the 8-bit register. The clock f1 simultaneously increases the counter and shifts left the 8-bit register. The comparator has f(t)/k (where k is an attenuation constant) and a ramp signal as the + and – inputs respectively. The output of the comparator drives the load line of the 4-bit latch. The comparator output is high until the ramp input becomes higher than f(t)/k. at that point, the last input to the latch stays for the rest of cycle until counter is reset to 0000. Synchronously with the counter reset the f2 clock signal goes high loading the output of the latch to the 8-bit register. Note that only the four counter bits are loaded from the latch, the other four are grounded and therefore loaded as 0.

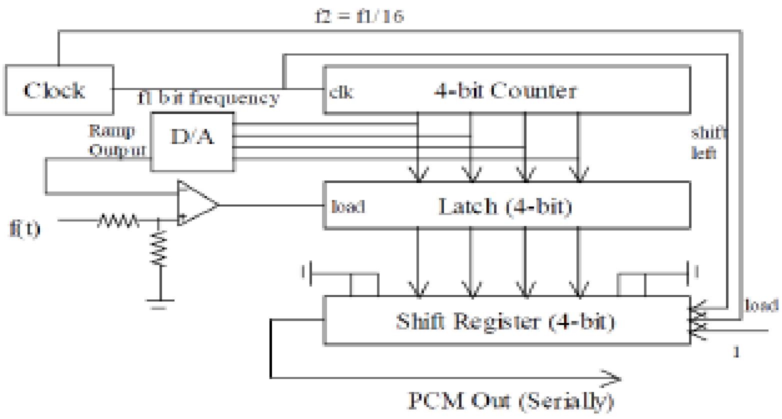


Figure 2.5: PCM Modulator

At this point the cycle starts again. The counter starts counting and the 8-bit register starts shifting out serially the 8 bits loaded in it. Note that every time a bit is shifted out serially, another bit is taken as input in the rightmost bit of the registers from the serial input. In our cases this input is connected to a logic “1” and are the synchronization bits. An example is given in the Figure 2.6 (a) & (b).

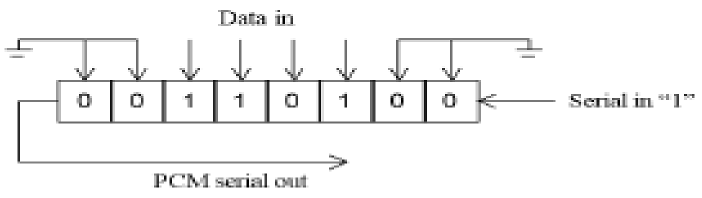


Figure 2.6 (a): Status of the shift register as it shifts the bits

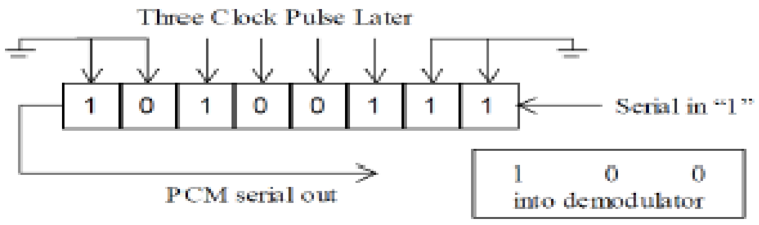


Figure 2.6 (b): Status of the shift register as it shifts the bits

Demodulator

The synchronization circuit found in Figure 2.7 recognizes the eight "1" used to synchronize and with the last "1" turns the clock on (same frequency as f1). The 8-bit register on the demodulator will load serially the next eight bits coming from the modulator. The same clock runs a 3-bit counter with its "carry" line connected to a load control line of latch. The latch will load the 4 counter bits on the 8-bit register when the counter is reset to 0000. Note that it takes nine clock pulses to reset instead of eight. In order to avoid an extra bit to be shifted-in three outputs from the counter are NAND together and tied to the clock. This avoids the 8 bit register to shift in another bit when the counter output is "111".

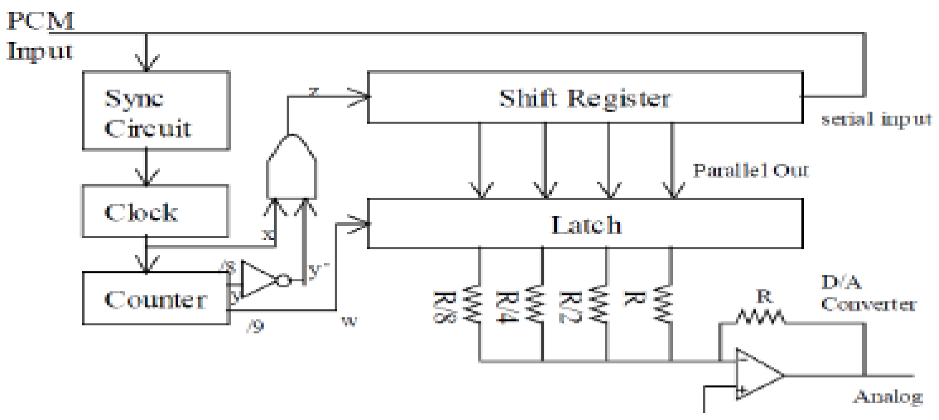


Figure 2.7: PCM Demodulator

Hardware Implementation

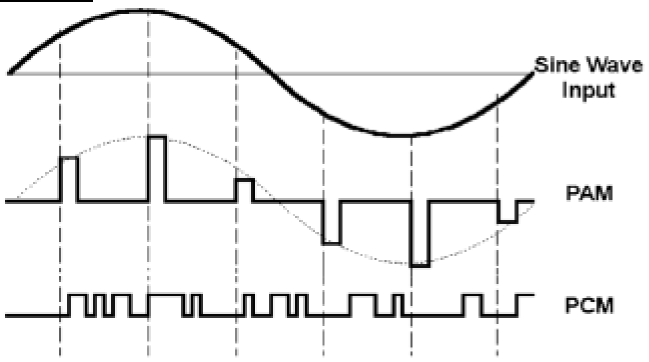
In the kit this is implemented differently. The analog signal is passed through a ADC (Analog to Digital Converter) and then the digital code word is passed through a parallel to serial converter block. This is modulated PCM. This is taken by the Serial to Parallel converter and then through a DAC to get the demodulated signal. The clock is given to all these blocks for synchronization. The input signal can be either DC or AC according to the kit. The waveforms can be observed on a CRO for DC without problem.AC also can be observed but with poor resolution.

TABULATION:

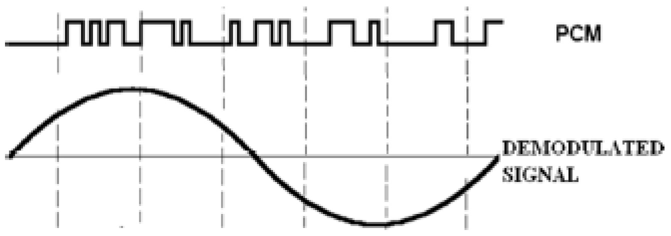
S.No	Name of the signal	Amplitude (V)	Time period (msec)	Frequency
1	Message			
2	Carrier			
3	Modulated			
4	Demodulated			

MODEL GRAPH

Pulse code modulation:



Pulse Code demodulation:



Post Lab

1. State A-law and µ-law in non-linear PCM.
2. What are the advantages and disadvantages of mid-rise and mid-tread quantization schemes?
3. What are the ways PCM systems can be improvised?

RESULT:

Thus the PCM signal was generated using PCM modulator and the message signal was detected from PCM signal by using PCM demodulator.

8. SIMULATION OF DATA REFORMATING BY LINE CODING TECHNIQUES

AIM:

In this experiment you will investigate how binary information is serially coded for transmission at baseband frequencies. In particular, you will study:

a) Line coding methods which are currently used in data communication applications;
b) Power spectral density functions associated with various line codes;
c) Causes of signal distortion in a data communications channel;
d) Effects of intersymbol interference (ISI) and channel noise by observing the eye pattern.

Prelab:

1. Given the binary sequence $b = [1;\ 0;\ 1;\ 0;\ 1;\ 1];$ sketch the waveforms representing the sequence b using the following line codes: a. unipolar NRZ; b. polar NRZ; c. unipolar RZ; d. bipolar RZ; e. Manchester. Assume unit pulse amplitude and use binary data rate R_b = 1 kbps.
2. Determine and sketch the power spectral density (PSD) functions corresponding to the above line codes. Use R_b = 1 kbps*:* Let $f_1 > 0$ be the location of the first spectral null in the PSD function. If the transmission bandwidth BT of a line code is determined by f_1, determine B_T for the line codes in question 1 as a function of R_b*:*

THEORY:

A **line code** is the code used for data transmission of a digital signal over a transmission line. This process of coding is chosen so as to avoid overlap and distortion of signal such as inter-symbol interference. The properties of Line Coding:

- As the coding is done to make more bits transmit on a single signal, the bandwidth used is much reduced.
- For a given bandwidth, the power is efficiently used.
- The probability of error is much reduced.
- Error detection is done and the bipolar too has a correction capability.
- Power density is much favourable.
- The timing content is adequate.
- Long strings of **1s** and **0s** is avoided to maintain transparency.

PCM binary digits should be represented by electrical pulses in order to transmit them through a base band channel. The most commonly used PCM popular data formats are being realized here. Line coding refers to the process of representing the bit stream (**1**"s and **0**"s) in the form of voltage or current variations optimally tuned for the specific properties of the physical channel being used. The block diagram for base band digital communication using data formatted codes is shown in figure 3.1.

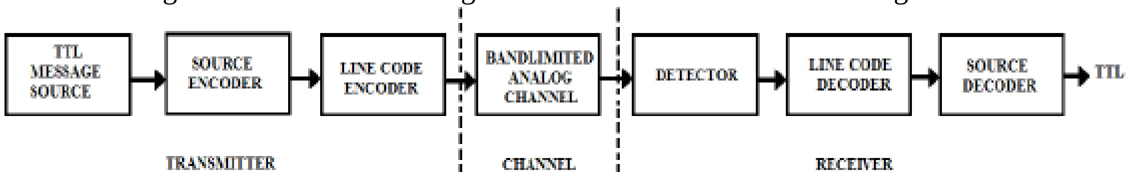


Figure 3.1. Baseband transmission system

The selection of a proper line code can help in so many ways: One possibility is to aid in clock recovery at the receiver. A clock signal is recovered by observing transitions in the received bit sequence, and if enough transitions exist, a good recovery of the clock is guaranteed, and the signal is said to be self-clocking. Another advantage is to get rid of DC shifts. The DC component in a line code is called the bias or the DC coefficient. Unfortunately, most long-distance communication channels cannot transport a DC component. This is why most line codes try to eliminate the DC component before being transmitted on the channel. Such codes are called DC balanced, zero- DC, zero-bias, or DC equalized.

Some common types of line encoding in common-use nowadays are unipolar, polar, bipolar, Manchester, MLT-3 and Duo-binary encoding as explained here:

Unipolar (Unipolar NRZ and Unipolar RZ):

Unipolar is the simplest line coding scheme possible. It has the advantage of being compatible with TTL logic. Unipolar coding uses a positive rectangular pulse p (t) to represent binary **1**, and the absence of a pulse (i.e., zero voltage) to represent a binary **0**. Two possibilities for the pulse p (t) exist: Non-Return-to-Zero (NRZ) rectangular pulse and Return-to-Zero (RZ) rectangular pulse. The difference between Unipolar NRZ and Unipolar RZ codes is that the rectangular pulse in NRZ stays at a positive value (e.g., +5V) for the full duration of the logic **1** bit, while the pule in RZ drops from +5V to 0V in the middle of the bit time.

A drawback of unipolar (RZ and NRZ) is that its average value is not zero, which means it creates a significant DC-component at the receiver (see the impulse at zero frequency in the corresponding power spectral density (PSD) of this line code. The disadvantage of unipolar RZ compared to unipolar NRZ is that each rectangular pulse in RZ is only half the length of NRZ pulse. This means that unipolar RZ requires twice the bandwidth of the NRZ code.

Polar NRZ and RZ: *Understand yourself from the Unipolar*

Bipolar (Bipolar NRZ and Bipolar RZ):

Binary 1's are represented by alternating positive or negative values. The binary 0 is represented by a zero level. The term pseudo-ternary refers to the use of 3 encoded signal levels to represent two–level (binary) data. This is also called alternate mark inversion (AMI) signalling.

Bipolar RZ: Also called RZ–AMI, where AMI denotes alternate mark (binary 1) inversion

Bipolar NRZ: Also called NRZ–M, where M denotes inversion on mark (binary 1)

Manchester Encoding:

In Manchester code each bit of data is signified by at least one transition. Manchester encoding is therefore considered to be self-clocking, which means that accurate clock recovery from a data stream is possible. In addition, the DC component of the encoded signal is zero. Although transitions allow the signal to be self-clocking, it carries significant overhead as there is a need for essentially twice the bandwidth of a simple NRZ or NRZI encoding.

Post lab 1:

For the above set of line codes determine which will generate a waveform with no *dc* component regardless of binary sequence represented. Why is the absence of a *dc* component of any practical significance for the transmission of waveforms?

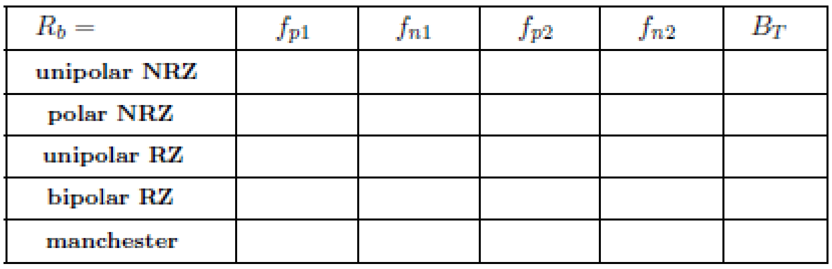
b) Power spectral density (PSD) functions of line codes: Generate a 1,000 sample binary sequence:

b = binary(1000);

Display the PSD function of each line code used

psd(wave gen(b, 'line code name'))

Let: *fp*1: ¯rst spectral peak; *fn*1: ¯rst spectral null; *fp*2: second spectral peak; *fn2*: second spectral null; such that all *f*(.) > 0*:* Record your observations in Table.



$R_b =$	f_{p1}	f_{n1}	f_{p2}	f_{n2}	B_T
unipolar NRZ					
polar NRZ					
unipolar RZ					
bipolar RZ					
manchester					

To illustrate the dependence of the PSD function on the underlying binary data rate, use the Manchester line code and vary *Rb*:

psd(wave gen(b, 'manchester', Rb))

where Rb = *(* 5 kbps*;* 10 kbps*;* 20 kbps*)*. You may replace Manchester by any other line code used in part. Observe the location of spectral peaks and nulls and relate them to *Rb*.

Post Lab 2:

For a baseband data communications channel with usable bandwidth of 10 kHz, what is the maximum binary data rate for each of the line codes examined in a)

c) Channel Characteristics

The MATLAB function that represents the channel response is *channel* which is called with the following arguments:

channel(input, gain, noise power, bandwidth)

Create a 10 sample binary sequence *b* and generate a waveform representing *b* in polar NRZ signalling format. Use *Rb* = 1 kbps*:*

b = binary(10);
x = wave gen(b, 'polar nrz', 1000);

From your observation in part A, determine the transmission bandwidth *BT* of x:

BT = Hz.

Consider a baseband data transmission channel with unity gain and additive white Gaussian noise (AWGN) where the noise power is 10^{-2}W and the channel bandwidth is 4.9 kHz*:* Transmit waveform x over this channel. Display the channel input and output waveforms:

y = channel(x, 1, 0.01, 4900);
subplot(211), waveplot(x)
subplot(212), waveplot(y)

If the signalling format is polar NRZ at *Rb* = 1 kbps, estimate the transmitted sequence from the display of the channel output waveform.

^b =

Compare your estimate with the original sequence b.

Effect of channel noise on the transmitted waveform: Gradually increase the channel noise power while keeping the channel bandwidth at 4*:*9 kHz and observe changes in the channel output.

y = channel(x,1,sigma,4900);
waveplot(y)

where sigma *2 f*0*:*1*;* 0*:*5*;* 1*;* 2*;* 5*g:* At what noise power level, does the channel output waveform becomes indistinguishable from noise? You can also observe e®ects of increasing channel noise power by looking at the PSD of the channel output waveform.

b = binary(1000);
x = wave gen(b, 'polar nrz', 1000);
clf; subplot(121); psdf(x);
subplot(122); psdf(channel(x,1,0.01,4900))
hold on
subplot(122); psdf(channel(x,1,1,4900))

subplot(122); psdf(channel(x,1,5,4900))

Post Lab 3:

Since the channel noise is additive and uncorrelated with the channel input, determine an expression that will describe the PSD of the channel output in terms of the input and noise PSD functions.

Effects of channel bandwidth on transmitted waveform: Distortion observed in the time display of the channel output is due to ¯nite bandwidth of the channel and due to noise. To study distortion due to channel bandwidth only, set noise power to zero and regenerate the channel output waveform:

```
hold off; clf;
b = binary(10);
x = wave gen( b, 'polar nrz', 1000 );
subplot(211), waveplot(x)
subplot(212), waveplot(channel(x,1,0,4900))
```

Investigate effects of channel bandwidth on the output waveform.

```
subplot(212), waveplot( channel(x,1,0,bw)
```

where bw *2 f*3000*;* 2000*;* 1000*;* 500*g:* Observe the delay in the output waveform due to ¯ltering characteristics of the channel. Plot the input and output waveforms. Determine the appropriate sampling instants for the decoding of the waveform for the case bw = 500.

d) Eye Diagram

Effects of channel filtering and noise can be best seen by observing the output waveform in the form of an *eye diagram.* The eye diagram is generated with multiple sweeps where each sweep is triggered by a clock signal and the sweep width is slightly larger than the binary data period *Tb* = 1=*Rb:* In this simulation the eye diagram is based on a sweep width of 2*Tb:*

Generation of Eye Diagram:

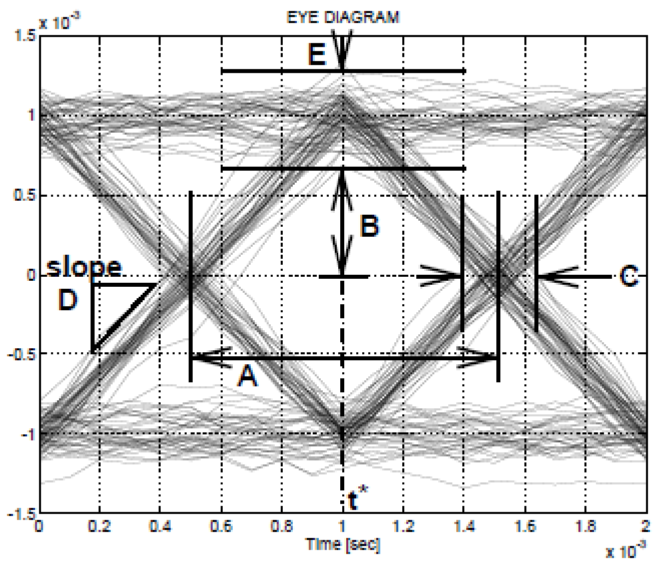
```
b = [ 1 0 0 1 0 1 1 0 ];
x = wave gen( b, 'polar nrz', 1000);
clf
subplot(221), waveplot(x)
subplot(223), eye diag(x)
```

The eye diagram for the waveform x represents what you should expect to see for an undistorted signal. To observe how the eye diagram is generated and to observe e®ects of signal distortion as the signal x is transmitted over a ¯nite bandwidth channel with no noise component:

```
y = channel( x, 1, 0, 4000 );
subplot(222), waveplot(y)
subplot(224), eye diag(y,-1)
```

If the second argument to the function *eye diag* is negative, you have to hit the Return key for the next trace to be displayed. This will assist you to understand how the eye diagram is generated.

Key parameters to be measured with an eye diagram are shown below.



A time interval over which the waveform can be sampled;
B margin over noise;
C distortion of zero crossings;
D slope: sensitivity to timing error;
E maximum distortion;

t^{o} optimum sampling instant measured with respect to the time origin. If the binary data period is *Tb*, then the waveform will be sampled at t^{o}; t^{o} + *Tb*; t^{o} + 2*Tb*;,...for signal detection. Generate the eye diagram from a polar NRZ waveform at the channel output for values of noise variance s2 and channel bandwidth bw shown in Table. Record *t*¤, *A* and *B* for each set of s2 and bw.

clf
b = binary(100);
x = wave gen(b, 'polar nrz', 1000);
eye diag(channel(x, 1, s2, bw))

Polar NRZ Line Code				
s2	bw	t^*	**A**	**B**
0.01	3000			
	2000			
	1000			
0.02	4000			
0.08				
0.10				

Generate eye diagrams as above for Manchester, polar RZ and unipolar RZ and unipolar NRZ line codes and observe how the line code dictates the shape and the symmetry of the eye diagram.

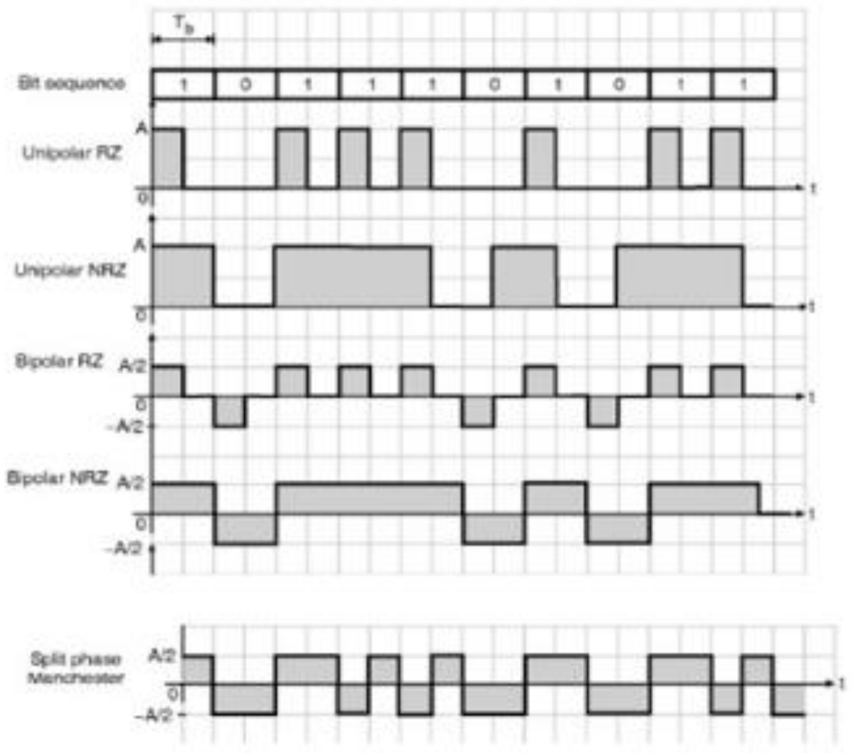
Model Code for Line code generation:
```
bits = [1 0 1 0 0 0 1 1 0];
bitrate = 1; % bits per second
figure;
[t,s] = unrz(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -0.1 1.1])
grid on;
title(['Unipolar NRZ: [' num2str(bits) ']']);
```

```
figure;
[t,s] = urz(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -0.1 1.1])
grid on;
title(['Unipolar RZ: [' num2str(bits) ']']);

figure;
[t,s] = prz(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -1.1 1.1])
grid on;
title(['Polar RZ: [' num2str(bits) ']']);

figure;
[t,s] = manchester(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -1.1 1.1])
grid on;
title(['Manchester: [' num2str(bits) ']']);
```

TABULATION:

S.No	Name of the signal	Amplitude (V)	Time period (msec)	Frequency
1	Unipolar RZ			
2	Unipolar NRZ			
3	Bipolar RZ			
4	Bipolar NRZ			
5	Manchester			

MODEL GRAPH:



RESULT:
Thus the different line coding techniques are simulated and studied successfully as per the objectives.