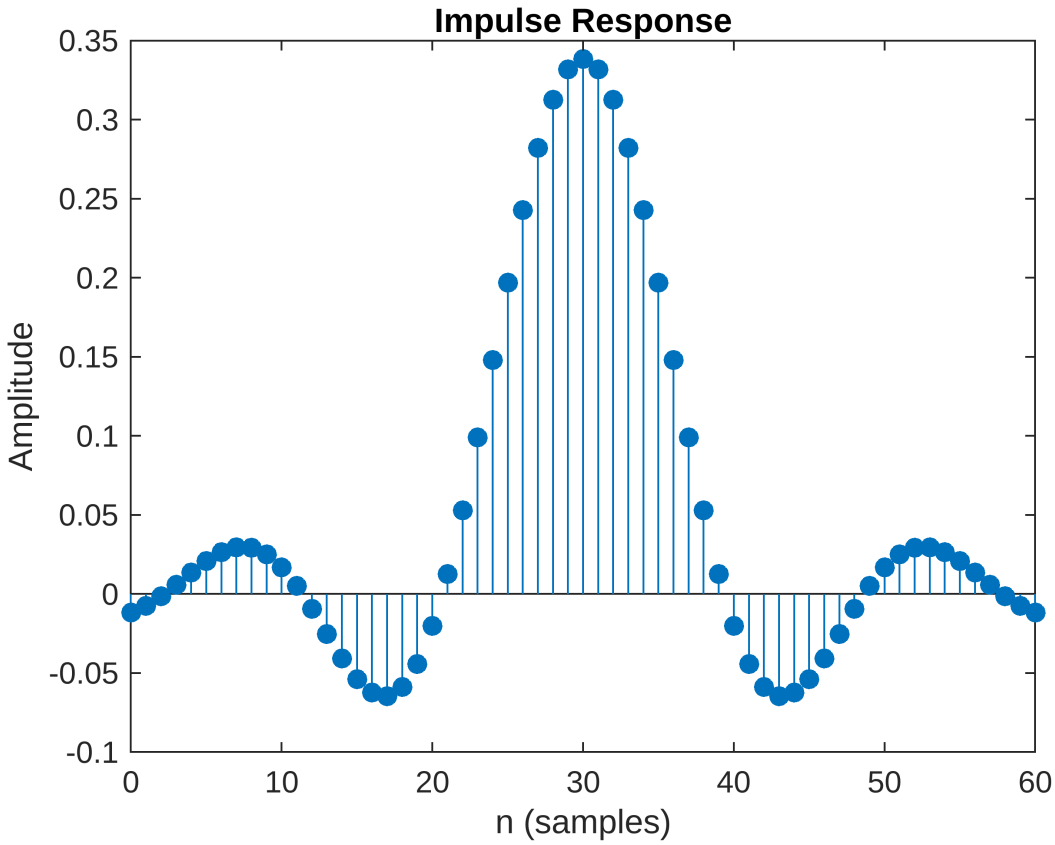
Communication System Lab 5

By: Saurabh Kumar (SC22B146)

SRRC and frequency selective channel

1. Generate the line code brc(t) corresponding to a random sequence of 100 bits for Tb = 0.1s using the SRRC pulse shape. Truncate and time delay the pulse shape so that the pulse extends from 0 to 6Tb and the peak of the pulse is at 3Tb. Record the parameters used for the SRRC pulse shape (excess bandwidth and or rolloff factor).

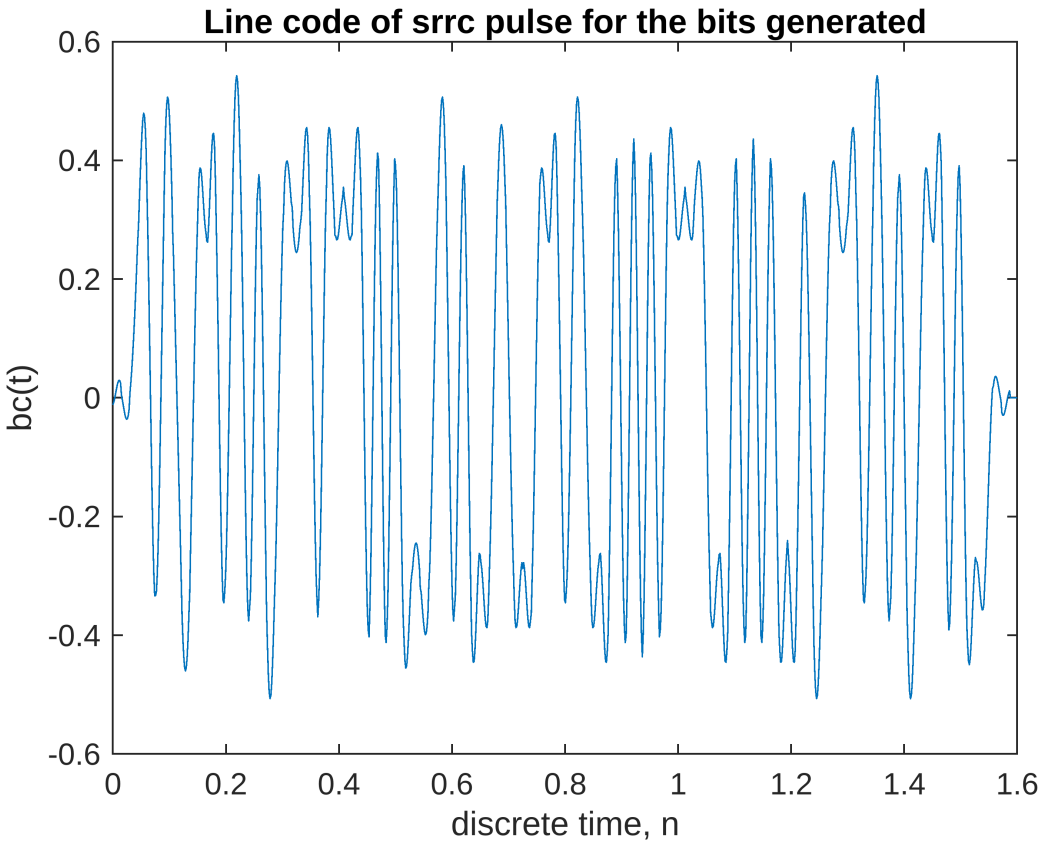
```
% paramters
Tb = 0.1;
fs = 100;
beta = 0.25;
N = 100;

brc = rcosdesign(beta, 6, Tb*fs,'sqrt');
impz(brc);
```




```
% source bits
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_brc = zeros(1,sig_len*fs);
for i=1:(length(source))
    signal_brc(round((i-1)*Tb*fs+1):round((i+5)*Tb*fs+1)) =
signal_brc(round((i-1)*Tb*fs+1):round((i+6-1)*Tb*fs+1)) + source(i)*brc;
end

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

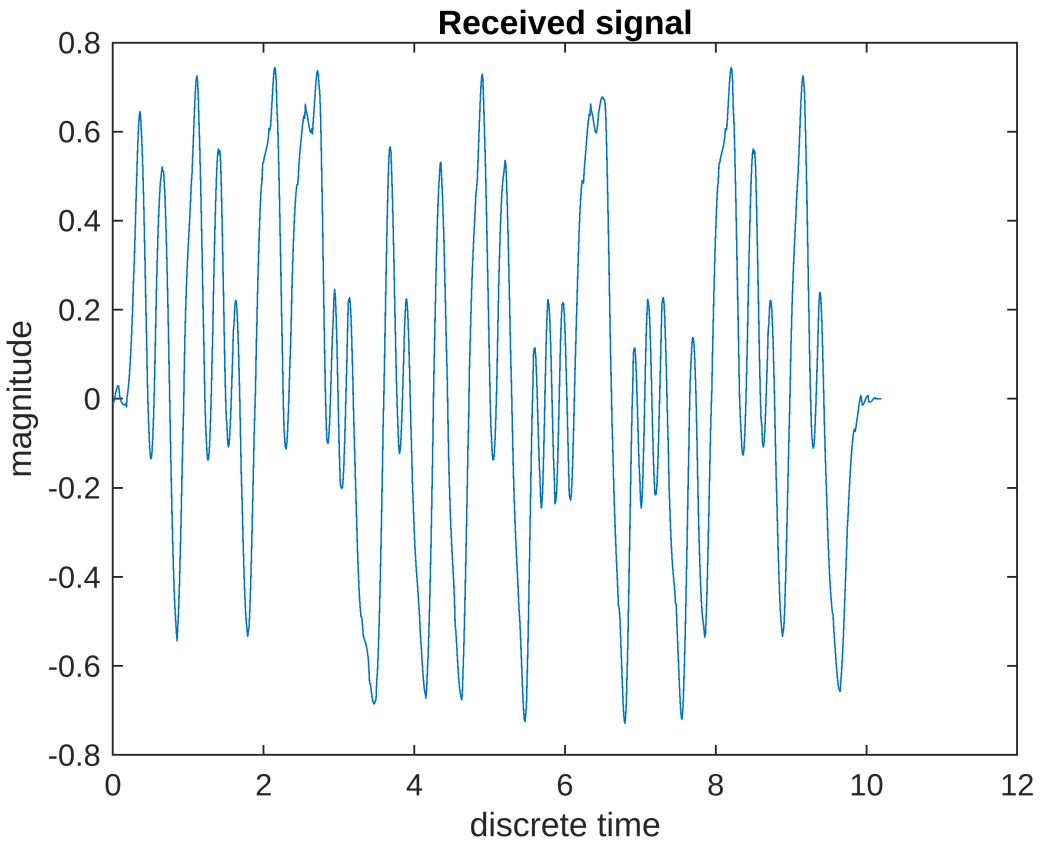


Inference: The pulse is generated using the built-in function 'rcosdesign' and the line code is plotted for 100 bits.

2. Simulate transmission of brc(t) through a channel which has the following impulse response: $\delta(t) + 0.75\delta(t - Tb) + 0.25\delta(t - 2Tb)$.

```
% channel response
channel = zeros(1,2*Tb*fs);
channel(1) = 1;
```

```
channel(Tb*fs) = 0.75;
channel(2*Tb*fs) = 0.25;

received_signal = conv(signal_brc, channel);

figure;
t = linspace(0,(100*Tb+2*Tb),length(received_signal));
plot(t, received_signal);
title("Received signal");
xlabel("discrete time");
ylabel("magnitude");
```

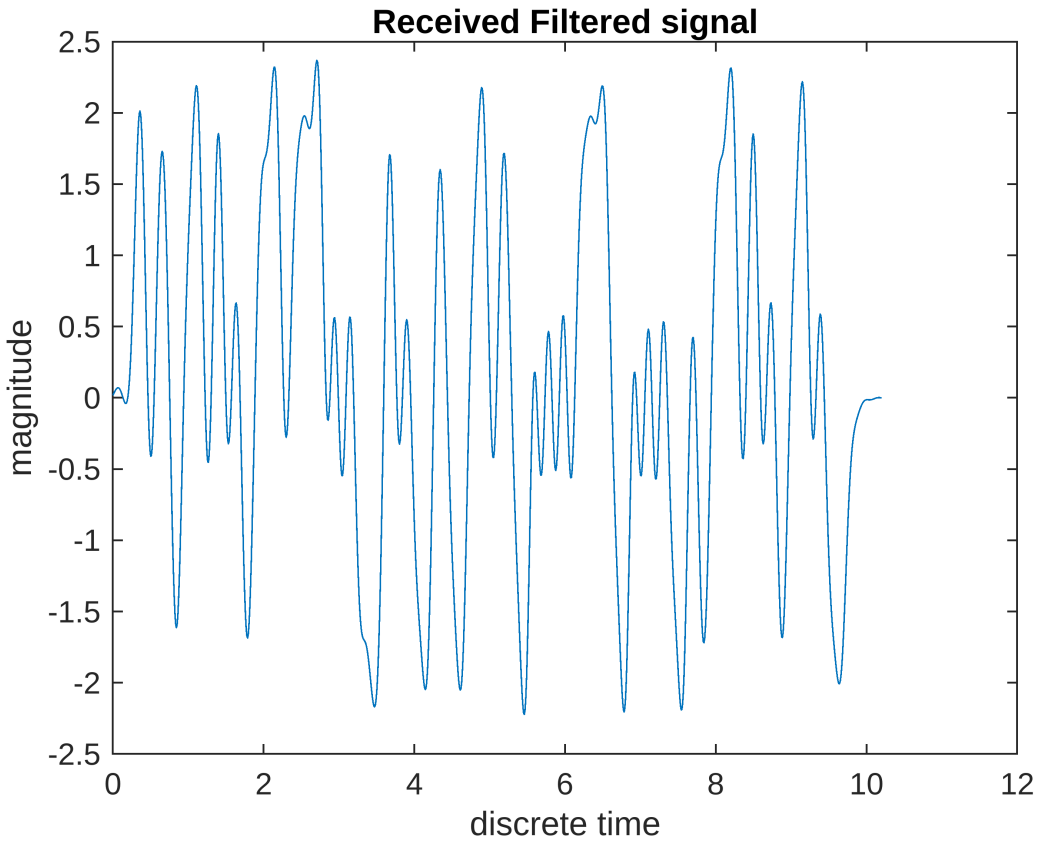


Inference: The channel is created is the signal is passed through it. We observe intersymbol interference.

3. Implement a receive filter which has a SRRC impulse response which is matched to the pulse shape using for generating brc(t). Filter the channel output using this SRRC receive filter.

```
% matched_filter
n = Tb*fs;
filter_matched = fliplr(brc);
received_signal_filtered = conv(received_signal, filter_matched, 'same');

figure;
t = linspace(0,(100*Tb+2*Tb),length(received_signal_filtered));
plot(t, received_signal_filtered);
```

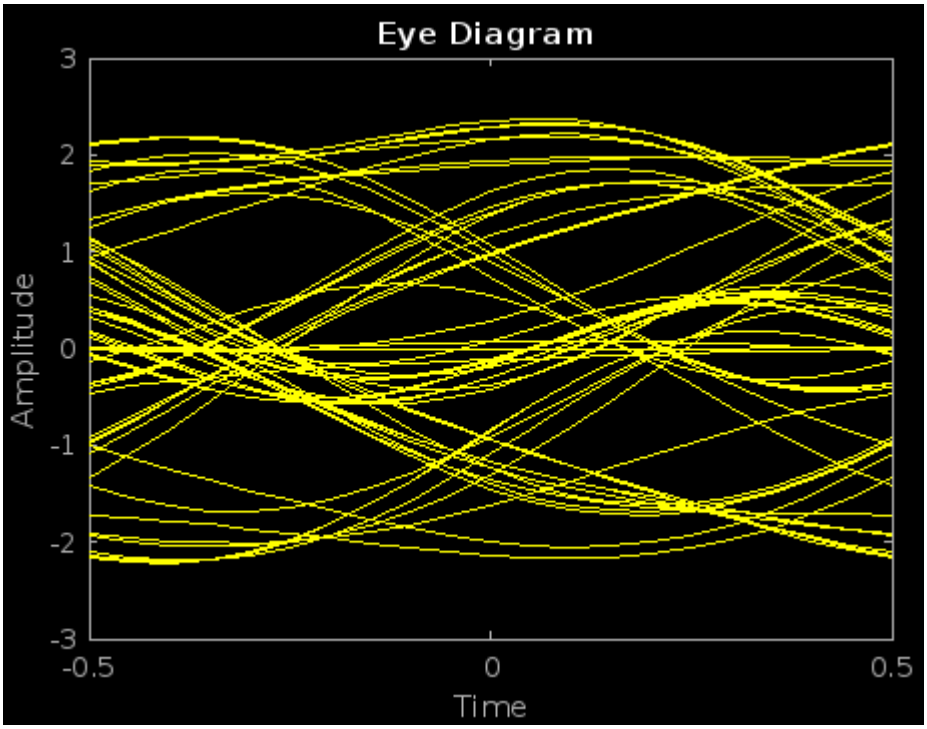
```
title("Received Filtered signal");
xlabel("discrete time");
ylabel("magnitude");
```



Inference: The signal at the receiver is passed through receiver filter which is a matched filter.

4. Plot the eye diagram of the output of this receive filter. What do you observe? Compare this with the eye diagram that you have obtained in the last labsheet for the End-To-End system with SRRC. Comment on the characteristics of the eye opening that is obtained now.

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



Inference: As compared to the case with reciever having low pass response, this channel produces the signal which is more distorted, as indicated by the much less opening in the eye diagram.

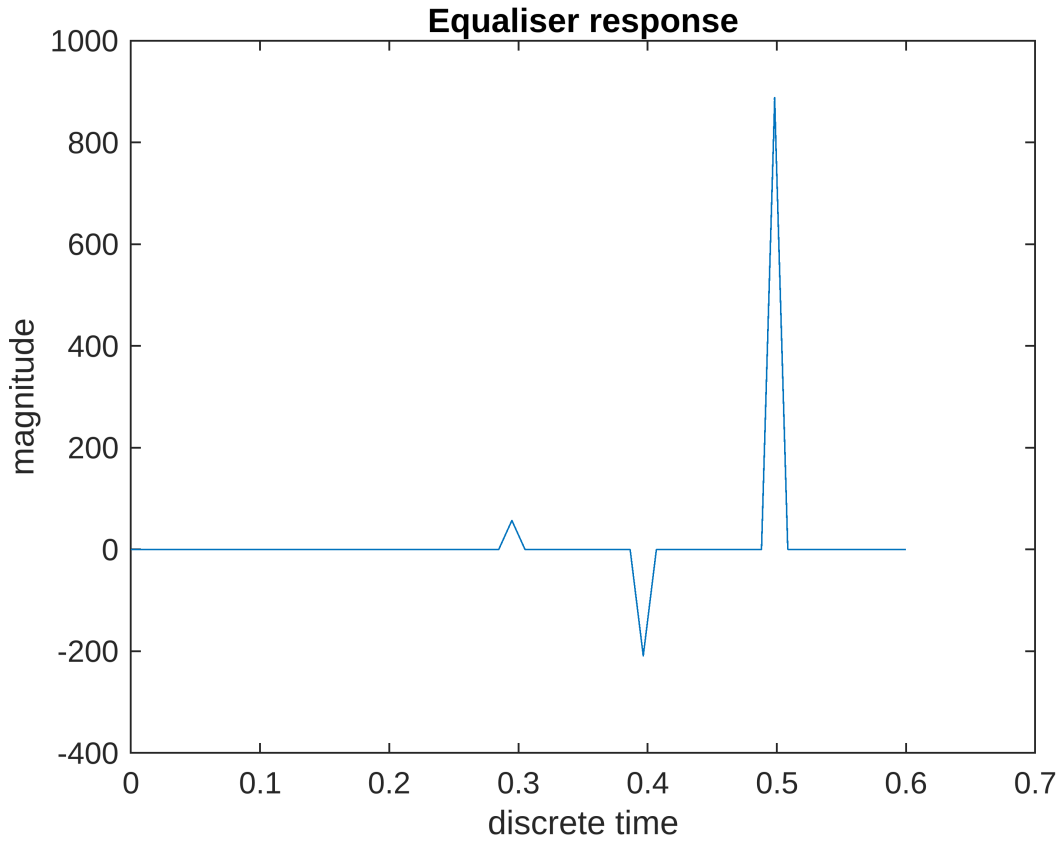
5. Implement a linear transversal zero forcing equalizer that tries to mitigate ISI. Consider the output of the zero forcing equalizer for a single input pulse. Design the filter such that the sample value of this signal at 3Tb is 1, while other samples (taken at 3Tb ± mTb, m ∈ {1, 2, 3}) are 0. You should record all the steps that you have used in the design of this zero forcing equalizer.

```
% Getting singnal points
u0 = received_signal_filtered(round(1));
u1 = received_signal_filtered(round(Tb*fs));
u2 = received_signal_filtered(round(2*Tb*fs));
u3 = received_signal_filtered(round(3*Tb*fs));
u4 = received_signal_filtered(round(4*Tb*fs));
u5 = received_signal_filtered(round(5*Tb*fs));
u6 = received_signal_filtered(round(6*Tb*fs));

% Forming and solving linear equations
syms w0 w1 w2 w3 w4 w5 w6
eqn1 = w0*u0 == 0; % g0
eqn2 = w0*u1 + w1*u0 == 0; % g1
eqn3 = w0*u2 + w1*u1 + w2*u0 == 0; % g2
eqn4 = w0*u3 + w1*u2 + w2*u1 + w3*u0 == 1; % g3 (sample value is 1)
eqn5 = w0*u4 + w1*u3 + w2*u2 + w3*u1 + w4*u0 == 0; % g4
eqn6 = w0*u5 + w1*u4 + w2*u3 + w3*u2 + w4*u1 + w5*u0 == 0; % g5
eqn7 = w0*u6 + w1*u5 + w2*u4 + w3*u3 + w4*u2 + w5*u1 + w6*u0 == 0; % g6
[A,B] = equationsToMatrix([eqn1,eqn2,eqn3,eqn4,eqn5,eqn6],
[w0,w1,w2,w3,w4,w5,w6]);
```

```
X = linsolve(A,B);
```

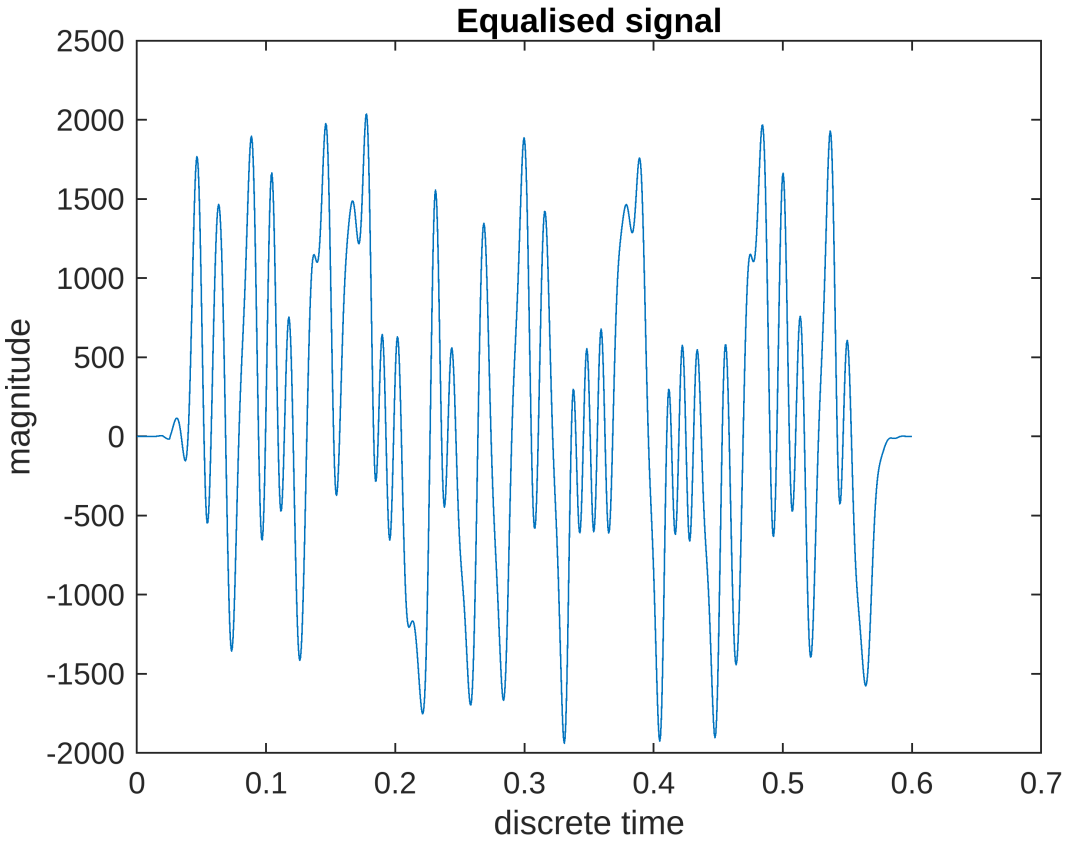
```
Warning: symbolic:mldivide:RankDeficientSystem⁄Solution is not unique because the system is rank-
deficient.⁄
```

```
% Weights
w0 = round(X(1));
w1 = round(X(2));
w2 = round(X(3));
w3 = round(X(4));
w4 = round(X(5));
w5 = round(X(6));
w6 = round(X(7));

% Equalizer response
equalizer = zeros(1,round(6*Tb*fs));
equalizer(round(1)) = w0;
equalizer(round(Tb*fs)) = w1;
equalizer(round(2*Tb*fs)) = w2;
equalizer(round(3*Tb*fs)) = w3;
equalizer(round(4*Tb*fs)) = w4;
equalizer(round(5*Tb*fs)) = w5;
equalizer(round(6*Tb*fs)) = w6;

% Zero Forcing Equalizer Plot
figure;
equalizer_t = linspace(0,6*Tb,length(equalizer));
plot(equalizer_t,equalizer);
title("Equaliser response");
xlabel("discrete time");
ylabel("magnitude");
```

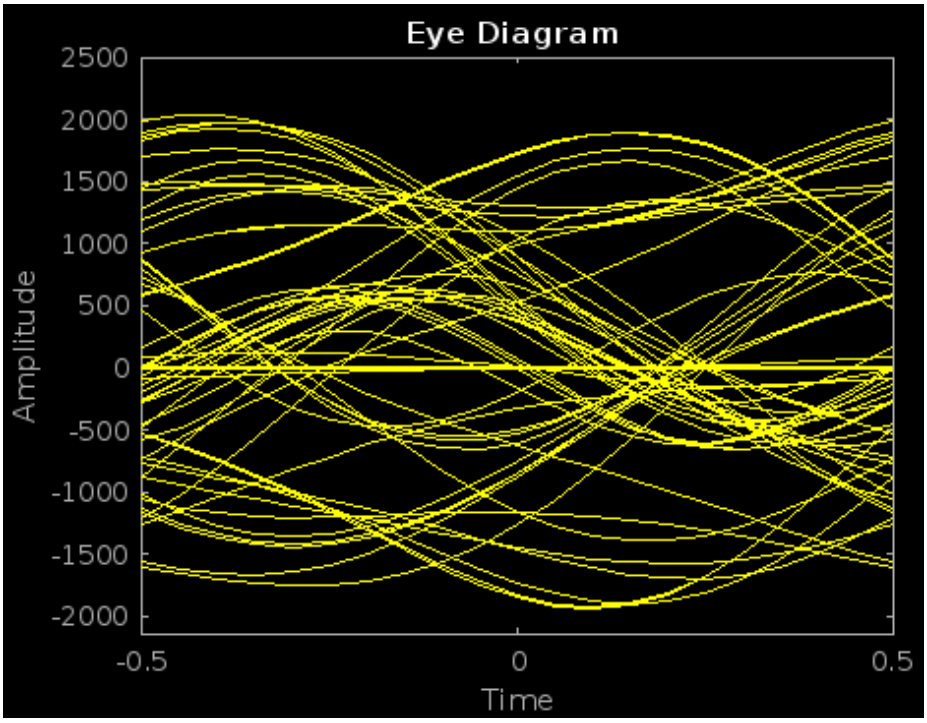


6. Filter the output of the receive filter (in (3) above) using this equalizer. Plot the time domain response as well as the eye diagram of the channel output. What do you observe? Compare the eye diagram with the one in (4) above.

```
equalized_sig = conv(received_signal_filtered,equalizer);
figure;
equalizer_t = linspace(0,6*Tb,length(equalized_sig));
plot(equalizer_t,equalized_sig);
title("Equalised signal");
xlabel("discrete time");
ylabel("magnitude");
```




```
eyediagram(equalized_sig, 2*Tb*fs, 1, round(Tb*fs/2));
```



Inference: The eye diagram of equalized filtered signal has more opening, indicating the less distortion.