COMMUNICATION THEORY COURSE PROJECT
(SCHEME: QPSK TYPE-1)

Himani Sharma (2022102032)

Kripi Singla (2022102063)

1. INTRODUCTION

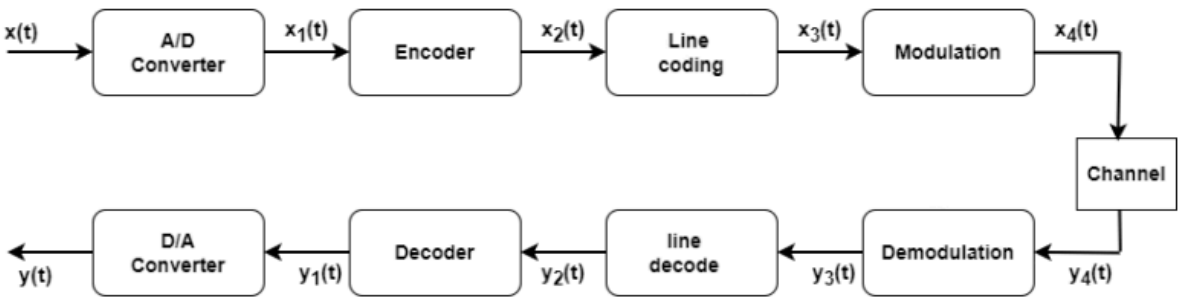
In this project, we are required to design our own communication model based on the QPSK modulation scheme.

QPSK modulation scheme

Quadrature Phase Shift Keying (QPSK) is a digital modulation scheme used in communication systems to transmit data over a radio frequency carrier wave. In QPSK, two bits of data are combined to form one symbol, and each symbol is then represented by one of four possible phase shifts of the carrier signal.

QPSK maps two bits of data into one of four possible phase shifts of the carrier signal. These phase shifts are typically separated by 90 degrees, hence the "quadrature" in the name. The four phase shifts are usually 0°, 90°, 180°, and 270°.

2. PROCEDURE



The plots for subsequent stages are generated for random and small wave data for better understanding and visualization.

```
Fs = 48000;
y_mono = [119/128,-9/128,10/128,115/128];%.........................x(t)
```

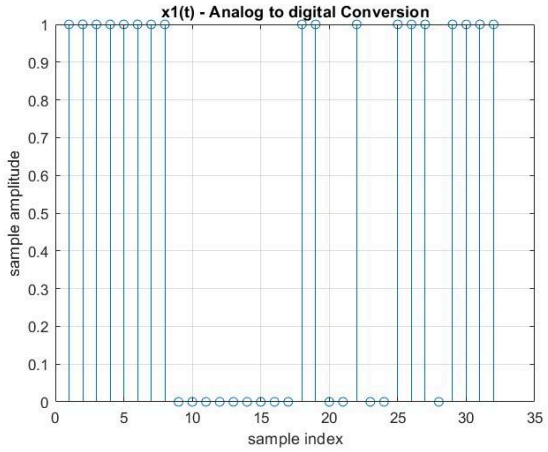
2.1) A/D CONVERTER

The signal is quantized to 256 levels by dividing a predefined range into 2 raised to the power 8 levels. Each value from the input data is then rounded to the nearest Level, and hence **quantized.**

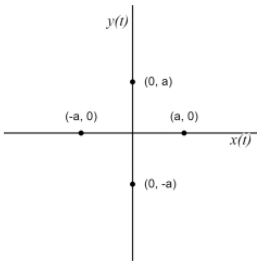
Each quantized value is then normalized and scaled by a factor of 255 so that each of the level values range from 0 to 255.

Each quantised value of the original audio is represented by a 8-bit binary number allowing for digital processing and transmission.

```
%% ADC conversion
levels = 2^8;
levels_found = linspace(min(y_mono), max(y_mono), levels);
y_quantised = quantize_signal(y_mono,levels_found);
y_normalized = (y_quantised - min(y_quantised)) / (max(y_quantised) - min(y_quantised));
y_scaled = round((y_normalized)* 255);
y_check = de2bi(y_scaled);
y_binary = de2bi(y_scaled, 8);
y_plot = reshape(y_binary,1,[]);
figure; %.........................x1(t)
stem(y_plot);
title('x1(t) - Analog to digital Conversion');
xlabel('sample index');
ylabel('sample amplitude');
grid on;
```



2.2) ENCODER



Here we have done mapping according to gray encoding where every symbol has only 1 bit difference as shown below:

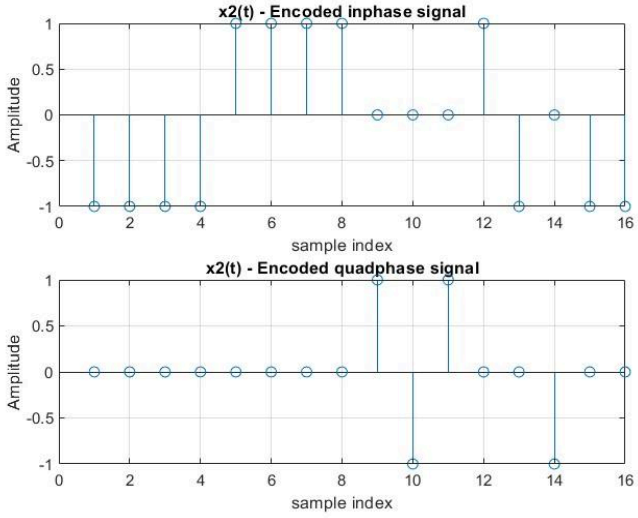
00 -> (a,0)
01 -> (0,a)
11 -> (-a,0)
10 -> (0,-a)

$$\theta_m = \theta_0 + \frac{2\pi}{M}(m-1) \qquad m = 1, 2, \ldots, M$$

Here M = 4, θ = 0

Also we have divided our encoded signal into two parts - inphase signal representing the first coordinate and quadrature phase for the second coordinate, both of which are perpendicular to each other.

Here, the value of a = 1 as shown below:



2.3) LINE CODING

$$\mathbf{x}_3(t) = \sum_k a_k p(t - kT_b),$$

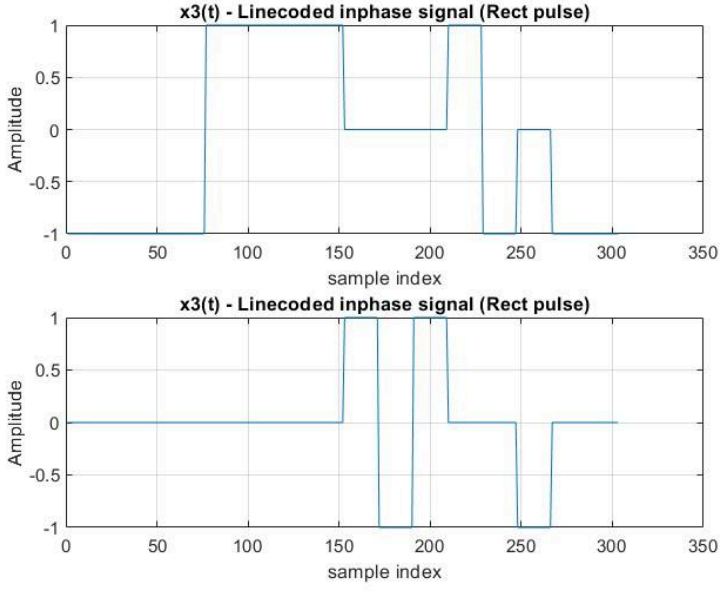
Here we are implementing line coding by convoluting the symbols with rectangular pulse and raised cosine respectively.

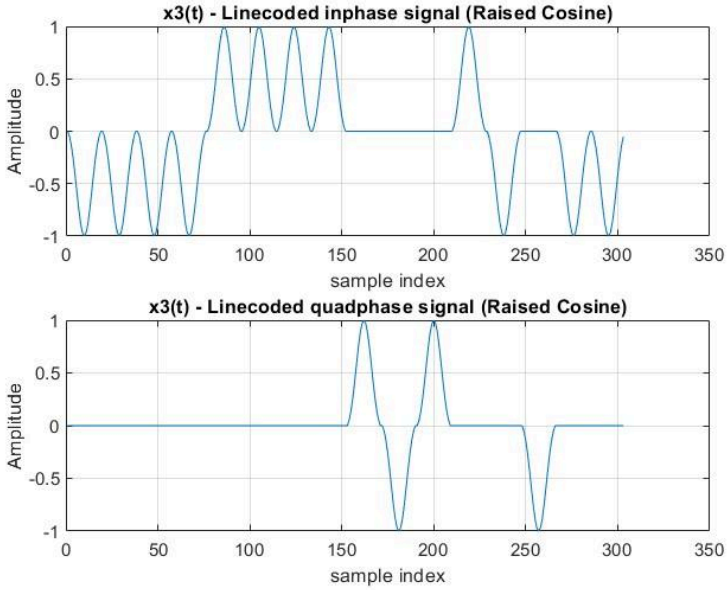
Parameters for rectangular pulse:-
Length of rectangular pulse = 19

Parameters for raised cosine:-
Upsampling_factor = 9
Length = 1
Roll off factor = 1

The raised cosine pulse also eventually has length = 19.

Digital data is represented by binary bits (0s and 1s), while communication channels typically transmit analog signals. Line coding converts binary data into a digital signal that can be transmitted over the channel.
Line coding techniques often ensure synchronization between the transmitter and receiver. This synchronization is crucial for the receiver to correctly interpret the transmitted signal and recover the original binary data.
Line coding techniques incorporate error detection and correction mechanisms. For example, specific patterns in the encoded signal may be used to detect errors caused by noise or interference in the communication channel.





2.4) MODULATION

Here the modulation scheme involves multiplication with a carrier signal with very high frequency - fc, here 1MHz to get a modulated signal.

We need to multiply the inphase signal with cosine wave and quad phase signal with sine wave individually for modulation.

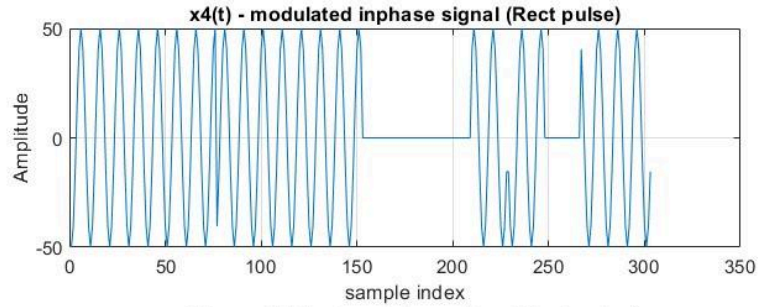
Final demodulated signal can be obtained by the addition of (inphase) and (i)*(quad phase)

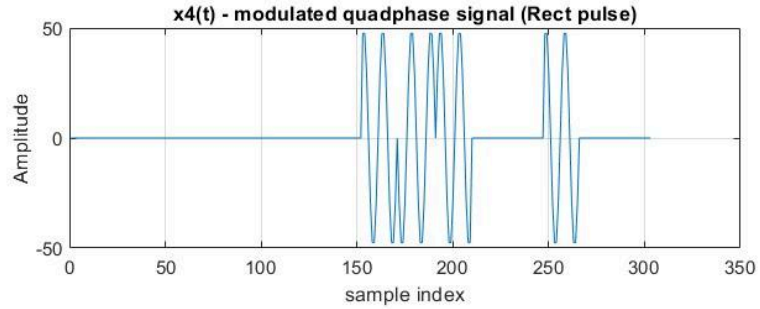
$$\varphi_{\text{PSK}}(t) = a_m\sqrt{\frac{2}{T_b}}\cos\omega_c t + b_m\sqrt{\frac{2}{T_b}}\sin\omega_c t \qquad 0 \le t < T_b$$

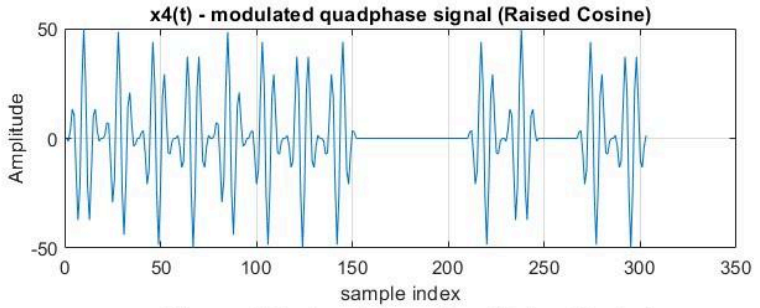
where $a_m = A\cos\theta_m$ and $b_m = -A\sin\theta_m$

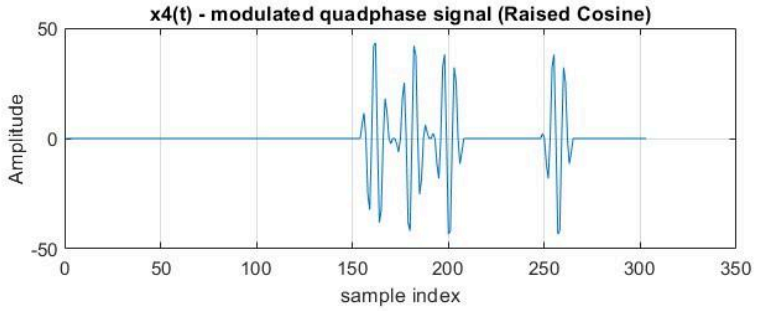
```
%% modulation
%rect pulse

fc = 1e6;
t_rect = 0:1/(10*fc):(length(inphase_linecoded_signal)-1)/(10*fc);
A = 5;
Tb = 19;
factor = sqrt(2/Tb);
in_cos_rect = cos(2*pi*fc*t_rect);
quad_sin_rect = sin(2*pi*fc*t_rect);

modulated_rect_inphase = A*inphase_linecoded_signal.*in_cos_rect';
modulated_rect_quadphase = A*quadphase_linecoded_signal.*quad_sin_rect';

%raised cosine

t_rc = 0:1/Fs:(length(inphase_rc_linecoded_signal)-1)/Fs;
in_cos_rc = cos(2*pi*fc*t_rc);
quad_sin_rc = sin(2*pi*fc*t_rc);

modulated_rc_inphase = A*inphase_rc_linecoded_signal.*in_cos_rc';
modulated_rc_quadphase = A*quadphase_rc_linecoded_signal.*quad_sin_rc';

final_modulated_rect = modulated_rect_inphase + (1i)*modulated_rect_quadphase;
final_modulated_rc = modulated_rc_inphase + (1i)*modulated_rc_quadphase;
```

Since in QPSK, there is phase difference of $\pi/2$ with every symbol change, we can observe the same in the figures.









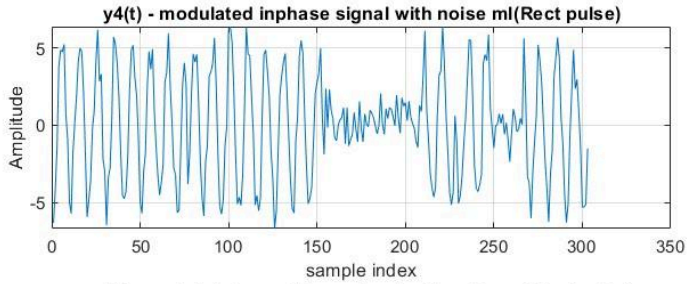
2.5) CHANNEL

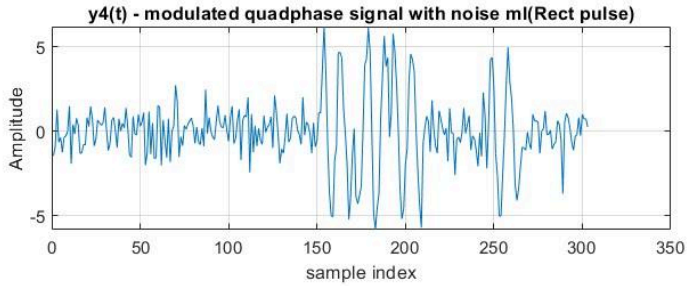
Memoryless AWGN Channel

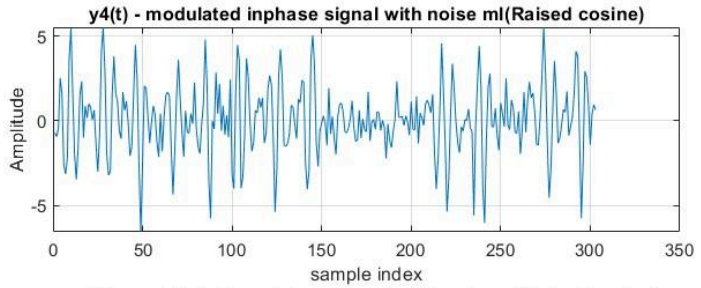
The memoryless AWGN channel introduces white Gaussian noise with a constant spectral density at each instant, independent of other times.

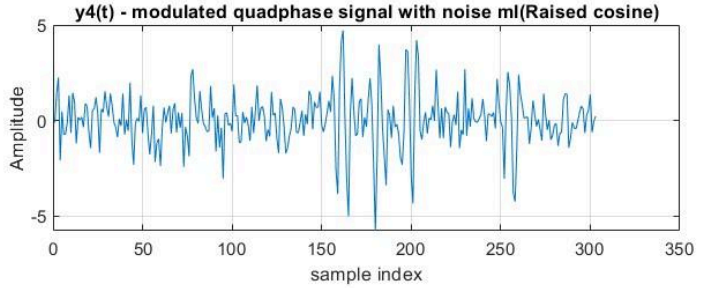
The received signal r(t) = s(t)+n(t) where n(t) represents the noise component.

```
%% Memoryless AWGN Channel

sigma_square = 1;
j = sqrt(-1);
noise_rect = sqrt(sigma_square)*(randn((size(final_modulated_rect)))+ j*(randn(size(final_modulated_rect))));
noise_rc = sqrt(sigma_square)*(randn((size(final_modulated_rc)))+ j*(randn(size(final_modulated_rc))));
```





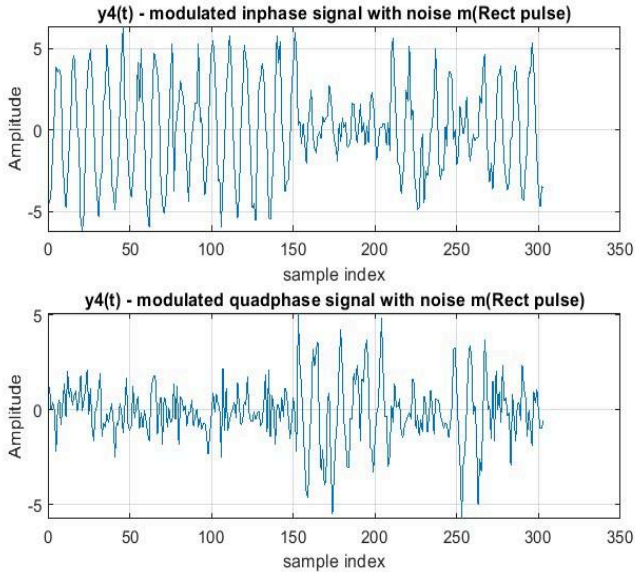


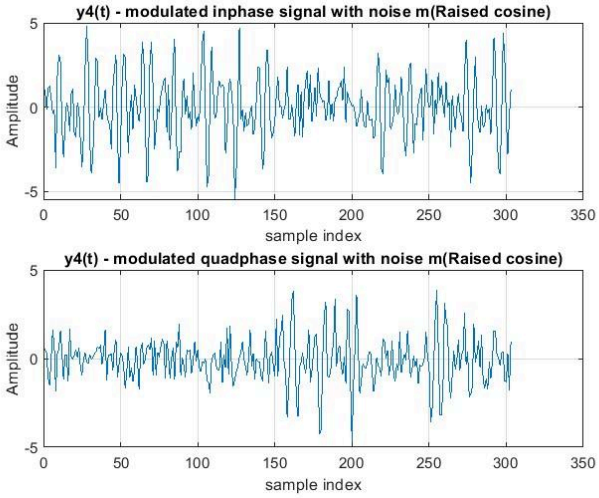


AWGN Channel with Memory

This channel type exhibits memory by introducing dependencies in noise across different times via convolution $r(t) = h(t) * s(t) + n(t)$.
The impulse response $h(t) = a\delta(t)+(1-a)\delta(t-bTb)$ shows that noise affects the signal at current and specific past times.

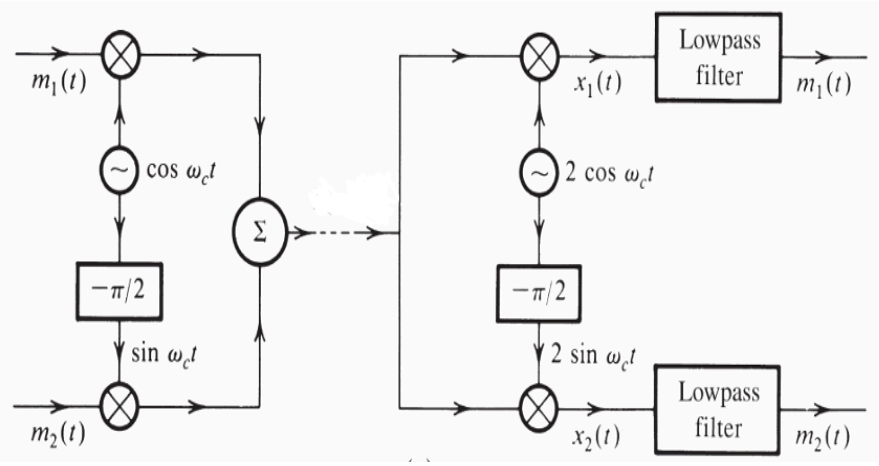
```
%% AWGN Channel with Memory
del = zeros(length(rect_pulse)+1,1);
a = 0.8;
del(1) = a;
del(end) = (1-a);
rect_mem = conv(final_modulated_rect,del);
received_signal_rect_mem = rect_mem(1:end-length(rect_pulse)) + noise_rect;

rc_mem = conv(final_modulated_rc,del);
received_signal_rc_mem = rc_mem(1:end-length(transmit_filter)) + noise_rc;
```





Memoryless channels treat each symbol independently concerning noise, whereas channels with memory introduce correlations.
Pe is generally lower in memoryless channels under similar conditions due to simpler noise characteristics. In channels with memory, correlated noise tends to increase Pe.

2.6) DEMODULATION



For demodulation, we multiply inphase component of the received signal with 2*cos(2*pi*fc*t) and the quadrature component with 2*sin(2*pi*fc*t).

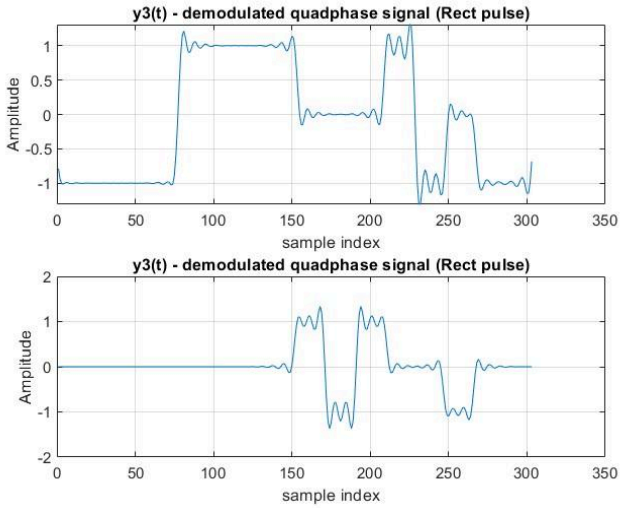
We then pass each of the received signals after multiplication through a low pass filter with cutoff frequency very low as compared to fc, since we require only the component centered at f = 0.

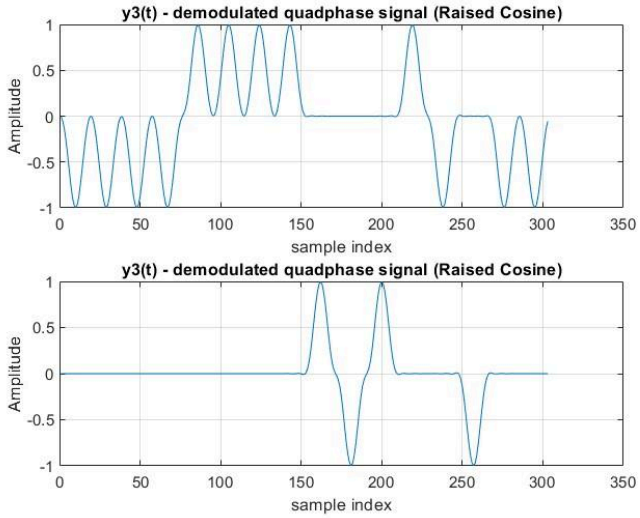
The filtered signal is the final demodulated inphase and quadrature phase component of the signal.

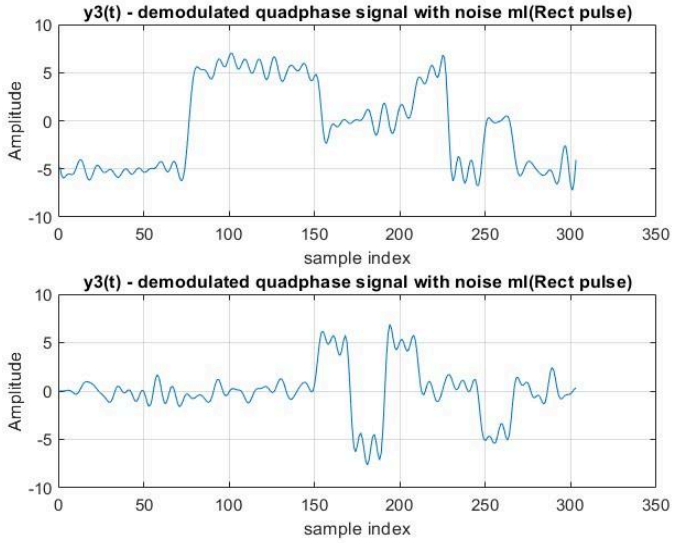
This demodulated inphase and quadrature component of the signal should resemble the linecoded inphase and quadrature phase components of the signal as we can observe in the following plots.

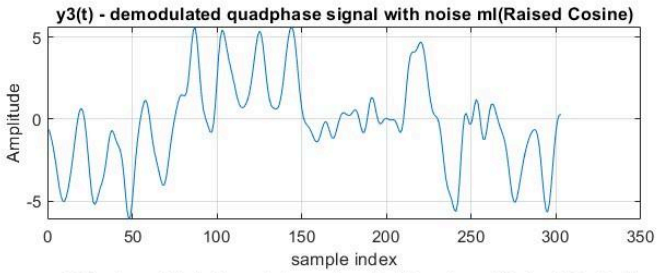
```
%% Demodulation
```

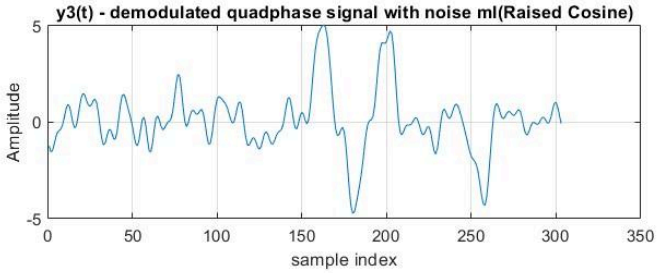
```
%% Memoryless Channel
%Rect pulse

received_signal_rect_cos = 2*received_signal_rect.*cos(2*pi*fc*t_rect)';
received_signal_rect_sine = 2*received_signal_rect.*sin(2*pi*fc*t_rect)';

received_inphase = lowpass(real(received_signal_rect_cos),5500,Fs);
received_quadphase = lowpass(imag(received_signal_rect_sine),5500,Fs);

%Raised Cosine pulse

received_signal_rc_cos = 2*received_signal_rc.*cos(2*pi*fc*t_rc)';
received_signal_rc_sine = 2*received_signal_rc.*sin(2*pi*fc*t_rc)';

received_rc_inphase = lowpass(real(received_signal_rc_cos),5500,Fs);
received_rc_quadphase = lowpass(imag(received_signal_rc_sine),5500,Fs);
```

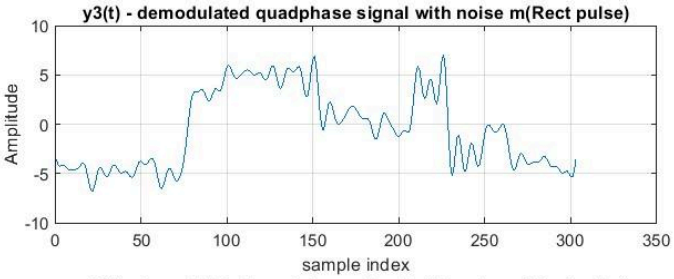


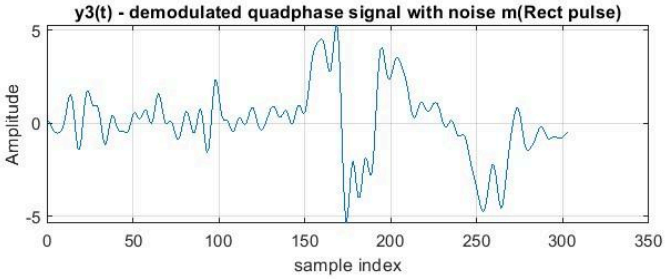

y3(t) - demodulated quadphase signal (Raised Cosine)
Amplitude
sample index
y3(t) - demodulated quadphase signal (Raised Cosine)
Amplitude
sample index

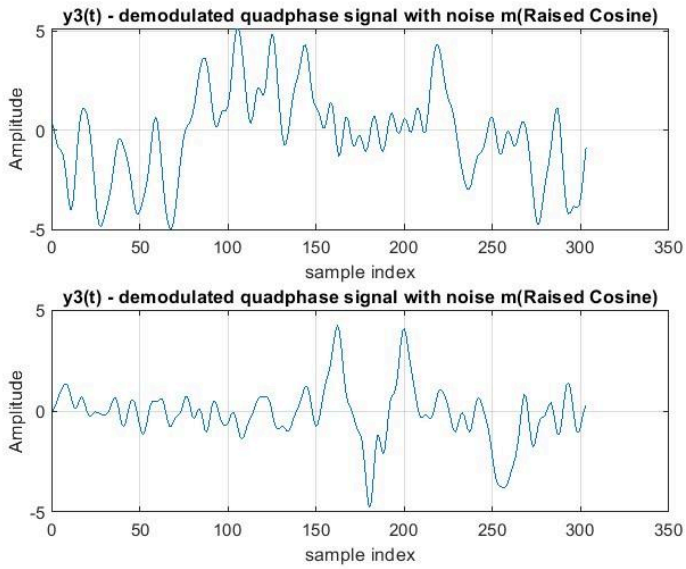

y3(t) - demodulated quadphase signal with noise ml(Rect pulse)
Amplitude
sample index
y3(t) - demodulated quadphase signal with noise ml(Rect pulse)
Amplitude
sample index


y3(t) - demodulated quadphase signal with noise ml(Raised Cosine)
Amplitude
sample index


y3(t) - demodulated quadphase signal with noise ml(Raised Cosine)
Amplitude
sample index


y3(t) - demodulated quadphase signal with noise m(Rect pulse)
Amplitude
sample index


y3(t) - demodulated quadphase signal with noise m(Rect pulse)
Amplitude
sample index

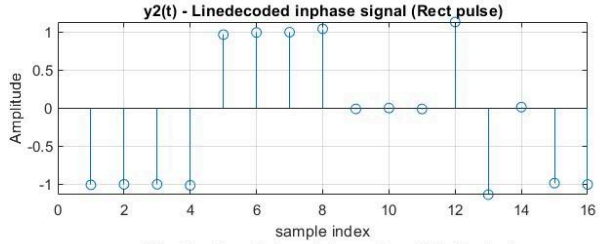


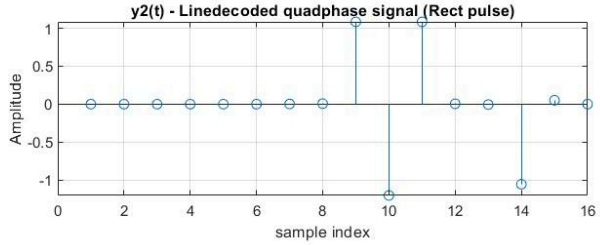
2.7) LINE DECODING

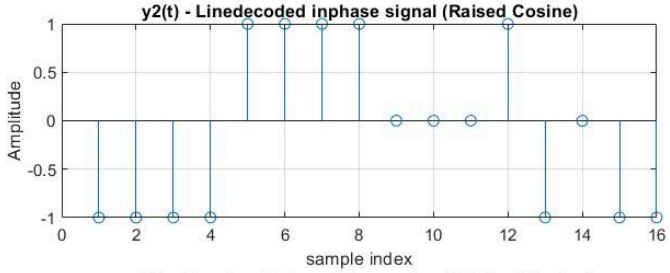
Line decode makes the decision for the output statistics.

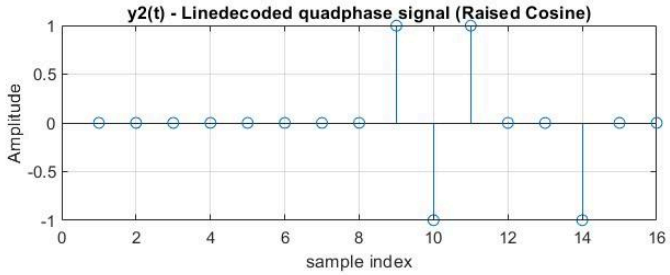
In line decoding, when working with rectangular pulses and raised cosine waveforms, the decision for each bit is made at sample index 10 (because the length of both the pulses is 19). This choice corresponds to the midpoint of the encoded pulse. In the case of the raised cosine waveform, sample index 10 represents the point where the waveform reaches its peak amplitude.

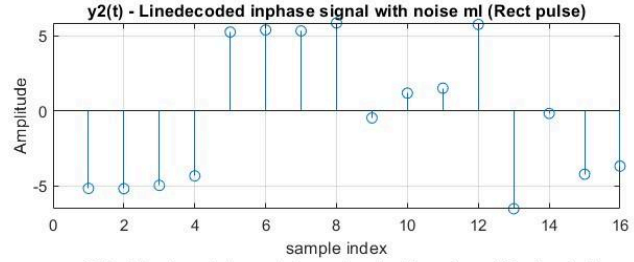
```
line_decoded_inphase_mem = zeros(length(inphase),1);
line_decoded_quadphase_mem = zeros(length(quadphase),1);

t_seq_rect = 10:19:length(received_quadphase_mem)-1;
k=1;
for i = t_seq_rect
    line_decoded_inphase_mem(k) = received_inphase_mem(i);
    line_decoded_quadphase_mem(k) = received_quadphase_mem(i);
    k = k+1;
end
```

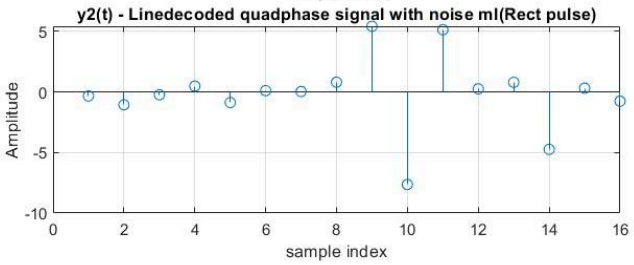

y2(t) - Linedecoded inphase signal (Rect pulse)
Amplitude
sample index

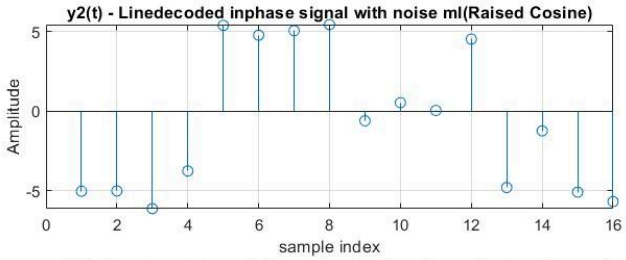

y2(t) - Linedecoded quadphase signal (Rect pulse)
Amplitude
sample index

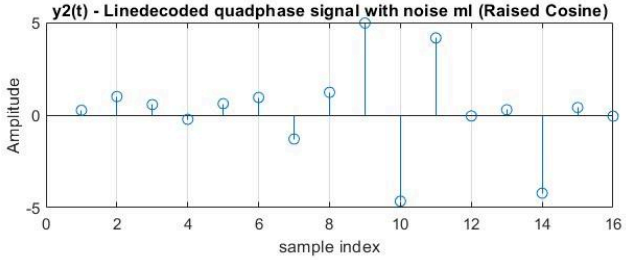

y2(t) - Linedecoded inphase signal (Raised Cosine)
Amplitude
sample index

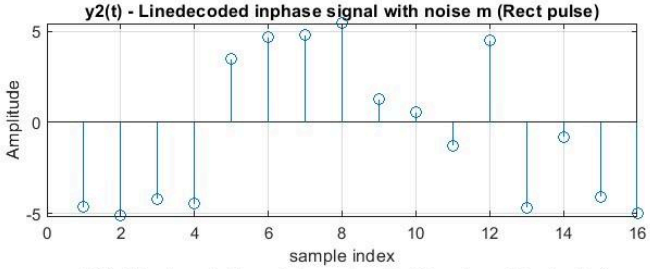

y2(t) - Linedecoded quadphase signal (Raised Cosine)
Amplitude
sample index

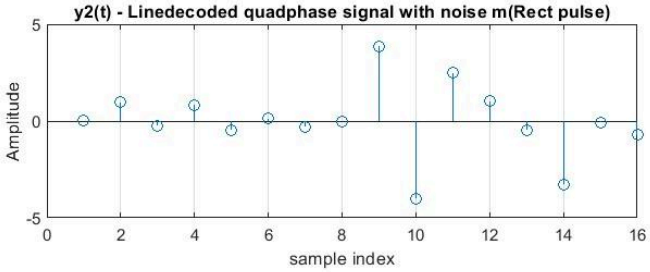

y2(t) - Linedecoded inphase signal with noise ml (Rect pulse)
Amplitude
sample index

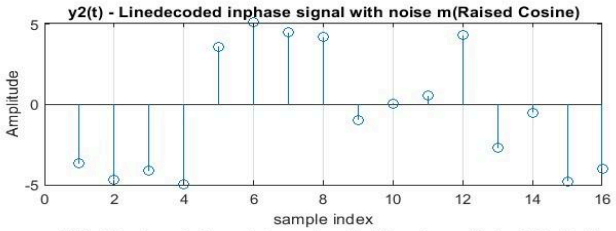

y2(t) - Linedecoded quadphase signal with noise ml(Rect pulse)
Amplitude
sample index

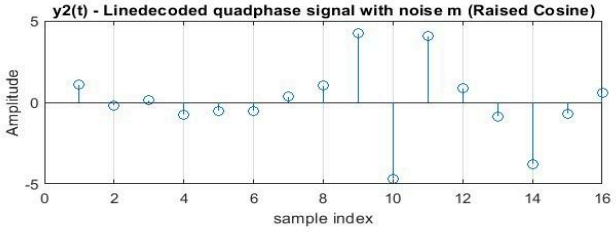

y2(t) - Linedecoded inphase signal with noise ml(Raised Cosine)
Amplitude
sample index


y2(t) - Linedecoded quadphase signal with noise ml (Raised Cosine)
Amplitude
sample index


y2(t) - Linedecoded inphase signal with noise m (Rect pulse)
Amplitude
sample index


y2(t) - Linedecoded quadphase signal with noise m(Rect pulse)
Amplitude
sample index


y2(t) - Linedecoded inphase signal with noise m(Raised Cosine)
Amplitude
sample index


y2(t) - Linedecoded quadphase signal with noise m (Raised Cosine)
Amplitude
sample index

2.8) DECODING

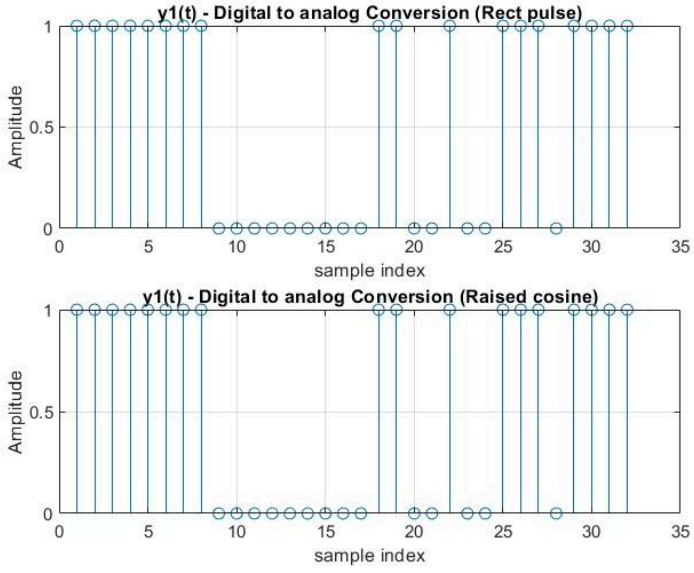
This decodes the decision statistics to give the final decoded bits to be converted back to analog.

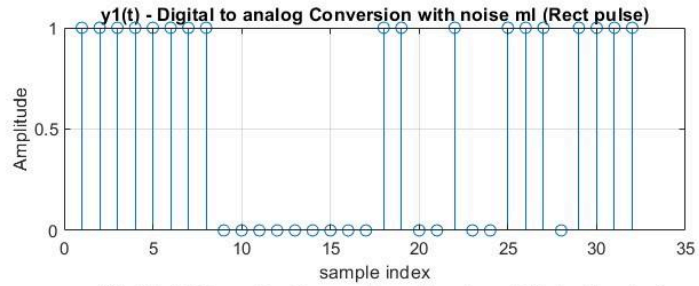
Decision Rule for QPSK

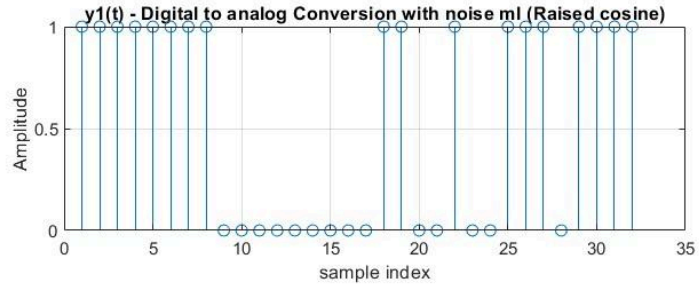
In QPSK (Quadrature Phase Shift Keying) modulation, the decision rule involves dividing the constellation diagram, which represents all possible signal points, into four regions corresponding to the four possible phase shifts. Each region represents a specific pair of bits.
The constellation diagram is divided into four quadrants or regions, each corresponding to one of the four possible phase shifts (0°, 90°, 180°, and 270°). These phase shifts are typically represented by points on the unit circle at angles of 0°, 90°, 180°, and 270°.
Depending on which quadrant or region the received signal point falls into, the demodulator determines the corresponding pair of bits, that is :-

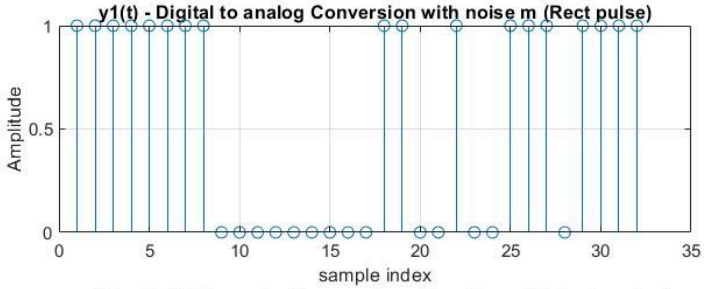
- If the received signal point falls in the region between -45° and 45°, it corresponds to the pair of bits "00".
- If the received signal point falls in the region between 45° and 135°, it corresponds to the pair of bits "01".
- If the received signal point falls in the region between 135° and 225°, it corresponds to the pair of bits "11".
- If the received signal point falls in the region between 225° and -45° (or 0°), it corresponds to the pair of bits "10".

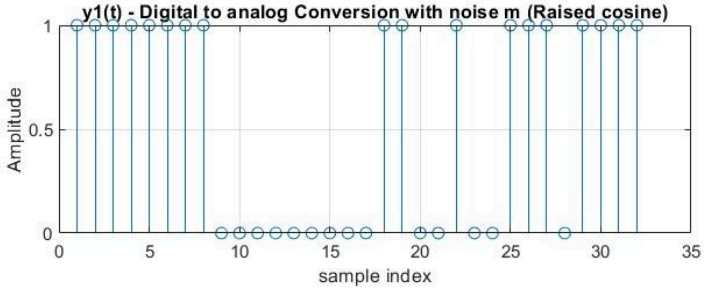
```
function decision = decision_making(input_signal)
    decision = zeros(length(input_signal),1);
    for i = 1:length(input_signal)
        check = angle(input_signal(i));
        if(check > -pi/4 && check <= pi/4)
        decision(i) = 1 + (1i)*0;
        elseif(check > pi/4 && check <= 3*pi/4)
        decision(i) = 0 + (1i)*1;
        elseif(check > 3*pi/4 && check <= pi)
        decision(i) = -1 + (1i)*0;
        elseif(check > -pi && check <= -3*pi/4)
        decision(i) = -1 + (1i)*0;
        elseif(check > -3*pi/4 && check <= -pi/4)
        decision(i) = 0 + (1i)*(-1);
        end
    end
end
```











2.9) D/A CONVERTER

This section of the code handles the conversion of the decoded binary data back into analog format.

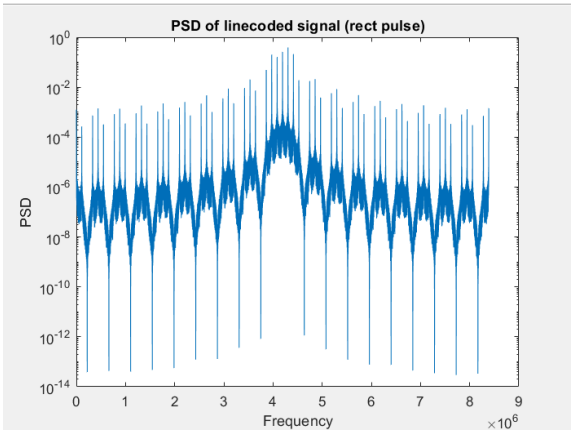
The binary data is reshaped into a matrix with 8 columns.This reshaping prepares the binary data for grouping into 8-bit words, which corresponds to the format used during the encoding process.

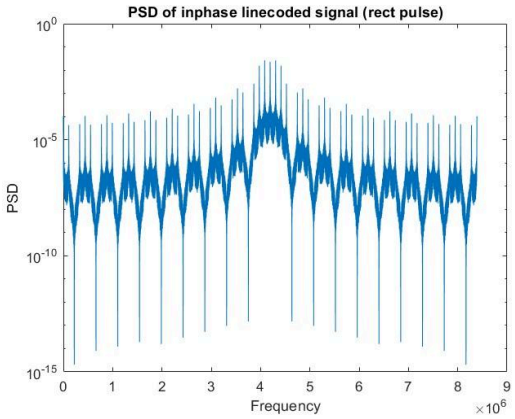
These values obtained are converted into decimal and normalized.

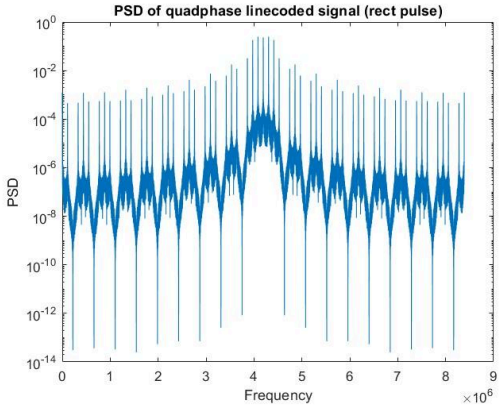
This method effectively converts the digital data, which was modified for transmis sion as a binary stream, back into a format resembling its original analog form.

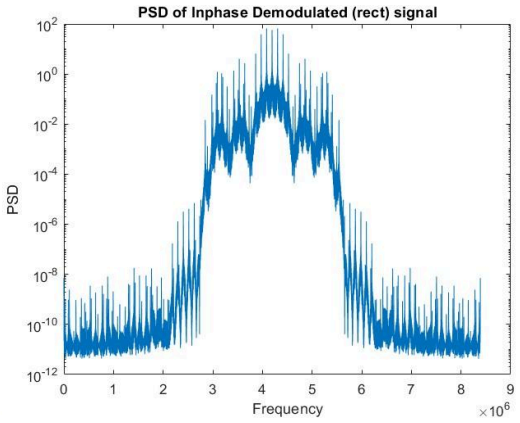
3. PSD - WITHOUT NOISE

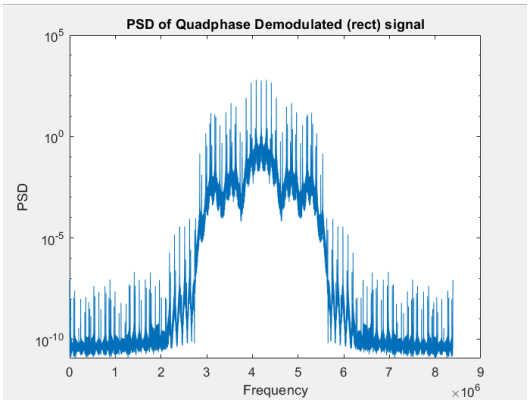
(a) Rect pulse



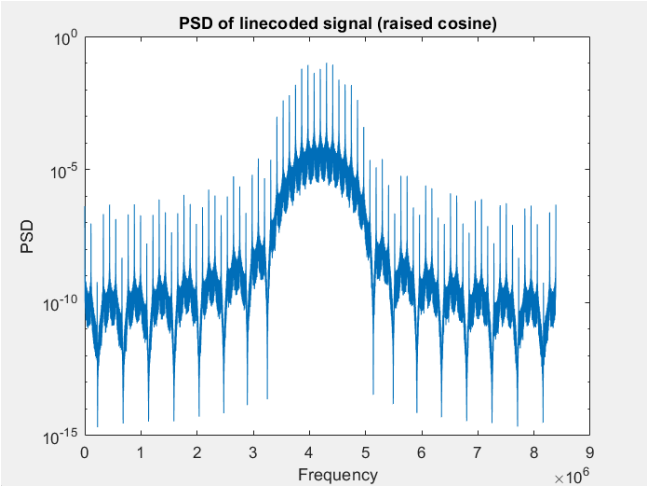


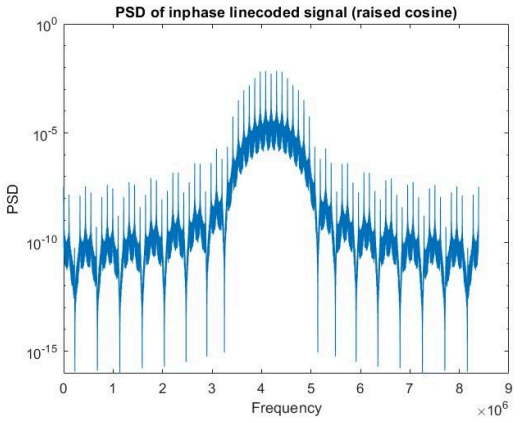


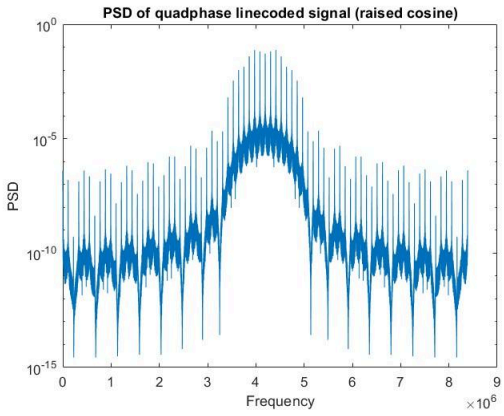


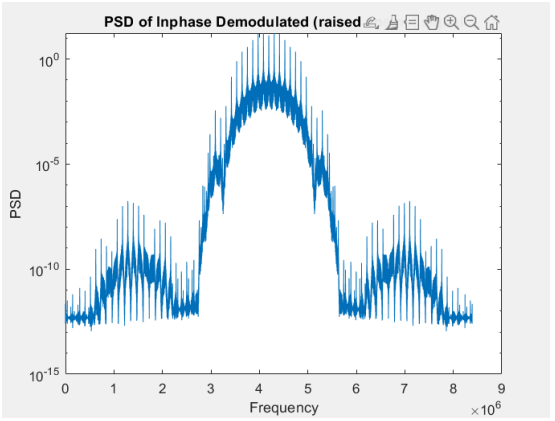


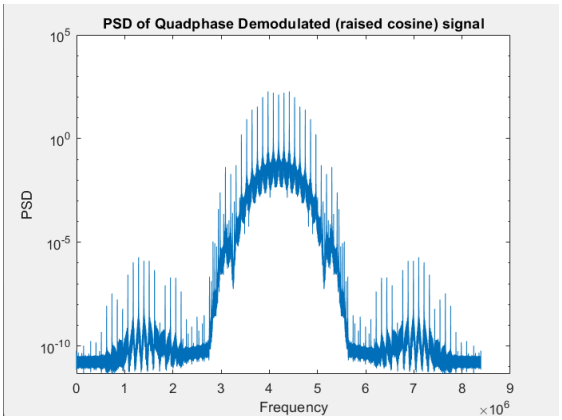
(b) Raised cosine





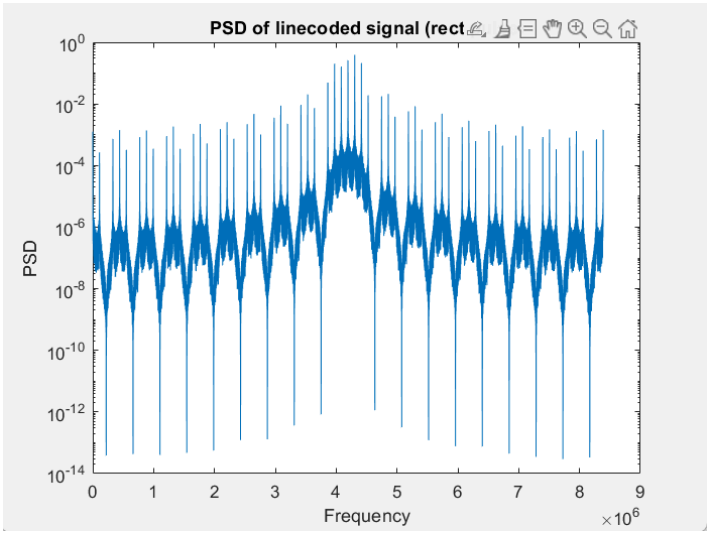


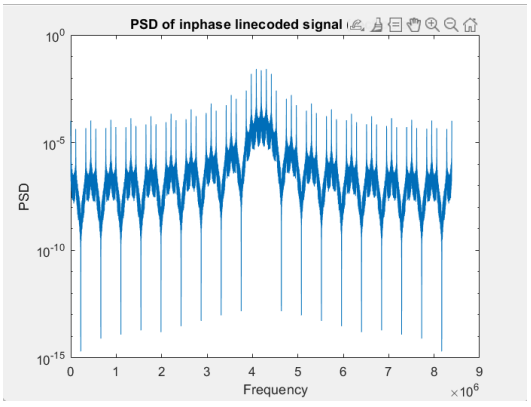


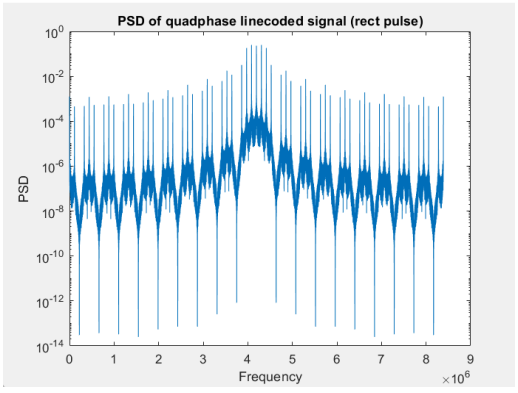


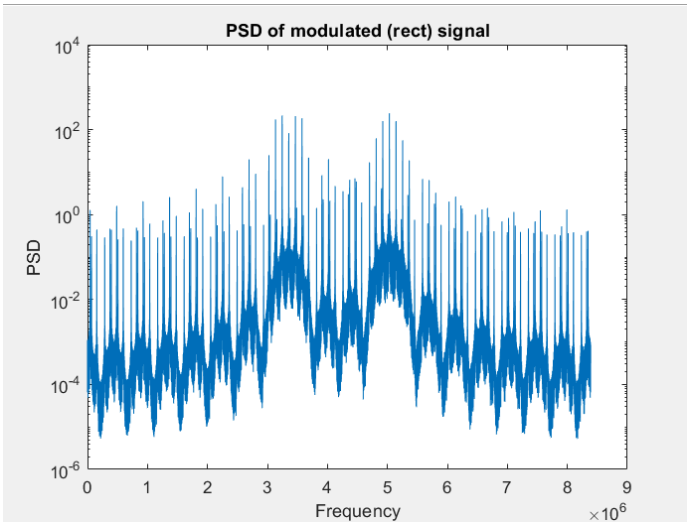
4. PSD - WITH MEMORYLESS NOISE

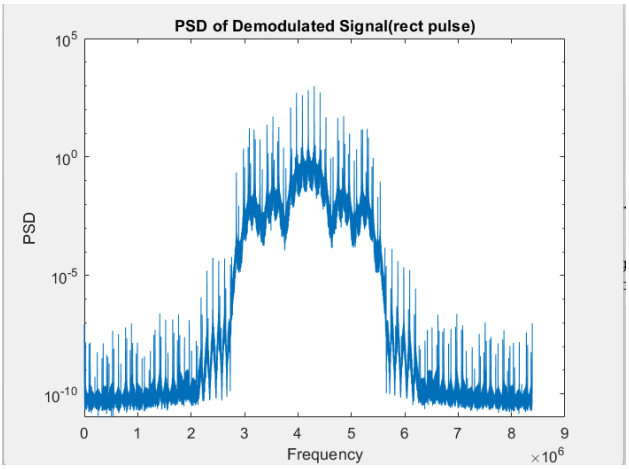
(a) Rect pulse

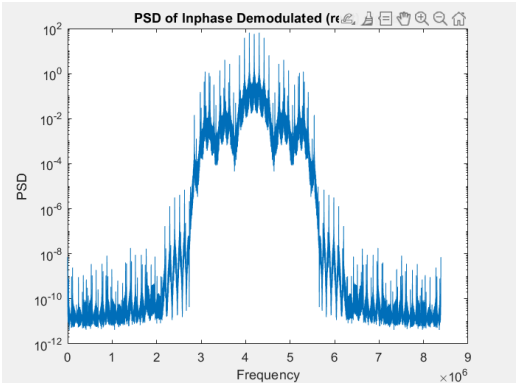


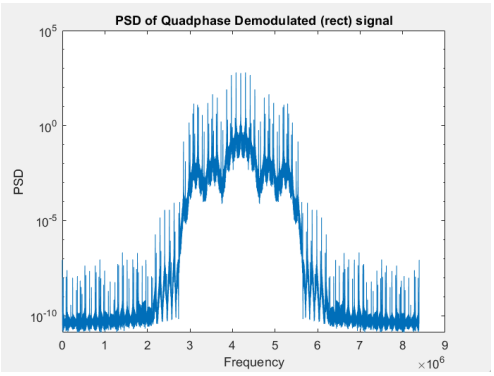


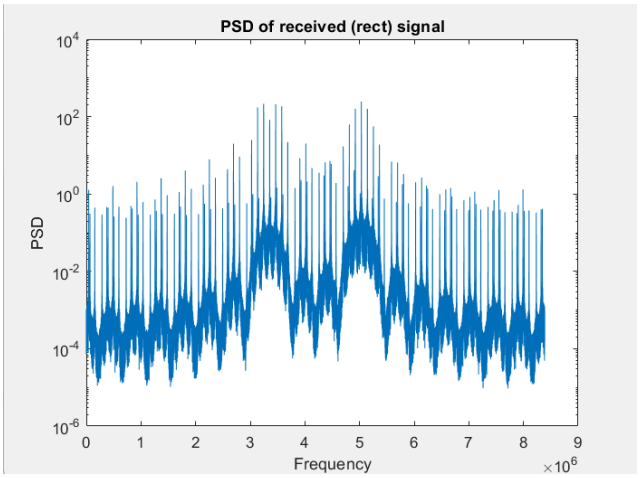



PSD of modulated (rect) signal
PSD
Frequency
×10^6

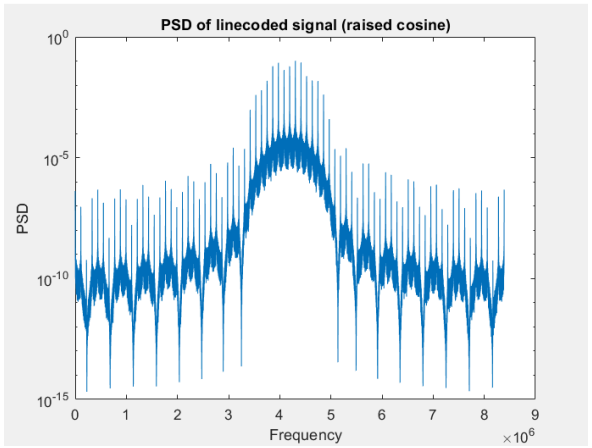

PSD of Demodulated Signal(rect pulse)
PSD
Frequency
×10^6

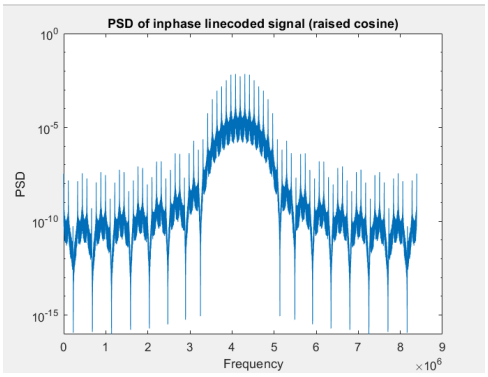

PSD of Inphase Demodulated (re
PSD
Frequency
×10^6

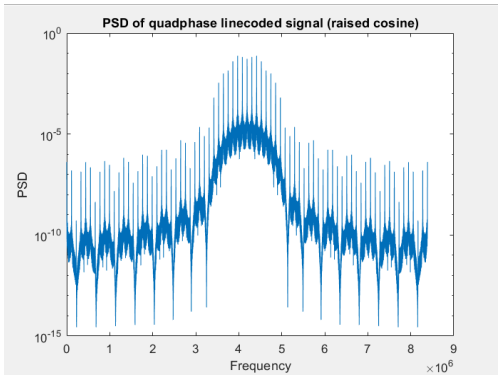

PSD of Quadphase Demodulated (rect) signal
PSD
Frequency
×10^6

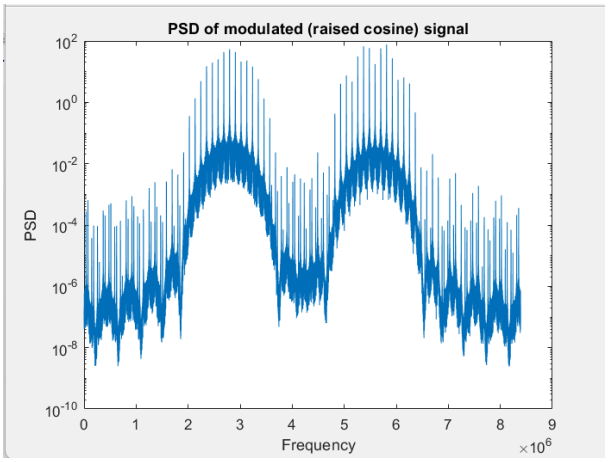


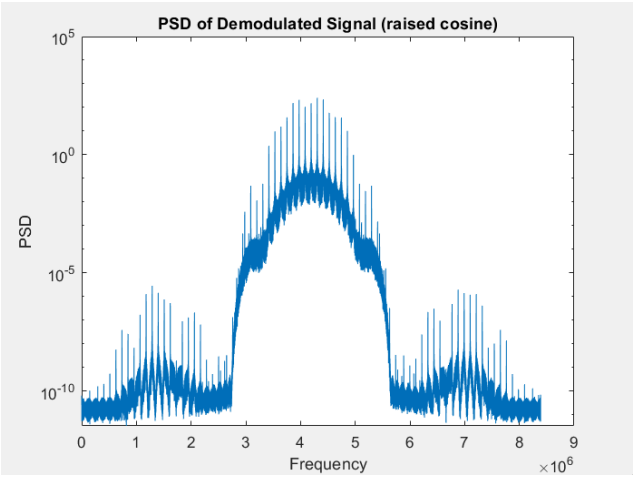
(b) Raised cosine

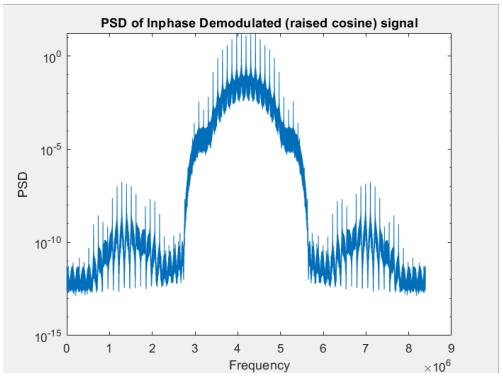


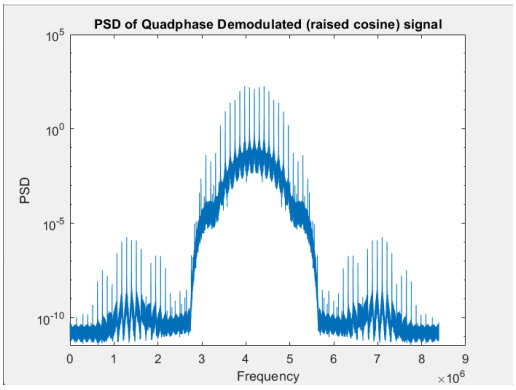


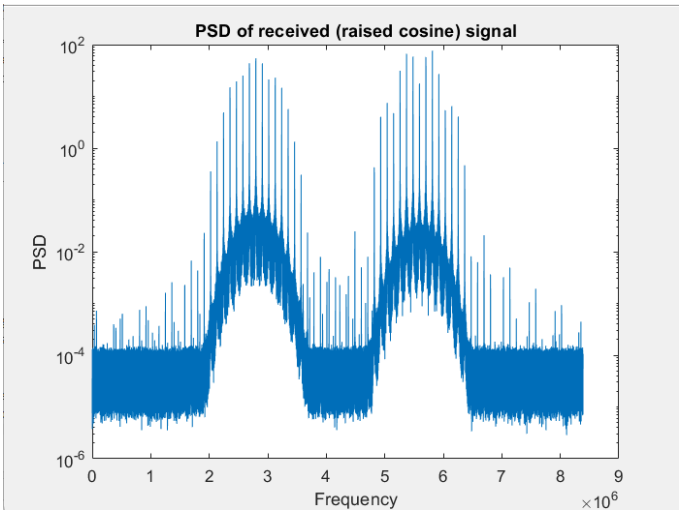



PSD of modulated (raised cosine) signal
PSD
Frequency
×10^6


PSD of Demodulated Signal (raised cosine)
PSD
Frequency
×10^6

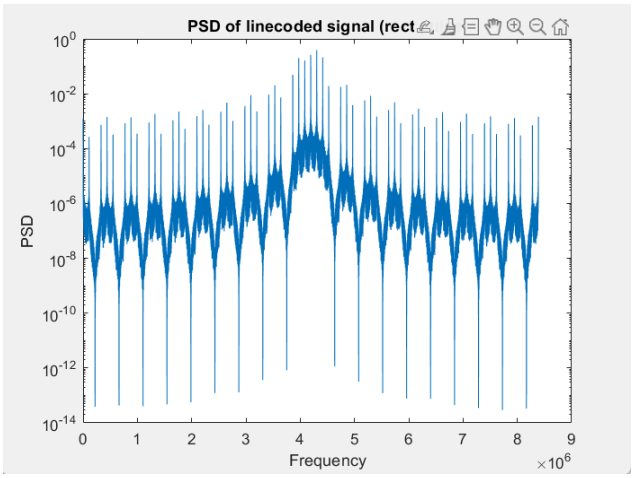

PSD of Inphase Demodulated (raised cosine) signal
PSD
Frequency
×10^6

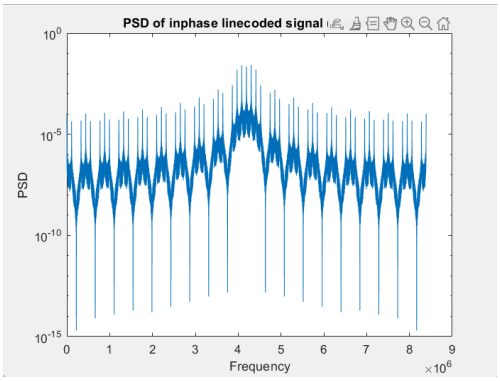

PSD of Quadphase Demodulated (raised cosine) signal
PSD
Frequency
×10^6

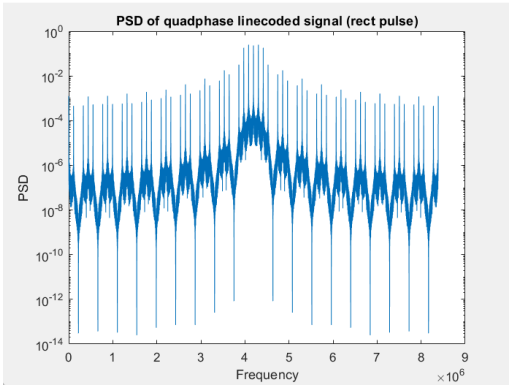


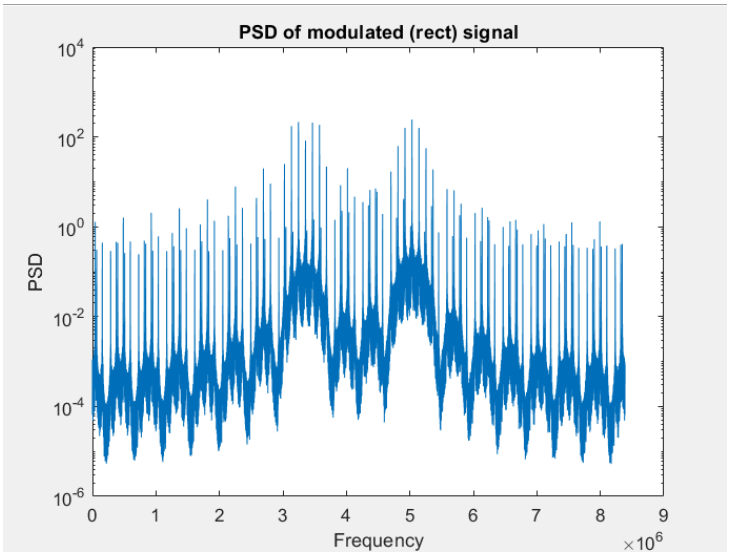
5. PSD - WITH MEMORY NOISE

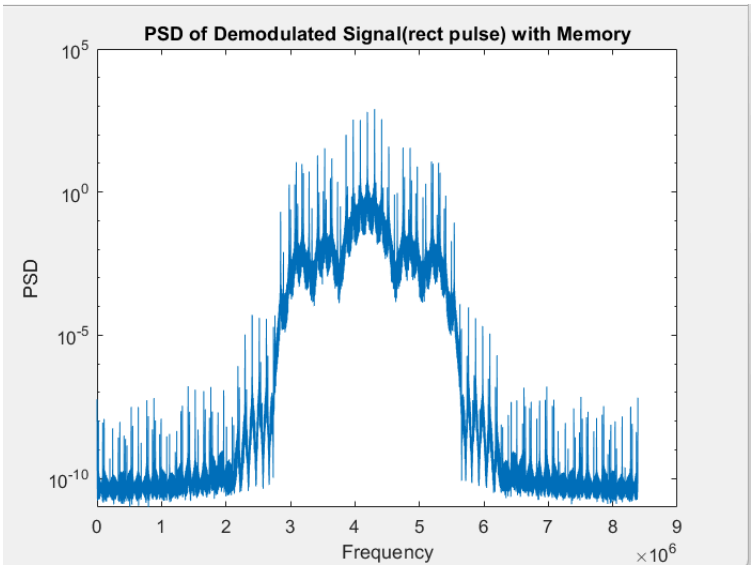
(a) Rect pulse

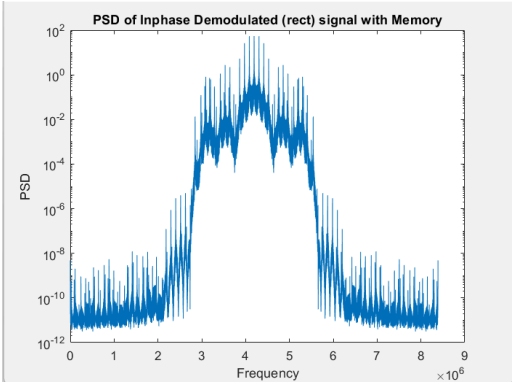


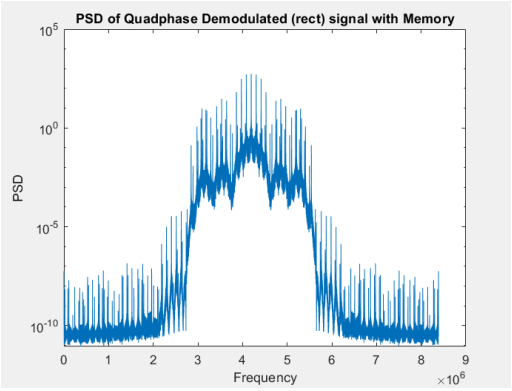


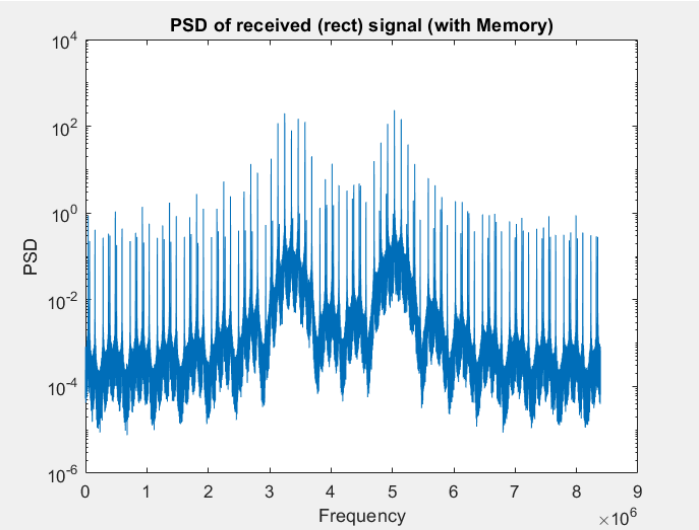



PSD of modulated (rect) signal
PSD
Frequency
×10^6

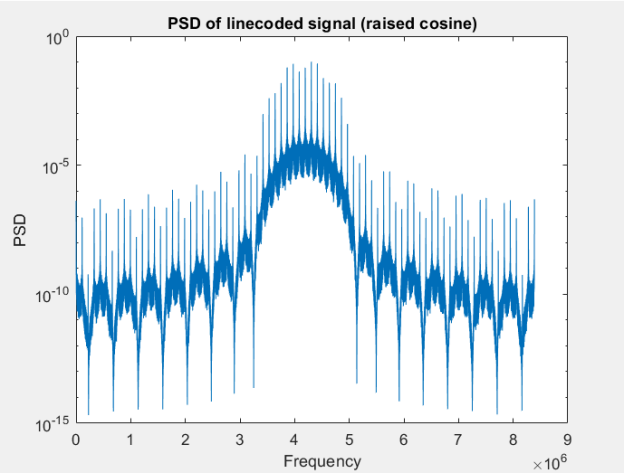

PSD of Demodulated Signal(rect pulse) with Memory
PSD
Frequency
×10^6

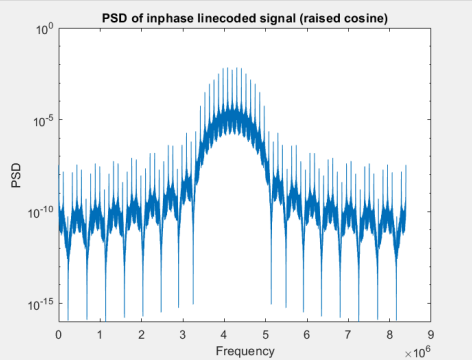

PSD of Inphase Demodulated (rect) signal with Memory
PSD
Frequency
×10^6

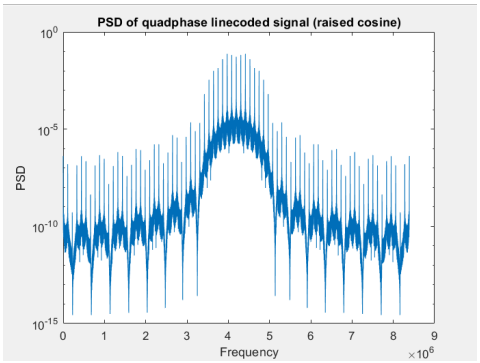

PSD of Quadphase Demodulated (rect) signal with Memory
PSD
Frequency
×10^6

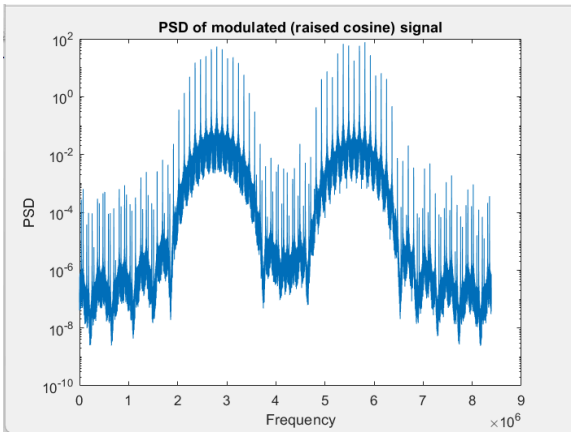


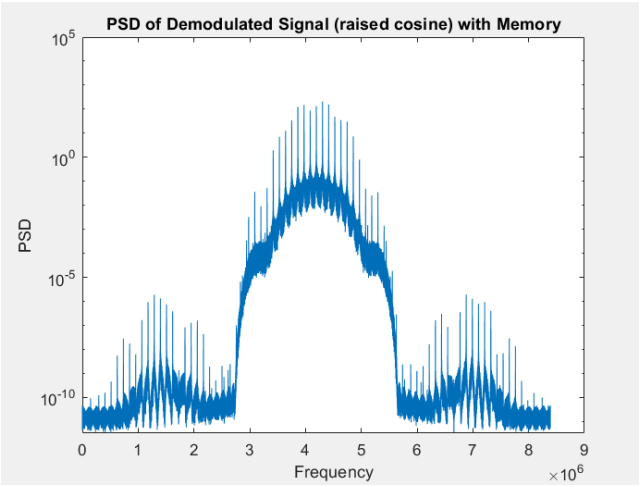
(b) Raised cosine

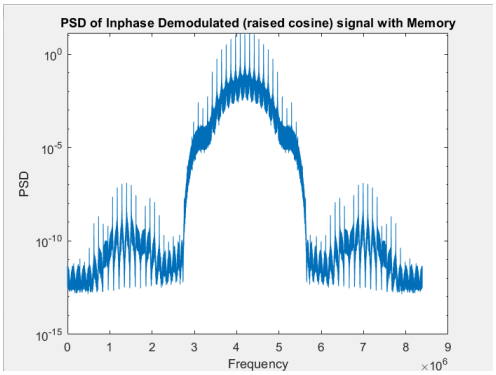


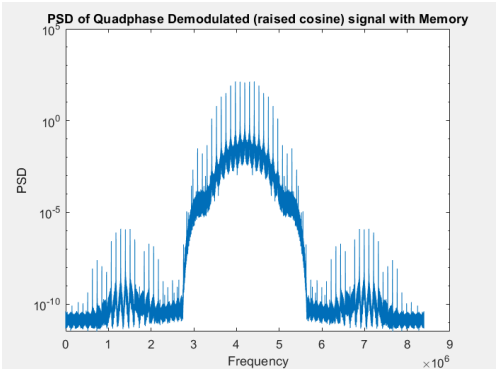


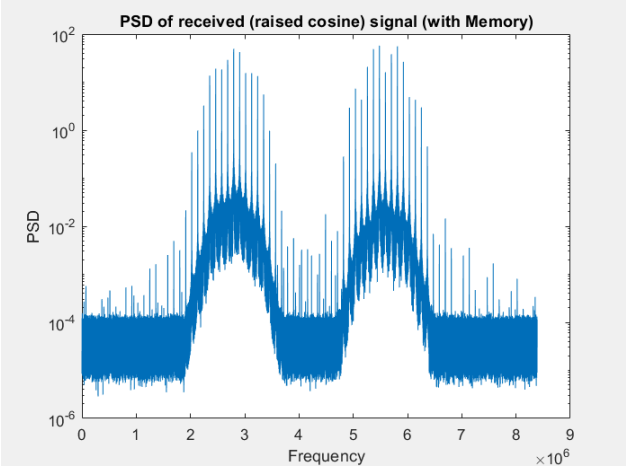



PSD of modulated (raised cosine) signal
PSD
Frequency
×10^6


PSD of Demodulated Signal (raised cosine) with Memory
PSD
Frequency
×10^6


PSD of Inphase Demodulated (raised cosine) signal with Memory
PSD
Frequency
×10^6

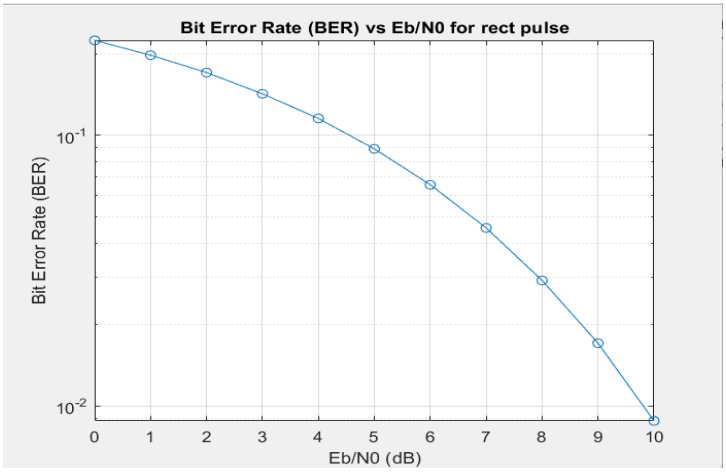

PSD of Quadphase Demodulated (raised cosine) signal with Memory
PSD
Frequency
×10^6



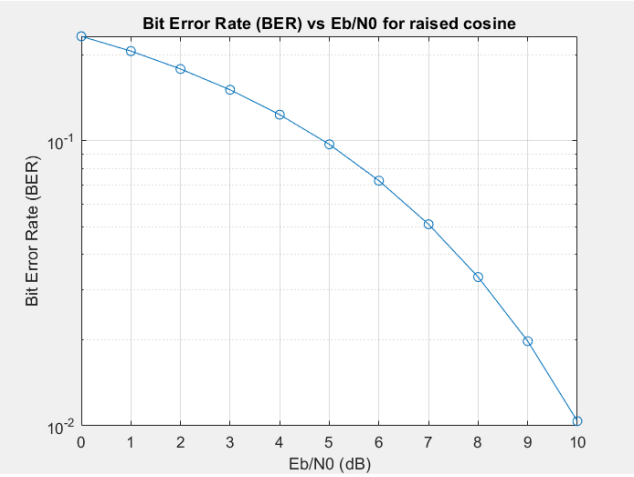
6. BER vs SNR

(a) WITHOUT MEMORY NOISE

(i) Rect pulse

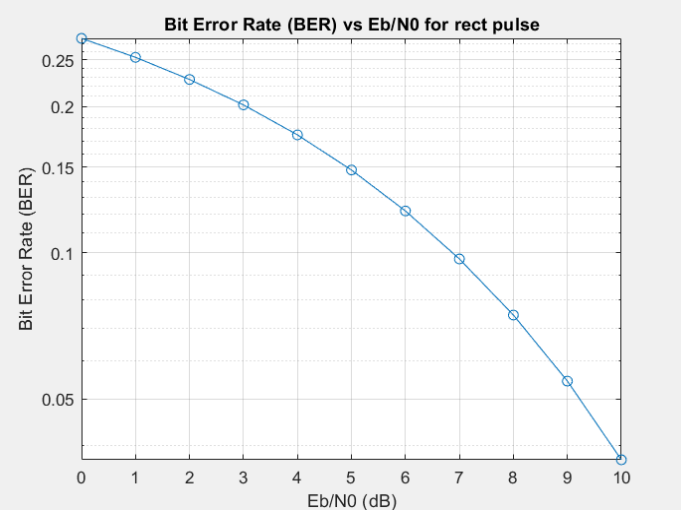


(ii) Raised cosine

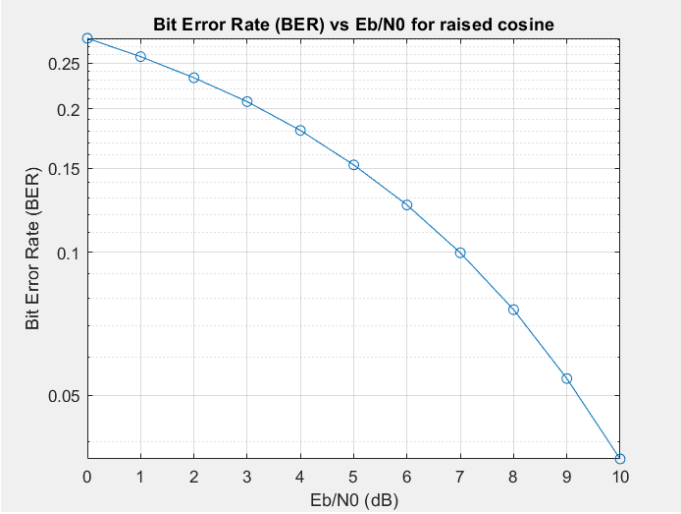


(b) WITH MEMORY NOISE

(i) Rect pulse



(ii) Raised cosine



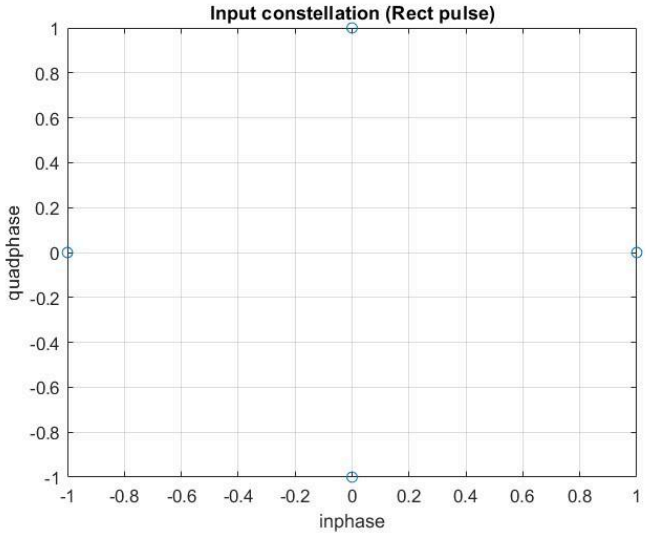
Typically, the probability of error (Pe) is lower when using a raised cosine pulse compared to a rectangular pulse.

7. CONSTELLATIONS

Input: Here we can observe that for input signals without any modulation, there is mapping directly to the points mentioned as (a,0),(0,a),(-a,0) & (0,-a).

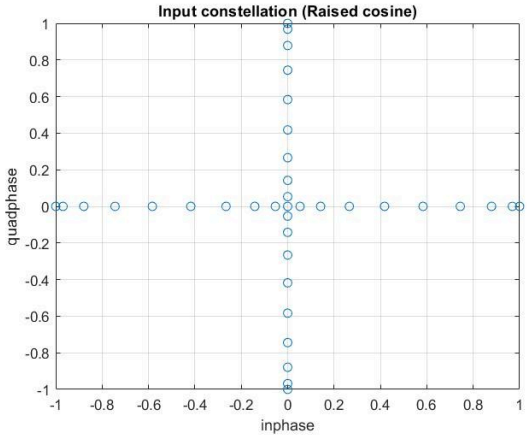
(a) Rect pulse

The observation is pretty clear in the case of rect as we can see in the figure below:



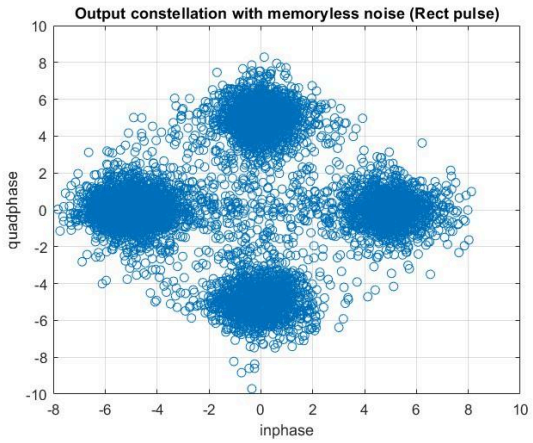
(b) Raised cosine

In the case of raised cosine, since we are multiplying with a raised cosine wave, the values obtained are not exactly as +a/-a, hence we can see many points in the range of -a to +a

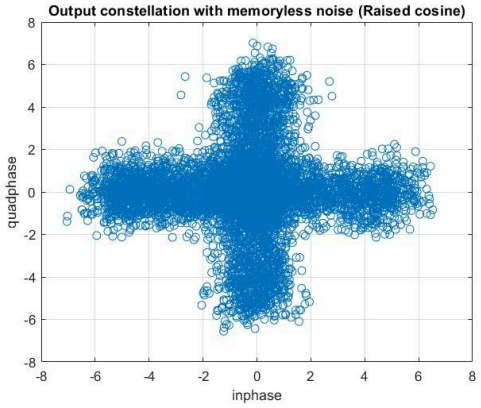


(a) **Without memory noise -** Here we can observe clusters of samples centered at points (a,0),(0,a),(-a,0) & (0,-a), but due to the presence of noise, scattering can be observed

(i) Rect pulse

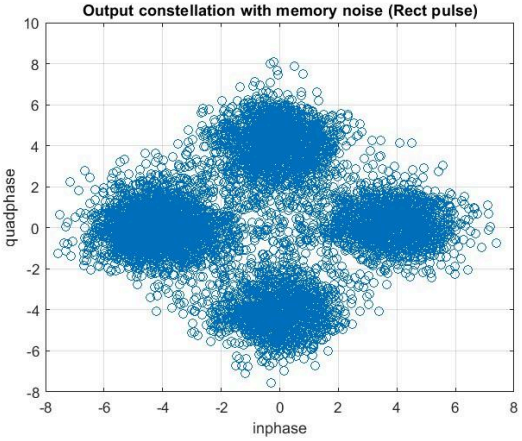


(ii) Raised cosine

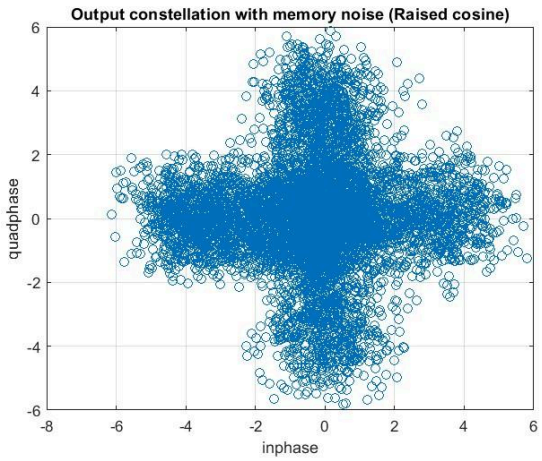


(b) With memory noise

(i) Rect pulse



(ii) Raised cosine



Here we can observe that for raised cosine the samples are pretty close to the axis while in the case of rect pulse, the samples are scattered more towards bigger angles. These scattered samples are difficult to map to a specific symbol.
Hence, we can conclude that raised cosine gives better decision making output as compared to rect pulse.
Also, comparing the constellations of memoryless channel and with memory channel, we can observe that less scattering is present for memoryless channel.
Therefore, Pe(Probability of error) for AWGN memoryless channel is much less than that of AWGN with memory channel.