Task 1

In Task I, you are expected to generate eye diagrams for baseband binary phase shift keying (BPSK) signaling with different pulse shaping filters

```
br = 10;                % bit rate of the generator 10 bits/second
Tb = 1/br;              % bit duration
A = 1;                  % amplitude of the impulses
time_resolution = 100;  % number of data points between two impulses
delta_t = Tb/100;       % smallest unit of time
t_lim = 2;              % limit for the visualization in seconds
L = 1000;               % length of the binary sequence
time_duration = L*Tb;   % time to emmit all the bits in seconds

bin_seq = randi([0 1], 1, L);   % random binary sequence
t_com = 0:delta_t:time_duration;% common time vector

% Mapping binary data into impluses
imp_train1  = zeros(1,L); % impulse train representing BPSK symbols
for index = 1:L
    if bin_seq(index) == 1
        imp_train1(index) = +A;
    else
        imp_train1(index) = -A;
    end
end

t_impulse = 0:Tb:time_duration -Tb;
stem(t_impulse,imp_train1, 'LineWidth',1); grid on;
axis([0 t_lim -(3*A/2) (3*A/2)]);
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');
```

Generating the sinc filter

```
t_sinc = -time_duration: delta_t: time_duration;
sinc_pulse = sinc(t_sinc/Tb);
plot(t_sinc, sinc_pulse, 'LineWidth',1);
grid on; xlim([-t_lim/2 t_lim/2]);
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');

% for the convolution we need to take impulse train
% and the sinc filter to a common time scale.
imp_train2 = zeros(1, length(t_com));
for index = 0:L-1
    imp_train2(index*time_resolution +1) = imp_train1(index +1);
end
stem(t_com,imp_train2, 'LineWidth',1); grid on;
axis([0 t_lim -(3*A/2) (3*A/2)]);
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');
```

```
% convolution of impulse train and the sinc filter
tx_signal1 = conv(imp_train2, sinc_pulse, 'same');
plot(t_com, tx_signal1); hold on;
stem(t_com,imp_train2,'--' ,'LineWidth',1); grid on;
xlim([0 t_lim]);
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex'); hold off;

% generating the eyediagram of the transmitted signal 1
eyediagram(tx_signal1,2*Tb/delta_t, 2*Tb); grid on;
```

Generating the raised cosine filter with alpha 0.5

```
t_rcos = t_sinc;
rcos_pulse0point5 = sinc(t_rcos/Tb).*( cos(pi*0.5*t_rcos/Tb) ./ (1 - (2*0.5*t_rcos/Tb).^2) );
figure; plot(t_rcos, rcos_pulse0point5, 'LineWidth',1);
grid on; xlim([-t_lim/2 t_lim/2]);
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');

% convolution of impulse train and the raised cosine filter
tx_signal2 = conv(imp_train2, rcos_pulse0point5, 'same');
plot(t_com, tx_signal2); hold on;
stem(t_com,imp_train2,'--' ,'LineWidth',1); grid on;
xlim([0 t_lim]);hold off;
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');

% generating the eyediagram of the transmitted signal 2
eyediagram(tx_signal2,2*Tb/delta_t, 2*Tb); grid on;
```

Generating the raised cosine filter with alpha 1

```
rcos_pulse1 = sinc(t_rcos/Tb).*( cos(pi*1*t_rcos/Tb) ./ (1 - (2*1*t_rcos/Tb).^2) );
figure; plot(t_rcos, rcos_pulse1, 'LineWidth',1);
grid on; xlim([-t_lim/2 t_lim/2]);
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');

% convolution of impulse train and the raised cosine filter
tx_signal3 = conv(imp_train2, rcos_pulse1, 'same');
plot(t_com, tx_signal3); hold on;
stem(t_com,imp_train2,'--' ,'LineWidth',1); grid on;
xlim([0 t_lim]); hold off;
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');

% generating the eyediagram of the transmitted signal 3
eyediagram(tx_signal3,2*Tb/delta_t, 2*Tb); grid on;
```

Task 2

In Task 2, you are required to repeat Task 1, in the presence of additive white Gaussian noise (AWGN). To generate noise, use ‘randn’ function.

```
Eb = (1/2)*((+A)^2 + (-A)^2);       % average bit energy
gamma = 10;                         % power efficiency(Eb/N0) in dB
variance = Eb/(2*10^(gamma/10));    % variance of the AWGN
```

Adding AWGN to the signal obtained by convolving impulse train and the sinc pulse

```
tx_signal1_with_awgn = tx_signal1 + sqrt(variance)*rand(1, length(tx_signal1));
figure; plot(t_com, tx_signal1_with_awgn); hold on;
stem(t_com,imp_train2,'--' ,'LineWidth',1); grid on;
xlim([0 t_lim]);hold off;
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');

% generating the eyediagram of the transmitted signal 1 with awgn
eyediagram(tx_signal1_with_awgn,2*Tb/delta_t, 2*Tb); grid on;
```

Adding AWGN to the signal obtained by convolving impulse train and the raised cosine pulse with alpha = 0.5

```
tx_signal2_with_awgn = tx_signal2 + sqrt(variance)*rand(1, length(tx_signal2));
figure; plot(t_com, tx_signal2_with_awgn); hold on;
stem(t_com,imp_train2,'--' ,'LineWidth',1); grid on;
xlim([0 t_lim]);hold off;
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');

% generating the eyediagram of the transmitted signal 2 with awgn
eyediagram(tx_signal2_with_awgn,2*Tb/delta_t, 2*Tb); grid on;
```

Adding AWGN to the signal obtained by convolving impulse train and the raised cosine pulse with alpha = 1

```
tx_signal3_with_awgn = tx_signal3 + sqrt(variance)*rand(1, length(tx_signal3));
figure; plot(t_com, tx_signal3_with_awgn); hold on;
stem(t_com,imp_train2,'--' ,'LineWidth',1); grid on;
xlim([0 t_lim]);hold off;
ylabel("$ Amplitude(A)$",'Interpreter','latex');
xlabel("$ Time(t)$",'Interpreter','latex');

% generating the eyediagram of the transmitted signal 3 with awgn
eyediagram(tx_signal3_with_awgn,2*Tb/delta_t, 2*Tb); grid on;
```

Task 3

Q1-Q11

```
bit_seq_len = 10^4; % number of bits
SNR_list = [0:10];  % snr from 0 to 10

for x1 = 1:length(SNR_list)

   % Transmitter
   bit_seq = randi([0 1],1 ,bit_seq_len);   % random bit sequence
   symbol= 2*bit_seq-1;                     %0 to -1 and 1 to 1
   h= [0.2 0.9 0.3];                        % channel response
   c_out = conv(symbol,h);
   Eb=1;
   No=Eb/(10^(SNR_list(x1)/10));
   AWGN=No*randn(1,max(size(c_out)));

   % Noise addition
   y = c_out + AWGN; % additive white gaussian noise
   yAWGN=symbol+AWGN(1:bit_seq_len);%sybols + noise (no channel response)
   for x2 = 1:4
       hMat = toeplitz([h([2:end]) zeros(1,2*x2+1-length(h)+1)],...
           [ h([2:-1:1]) zeros(1,2*x2+1-length(h)+1) ]);
       b  = zeros(1,2*x2+1);

       b(x2+1) = 1;
       c  = [inv(hMat)*b.'].';% coefficient set
       yFilt = conv(y,c);
       yFilt = yFilt(x2+2:end);
       ySamp = yFilt(1:1:bit_seq_len);
       size(ySamp);

       %decoding and calculating the number of errors
       ipHat = ySamp>0;
       yAWGN= yAWGN>0;
       nAWGN(1,x1)=size(find([bit_seq- yAWGN]),2);
       nErr(x2,x1) = size(find([bit_seq- ipHat]),2);
   end
end

simBer = nErr/bit_seq_len; % Bit error rate

% plot
figure;
semilogy(SNR_list,simBer(1,:),'bs-'); hold on;
semilogy(SNR_list,simBer(2,:),'gd-');
semilogy(SNR_list,simBer(3,:),'ks-');
semilogy(SNR_list,simBer(4,:),'mx-');
semilogy(SNR_list,nAWGN/bit_seq_len,'r*-');
axis([0 10 10^-3 0.5]); grid on;
legend('3tap', '5tap','7tap','9tap','AWGN');
xlabel('Eb/No, dB');
ylabel('Bit Error Rate');
title('Bit error probability curve for BPSK in ISI with ZF equalizers');
```