SIMULATION OF A BPSK MODULATED BASEBAND SYSTEM WITH PULSE SHAPING FILTER

Submitted by: **GROUP NO. 6**

Aparna Gupta	18ECE1005
Ritesh Yadav	18ECE1018
Sanket Zanwar	18ECE1023
Mayank Raj	18ECE1032

NATIONAL INSTITUTE OF TECHNOLOGY GOA

FARMAGUDI, GOA

Date: 21st July, 2021

SIMULATION OF A BPSK MODULATED BASEBAND SYSTEM WITH PULSE SHAPING FILTER

AIM: Simulate a BPSK modulated baseband system with pulse shaping filter. Find the effect of roll-off factor in raised cosine and root raised cosine pulse shaping filter by plotting eye diagram.

TOOLS: MATLAB

THEORY:

BINARY PHASE SHIFT KEYING

Binary phase shift keying (BPSK) is the simplest form of digital phase modulation. For BPSK, each symbol consists of a single bit. Accordingly, we must choose two distinct values of θ(t), one to represent 0, and one to represent 1. Since there are 2π radians per cycle of carrier, and since our symbols can only take on two distinct values, we can choose θ(t) as follows. Let θ1(t), the value of θ(t) that represents a one, be 0, and let θ0(t), the value of θ(t) that represents a zero, be π. Doing so, we obtain:

$$S0(t) = \sqrt{Es}\ \cos(2\pi fc\ t + \pi),$$

$$S1(t) = \sqrt{Es}\ \cos(2\pi fc\ t + 0),$$

where p Es is the peak amplitude of the modulated sinusoidal carrier, S0(t) is the BPSK signal that represents a zero, and S1(t) is the BPSK signal that represents a one.

Phase Modulation Equals Amplitude Modulation

The expressions for S(t) given clearly show BPSK as a form of phase modulation. However, since: $\cos(\theta+\pi) = -\cos(\theta)$, we can rewrite S0(t) and S1(t) as:

$$S0(t) = -\sqrt{Es}\ \cos(2\pi fc\ t\ ),$$

$$S1(t) = \sqrt{Es}\ \cos(2\pi fc\ t\ ),$$

These expressions for S(t) show BPSK as a form of amplitude modulation, where $A0(t) = -1$ and $A1(t) = +1$. This begs the question: Is BPSK phase modulation or amplitude modulation? Both possibilities are correct, since the two are equivalent, as demonstrated by the trigonometric identity we used to convert between the two forms.

The modulation process is probably easier to understand when viewed from the perspective of amplitude modulation. For the example above, the carrier signal is $\sqrt{Es}\cos(2\pi fc\ t)$, and the amplitude modulation is a square wave that has an amplitude of ±1 and a period of T , the duration of one symbol. Fig. illustrates how we create BPSK by multiplying a sinusoidal carrier by rectangular bit pulses.

An Alternate Choice of θ(t)

In the previous example, we chose $\theta0(t) = \pi$ and $\theta1(t) = 0$. We could also chose $\theta0(t) = +\pi/2$ and $\theta1(t) = -\pi/2$. This results in:

$$S0(t) = \sqrt{Es}\ \cos(2\pi fc\ t + \pi/2\ ),$$

$$S1(t) = \sqrt{Es}\ \cos(2\pi fc\ t - \pi/2\ ).$$

Using the identities $\sin(\theta) = \cos(\pi/2 - \theta)$, $\cos(-\theta) = \cos(\theta)$, and $\sin(-\theta) = -\sin(\theta)$, we can re-write as:

$$S0(t) = -\sqrt{Es}\ \sin(2\pi fc\ t),$$

$$S1(t) = \sqrt{Es}\ \sin(2\pi fc\ t).$$

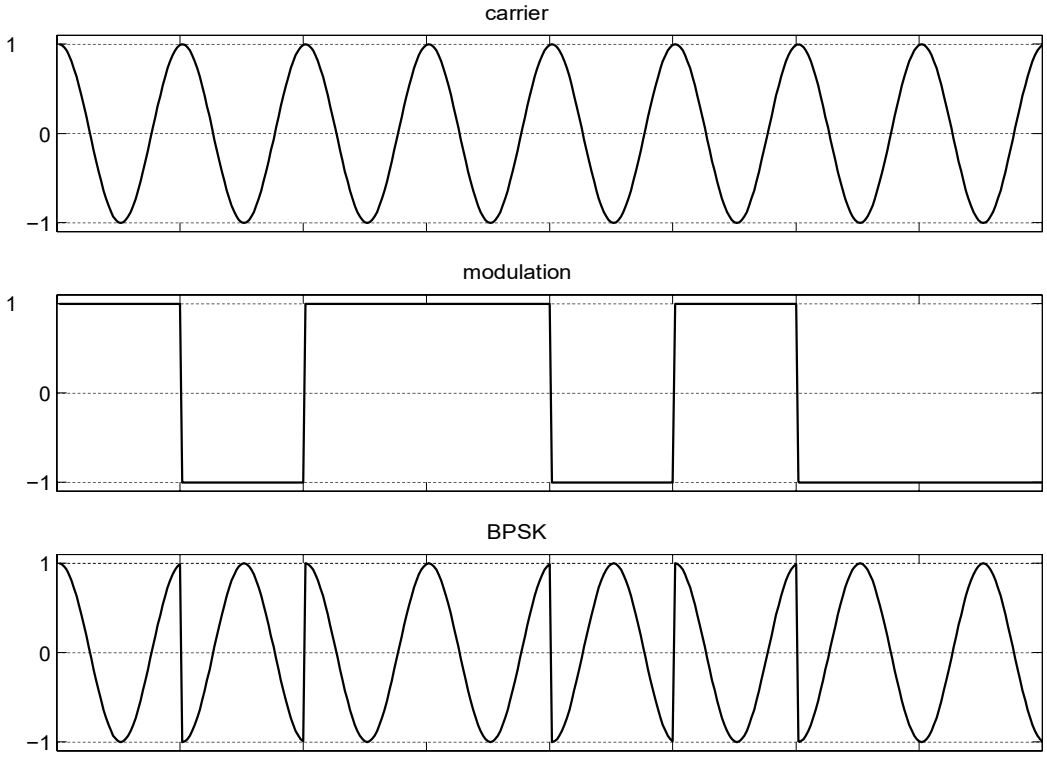


Figure1: BPSK Modulation

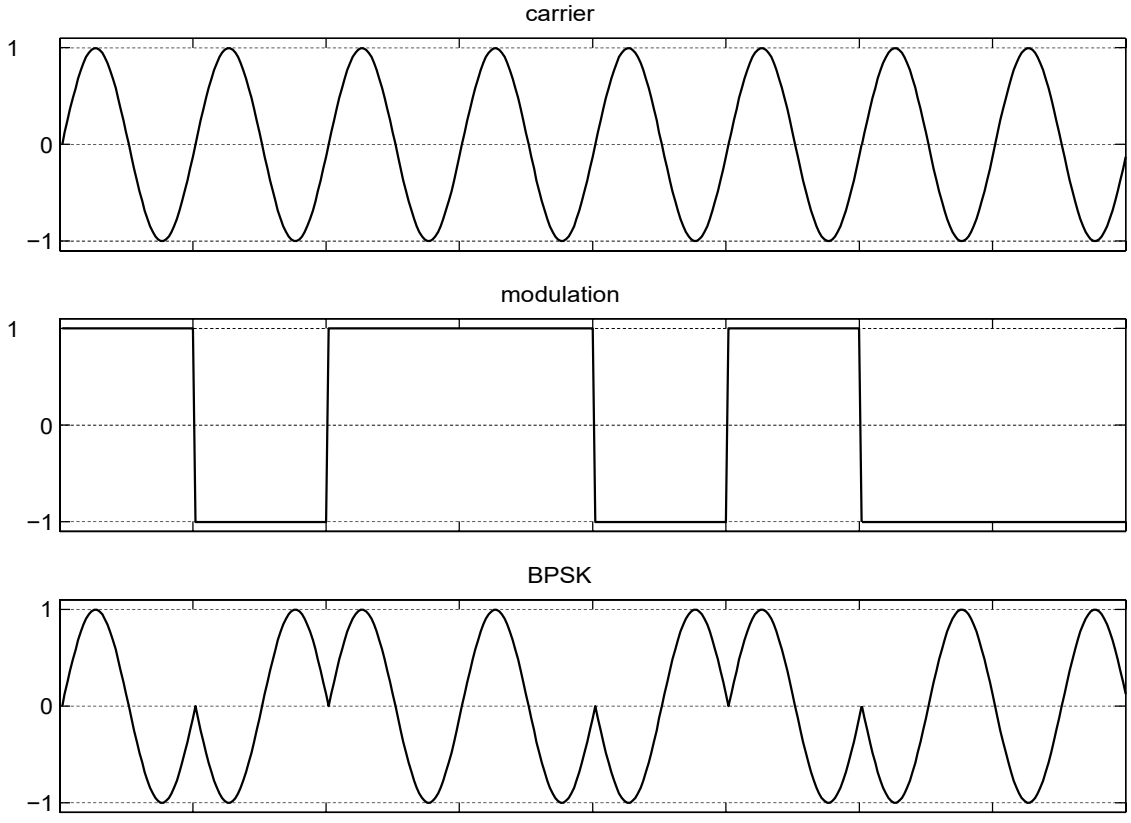


Figure2: BPSK Modulation

The result is once again a sinusoidal carrier multiplied by rectangular bit pulses, though the carrier is now sine instead of cosine, as shown in Fig. 2. Although this BPSK signal looks quite different from the one shown in Fig. 1, both represent the same bit pattern. The only difference is a carrier phase offset, caused by our choices of θ(t).

PULSE SHAPING

We represented the baseband symbols with rectangular pulses that had amplitudes of ±1 and widths of T . We may also think of the baseband symbols as weighted impulses to which we apply a pulse shape. The top graph of Fig. 3 shows a baseband information sequence consisting of weighted impulses. The middle graph of Fig. shows this same signal after applying a rectangular pulse shape to the impulses. The bottom graph of Fig. 3 shows the signal if we filter the impulses with a raised-cosine pulse shaping filter.

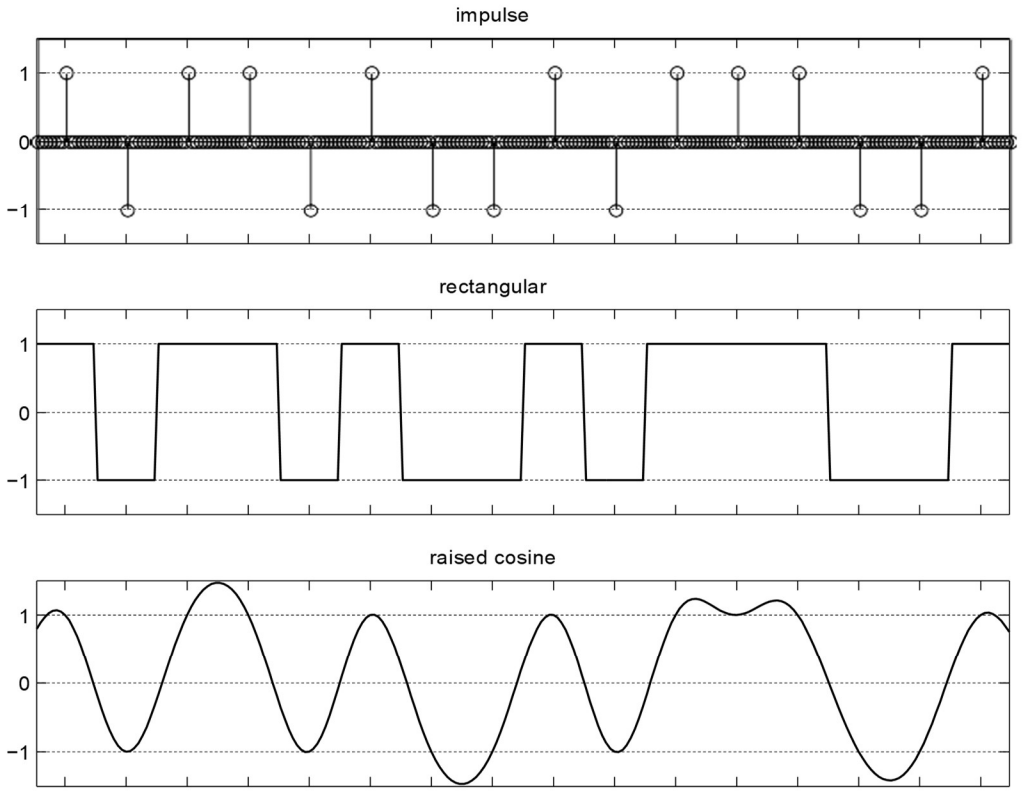


Figure 3: Baseband Pulse Shaping

The difference between the rectangular and raised-cosine pulse shapes is very easy to see in these time domain signals. Less evident, but more important, are the differences between the frequency spectrum of these signals. The smoother transitions of the raised-cosine pulse result in a signal that uses less bandwidth than those of the rectangular pulse. To illustrate this, Fig. 4 provides a comparison between the spectrum of the rectangular pulses and the raised-cosine filtered pulses. The dotted

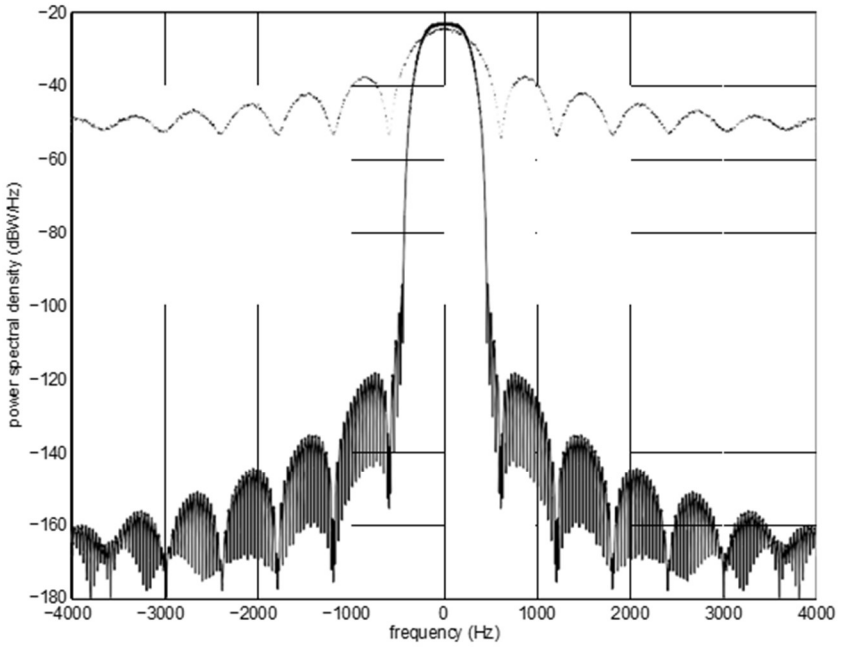


Figure 4: Shaped Spectrums

trace near the top of Fig. 4 is the spectrum of the rectangular pulses, while the solid trace near the bottom of Fig. 4 is the spectrum of the raised-cosine shaped pulses. The raised-cosine filter has not only narrowed the main spectral lobe, it has also nearly eliminated the sidelobes. In most digital communications systems, we split the task of pulse shaping equally between the transmitter and the receiver. In order to do this, we must use the squareroot of the raised-cosine filter response at both the transmitter and the receiver. This way, the product of the two filter responses will result in an overall raised-cosine response having zero ISI. Note that a signal which has only been filtered by one of the square-root raised-cosine filters (e.g. the signal on the channel) does not exhibit zero ISI.

THE EYE DIAGRAM

One consequence of pulse shaping is the need for accurate symbol timing recovery at the receiver. With rectangular pulse shaping, the symbol transitions are vertical lines. With raised-cosine pulse shaping, the symbol boundaries are hard to identify, since they are smooth and gradual. An eye diagram provides an easy way to observe the transitions between symbols and inspect the symbol timing. An eye diagram is simply the baseband signal repeatedly plotted over an interval of one symbol. The maximal opening of the eye indicates the center of the symbol, which is also the optimal time for the receiver to take a sample. The transitions between symbols cause the eye to close at the edges. Fig. 5 shows a typical eye diagram for BPSK with raised-cosine pulse shaping. At the center of the symbol, the eye converges to two points: ±1, while at the symbol edges, the eye takes on many different values. The exact shape of the eye is determined by the filter roll-off factor as well as the number of samples per symbol.

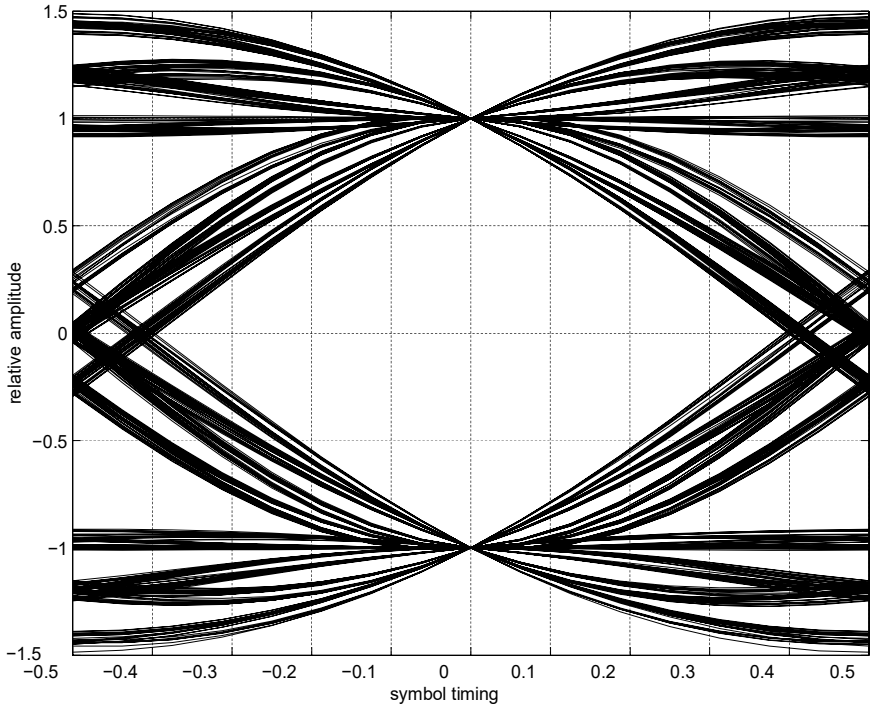


Figure 5: BPSK Eye Diagram

RAISED COSINE FILTER

The raised cosine filter is one of the most common pulse-shaping filters in communications systems. In addition, it is used to minimize intersymbol interference (ISI) by attenuating the starting and ending portions of the symbol period. Because these portions are most susceptible to creating interference from multi-path distortion, the shaping characteristics of the raised cosine filter helps reduce ISI. This impulse response for this filter is given by the equation shown below:

$$h_{RC}(n) = \frac{\pi}{4}\text{sinc}\left(\frac{\pi n}{R}\right) \cdot \left[\text{sinc}\left(\frac{\pi}{2} - \alpha\frac{\pi n}{R}\right) + \frac{\sin\left(\frac{\pi}{2} - \alpha\frac{\pi n}{R}\right)}{\left(\frac{\pi}{2} + \alpha\frac{\pi n}{R}\right)}\right]$$

As the equation shows, the sinc pulse is implemented in the creation of this filter. The filter rolloff parameter, alpha(α), can range between values of 0 and 1. The impulse response of the resulting filter is shown below:

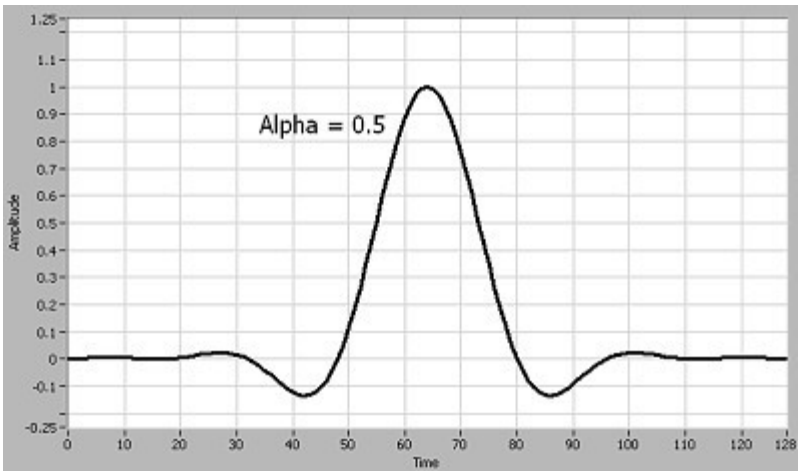


Figure 7: Impulse Response of Raised Cosine Filter

Again, the sinc pulse is shaped such that ideally, the resulting channel bandwidth will be described by the equation:

$$Bw = Rs\ (1 + \alpha)$$

Ideally, the frequency response (FFT) of the sinc pulse should yield a completely square response such that a specific frequency bandwidth (Bw is exactly half of the symbol rate (Rs). However, in the non-ideal programming environment, the actual frequency response differs slightly from the ideal response (due to estimation of infinity, ∞). Below, we show the non-ideal frequency response simply by taking the FFT (logarithmic) of the impulse response.

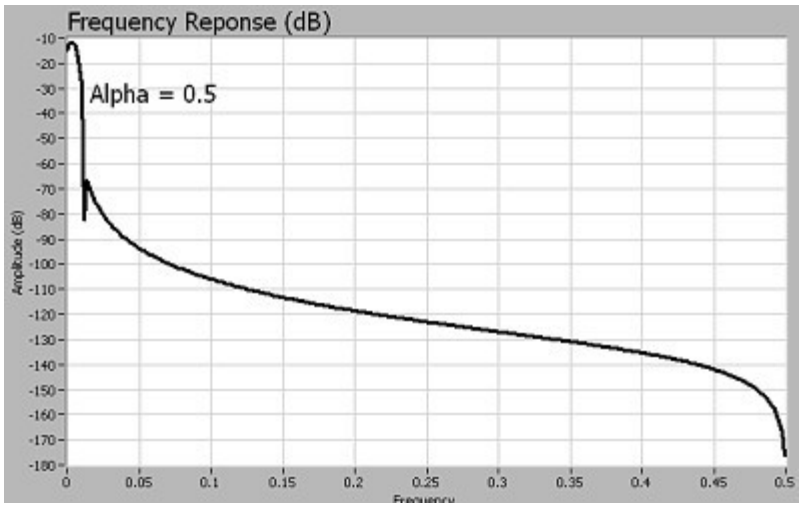


Figure 8: Frequency Response of Raised Cosine Filter

In this specific example, we've used a rolloff factor (α) of 0.5. Note that the bandwidth of the signal is concentrated in a specific frequency range. Again this is critical in a communications system because keeping channels bandlimited is necessary to prevent adjacent channel interference.

ROOT RAISED COSINE FILTER

The root raised cosine filter produces a frequency response with unity gain at low frequencies and complete at higher frequencies. It is commonly used in communications systems in pairs, where the transmitter first applies a root raised cosine filter, and then the receiver then applies a matched filter. Mathematically, the raised cosine filter can be defined by the following equation:

$$h_{SRC}(t) = \frac{\left[\frac{4\alpha}{\pi}\cos\left(\frac{(1+\alpha)\pi n}{R}\right)\right] + \left[(1-\alpha)\operatorname{sinc}\left(\frac{(1-\alpha)\pi n}{R}\right)\right]}{\sqrt{R}\left[1-\left(\frac{4\alpha n}{R}\right)^2\right]}$$

In this equation, α is the rolloff factor, which determines the sharpness of the frequency response. In addition, R is the number of samples per symbol. As the equation above illustrates, the sinc pulse is used to shape the filter so that it appears with a finite frequency response. The impulse response for this filter is shown below:

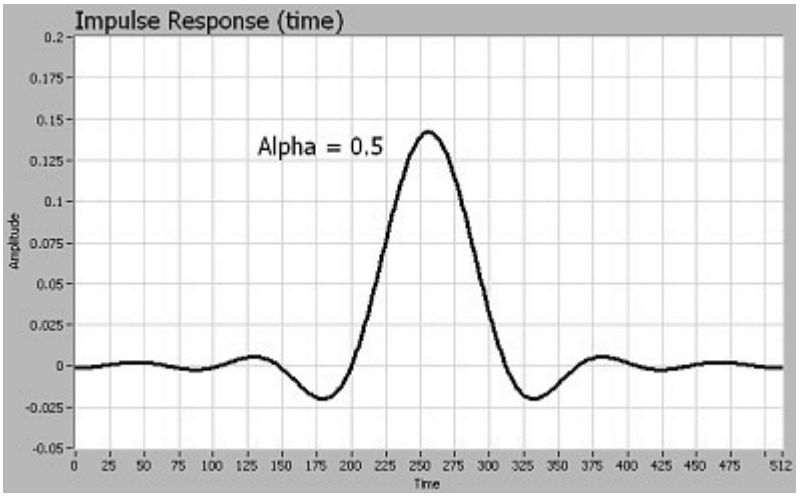


Figure 9: Impulse Response of Root Raised Cosine Filter

As the graph illustrates, the impulse response resembles the sinc pulse described previously. As mentioned earlier, this should ideally have a perfectly square frequency response. We show the actual frequency response of the root raised cosine filter below:

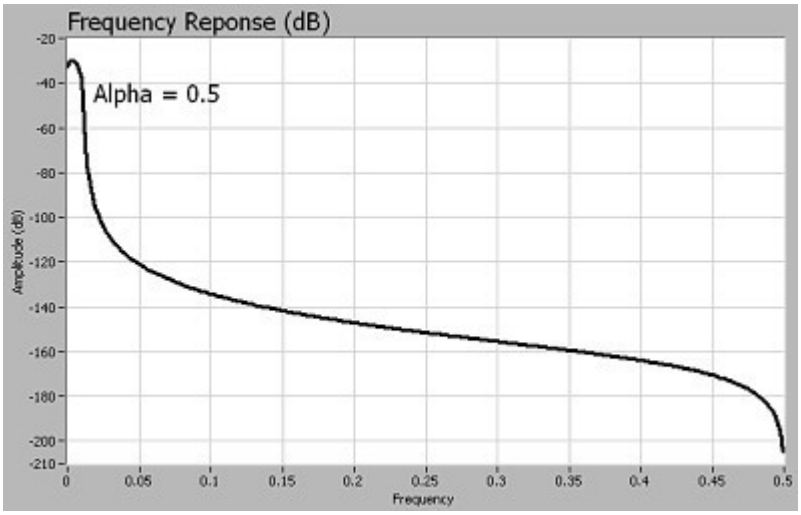


Figure 10: Frequency Response of Root Raised Cosine Filter

Again, note from the figure above that an FFT of the impulse response is not completely ideal due to estimation of infinity. Again, we've used a rolloff factor (α) of 0.5. Also note that like the raised cosine filter, the bandwidth of the signal is concentrated in a specific frequency range.

TRUNCATION

The good thing about the square-root raised cosine pulse shape is that the corresponding matched filter output has no ISI. The bad thing is that the pulse shape has infinite support in time. In a practical system, pulses cannot last indefinitely. So the pulse shape is truncated. The result of truncation is the presence of non-zero side lobes in the frequency domain — the spectrum is no longer zero for jf j > 1+

2Ts . This is illustrated in Figures 3 through 5. In Figure 3, the pulse given by (5) is sampled at N = 4 samples/symbol and is truncated to span only 4 symbols as shown in the upper plot. The lower plot of Figure 3 shows the consequence in the frequency domain: high sidelobes and a significant pass-band ripple. The stop band attenuation is only 18 dB which is not enough for practical applications. In Figure 4, the pulse given by (5) is sampled at N = 4 samples/symbol and is truncated to span 8 symbols as shown in the upper plot. In the frequency domain, we see that the pass band ripple has been eliminated but and the out-of-band sidelobes are now about 25 dB down. In Figure 5, the pulse given by (5) is sampled at N = 4 samples/symbol and is truncated to span 16 symbols as shown in the upper plot. Now the out-of-band sidelobes are about 32 dB down. Clearly, as the time span of the pulse is increased, the spectrum approaches the ideal spectrum. In general, the smaller the roll-off factor, the longer the pulse shape needs to be in order to achieve a desired stop-band attenuation. Current practice requires a stop band attenuation of about 40 dB. A good rule-of-thumb that achieves this is

$$L_{\text{symbol}} = -44\beta + 33 \qquad (6)$$

for $0{:}2 < \beta << 0{:}75$ where L_{symbol} is the length of the filter measured in symbols. Clearly, the prediction of $L_{\text{symbol}} = 0$ for $\beta = 0{:}75$ is an understatement of the required filter length. The values generated by this formula are intended to be starting points. The resulting filter characteristics should be verified using the DFT.

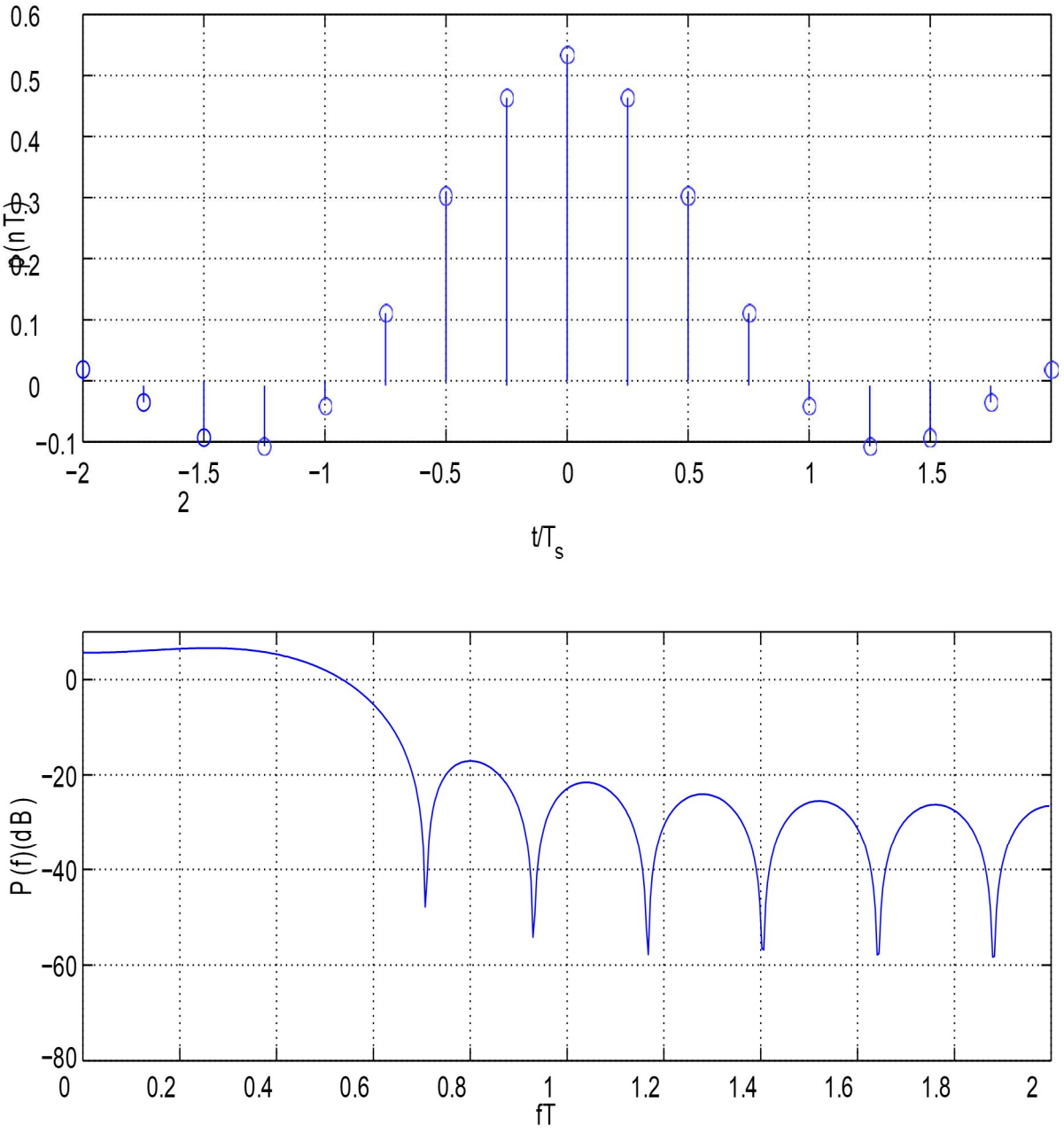


Figure 3: The effects of truncation on the square-root raised cosine pulse shape. Top plot: the square-root raised cosine pulse shape sampled at N = 4 samples/symbol with β= 0:5 and truncated to span 4 symbols. Lower plot: the corresponding spectrum.

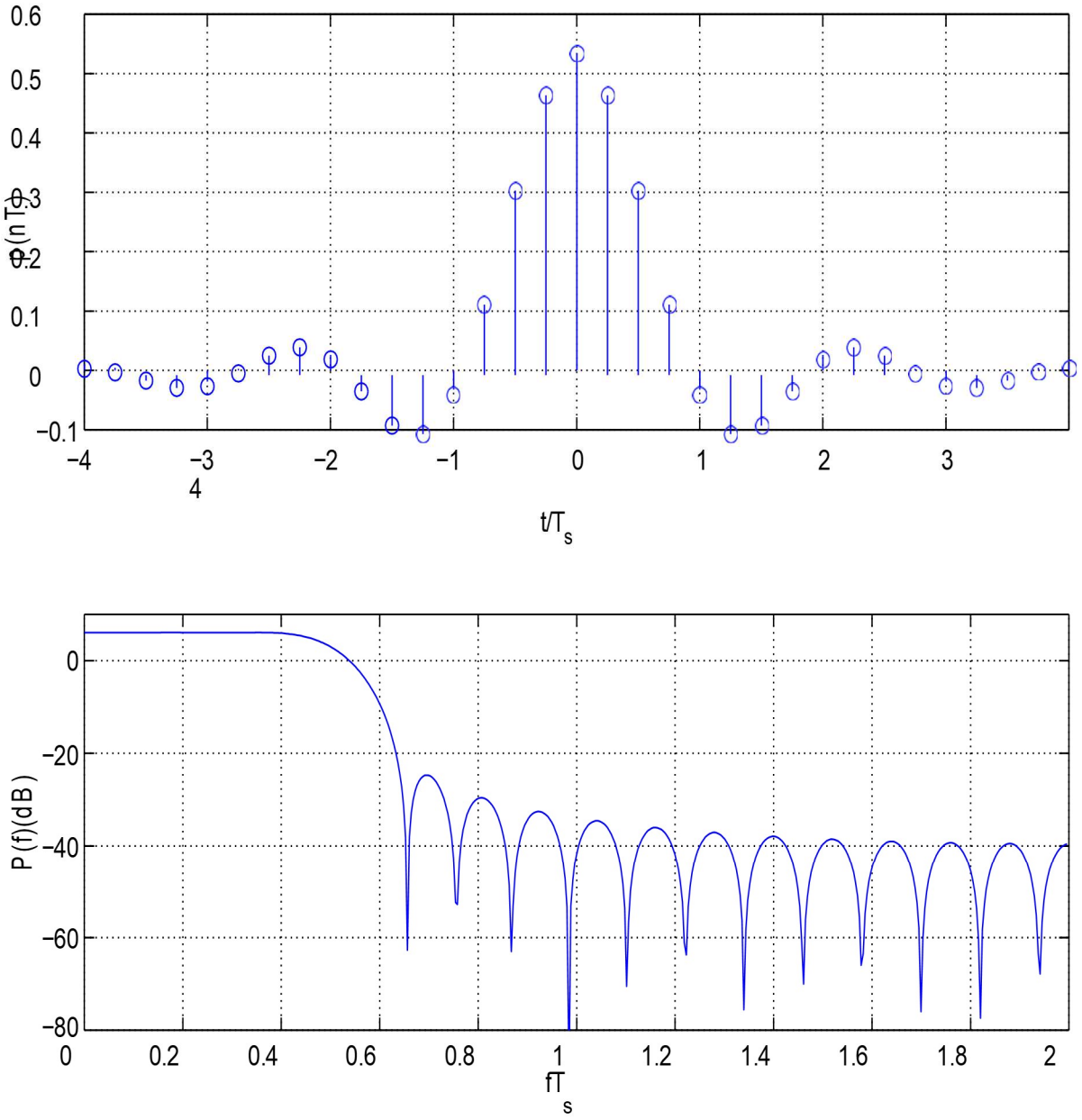


Figure 5: The effects of truncation on the square-root raised cosine pulse shape. Top plot: the square-root raised cosine pulse shape sampled at N = 4 samples/symbol with β= 0:5 and truncated to span 16 symbols. Lower plot: the corresponding spectrum.

PRINCIPLE:

Unlike the rectangular pulse, the raised cosine pulse takes on the shape of a sinc pulse, as indicated by the leftmost term of p(t). Unfortunately, the name “raised cosine” is misleading. It actually refers to the pulse’s frequency spectrum, $P(\omega)$, not to its time domain shape, p(t). The precise shape of the raised cosine spectrum is determined by the parameter, α, where $0 \leq \alpha \leq 1$.

Specifically, α governs the bandwidth occupied by the pulse and the rate at which the tails of the pulse decay. A value of $\alpha = 0$ offers the narrowest bandwidth, but the slowest rate of decay in the time domain. When $\alpha = 1$, the bandwidth is $1/\tau$, but the time domain tails decay rapidly. It is interesting to note that the $\alpha = 1$ case offers a double-sided bandwidth of $2/\tau$. This exactly matches the bandwidth of the main lobe of a rectangular pulse, but with the added benefit of rapidly decaying time-domain tails. Conversely, inverse when $\alpha = 0$, the bandwidth is reduced to $1/\tau$, implying a factor-of-two increase in data rate for the same bandwidth occupied by a rectangular pulse. However, this comes at the cost of a much slower rate of decay in the tails of the pulse. Thus, the parameter α gives the system designer a trade-off between increased data rate and time-domain tail suppression. The latter is of prime importance for systems with relatively high timing jitter at the receiver.

APPLICATIONS OF RRC AND RC FILTER:

Root Raised Cosine (RRC) pulse shaping is used in the space telecommunication. Use of the RRC filtering (i.e., pulse shaping) is adopted in commercial communications, such as cellular technology, and used extensively.

The raised-cosine filter is a filter frequently used for pulse shaping in digital modulation due to its ability to minimise Intersymbol Interference (ISI).

PROCEDURE:

Part 1: BPSK modulated baseband system with pulse shaping filter

1. Define number of samples, frequency, filter order and filter span in symbols.
2. Produce random binary data using random function.
3. Modulate the random binary data using pskmod function with modulation order i.e. 2 for BPSK.
4. Design the root raised cosine filter with roll off factor.
5. Calculate snr.
6. Add AWGN.
7. Plot the impulse response.
8. Repeat steps 2 to 7. This time design raised cosine filter with roll off factor.
9. Repeat steps 2 to 7. This time design raised cosine Nyquist filter.

Part 1: MATLAB CODE:

```
close all;
clear all;
clc;
M = 2;
k = log2(M);
fs = 1;
T = 1/fs;
t = 0:T:100;
nsamp = 5;
n=20000
fftlength = 2^nextpow2(length(t));
ebno = 5;
filtorder = 20;
delay = filtorder/(nsamp*2);
rolloff = 0.5;
span = 10; % Filter span in symbols
filtDelay = k*span;

%% square root raised cosine filter.
rrcfilter = rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
x = randi([0,M-1],n,1); % Random binary data
ytx = pskmod(x,M);
ytx = rcosflt(ytx,1,nsamp,'filter',rrcfilter);
```

```
snr = ebno + 10*log10(k) - 10*log10(nsamp);
ynoisy = awgn(ytx, snr, 'measured');
fvtool(rrcfilter,'impulse')
title('Impulse Response for Root raised cosine filter');


%% raised cosine filter.
rcfilter = rcosine(1,nsamp,'fir',rolloff,delay);
x1 = randi([0,M-1],length(t),1); % Random binary data
ytx1 = pskmod(x1,M);
ytx1 = rcosflt(ytx1,1,nsamp,'filter',rcfilter);
snr = ebno + 10*log10(k) - 10*log10(nsamp);% add some
AWGN noise
ynoisy = awgn(ytx1, snr, 'measured');
fvtool(rcfilter,'impulse')
title('Impulse Response for Raised cosine filter');


%% Create a raised cosine nyquist filter.
rolloff = 0; % Rolloff factor of filter
rcnfilter = rcosine(1,nsamp,'fir',rolloff,delay);
x1 = randi([0,M-1],length(t),1); % Random binary data
ytx2 = pskmod(x1,M);
ytx2 = rcosflt(ytx2,1,nsamp,'filter',rcnfilter);
snr = ebno+10*log10(k) - 10*log10(nsamp);
ynoisy = awgn(ytx2, snr, 'measured');
fvtool(rcnfilter,'impulse')
title('Impulse Response for Raised cosine nyquist
filter');

figure
stem(rcfilter, 'Linewidth',2, 'color','b');
hold on
plot(rrcfilter);
hold on
stem(rcnfilter,'--ro', 'Linewidth',2);
hold on
plot(rcnfilter);
hold on
stem(rrcfilter,'-.k*','Linewidth',2);
hold on
plot(rcfilter);
grid on;
axis([1 21 -0.3 1.1]);
legend('RRC','RCN','RC');
title('RCC vs RCN vs RC');
```

```
ylabel('Amplitude');
xlabel('Samples');
```

Part 2: Effect of different roll off factors in raised cosine and root raised cosine pulse shaping filter

1. Define number of samples.
2. Produce random binary data using random function.
3. Modulate the random binary data using pskmod function with modulation order i.e. 2 for BPSK.
4. Define number of symbols and frequency.
5. Define sinc filter.
6. Define root raised cosine filter with different alpha values.
7. Unsample the transmit sequence.
8. Filter the sequence thus obtained.
9. Plot the eye diagram for few values of roll off factors and compare.
10. Repeat steps 1 to 9 for raised cosine filter.
11. Repeat steps 1 to 10 by adding noise.

Part 2: MATLAB CODE:

```
close all;
clear all;
clc;
M  = 2;
n = 1000;
x = randi([0,M-1],n,1); % Random binary data
am = pskmod(x,M)

% adding noise
%am = awgn(am,15,'measured');  % comment this line for
without noise

N  = 10^3; % number of symbols
fs = 10; % sampling frequency in Hz

% defining the sinc filter
```

```
sincNum = sin(pi*[-fs:1/fs:fs]); % numerator of the sinc
function
sincDen = (pi*[-fs:1/fs:fs]); % denominator of the sinc
function
sincDenZero = find(abs(sincDen) < 10^-10);
sincOp = sincNum./sincDen;
sincOp(sincDenZero) = 1; % sin(pix/(pix) =1 for x =0


%% root raised cosine filter

alpha = 0.02;
cosNum = sin(pi*[-1/fs:fs:1/fs]*(1-alpha)) + 4*alpha*[-
1/fs:fs:1/fs]*cos(pi*[-1/fs:fs:1/fs]*(1+alpha));
cosDen = pi*[-1/fs:fs:1/fs]*(1-(4*[-
1/fs:fs:1/fs]*alpha).^2);
cosDenZero = find(abs(cosDen)<10^-10);
cosOp = cosNum./cosDen;
cosOp(cosDenZero) = pi/4;

gt_alpha2 = sincOp.*cosOp;



alpha = 0.5;
cosNum = sin(pi*[-1/fs:fs:1/fs]*(1-alpha)) + 4*alpha*[-
1/fs:fs:1/fs]*cos(pi*[-1/fs:fs:1/fs]*(1+alpha));
cosDen = pi*[-1/fs:fs:1/fs]*(1-(4*[-
1/fs:fs:1/fs]*alpha).^2);
cosDenZero = find(abs(cosDen)<10^-10);
cosOp = cosNum./cosDen;
cosOp(cosDenZero) = pi/4;

gt_alpha5 = sincOp.*cosOp;

alpha = 0.75;
cosNum = sin(pi*[-1/fs:fs:1/fs]*(1-alpha)) + 4*alpha*[-
1/fs:fs:1/fs]*cos(pi*[-1/fs:fs:1/fs]*(1+alpha));
cosDen = pi*[-1/fs:fs:1/fs]*(1-(4*[-
1/fs:fs:1/fs]*alpha).^2);
cosDenZero = find(abs(cosDen)<10^-10);
cosOp = cosNum./cosDen;
cosOp(cosDenZero) = pi/4;

gt_alpha75 = sincOp.*cosOp;
```

```
alpha = 1;
cosNum = sin(pi*[-1/fs:fs:1/fs]*(1-alpha)) + 4*alpha*[-
1/fs:fs:1/fs]*cos(pi*[-1/fs:fs:1/fs]*(1+alpha));
cosDen = pi*[-1/fs:fs:1/fs]*(1-(4*[-
1/fs:fs:1/fs]*alpha).^2);
cosDenZero = find(abs(cosDen)<10^-10);
cosOp = cosNum./cosDen;
cosOp(cosDenZero) = pi/4;
gt_alpha1 = sincOp.*cosOp;

% upsampling the transmit sequence
am = transpose(am);
amUpSampled = [am;zeros(fs-1,length(am))];
amU = amUpSampled(:).';

% filtered sequence
st_alpha5 = conv(amU,gt_alpha5);
st_alpha1 = conv(amU,gt_alpha1);
st_alpha75 = conv(amU,gt_alpha75);
st_alpha2 = conv(amU,gt_alpha2);

% taking only the first 10000 samples
st_alpha5 = st_alpha5([1:10000]);
st_alpha1 = st_alpha1([1:10000]);
st_alpha2 = st_alpha2([1:10000]);
st_alpha75 = st_alpha75([1:10000]);

st_alpha5_reshape = reshape(st_alpha5,fs*2,N*fs/20).';
st_alpha1_reshape = reshape(st_alpha1,fs*2,N*fs/20).';
st_alpha2_reshape = reshape(st_alpha2,fs*2,N*fs/20).';
st_alpha75_reshape = reshape(st_alpha75,fs*2,N*fs/20).';

close all
subplot(2,2,1)
plot([0:1/fs:1.99],real(st_alpha2_reshape).','b');
title('Eye diagram with alpha=0.02 for  root raised
cosine');
xlabel('time')
ylabel('amplitude')
axis([0 2 -1.5 1.5])
grid on

subplot(2,2,2)
plot([0:1/fs:1.99],real(st_alpha5_reshape).','b');
title('Eye diagram with alpha=0.5  for root raised
cosine');
xlabel('time')
```

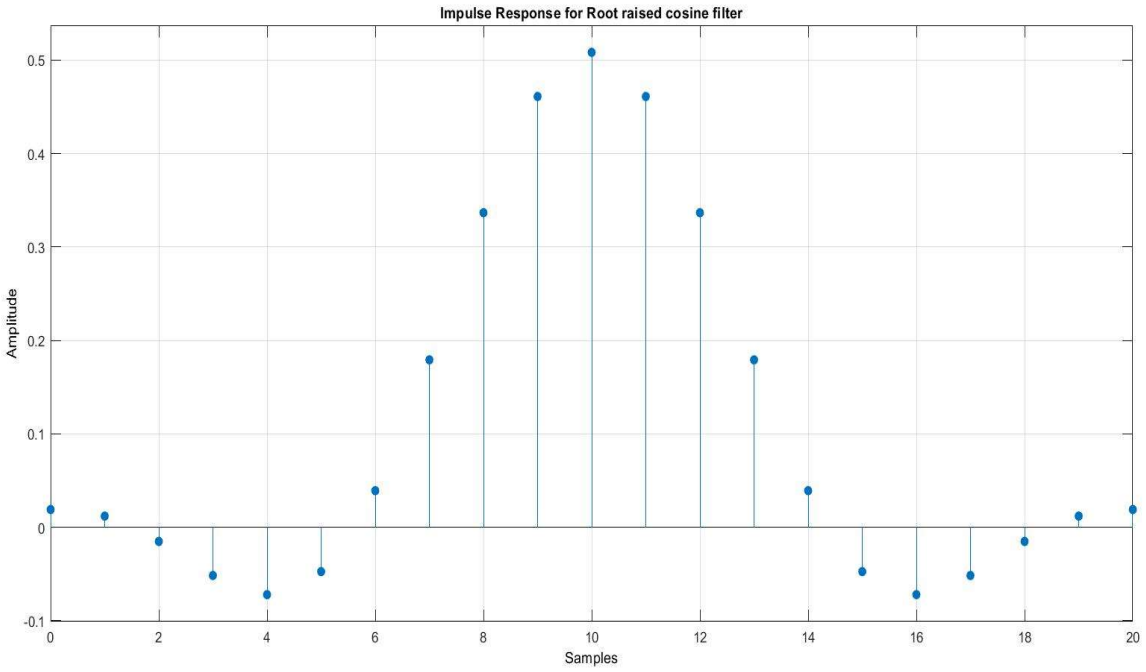
```
ylabel('amplitude')
axis([0 2 -1.5 1.5])
grid on

subplot(2,2,3)
plot([0:1/fs:1.99],real(st_alpha75_reshape).','b');
title('Eye diagram with alpha=0.75 for root raised
cosine');
xlabel('time')
ylabel('amplitude')
axis([0 2 -1.5 1.5])
grid on

subplot(2,2,4)
plot([0:1/fs:1.99],real(st_alpha1_reshape).','b');
title('Eye diagram with alpha=1 for root raised cosine')
xlabel('time')
ylabel('amplitude')
axis([0 2 -1.5 1.5 ])
grid on

%% raised cosine filter
% raised cosine filter
alpha = 0.5;
cosNum = cos(alpha*pi*[-fs:1/fs:fs]);
cosDen = (1-(2*alpha*[-fs:1/fs:fs]).^2);
cosDenZero = find(abs(cosDen)<10^-10);
cosOp = cosNum./cosDen;
cosOp(cosDenZero) = pi/4;

gt_alpha5 = sincOp.*cosOp;

alpha = 0.02;
cosNum = cos(alpha*pi*[-fs:1/fs:fs]);
cosDen = (1-(2*alpha*[-fs:1/fs:fs]).^2);
cosDenZero = find(abs(cosDen)<10^-10);
cosOp = cosNum./cosDen;
cosOp(cosDenZero) = pi/4;

gt_alpha2 = sincOp.*cosOp;

alpha = 0.75;
cosNum = cos(alpha*pi*[-fs:1/fs:fs]);
cosDen = (1-(2*alpha*[-fs:1/fs:fs]).^2);
cosDenZero = find(abs(cosDen)<10^-10);
cosOp = cosNum./cosDen;
cosOp(cosDenZero) = pi/4;
```

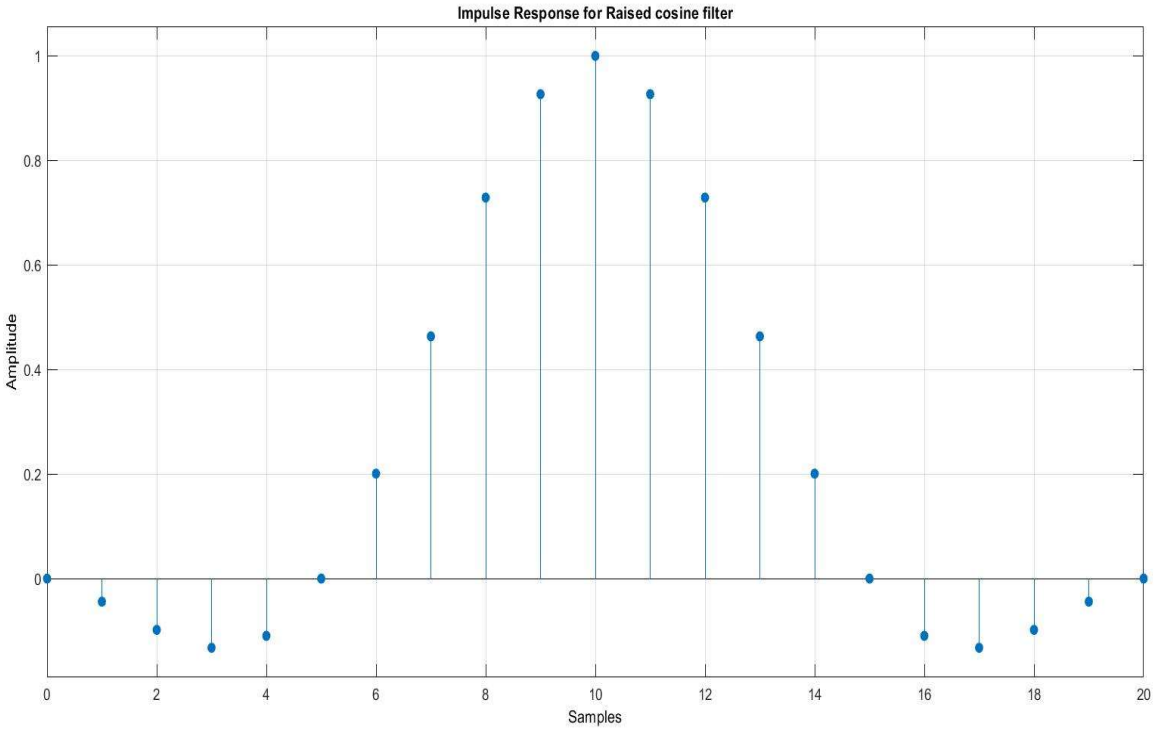
```
gt_alpha75 = sincOp.*cosOp;

alpha = 1;
cosNum = cos(alpha*pi*[-fs:1/fs:fs]);
cosDen = (1-(2*alpha*[-fs:1/fs:fs]).^2);
cosDenZero = find(abs(cosDen)<10^-10);
cosOp = cosNum./cosDen;
cosOp(cosDenZero) = pi/4;
gt_alpha1 = sincOp.*cosOp;

% upsampling the transmit sequence
amUpSampled = [am;zeros(fs-1,length(am))];
amU = amUpSampled(:).';

% filtered sequence
st_alpha5 = conv(amU,gt_alpha5);
st_alpha1 = conv(amU,gt_alpha1);
st_alpha2 = conv(amU,gt_alpha2);
st_alpha75 = conv(amU,gt_alpha75);

% taking only the first 10000 samples
st_alpha5 = st_alpha5([1:10000]);
st_alpha1 = st_alpha1([1:10000]);
st_alpha2 = st_alpha2([1:10000]);
st_alpha75 = st_alpha75([1:10000]);

st_alpha5_reshape = reshape(st_alpha5,fs*2,N*fs/20).';
st_alpha1_reshape = reshape(st_alpha1,fs*2,N*fs/20).';
st_alpha2_reshape = reshape(st_alpha2,fs*2,N*fs/20).';
st_alpha75_reshape = reshape(st_alpha75,fs*2,N*fs/20).';

figure
subplot(2,2,1)
plot([0:1/fs:1.99],real(st_alpha2_reshape).','b');
title('Eye diagram with alpha=0.02 for raised cosine');
xlabel('time')
ylabel('amplitude')
axis([0 2 -1.5 1.5])
grid on


subplot(2,2,2)
plot([0:1/fs:1.99],real(st_alpha5_reshape).','b');
title('Eye diagram with alpha=0.5 for raised cosine');
xlabel('time')
ylabel('amplitude')
axis([0 2 -1.5 1.5])
grid on
```

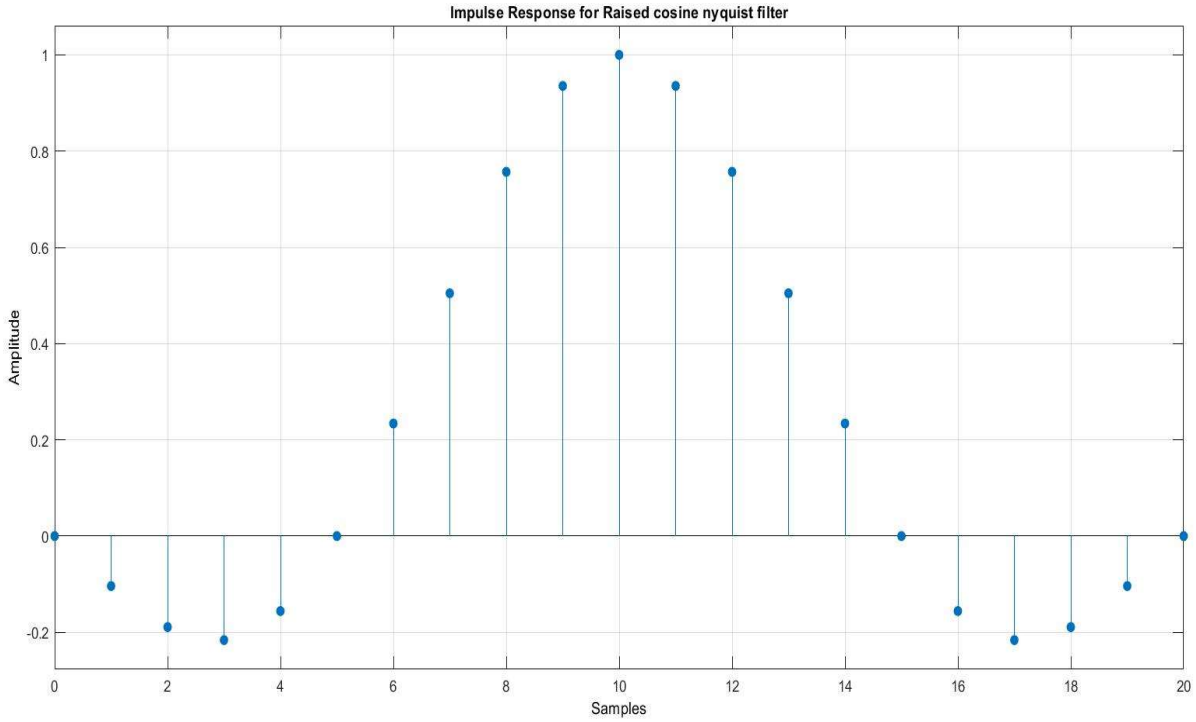
```
subplot(2,2,3)
plot([0:1/fs:1.99],real(st_alpha75_reshape).','b');
title('Eye diagram with alpha=0.75 for raised cosine ')
xlabel('time')
ylabel('amplitude')
axis([0 2 -1.5 1.5 ])
grid on

subplot(2,2,4)
plot([0:1/fs:1.99],real(st_alpha1_reshape).','b');
title('Eye diagram with alpha=1 for raised cosine ')
xlabel('time')
ylabel('amplitude')
axis([0 2 -1.5 1.5 ])
grid on
```

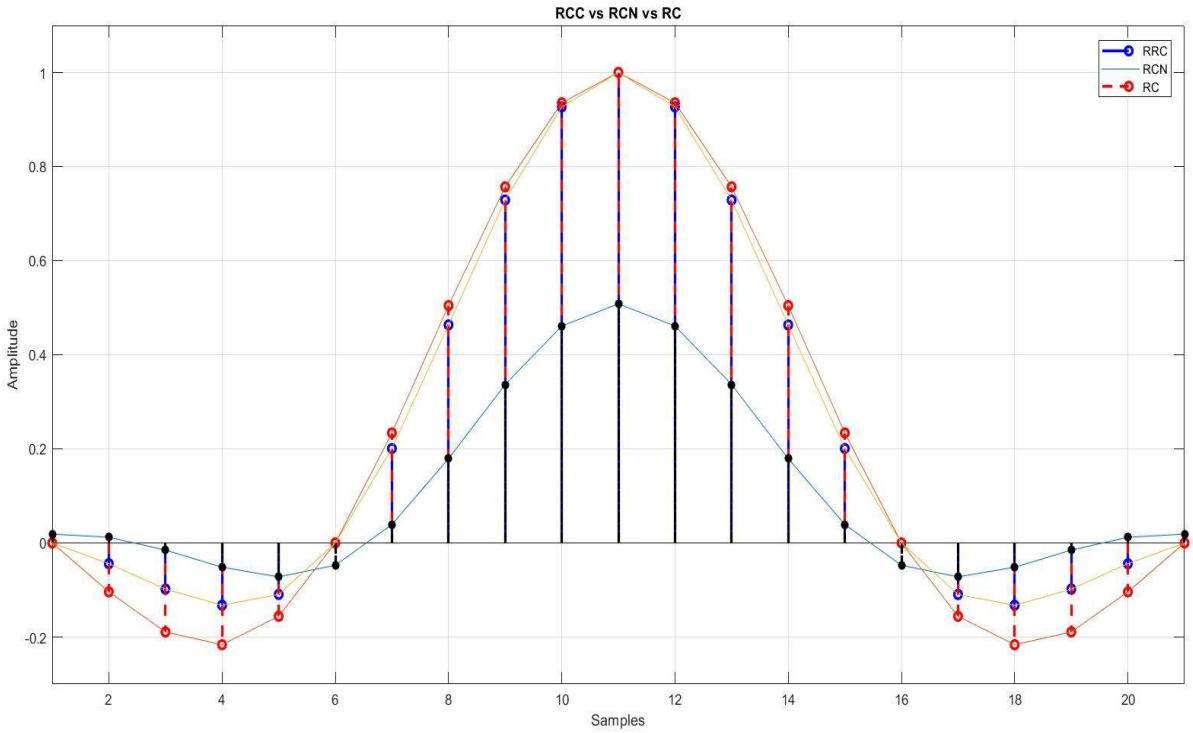
GRAPHS:

Part 1: BPSK modulated baseband system with pulse shaping filter





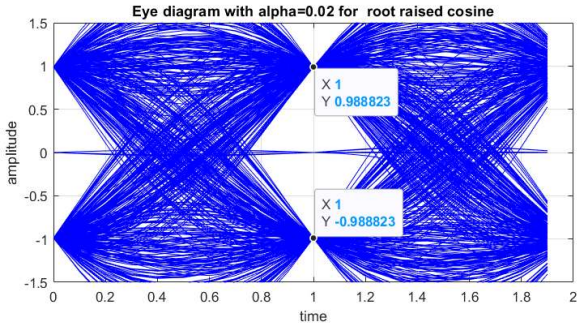

Impulse Response for Raised cosine nyquist filter
Amplitude
Samples
1
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8
10
12
14
16
18
20

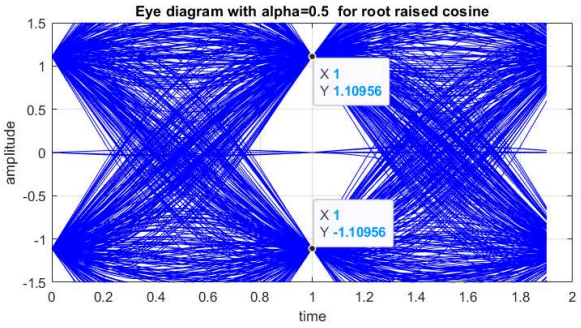

RCC vs RCN vs RC
RRC
RCN
RC
Amplitude
Samples
1
0.8
0.6
0.4
0.2
0
-0.2
2
4
6
8
10
12
14
16
18
20

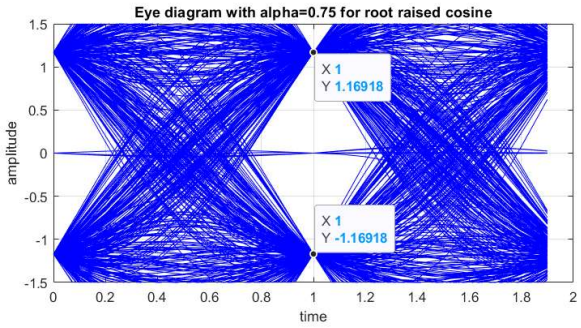
Part 2: Effect of different roll off factors in raised cosine and root raised cosine pulse shaping filter

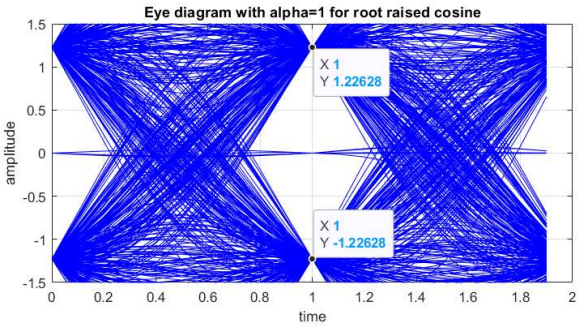
WITHOUT NOISE:

ROOT RAISED COSINE

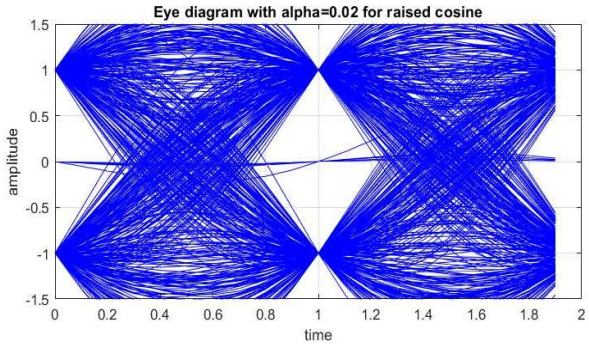


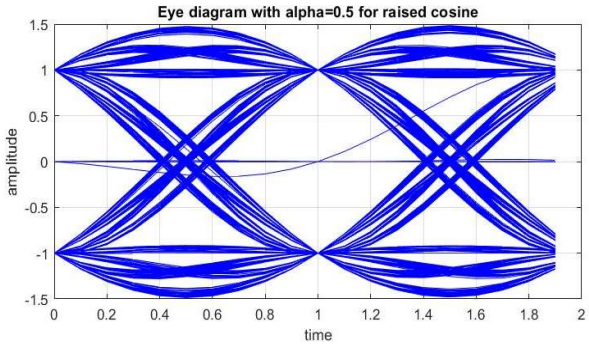


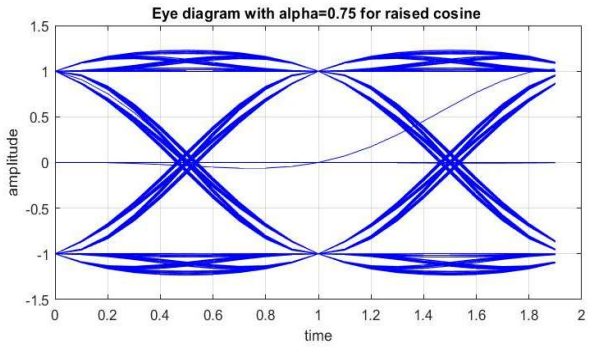


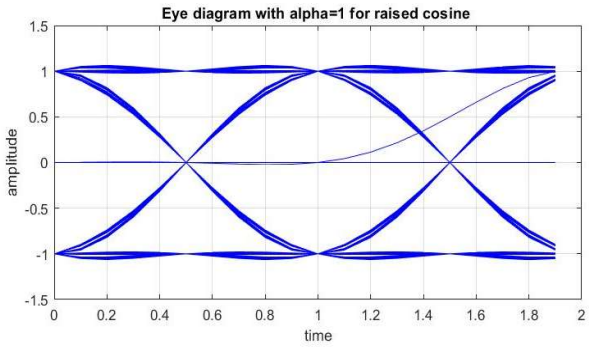


RAISED COSINE





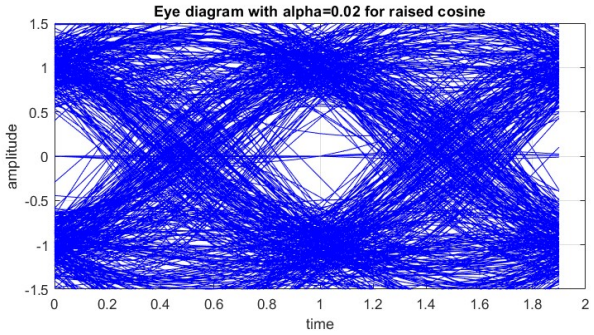


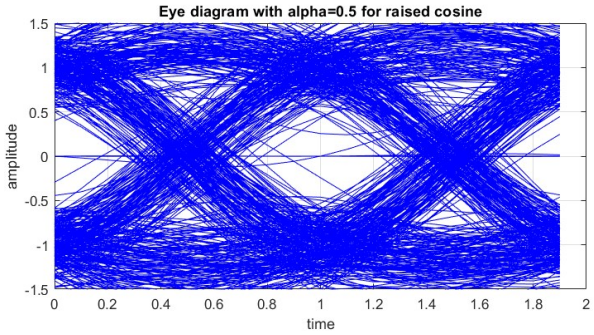


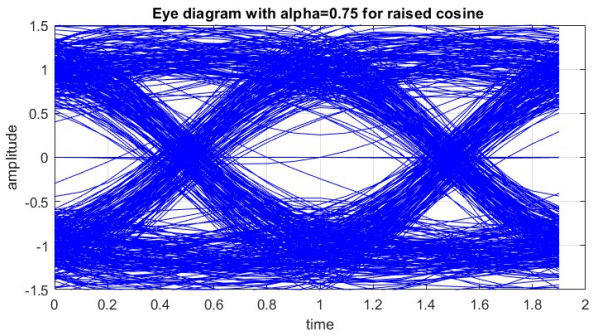
WITH NOISE:

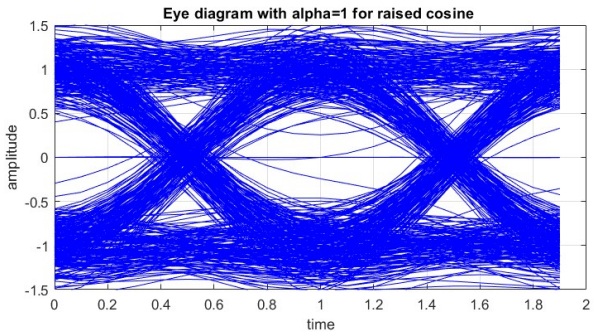
Case 1: am = awgn(am,10,'measured');

RAISED COSINE

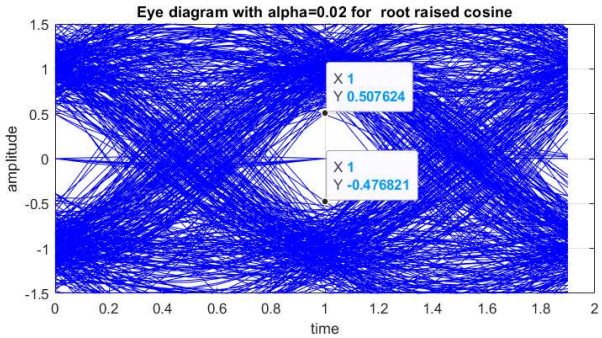


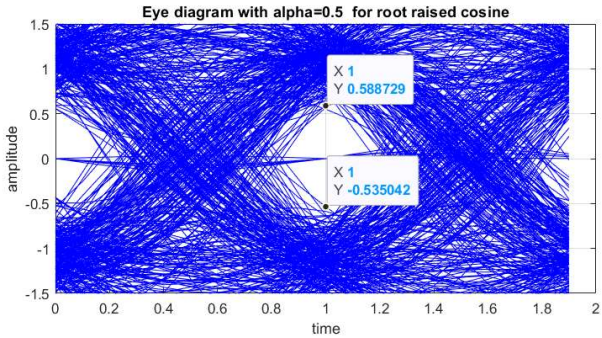


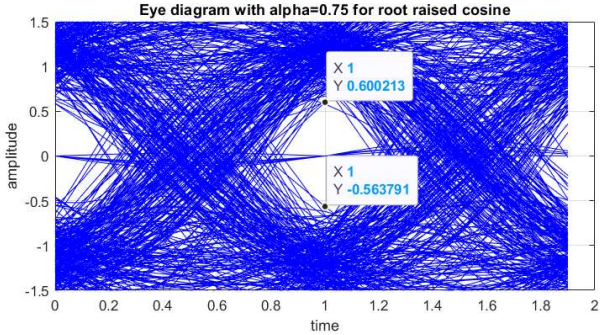


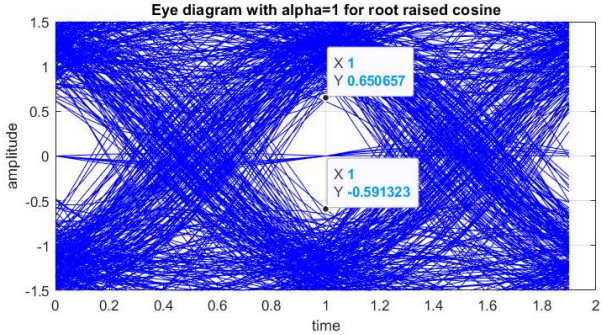


ROOT RAISED COSINE:



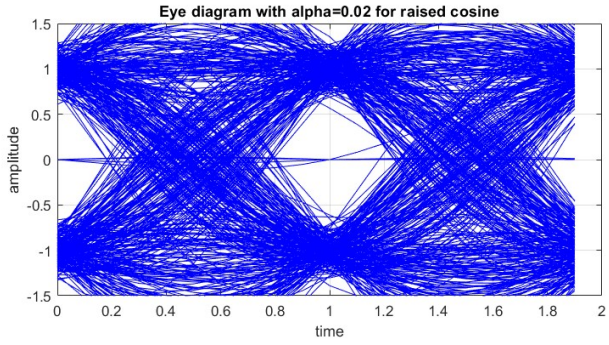


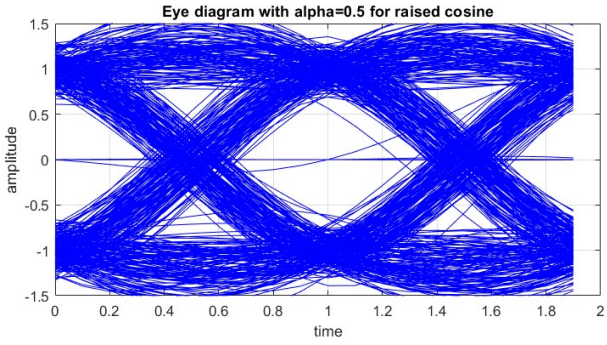


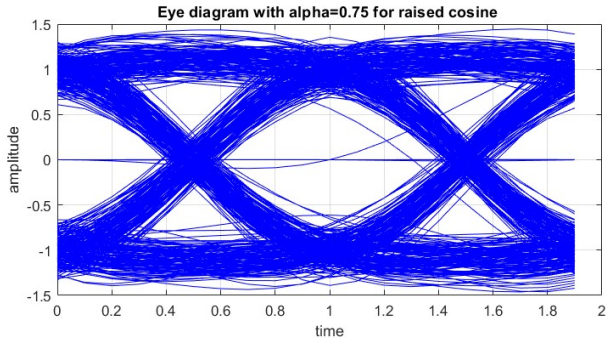


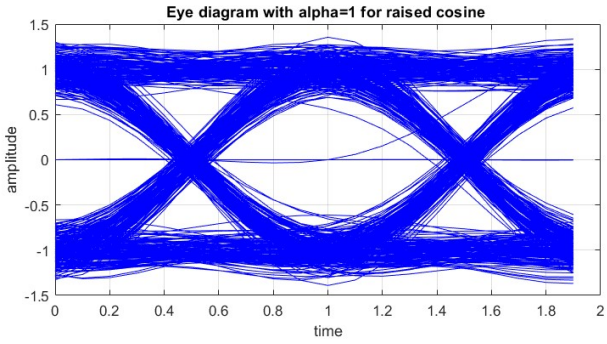
Case 2: am = awgn(am,15,'measured');

RAISED COSINE

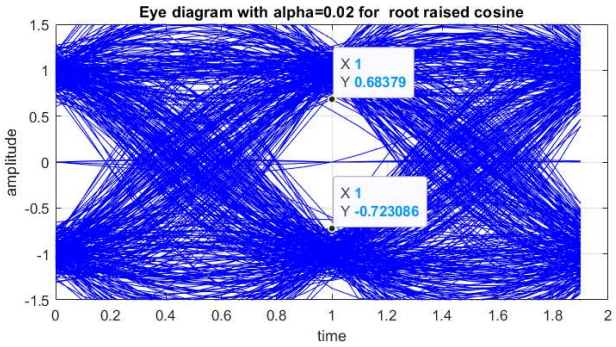


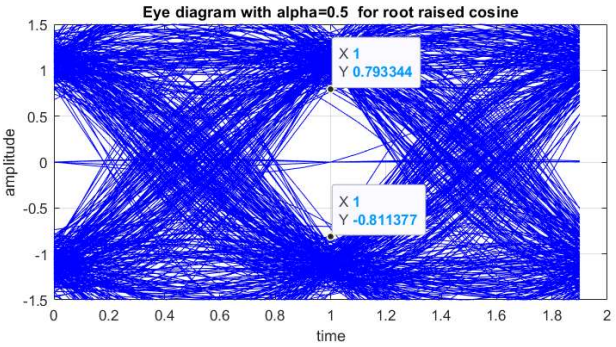


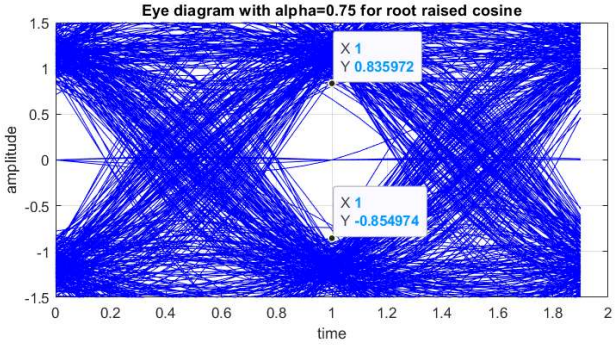


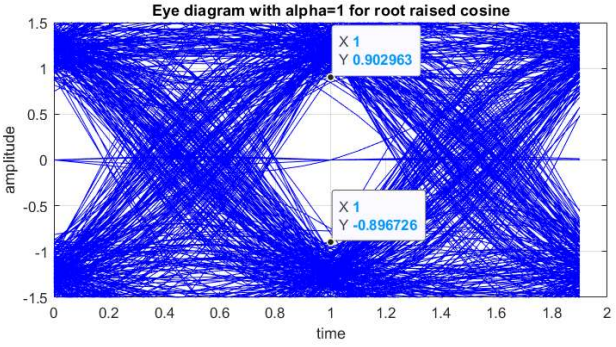


ROOT RAISED COSINE









RESULT: We successfully simulated BPSK modulated signal with pulse shaping filter. We designed raised cosine and root raised cosine signal and observed the difference by varying the roll off factor α.

INFERENCE:

1. The roll-off factor is a measure of the excess bandwidth of the filter, i.e., the bandwidth occupied beyond the Nyquist bandwidth of ½T, where 1/T is symbol rate.
2. As roll-off increases eye in the eye diagram opens up. This means that if there were no bandwidth restrictions it would be easier on the receiver if one used a large roll-off. (However, for bandwidth efficiency roll-off should be smaller.)
3. Smaller roll-off gives narrower bandwidth. However, its side lobes increases so attenuation in stop band is reduced.

REFERENCE:

1. **https://www.michael-joost.de/rrcfilter.pdf**
2. **https://engineering.purdue.edu/~ee538/SquareRootRaisedCosine.pdf**
3. **https://en.wikipedia.org/wiki/Pulse_shaping**