Communication System Lab 4

By: Saurabh Kumar (SC22B146)

Baseband channel modelling

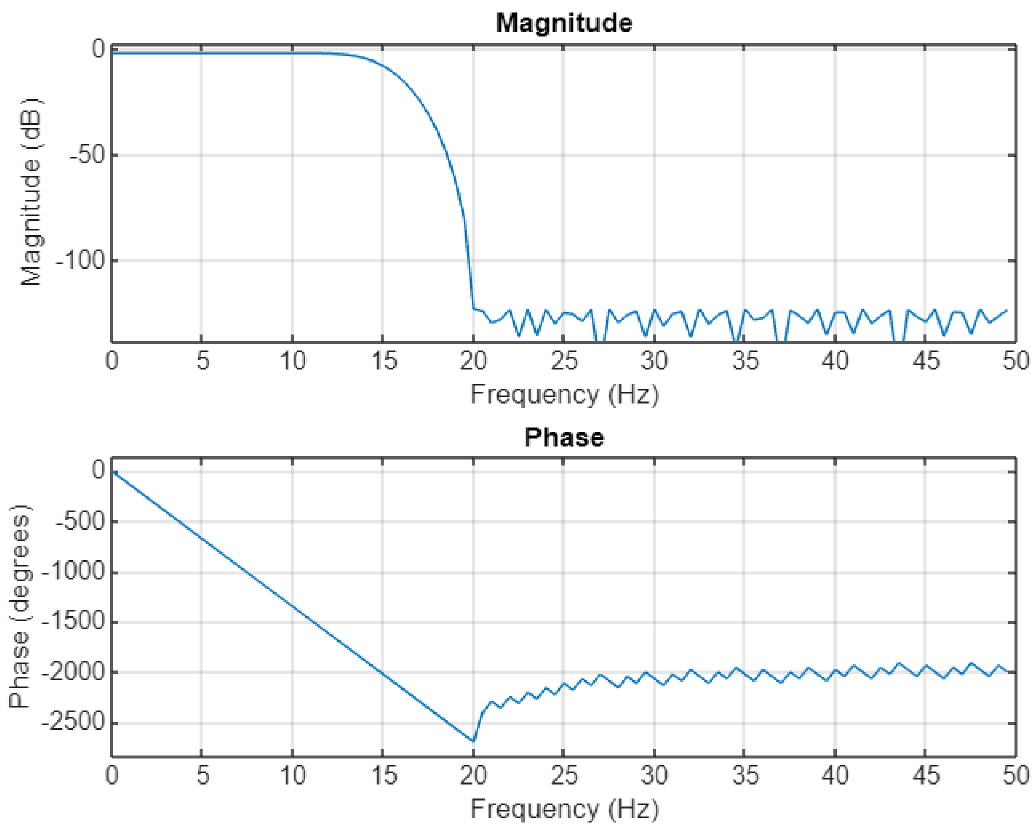
1. A baseband channel can be modelled as an LTI low-pass filter with a cutoff frequency of fc. The channel can be modelled as introducing attenuation and ISI. Your task is to write a Matlab function which can model a baseband channel.

(a) Write a Matlab channel function that takes as input a sampled signal x[n] and produces an output y[n].

(b) The function should low-pass filter the input x[n]; the low pass filter should have a gain g in the passband as input, and a passband cutoff frequency of fc.

2. Plot the frequency and phase response of the channel that you have modelled for your choice of channel parameters. Make sure that the axes are properly labelled.

```
% low-pass FIR filter function
function y = channel(x,fs,fc,g)
    h = firpm(75, [0 fc fc+10, fs/2]/(fs/2), [g g 0 0]);
    y = conv(x, h, 'same');

    % Plot the response
    figure;
    freqz(h, 1, 100, fs);
    grid on;
end

function y = channel_with_no_plot(x,fs,fc,g)
    h = firpm(75, [0 fc fc+10, fs/2]/(fs/2), [g g 0 0]);
    y = conv(x, h, 'same');
end

% channel with an arbitrary input signal
channel_fs = 100; channel_fc = 10; channel_g = 0.8;
channel([0 0 0],channel_fs,channel_fc,channel_g);
```



Inference: The function to take input signal and convolve it with the low pass filter impulse response is made, which also plots the output response. For further use, an additional function which only gives the output response is made.

Intersymbol interference - baseband channels

1. We will study and visualize the effect of intersymbol interference (ISI) using the baseband channel model developed above.

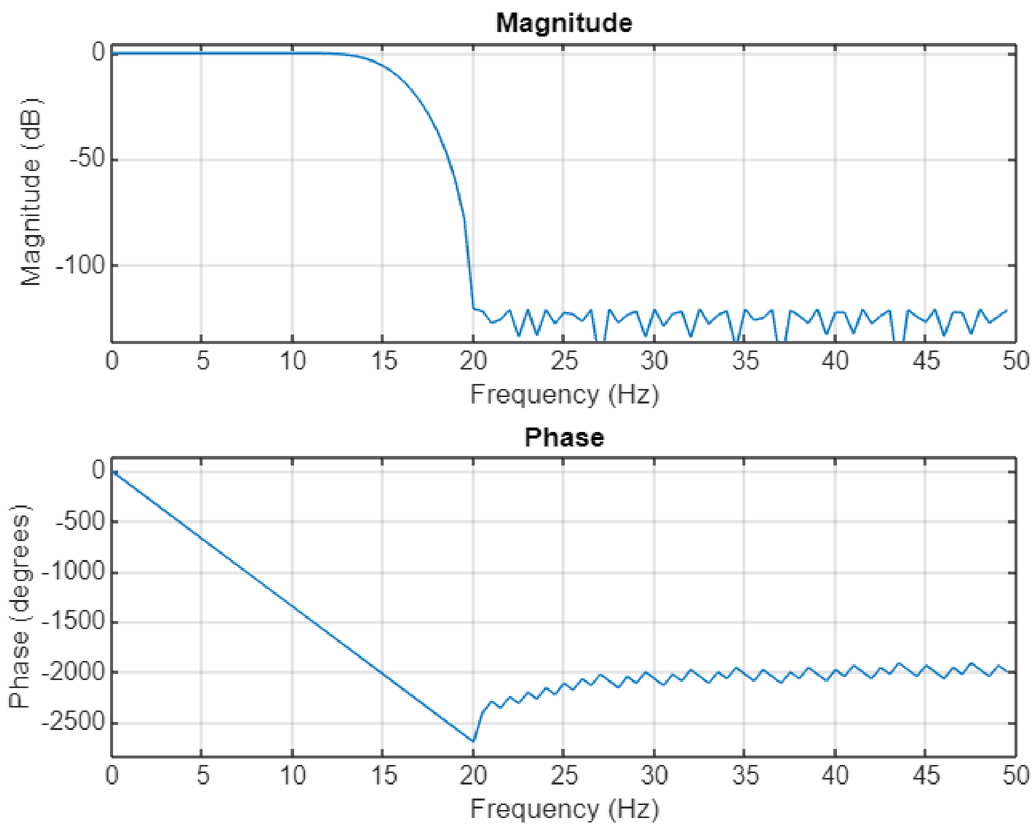
2. Consider the baseband channel modelled as a low pass filter. We will investigate what happens to a baseband waveform as it passes through a low pass or baseband channel.

3. We will use a sampling frequency of 100 Hz for this task.

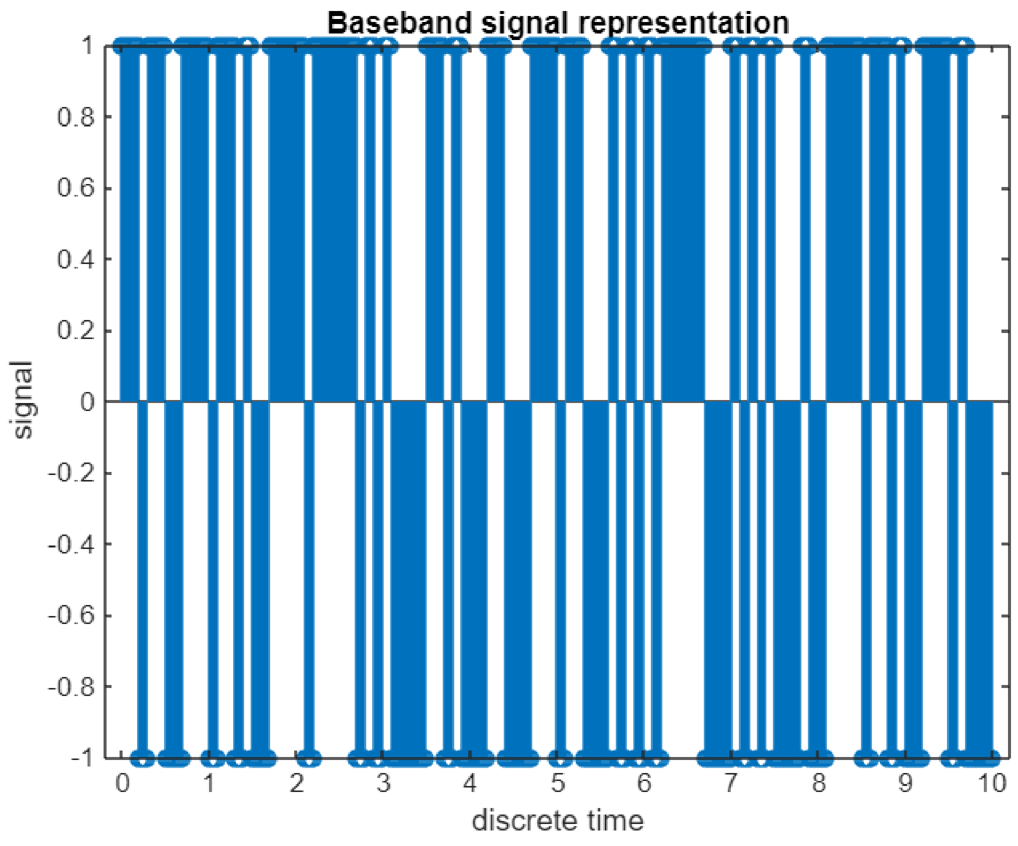
4. Use a low pass filter with a passband edge of 10 Hz with a passband gain of 1 to model the channel. Obtain and plot the magnitude spectrum of the channel.

5. Generate the line code b(t) corresponding to a random sequence of 100 bits for Tb = 0.1s.

```
% channel with an arbitrary input signal
channel_fs = 100; channel_fc = 10; channel_g = 1;
channel([0 0 0],channel_fs,channel_fc,channel_g);
```



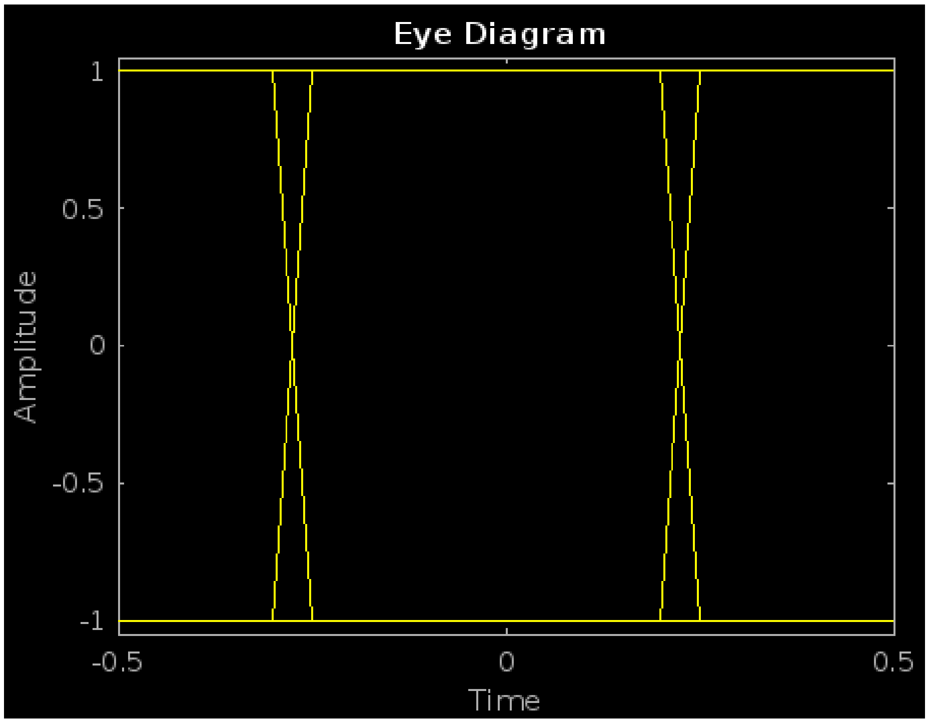
```
rng("default");
N = 100;
source = rand(1, N) > 0.5;
T = 0.1;
fs = 100;
n = T*fs;

% line code
signal = zeros(1,n*N);
for i=1:(length(source)-1)
    signal((i-1)*n+1:(i)*n) = repelem(source(i),n);
end
signal(signal==0) = -1;

figure;
t = linspace(0,N*T,N*fs*T);
stem(t,signal);
title("Baseband signal representation");
xlabel("discrete time");
ylabel("signal");
```



Inference: The channel response is again plotted, and a line code for the random sequence of 100 bits is generated by repeating the sequences.

6. Plot the eye diagram of this signal. You should observe that this is the eye diagram of a signal without ISI.

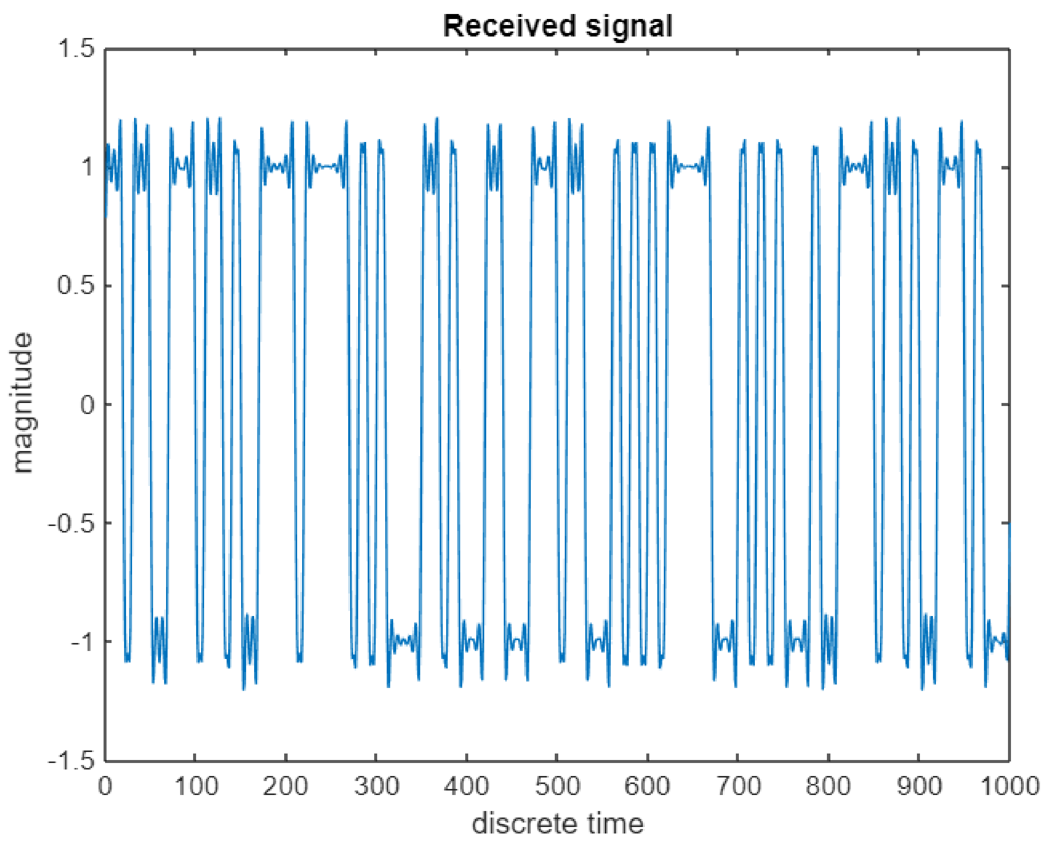
```
eyediagram(signal,2*n,1,n/2);
```



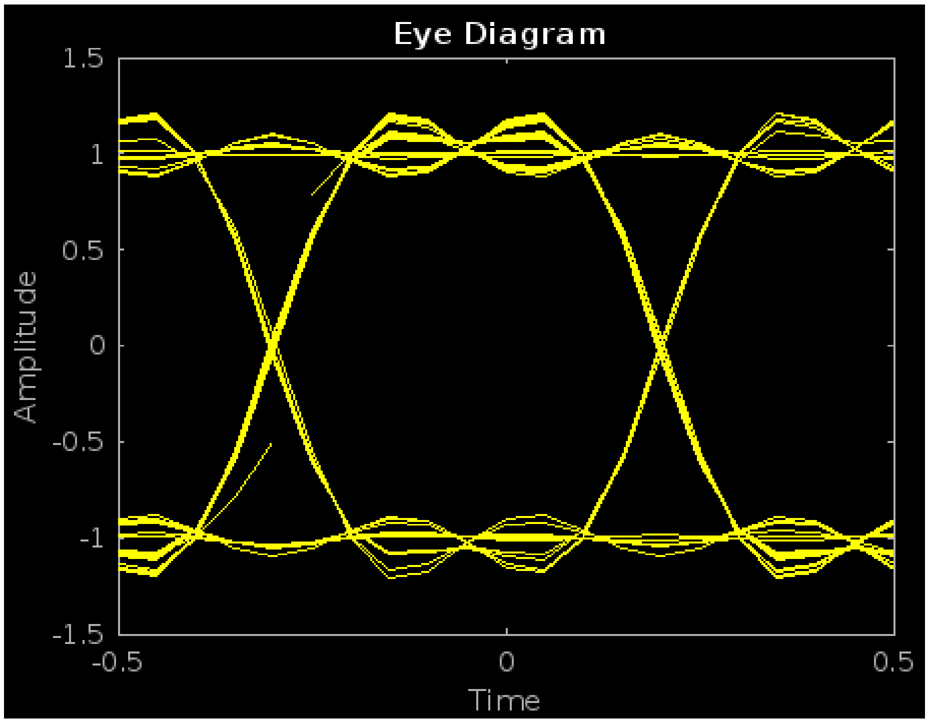
Inference: The eye diagram is plotted using the 'eyediagram' function. For this case, the signal is considered from the half of first T (bit time) (i.e., T + T/2) to the half of third T (i.e., 3T + T/2).

7. Pass the signal through the channel and plot the eye diagram of the channel output. Comment on what you have observed.

```
% received signal
channel_fs = 100; channel_fc = 10; channel_g = 1;
received_signal =
channel_with_no_plot(signal,channel_fs,channel_fc,channel_g);

figure;
plot(received_signal);
title("Received signal");
xlabel("discrete time");
ylabel("magnitude");
```



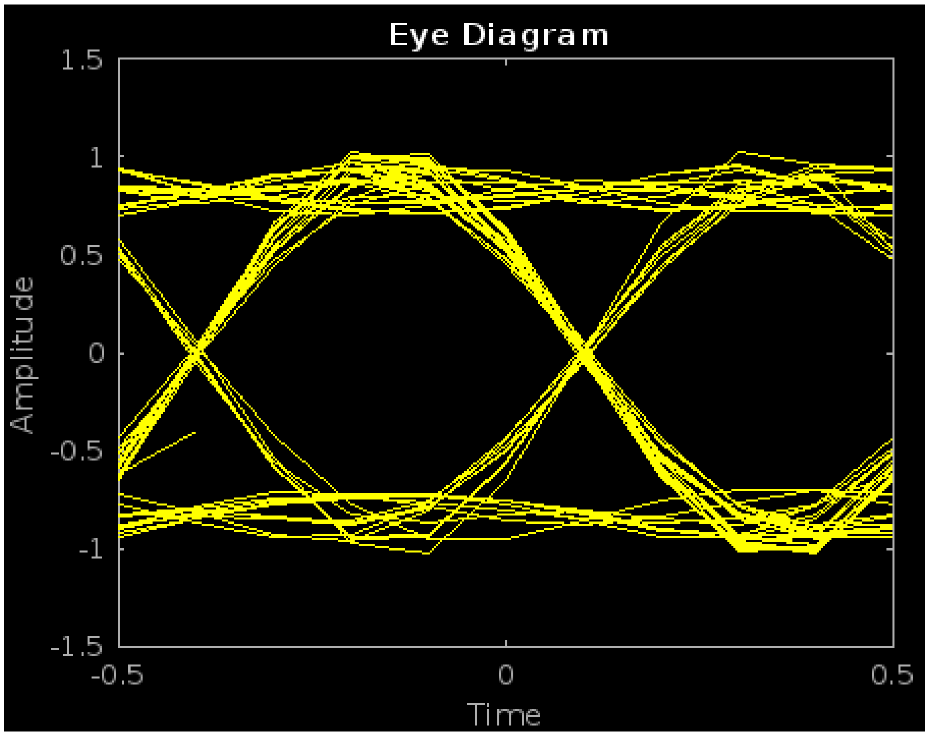
```
eyediagram(received_signal,2*n,1,n/2);
```

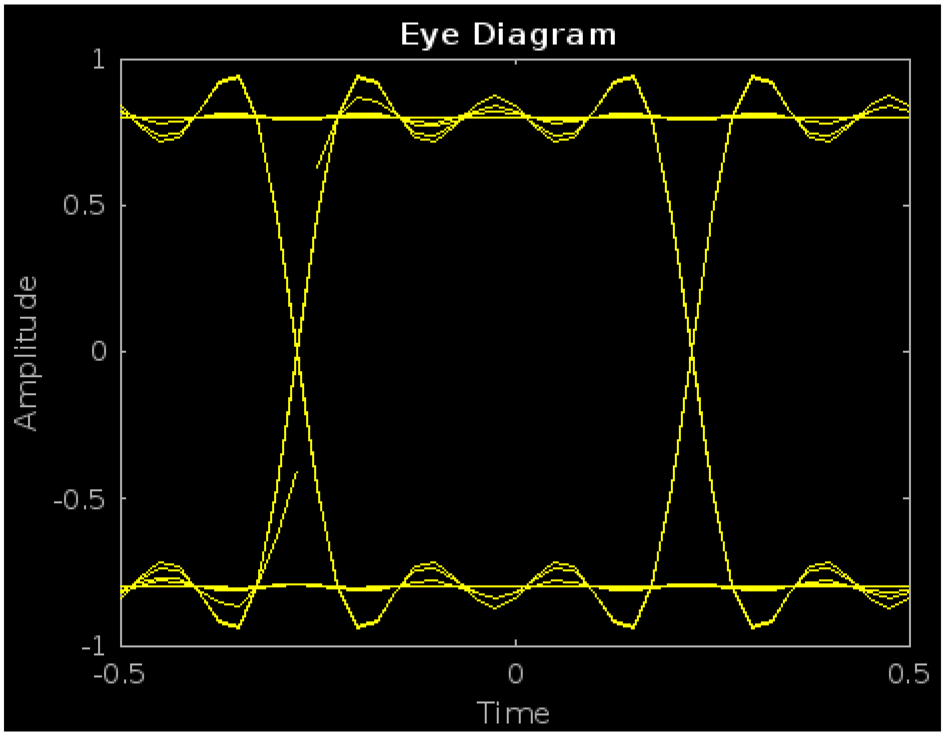


Inference: The linecoded signal is passed through the channel and the corresponding eye diagram is plotted as in the previous case.

8. Plot the eye diagrams for Tb = 0.05 and Tb = 0.2. Comment on your observations.

```
for T = [0.05, 0.2]
    rng("default");
    N = 100;
    source = rand(1, N) > 0.5;
    fs = 100;
    n = T*fs;

    % line code
    signal = zeros(1,n*N);
    for i=1:(length(source)-1)
     if i==1
     signal(1:n) = repelem(source(1),n);
     else
     signal((i-1)*n+1:(i)*n) = repelem(source(i),n);
     end
    end
    signal(signal==0) = -1;

    channel_fs = 100; channel_fc = 10; channel_g = 0.8;
    received_signal =
channel_with_no_plot(signal,channel_fs,channel_fc,channel_g);

    eyediagram(received_signal,2*n,1,round(n/2));
end
```



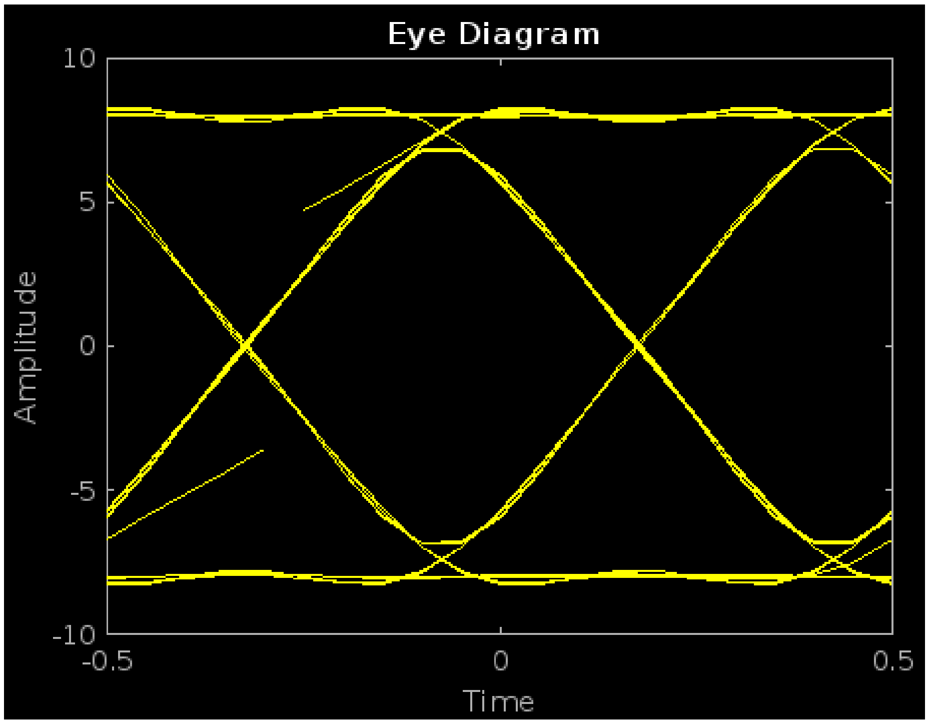


Inference: For Tb = 0.05, the transitions between bits will happen more quickly. This is reflected in the eye openings appearing narrower. However, for Tb = 0.2, longer bit duration results in more spread-out transitions in the eye diagram, and the eye opening appears wider.

9. Suppose the channel output is passed through a matched filter for the case of Tb = 0.1. Plot the eye diagram of the matched-filtered received signal. What differences do you observe?

```
rng("default");
N = 100;
source = rand(1, N) > 0.5;
T = 0.1;
fs = 100;
n = T*fs;

% line code
signal = zeros(1,n*N);
for i=1:(length(source)-1)
    signal((i-1)*n+1:(i)*n) = repelem(source(i),n);
end
signal(signal==0) = -1;

channel_fs = 100; channel_fc = 10; channel_g = 0.8;
received_signal =
channel_with_no_plot(signal,channel_fs,channel_fc,channel_g);

% matched_filter
filter_matched = fliplr(ones(1,n));
received_signal_filtered = conv(received_signal, filter_matched, 'same');
```

```
eyediagram(received_signal_filtered,2*n,1,round(n/2));
```



Inference: After passing the received signal though the matched filter, the signal's eye diagram appear wider indicating a low signal-to-noise ratio. Whereas, for the recieved signal, the eye openeings is narrower, meaning, more error-prone signal.

Effective pulse shapes

1. We will use a sampling frequency of 100 Hz for this task.

2. Generate a random sequence of bits (say 100 bits).e

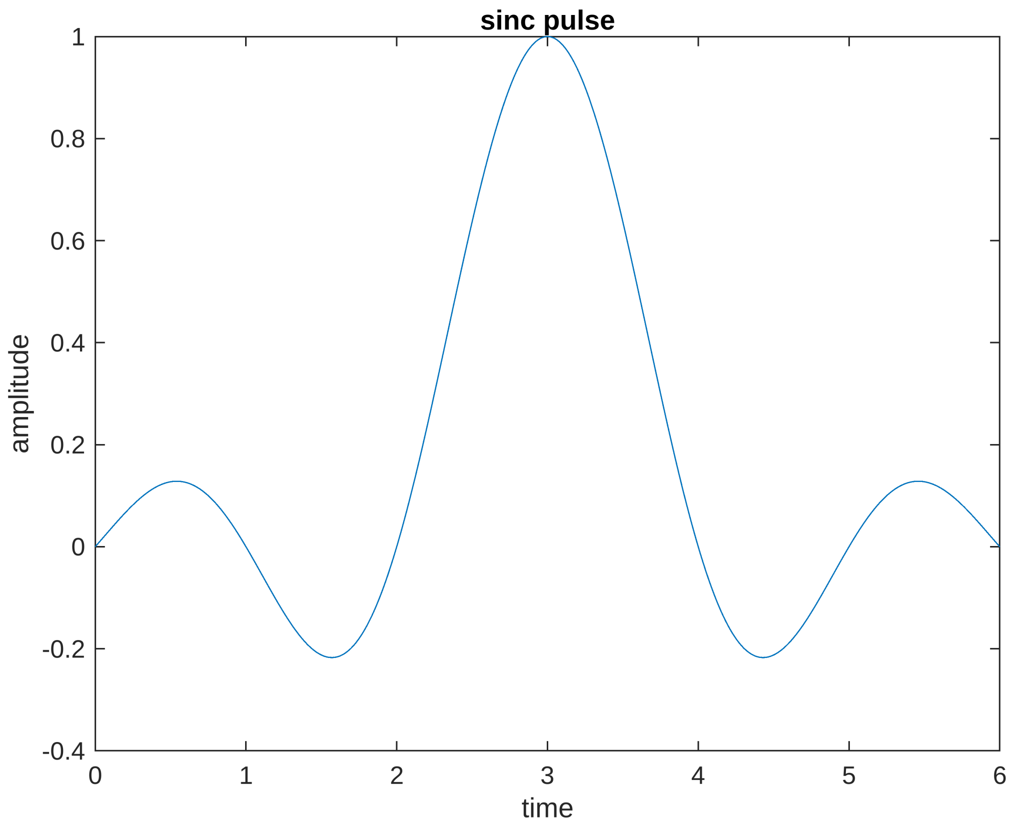
3. Generate the line code $b_s(t)$ corresponding to the above sequence of bits, but using the sinc pulse shape.

$$AT_b \frac{\sin\left(\pi \frac{t}{T_b}\right)}{\pi t}$$

Hint: The sinc pulse shape needs to be truncated and delayed so that the baseband signal can be generated.

```
% parameters
A = 1; Tb = 1; N = 100; fs = 100;
t = linspace(0,6*Tb,6*fs*Tb);
t1 = (t-3*Tb);

% line code
bs = A * Tb * (sin(pi*t1/Tb)./(pi*t1));
figure;
plot(t,bs);
```

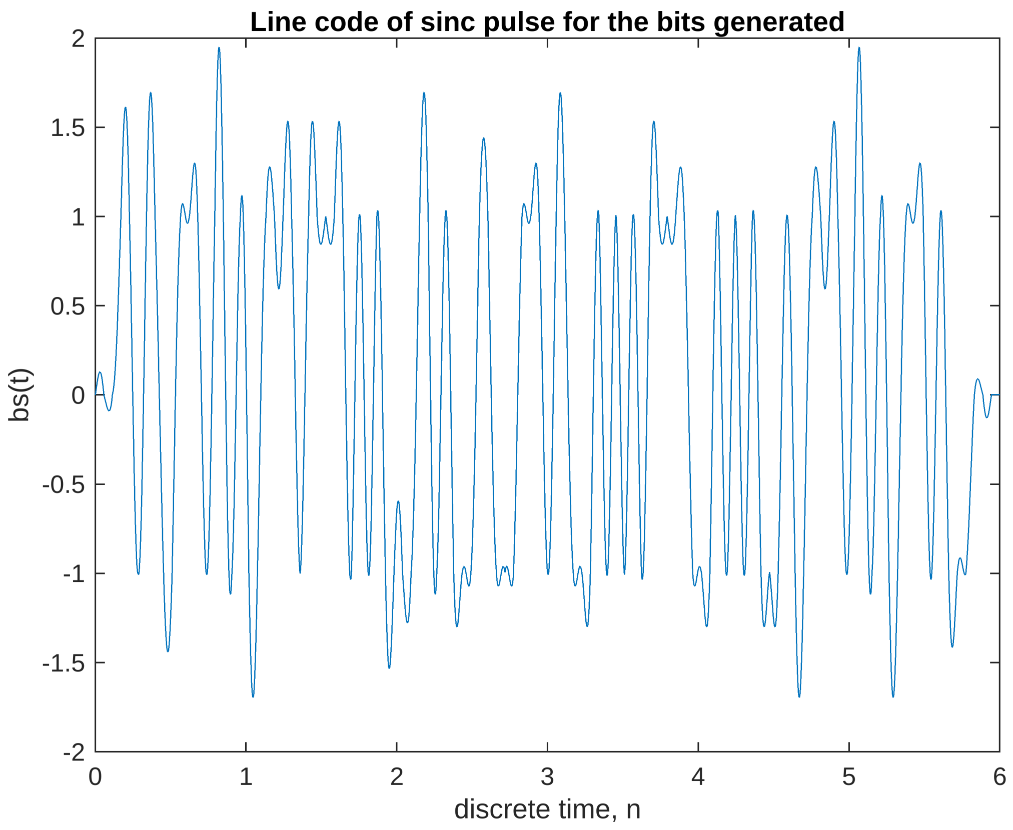
```
title("sinc pulse");
xlabel("time");
ylabel("amplitude");
```



```
rng("default");
source = rand(1, N) > 0.5;
source = double(source);
source(source==0) = -1;
disp(['Source bits: ', num2str(source)]);
```

```
Source bits: 1  1 -1  1  1 -1 -1  1  1  1 -1  1  1 -1  1 -1 -1  1  1  1  1 -1  1  1  1  1  1 -1  1 -1  1 -
```

```
sig_len = (N)*Tb+6*Tb;
signal_bs = zeros(1,sig_len*fs);
for i=1:(length(source))
    signal_bs((i-1)*Tb*fs+1:(i+6-1)*Tb*fs) = signal_bs((i-1)*Tb*fs+1:
(i+6-1)*Tb*fs) + source(i)*bs;
end

t = linspace(0,6*Tb,length(signal_bs));
figure;
plot(t,signal_bs);
title("Line code of sinc pulse for the bits generated");
xlabel("discrete time, n");
ylabel("bs(t)");
```



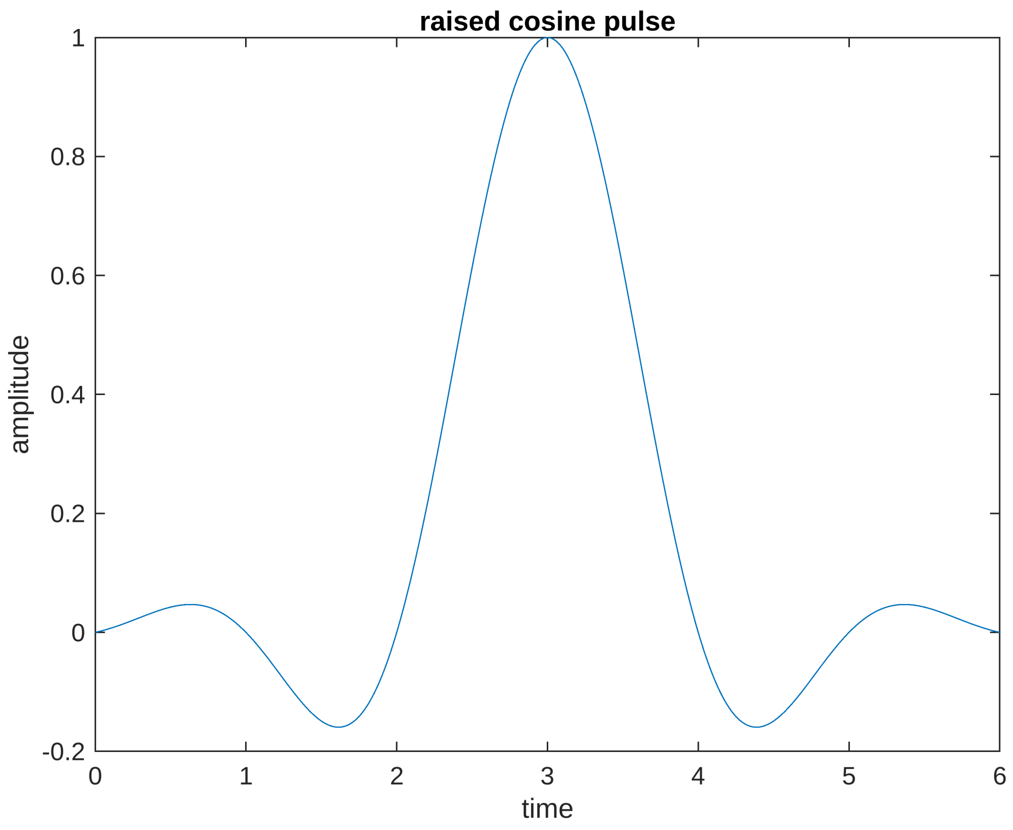
4. Generate the line code $b_c(t)$ corresponding to the same sequence of bits used above, but using the raised cosine pulse shape.

$$AT_b\frac{\sin\left(\pi\frac{t}{T_b}\right)}{\pi t}\frac{\cos\left(\pi\alpha\frac{t}{T_b}\right)}{\left(1-\left(2\alpha\frac{t}{T_b}\right)^2\right)}$$

Hint: Again, the raised cosine pulse shape needs to be truncated and delayed.

```
A = 1;
alpha = 0.4;
Tb = 1;
N = 100;
fs = 100;
t = linspace(0,6*Tb,6*fs*Tb);
t1 = (t-3*Tb);
% n = 10*Tb*fs;

% line code
bc = A * Tb * ( sin(pi*t1/Tb)./(pi*t1) ).*( cos(pi*alpha*t1/Tb)./( 1 -
(2*alpha*t1/Tb).^2) );
figure;
```

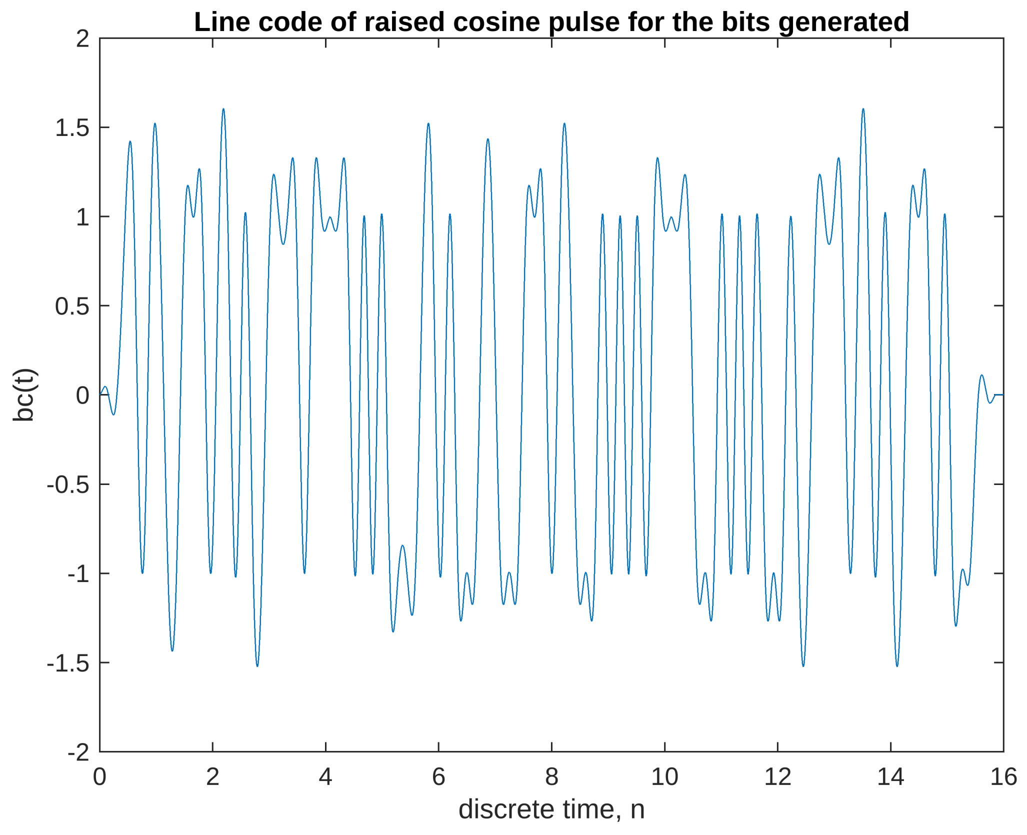
```
plot(t,bc);
title("raised cosine pulse");
xlabel("time");
ylabel("amplitude");
```



```
rng("default");
source = rand(1, N) > 0.5;
source = double(source);
source(source==0) = -1;
disp(['Source bits: ', num2str(source)]);
```

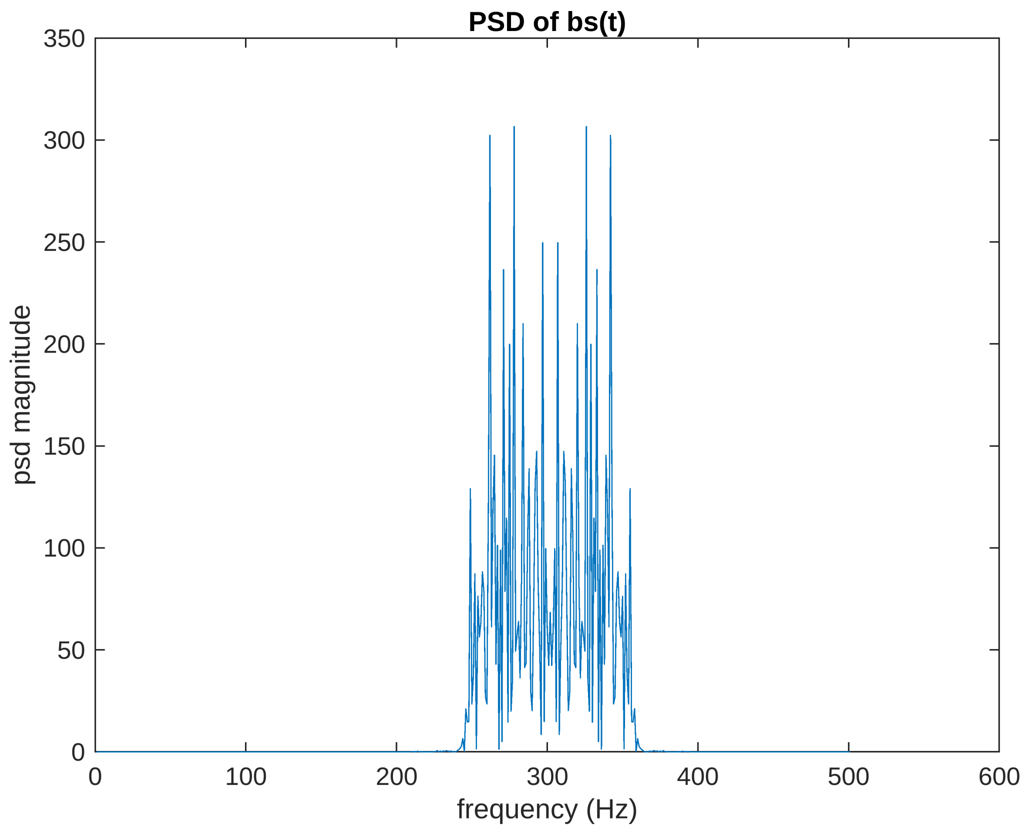
```
Source bits: 1   1 -1   1   1 -1  -1   1   1   1 -1   1   1 -1   1 -1  -1   1   1   1   1 -1   1   1   1   1   1 -1   1 -1   1 -
```

```
sig_len = (N)*Tb+6*Tb;
signal_bc = zeros(1,sig_len*fs);
for i=1:(length(source))
    signal_bc((i-1)*Tb*fs+1:(i+6-1)*Tb*fs) = signal_bc((i-1)*Tb*fs+1:
(i+6-1)*Tb*fs) + source(i)*bc;
end

t = linspace(0,16*Tb,length(signal_bc));
figure;
plot(t,signal_bc);
title("Line code of raised cosine pulse for the bits generated");
xlabel("discrete time, n");
ylabel("bc(t)");
```

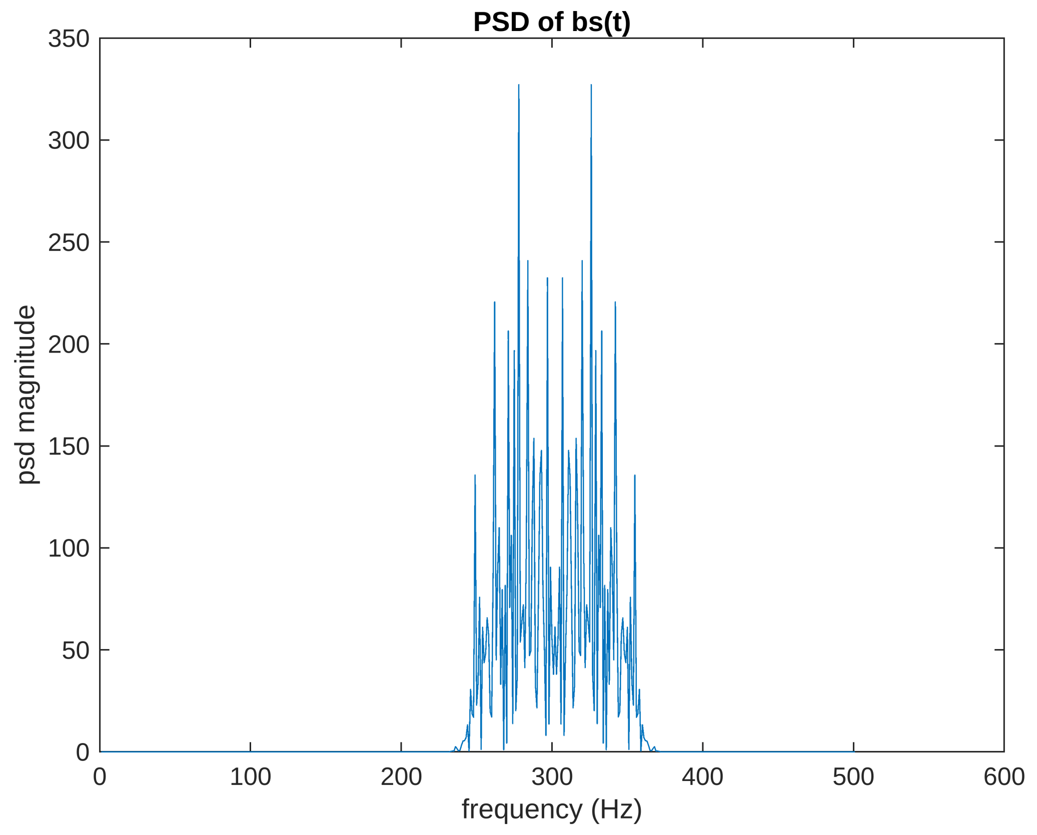


5. Plot the power spectral densities of the signals bs(t) and bc(t). Comment on the differences in the spectra.

```
psd_bs = abs(fftshift((fft(signal_bs))).^2/length(signal_bs));
figure;
plot(psd_bs(5000:5500));
title("PSD of bs(t)");
xlabel("frequency (Hz)");
ylabel("psd magnitude");
```

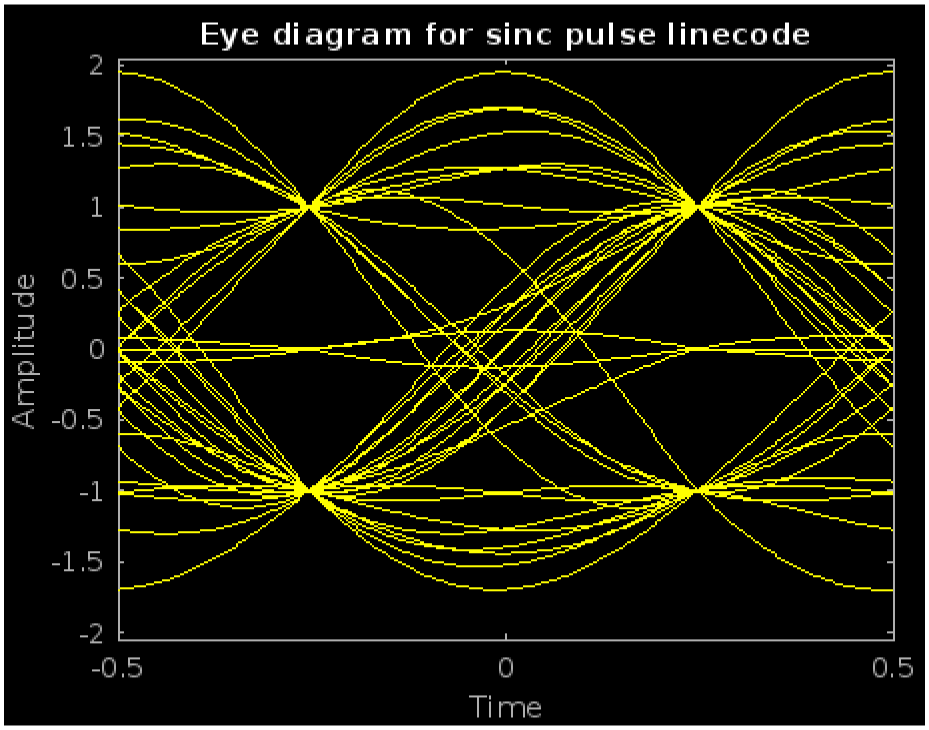


```
psd_bc = abs(fftshift((fft(signal_bc))).^2/length(signal_bc));
figure;
f = -(length(fs)/2):length(fs)/2-1;
plot(psd_bc(5000:5500));
title("PSD of bs(t)");
xlabel("frequency (Hz)");
ylabel("psd magnitude");
```

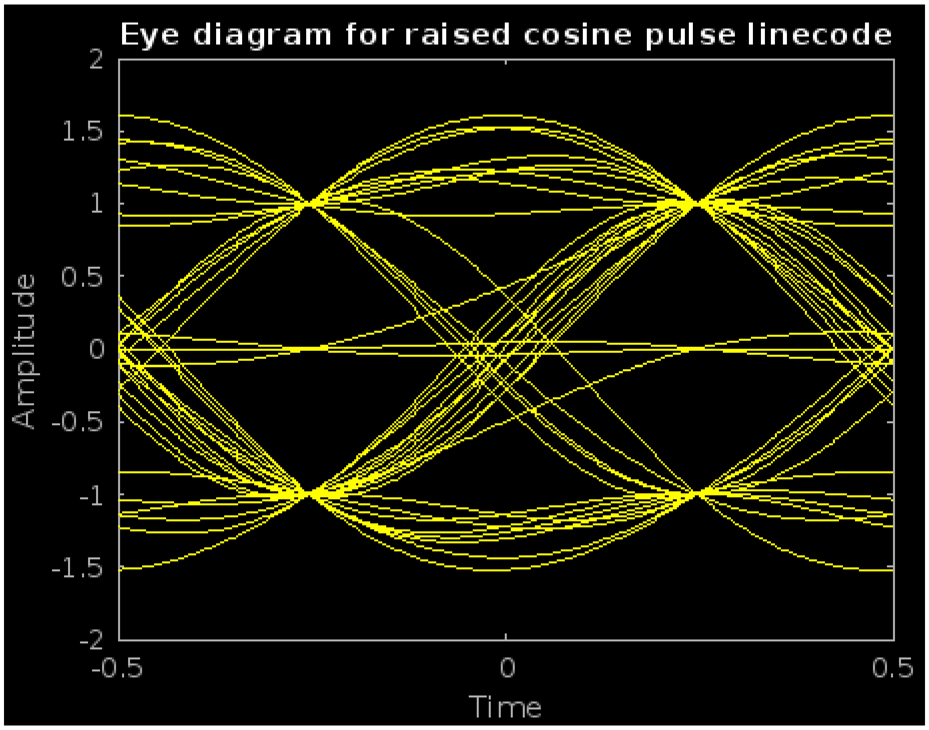


6. Plot the eye diagrams of bs(t) and bc(t). Comment on the differences and main features of each eye diagram. From the eye diagrams what can you conclude about ISI at the ideal sampling instants.

```
eyediagram(signal_bs, 2*Tb*fs, 1, round(Tb*fs/2));
title("Eye diagram for sinc pulse linecode");
```



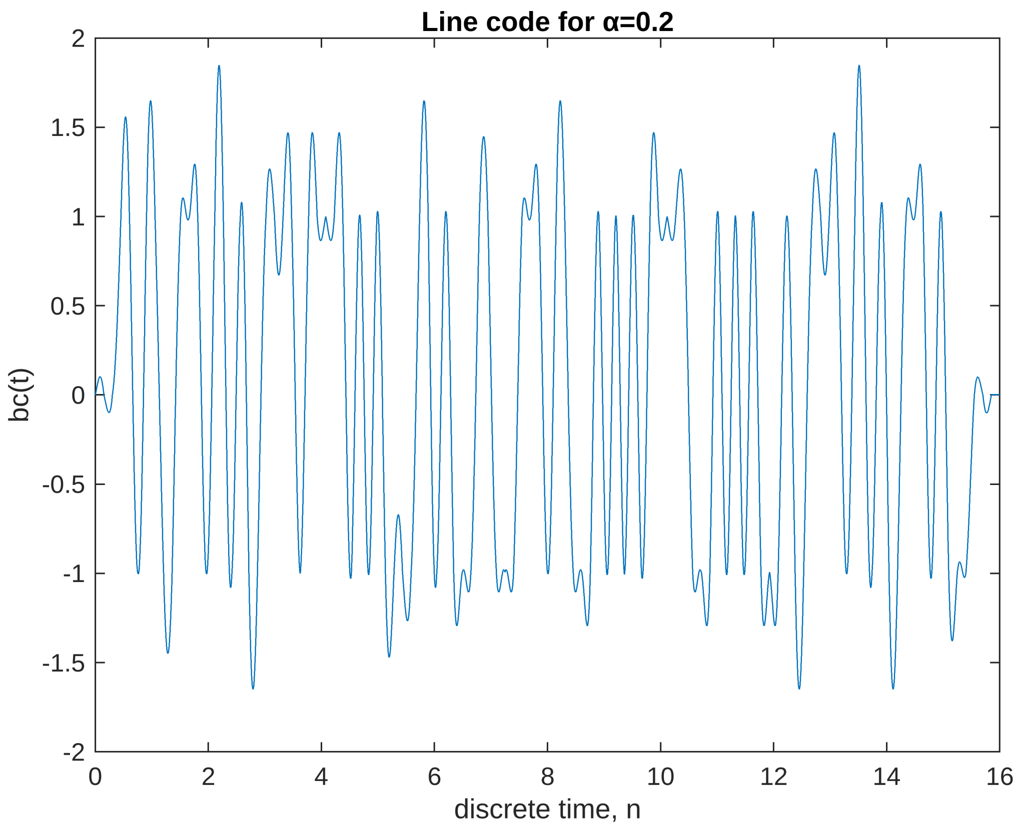
```
eyediagram(signal_bc, 2*Tb*fs, 1, round(Tb*fs/2));
title("Eye diagram for raised cosine pulse linecode");
```

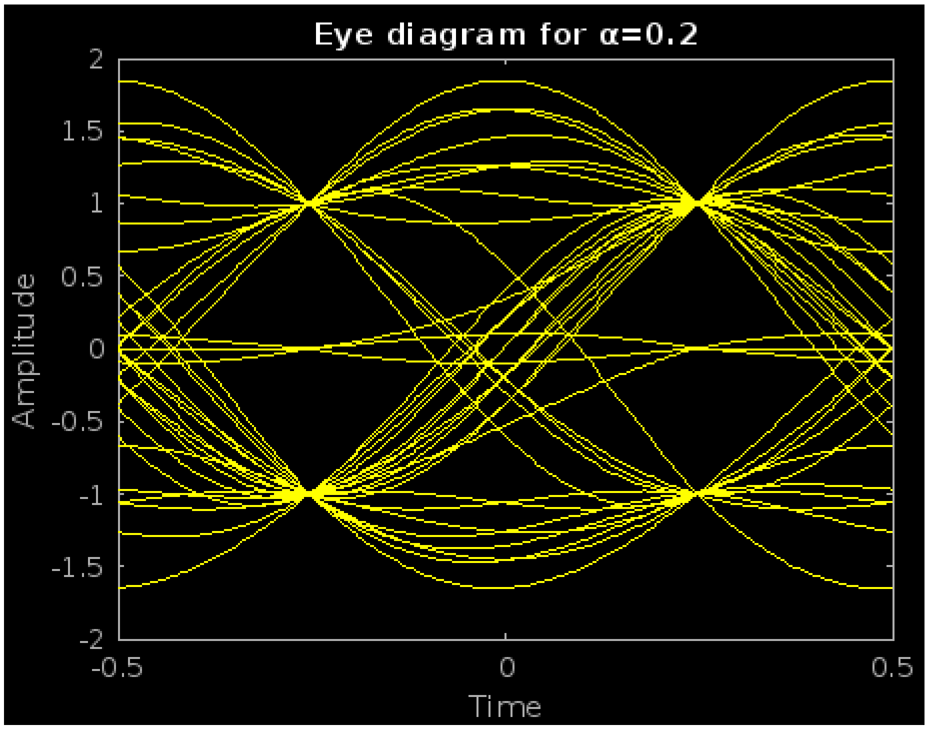


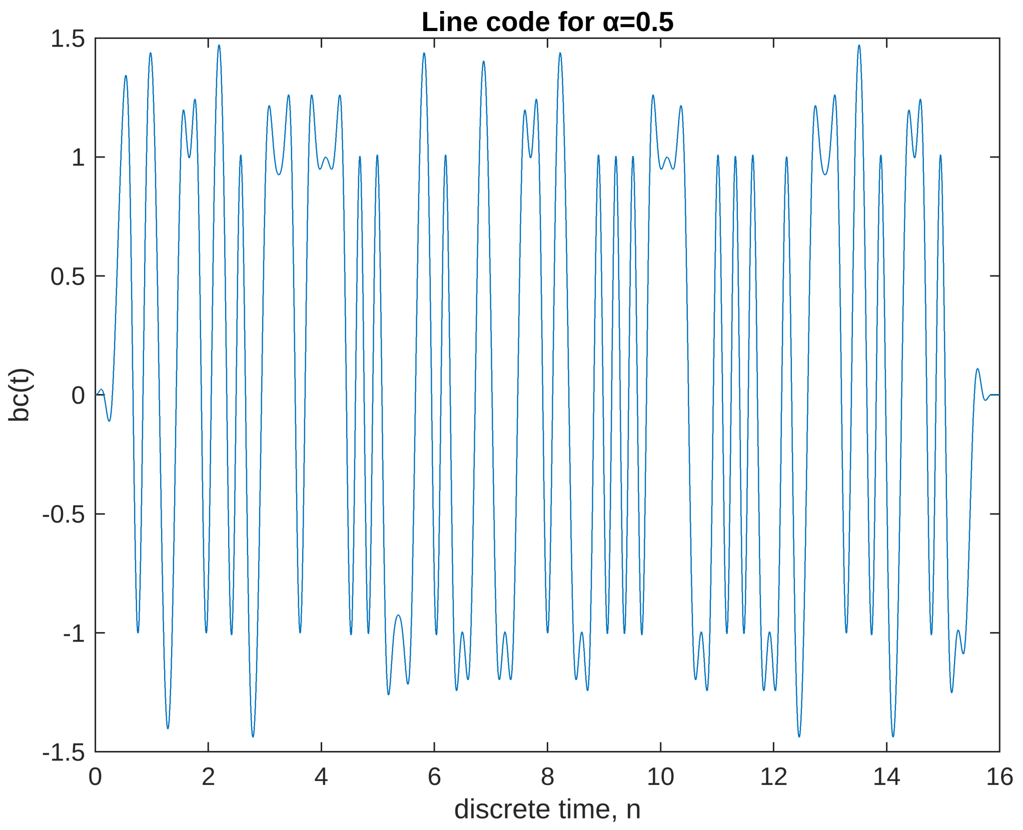
Inference: For raised cosine pulse linecode, the eye opening is sharper and perfectly defined at ideal sampling points, compared to the sinc pulse line code. That is, raised cosine pulse linecode has minimal ISI at ideal sampling instants,.

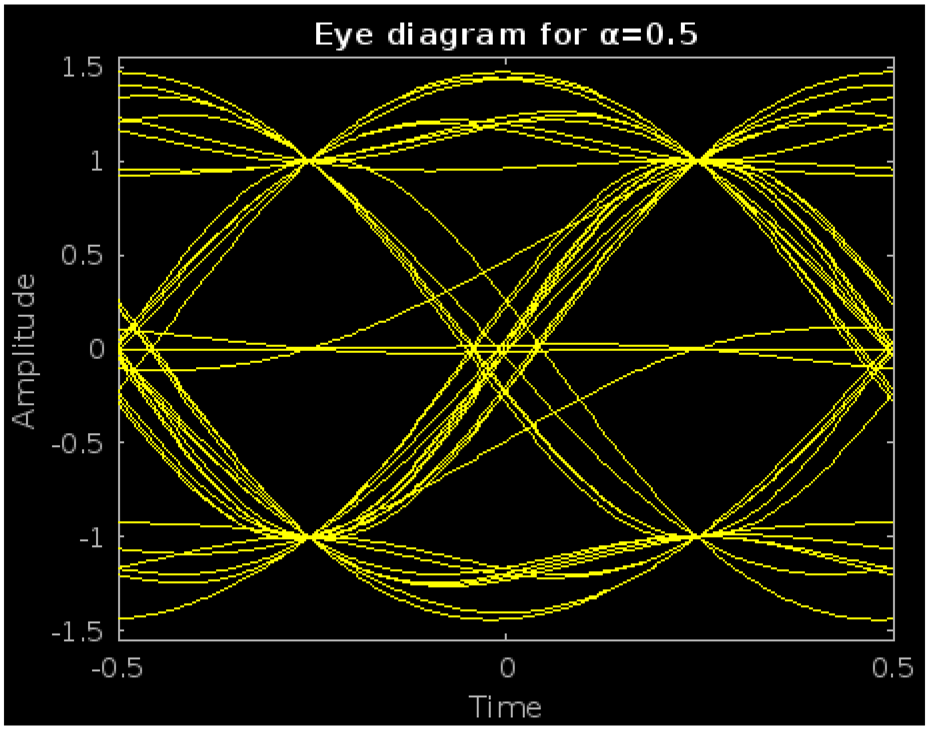
7. Plot the spectra and eye diagrams of bc(t) for different values of α. What do you observe?

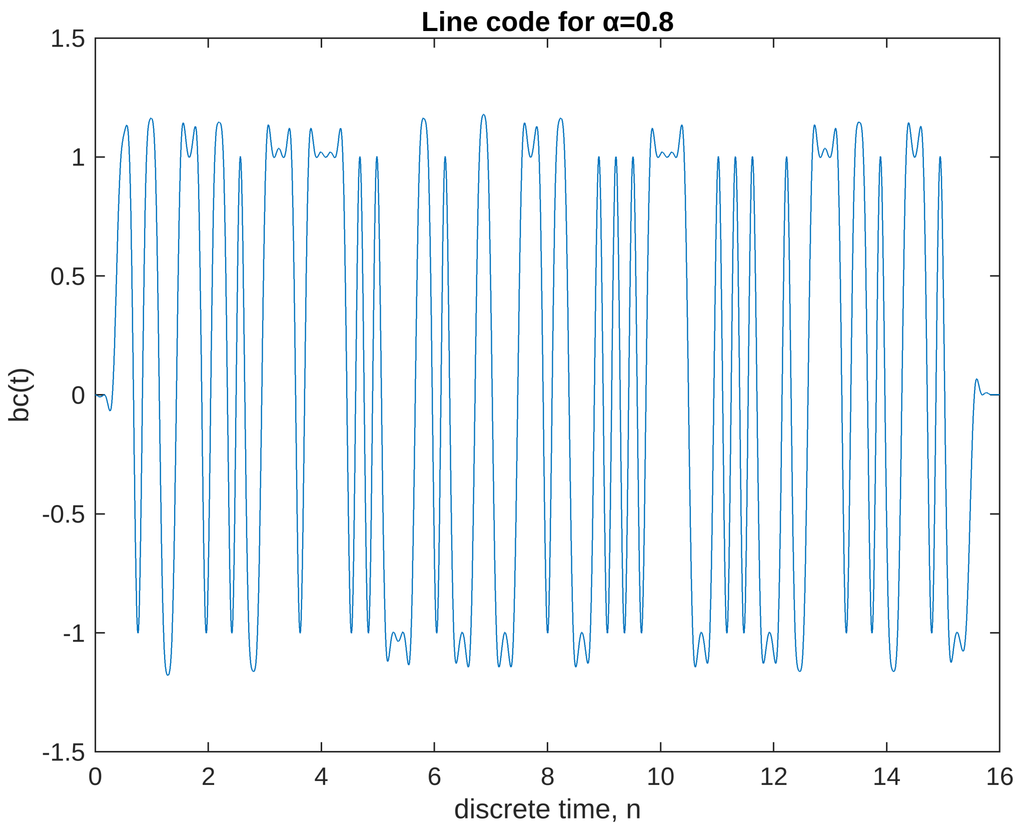
```
% parameters
A = 1; Tb = 1; N = 100; fs = 100;
t = linspace(0,6*Tb,6*fs*Tb);
t1 = (t-3*Tb);

% source bits
rng("default");
source = rand(1, N) > 0.5;
source = double(source);
source(source==0) = -1;

% figure;
k = 1; % iterator
for alpha = [0.2 0.5 0.8 1 2 5]
    % line code
    bc = A * Tb * ( sin(pi*t1/Tb)./(pi*t1) ).*( cos(pi*alpha*t1/Tb)./( 1 -
(2*alpha*t1/Tb).^2) );
    sig_len = (N)*Tb+6*Tb;
    signal_bc = zeros(1,sig_len*fs);
    for i=1:(length(source))
        signal_bc((i-1)*Tb*fs+1:(i+6-1)*Tb*fs) = signal_bc((i-1)*Tb*fs+1:
(i+6-1)*Tb*fs) + source(i)*bc;
    end

    % spectra
    figure;
    t = linspace(0,16*Tb,length(signal_bc));
    plot(t,signal_bc);
    title(['Line code for α=', num2str(alpha)]);
    xlabel("discrete time, n");
    ylabel("bc(t)");

    figure;
    % eyediagram
    eyediagram(signal_bc,2*Tb*fs,1,Tb*fs/2);
    title(['Eye diagram for α=', num2str(alpha)]);
    k = k + 1;
end
```

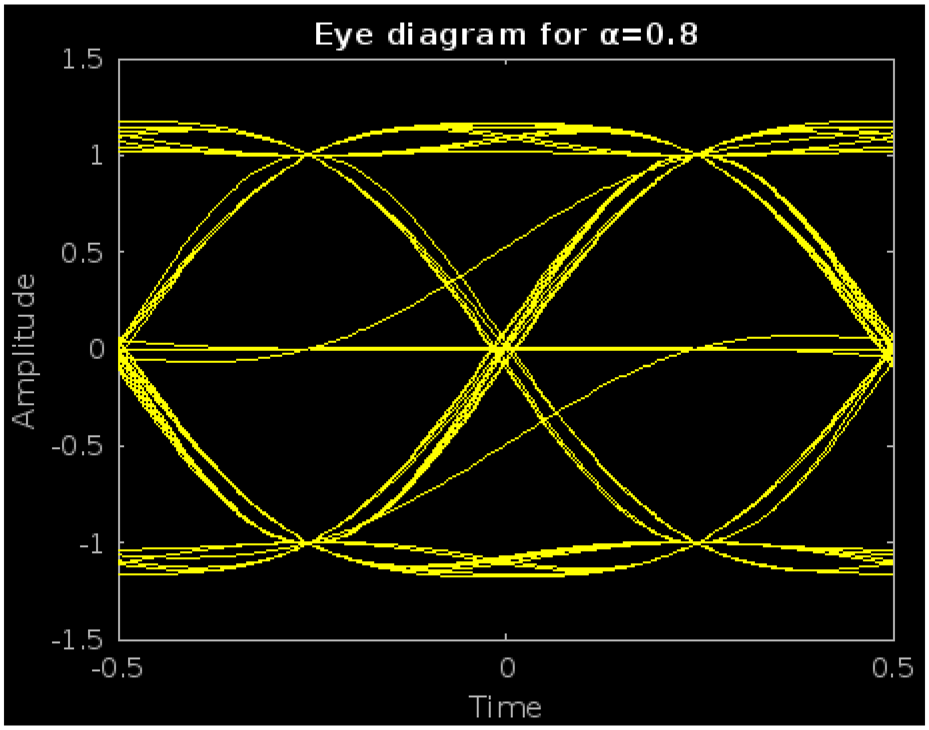

Line code for α=0.2
bc(t)
discrete time, n
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
2
4
6
8
10
12
14
16

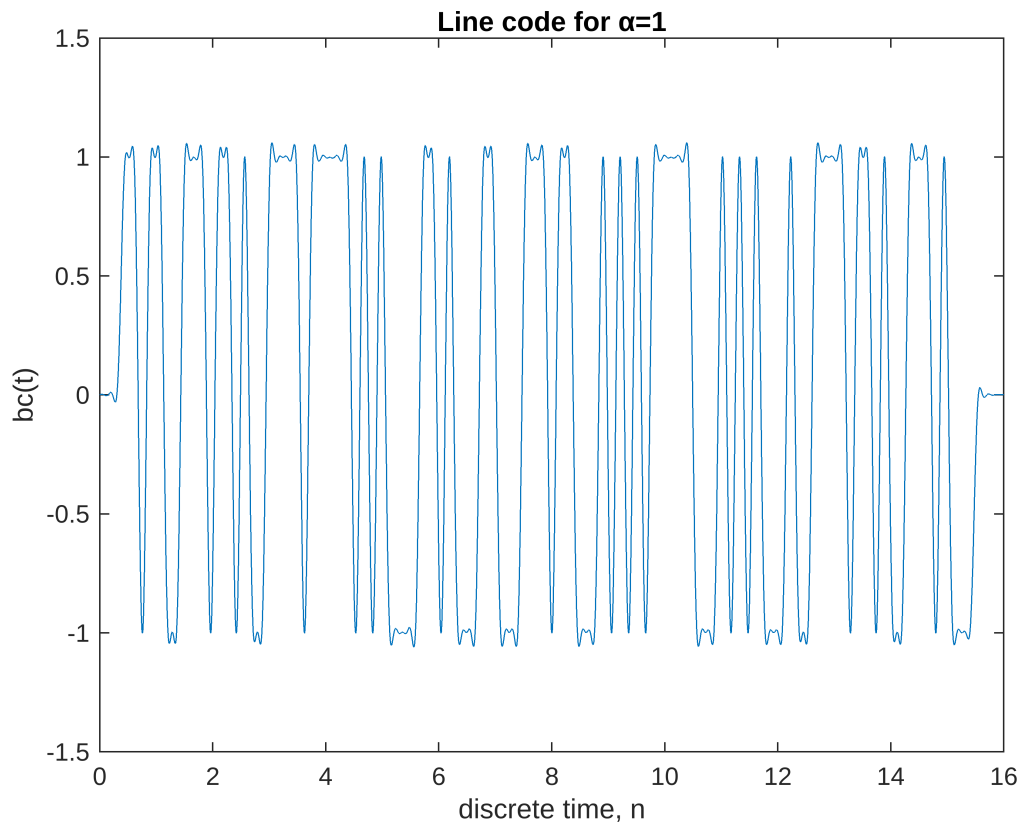

Eye diagram for α=0.2
Amplitude
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-0.5
0
0.5

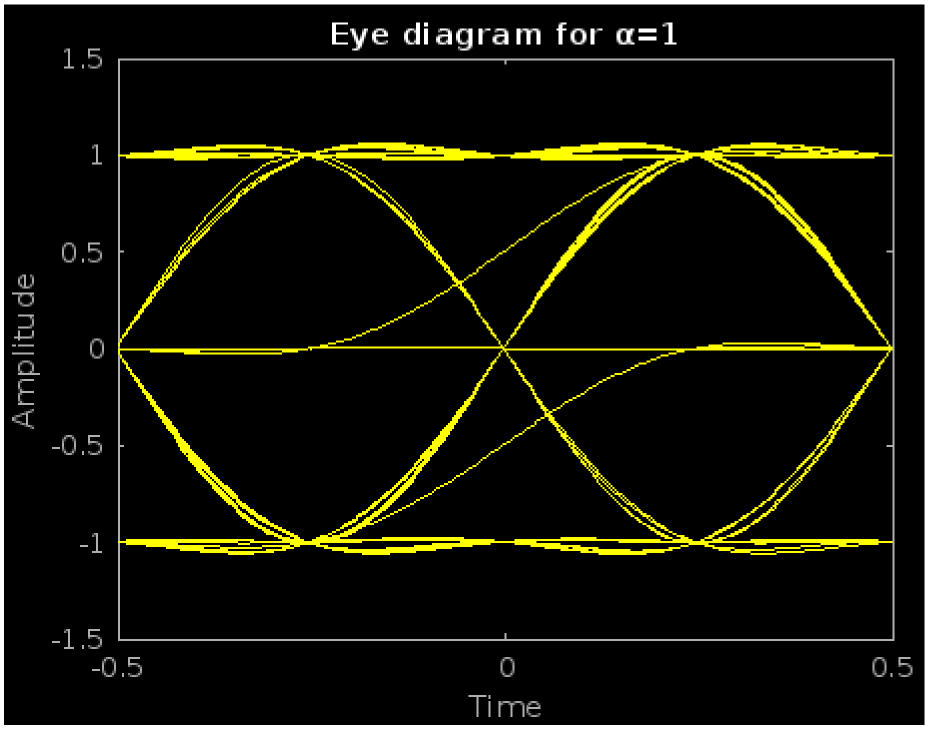

Line code for α=0.5
bc(t)
discrete time, n
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
14
16

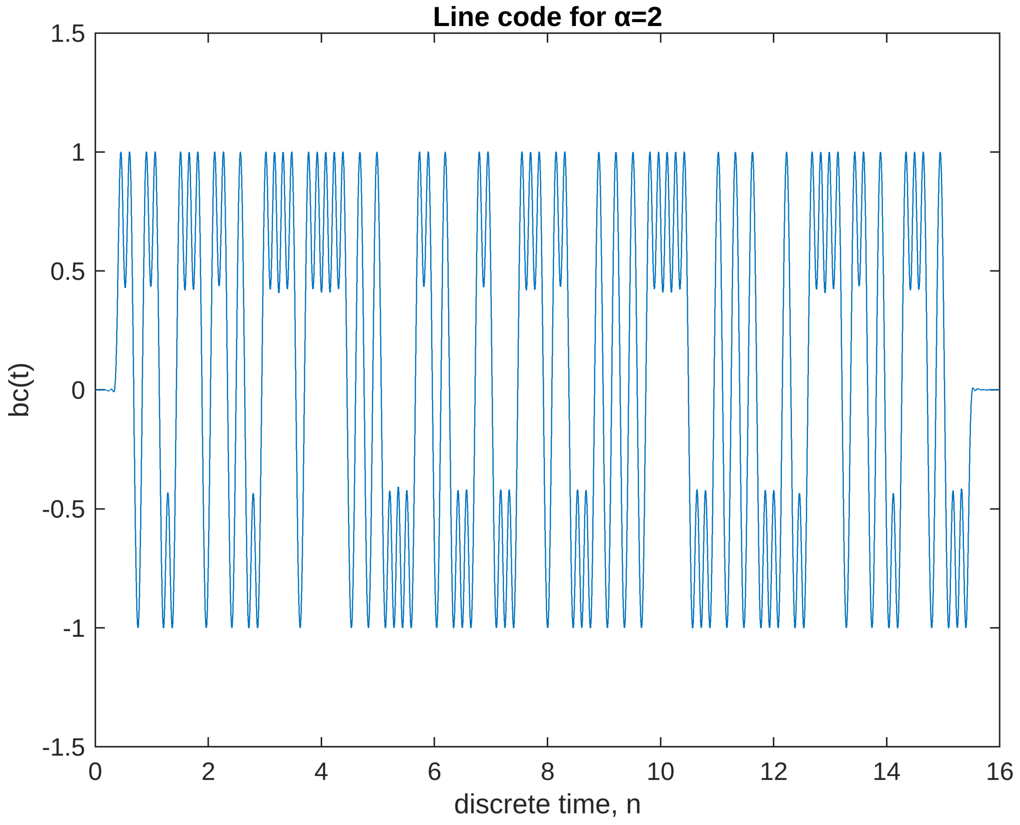

Eye diagram for α=0.5
Amplitude
Time
1.5
1
0.5
0
-0.5
-1
-1.5
-0.5
0
0.5

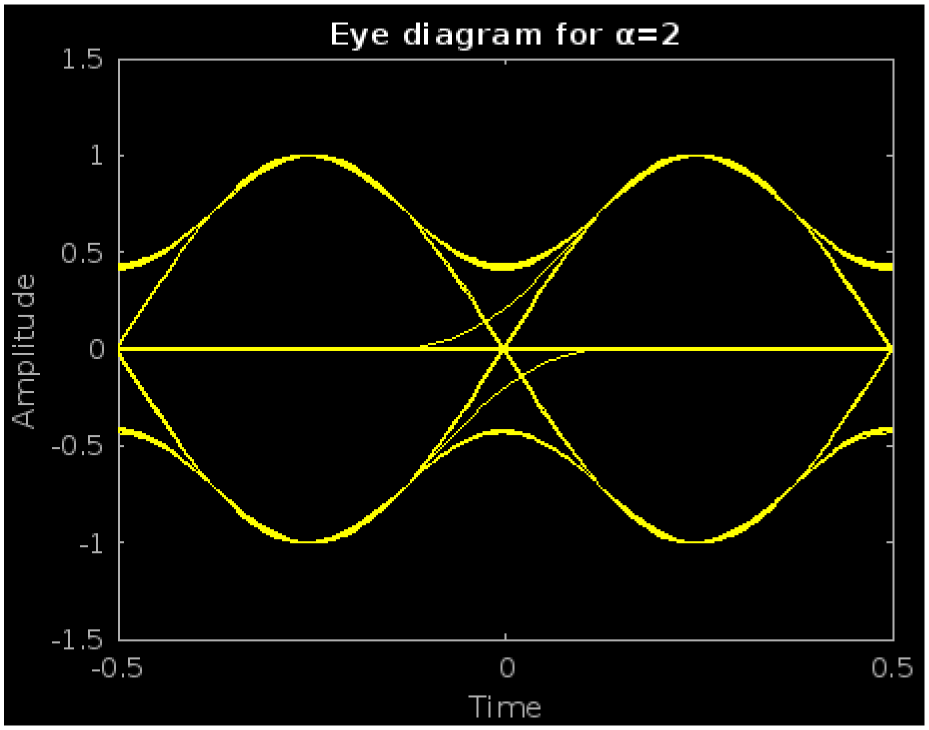

Line code for α=0.8
bc(t)
discrete time, n
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
14
16

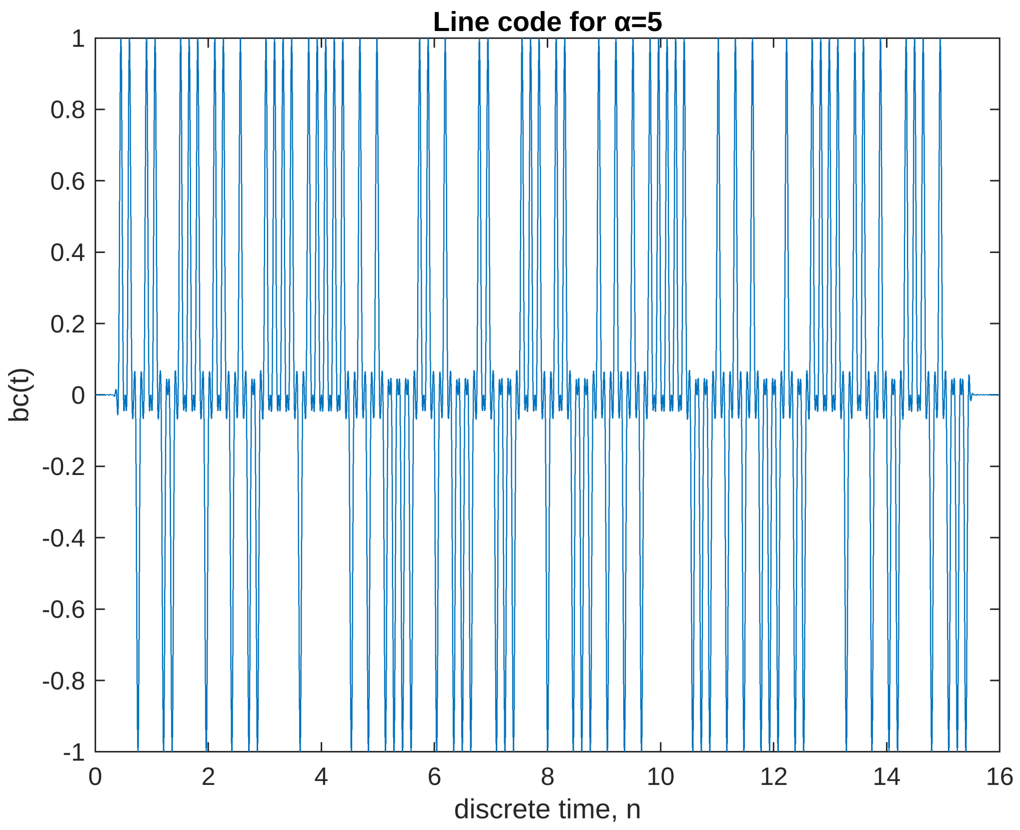

Eye diagram for α=0.8
Amplitude
Time
1.5
1
0.5
0
-0.5
-1
-1.5
-0.5
0
0.5

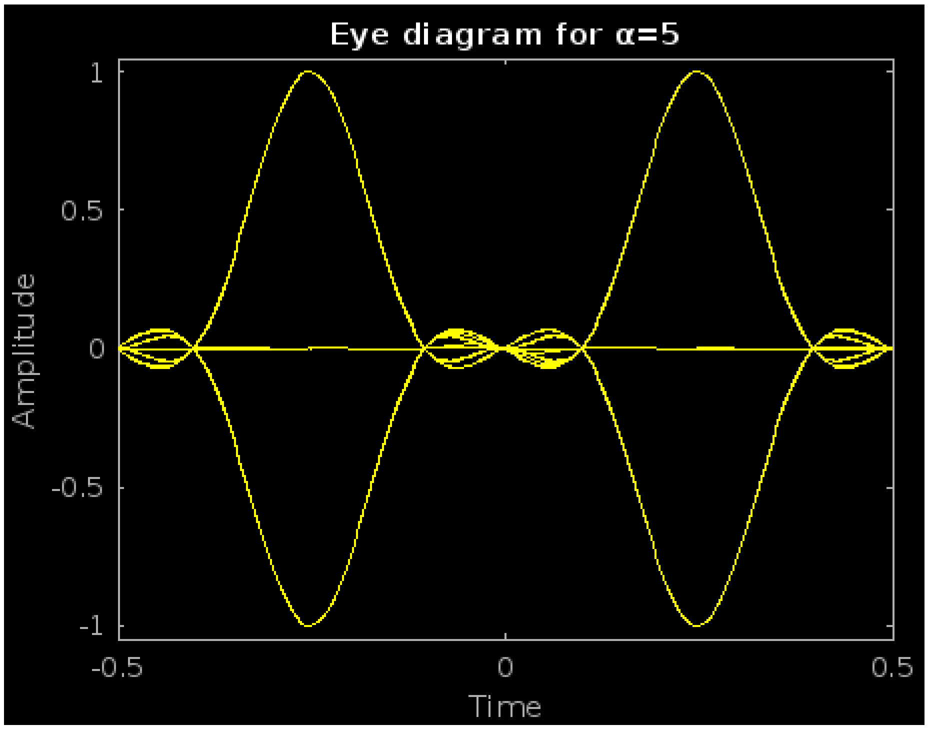

Line code for α=1
bc(t)
discrete time, n
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
14
16


Eye diagram for α=1
Amplitude
Time
1.5
1
0.5
0
-0.5
-1
-1.5
-0.5
0
0.5


Line code for α=2
bc(t)
discrete time, n
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
14
16


Eye diagram for α=2
Amplitude
Time
1.5
1
0.5
0
-0.5
-1
-1.5
-0.5
0
0.5





Inference: As alpha value increases, the signal become more clear, indicated by high signal-to-noise ratio as in the spectra and the eyediagram.