filter_pulse.m :

```
function filtered_time_pulse = filter_pulse(pulse_freq, pulse_number, color, Freq, B, time, type,
Fs,raised_cos_Freq)
    filteredNewPulseFreq = pulse_freq;
    if type == 1  % filter pulse using bandlimitied filter, B = 100 kHz
        filteredNewPulseFreq(Freq > B) = 0;
        filteredNewPulseFreq(Freq < -B) = 0;
    else  %filter pulse using raised cosine filter
        filteredNewPulseFreq = filteredNewPulseFreq.*raised_cos_Freq;  %conv in time =
multiplication in frequency
    end
    plot(Freq,abs(filteredNewPulseFreq)/Fs,color)  %plot filtered pulse in Freq domain
    title(['Pulse ', pulse_number, ' filtered in Freq Domain'])
    figure;

    filtered_time_pulse = ifft(ifftshift(filteredNewPulseFreq));  %inverse fourier transform
    plot(time/10^5,abs(filtered_time_pulse),color)  %plot filtered pulse in Time domain
    title(['Pulse ', pulse_number, ' filtered in Time Domain'])
    figure;
end
```

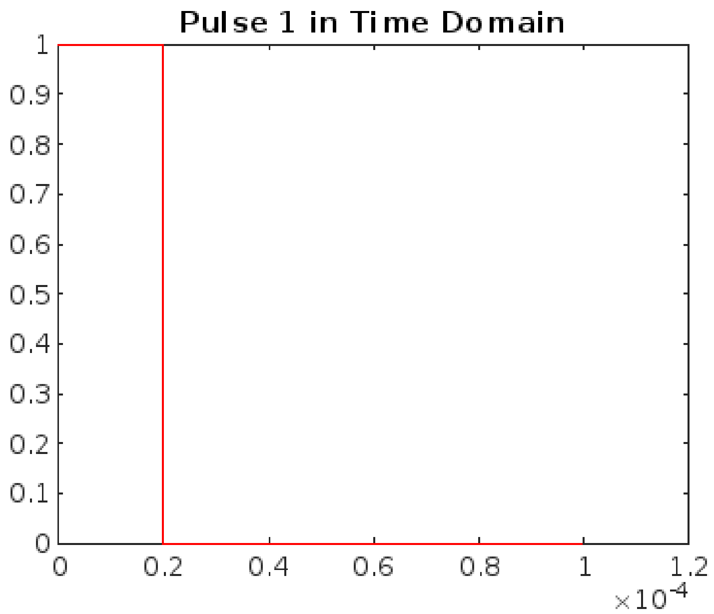
generate_pulse.m:

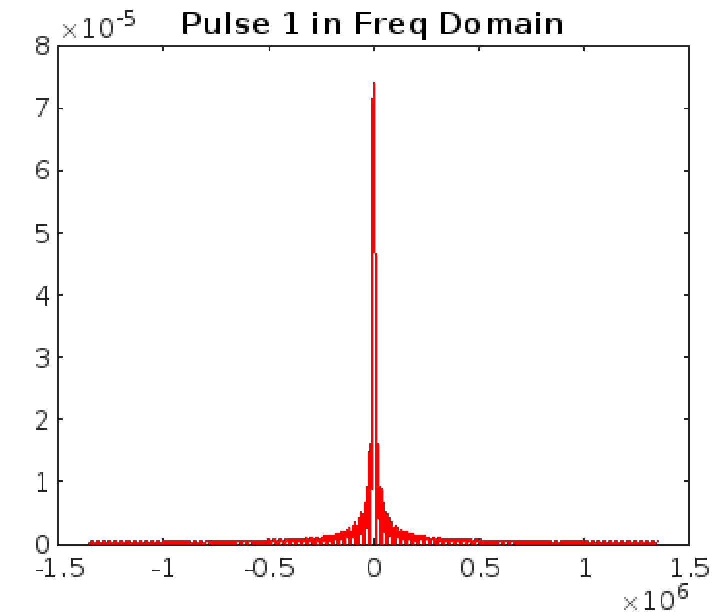
```
function pulse_freq = generate_pulse(limit1, step_size, limit2, time, Freq, pulse_number,
color,Fs)
    pulse = [];

    for i = limit1 : step_size : limit2
        pulse(end+1) = 1; %append ones to construct rect
    end

    %concatenating zeros at left and right of the rect to satisfy
    %dimensions and make graph clear
    for i = time
        if(i < limit1) %concatenate zeros to the left of rect
            pulse = cat(2,0,pulse);
        end
        if(i > limit2) %concatenate zeros to the right of rect
            pulse = cat(2,pulse,0);
        end
    end

    %the right limit of first pulse is the same the left limit of the second pulse
    %so, the right limit of first pulse should have value zero while the value will be preserved
in the left limit of the second pulse
    pulse(time == limit2) = 0;

    newPulse = pulse;
    plot(time/10^5,newPulse,color) %time/B for plotting (B = 10^5)
    title(['Pulse ', pulse_number, ' in Time Domain']) %plot pulse in time domain
    figure;

    pulse_freq = fftshift(fft(newPulse));  %fourier transform
    plot(Freq,abs(pulse_freq)/Fs,color);  %plot pulse in freq domain
    title(['Pulse ', pulse_number, ' in Freq Domain'])
    figure;
end
```

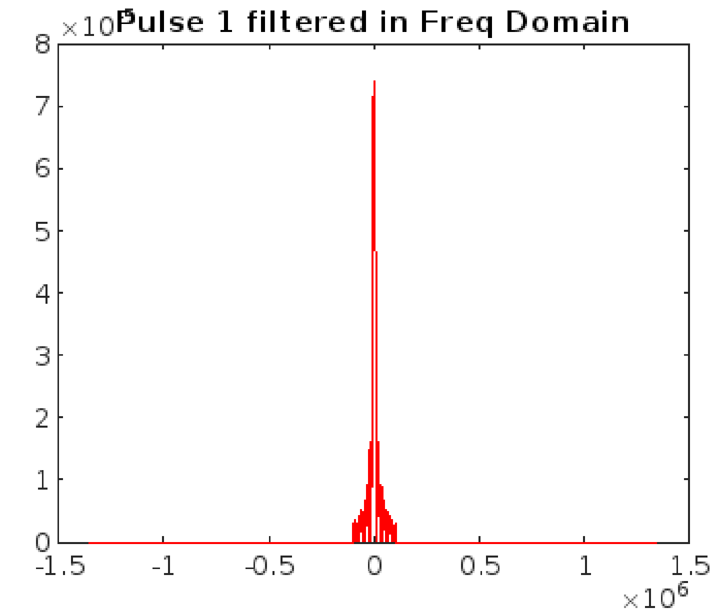
```
%Part 1 ISI LAB
close all;
clear all;
clc;

B=100*10^3; % bandwidth of band limited channel //used in band limit channel
 function
time=0:0.01:10; %random time
T=2/B; %duration of square pulse
Fs = 2700000; %assumed fs
Fn = Fs/2; % to plot in both sides // left& right
Freq = linspace(-Fn, Fn, numel(time)); %freq axis

pulse1_freq = generate_pulse(0, 0.01, 2, time, Freq, '1', 'r',Fs);%calling
 generate_pulse function to generate first pulse
filtered_time_pulse1 = filter_pulse(pulse1_freq, '1', 'r', Freq, B,  time,
 1,Fs);%calling filter_pulse function to filter first pulse using ideal filter

pulse2_freq = generate_pulse(2, 0.01, 4, time, Freq, '2', 'b',Fs); %calling
 generate_pulse function to generate second pulses
filtered_time_pulse2 = filter_pulse(pulse2_freq, '2', 'b', Freq, B,  time,
 1,Fs);%calling filter_pulse function to filter second pulse using ideal
 filter


plot(time/10^5,filtered_time_pulse1,'r') % plotting filtered first pulse in
 time domain
hold on;
plot(time/10^5,filtered_time_pulse2,'b')  % plotting filtered second pulse in
 time domain
title('Pulses 1&2 filtered in Time Domain')%both first and second pulse
 together
figure;

sum_filtered_time_pulses = filtered_time_pulse1+filtered_time_pulse2; % adding
 both filtered pulses
plot(time/10^5, sum_filtered_time_pulses, 'g') % plotting both filtered pulses
 in time domain
title('Sum of Pulses 1&2 filtered in Time Domain')
figure;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%


r = 1; %rolloff factor
f0 = (B)/(1+r); %6 db bw of filter
f_delta = B - f0; % difference between absolute BW and f0 // difference
 between f0 and f1

%Raised cosine rolloff Nequist filtering equation in time domain
```

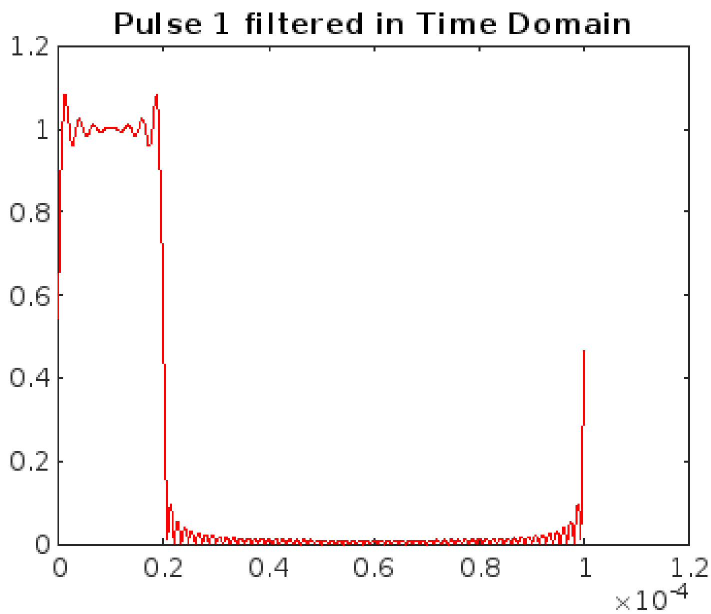
```
raised_cos_time = 2 .* f0 .*((sin(2 .* pi .* f0 .* (time/10^5))) ./ (2 .*
 pi .* f0 .* (time/10^5))) .* ((cos(2 .* pi .* f_delta .* (time/10^5))) ./ (1
 - (4 .* f_delta .* (time/10^5)).^2));

%handling nan & infinity values
infIndices = find(isinf(raised_cos_time));
nanIndices= find(isnan(raised_cos_time));
if ~isempty(infIndices)
    for i=infIndices
        raised_cos_time(i)=(raised_cos_time(i+1)+raised_cos_time(i-1))/2;
    end
end
if ~isempty(nanIndices)

    for i=nanIndices
        if i==1
            raised_cos_time(i)=0;
        else
            raised_cos_time(i)=(raised_cos_time(i+1)+raised_cos_time(i-1))/2;
        end
    end
end

raised_cos_Freq = fftshift(real(fft(raised_cos_time))); %converting raised cos
 to frequency domain

raised_cos_Freq=raised_cos_Freq/max(real(raised_cos_Freq));%normalization

plot(time/10^5,raised_cos_time) %plotting raised cosine filter in time domain
title('Raised Cosine in Time Domain')
figure;

plot(Freq,raised_cos_Freq) %plotting raised cosine filter in frequency domain
title('Raised Cosine in Freq Domain')
figure;

RC_filtered_time_pulse1 = filter_pulse(pulse1_freq, '1 RC', 'r', Freq, B,
 time, 2,Fs, raised_cos_Freq);%calling filter_pulse function to filter first
 pulse using raised cosine filter
RC_filtered_time_pulse2 = filter_pulse(pulse2_freq, '2 RC', 'b', Freq, B,
 time, 2,Fs, raised_cos_Freq);%calling filter_pulse function to filter second
 pulse using raised cosine filter


plot(time/10^5,RC_filtered_time_pulse1,'r')% plotting filtered first pulse in
 time domain
hold on;
plot(time/10^5,RC_filtered_time_pulse2,'b')% plotting filtered second pulse in
 time domain
title('Pulses 1&2 RC filtered in Time Domain')
figure;

RC_sum_filtered_time_pulses = RC_filtered_time_pulse1 +
 RC_filtered_time_pulse2;
```

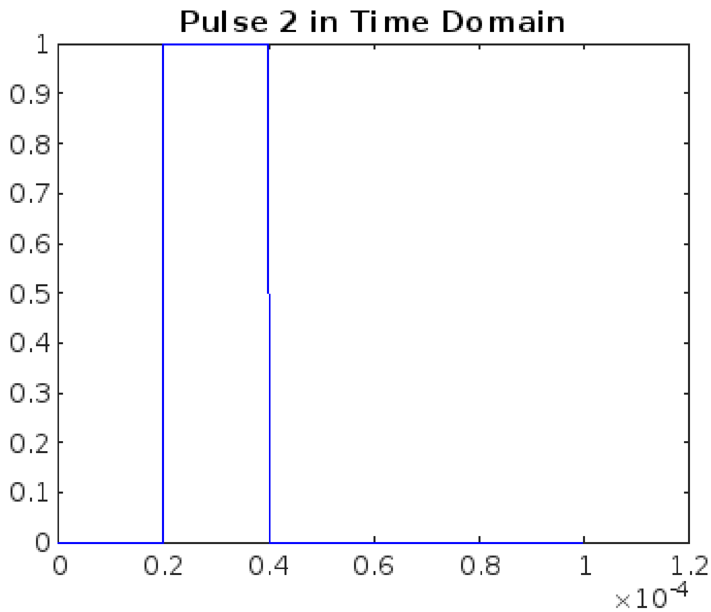
```
plot(time/10^5, RC_sum_filtered_time_pulses, 'g')
title('Sum of Pulses 1&2 RC filtered in Time Domain')
```

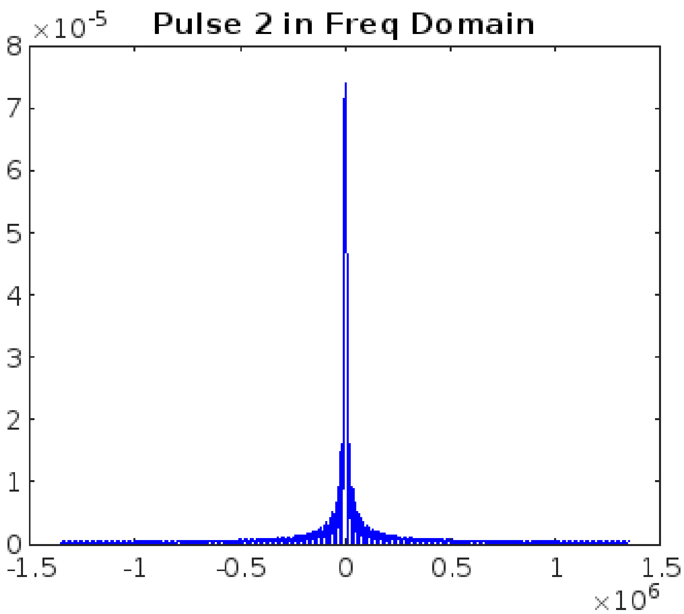


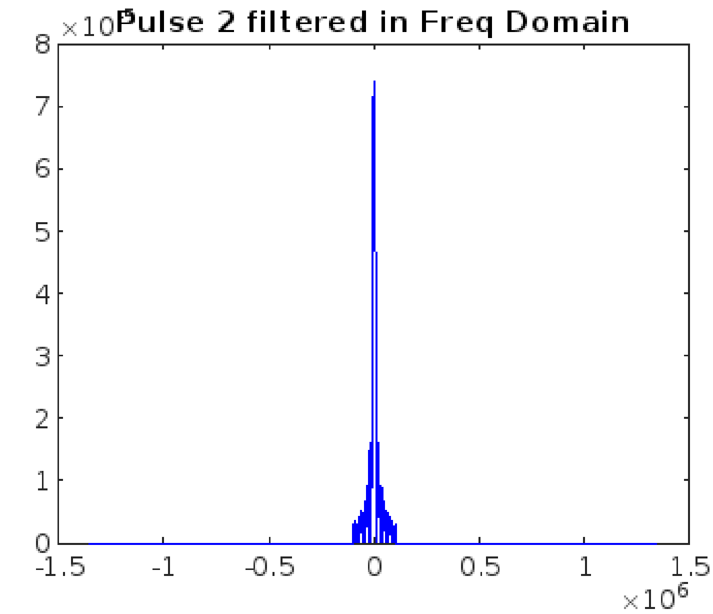


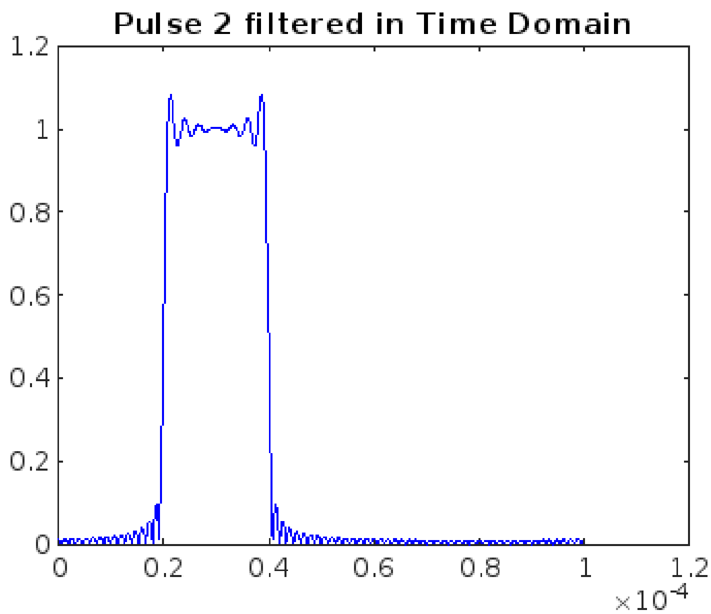

Pulse 1 filtered in Freq Domain
×10^5
8
7
6
5
4
3
2
1
0
-1.5
-1
-0.5
0
0.5
1
1.5
×10^6

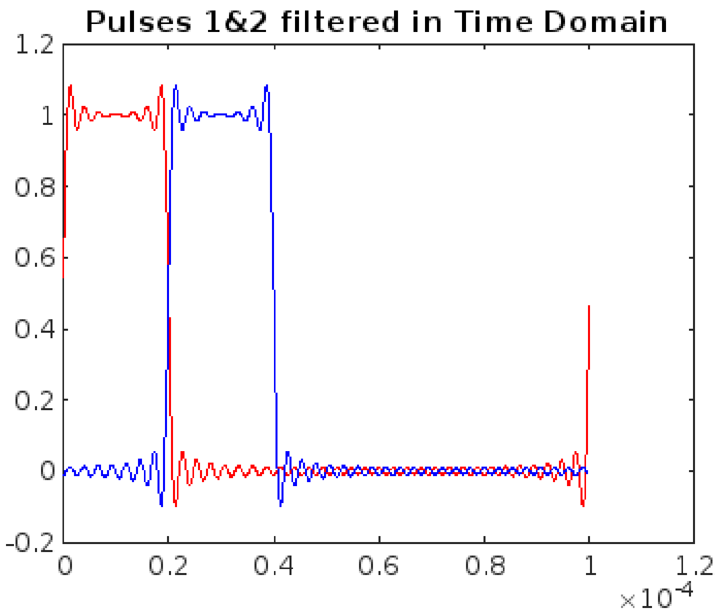

Pulse 1 filtered in Time Domain
1.2
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
×10^-4

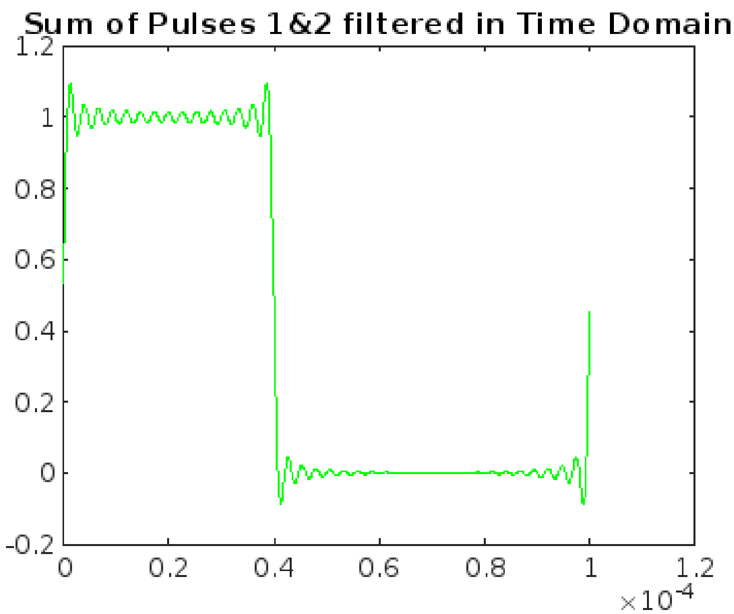

Pulse 2 in Time Domain
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
×10-4

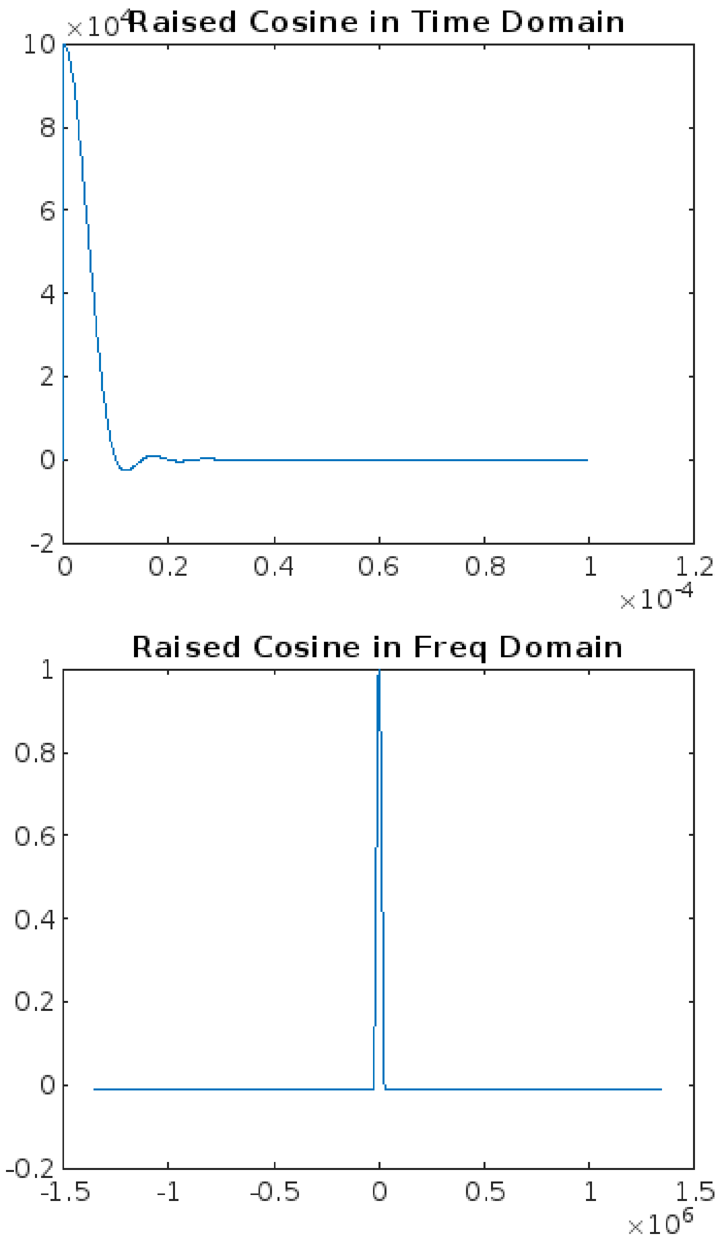

×10-5
Pulse 2 in Freq Domain
8
7
6
5
4
3
2
1
0
-1.5
-1
-0.5
0
0.5
1
1.5
×106


Pulse 2 filtered in Freq Domain
×10^5
8
7
6
5
4
3
2
1
0
-1.5
-1
-0.5
0
0.5
1
1.5
×10^6

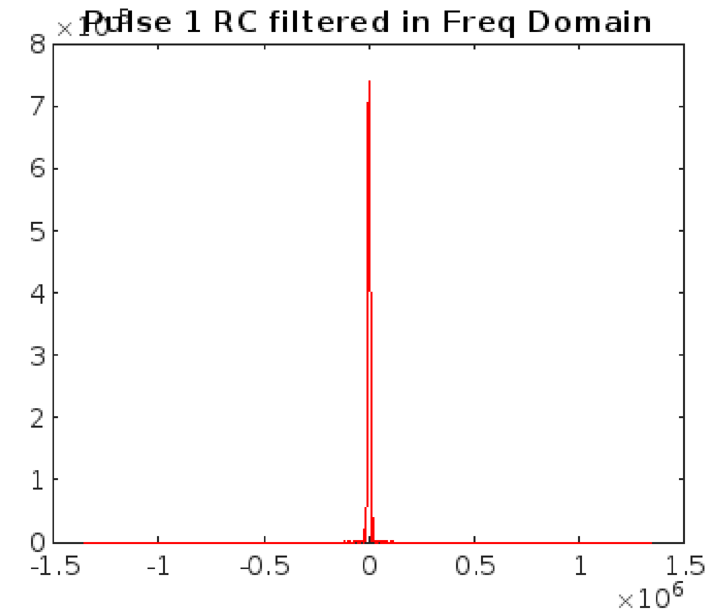

Pulse 2 filtered in Time Domain
1.2
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
×10^-4

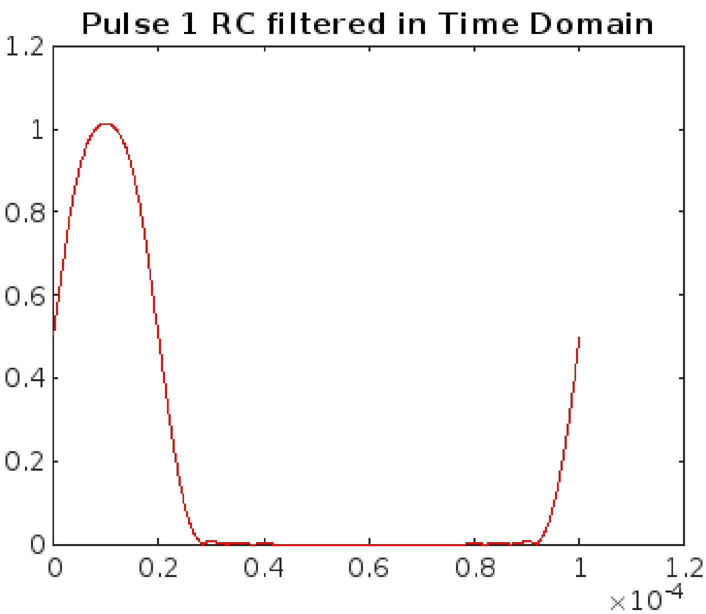

Pulses 1&2 filtered in Time Domain
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
×10⁻⁴

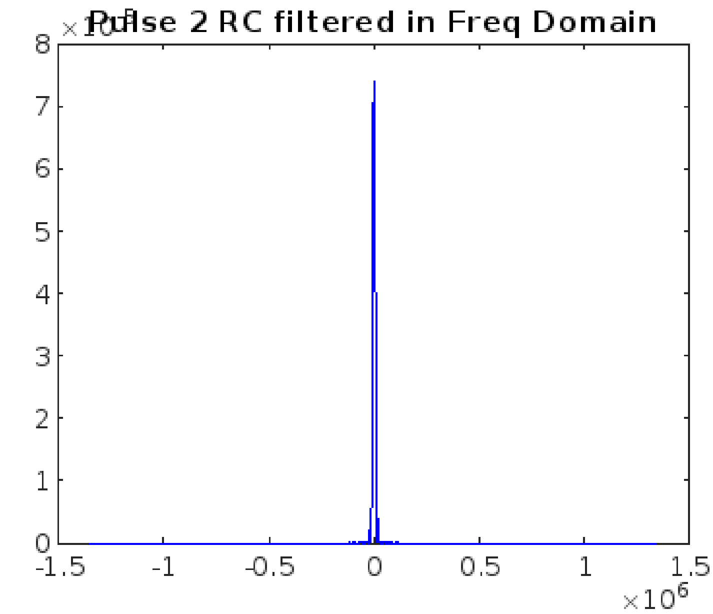

Sum of Pulses 1&2 filtered in Time Domain
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
×10⁻⁴

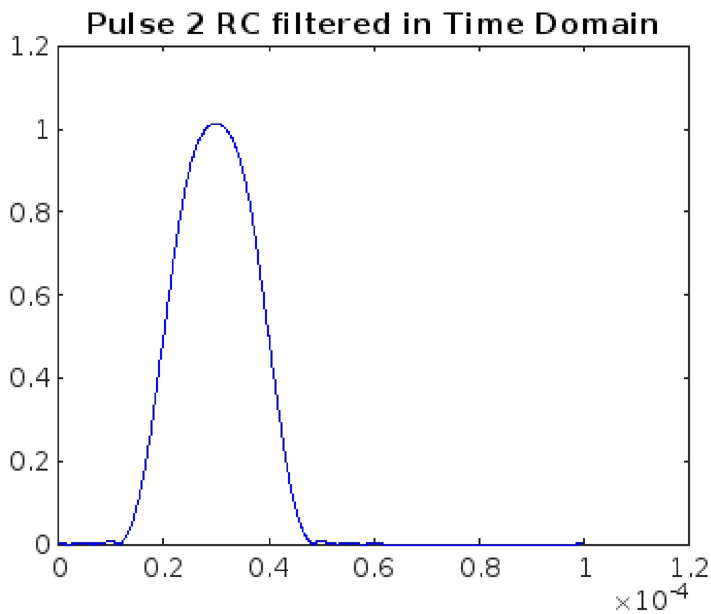


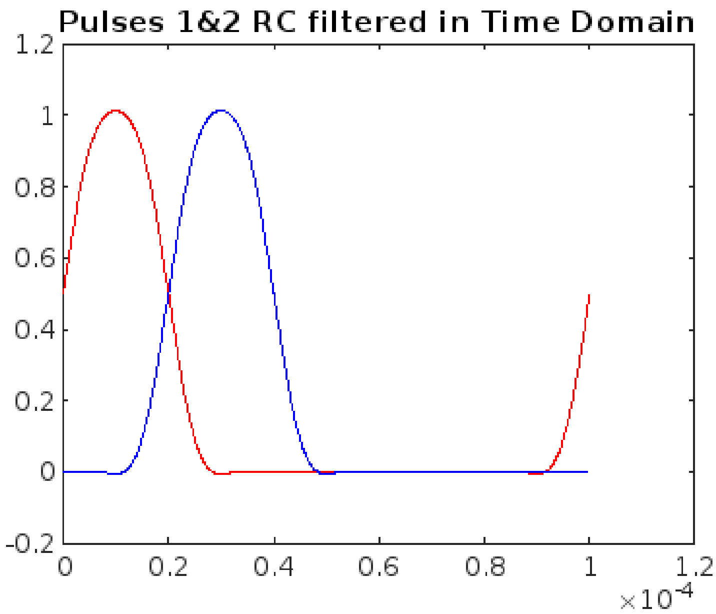
Raised Cosine in Time Domain
×10^-4
10
8
6
4
2
0
-2
0
0.2
0.4
0.6
0.8
1
1.2
×10^-4
Raised Cosine in Freq Domain
1
0.8
0.6
0.4
0.2
0
-0.2
-1.5
-1
-0.5
0
0.5
1
1.5
×10^6

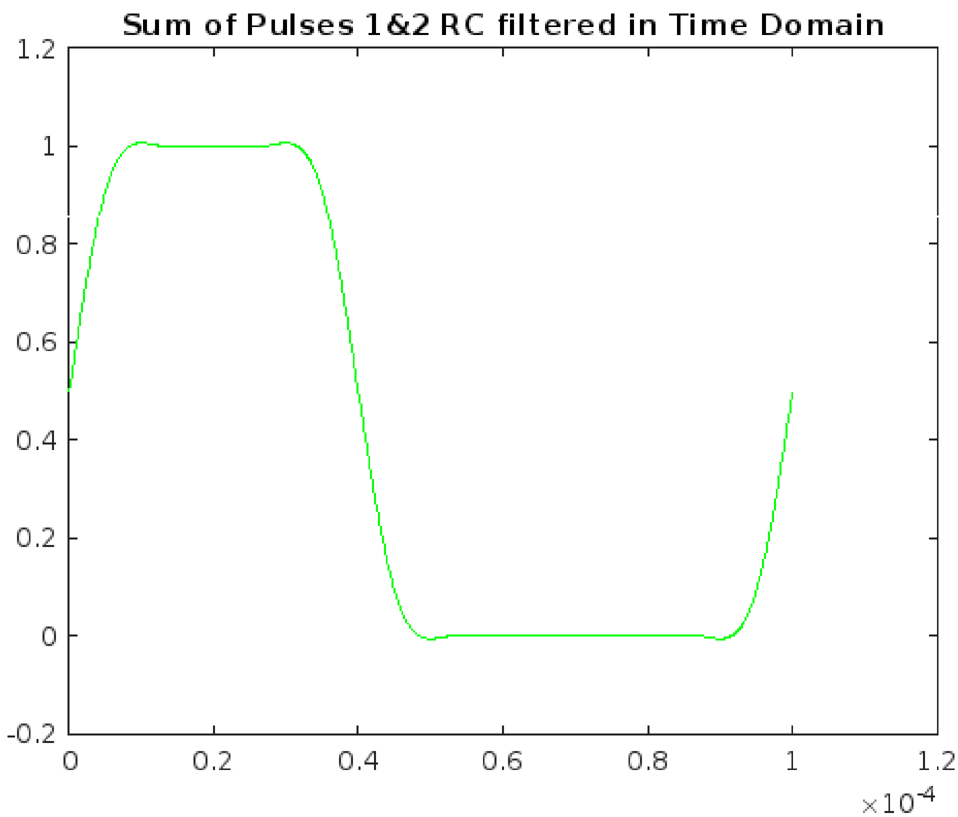

Pulse 1 RC filtered in Freq Domain
8
7
6
5
4
3
2
1
0
-1.5
-1
-0.5
0
0.5
1
1.5
×10^6


Pulse 1 RC filtered in Time Domain
1.2
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
×10^-4


Pulse 2 RC filtered in Freq Domain
8
7
6
5
4
3
2
1
0
-1.5
-1
-0.5
0
0.5
1
1.5
×10^6


Pulse 2 RC filtered in Time Domain
1.2
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
×10^-4

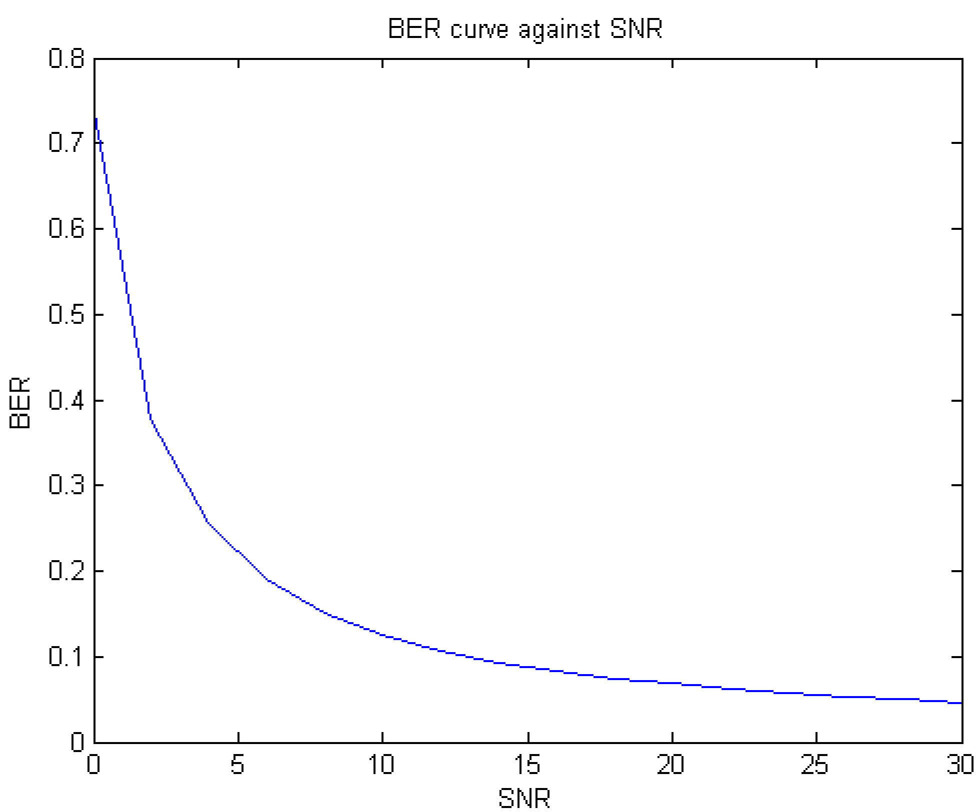




Published with MATLAB® R2022a

```
% Part 2
close all;
clear all;
clc;
% number of signals
L = 100;%Number of channel paths
snr = 0:2:30; %Signal To Noise Ratio
BER = zeros(size(snr)); %Generate BER and initialize to Zeros
numError = zeros(size(snr));
% L rows & 1 col
% h is the channel effect
h = randn(L,1);%H holds path effect
h1 = zeros(L,L);%h1 is row L x column L will be used to hold path effect
% making h1 matrix
index = 1;
%this loop to generate the h1 matrix which holds the channel path effect
for m = 1:L
    j = 1;
    for k = index:-1:1
        h1(m,j) = h(k);
        j = j + 1;
    end
    index = index +1;
end
%Y=HX+N
invH = inv(h1);%Getting H inverse will be used in restoring X
BER_temp = [];
%This loop is used to iterate through each snr
%inside it an inner loop which is used to get a mean of BER
for i = 1 : length(snr)
    for j = 1 : 20
        x = randi([0,1],L,1); %Generating random bits
        % multiply x and h
        xh = h1 * x;
        % add noise to x * h
        y = awgn(xh,snr(i),'measured');%measured
        %is used to measure signal power and apply snr according to it
        % get the received x
        xRec = invH * (y);
        % decide whether the RX sequence is 1 or 0 by comparing with threshold 0.5
        %xRecSeq = xRec > 0.5;
        %xRecSeq = xRecSeq./max(abs(xRecSeq));
        % calclate number of error and bit error rate for each SNR value
        %BPSK was mentioned in PDF
        %BPSK -1,1
        bpsk=[];
        for k=1:L
            if xRec(k)<=0
                bpsk(end+1)=-1;
            else
                bpsk(end+1)=1;
            end
```

```
        end
        numberOfChangedBits=0; %will hold the number of Changed Bits

        %xRecSeq = xRec > 0.5;
        %xRecSeq = xRecSeq./max(abs(xRecSeq));
        % calclate number of error and bit error rate for each SNR value
        for k = 1:L
            if bpsk(k) ~= x(k)
                numberOfChangedBits = numberOfChangedBits + 1;
            end
        end
        BER_temp = [BER_temp numberOfChangedBits]; %Concatenate numberOfChangedBits with BER_temp
    end
    BER_temp = BER_temp./(length(bpsk));
    BER(i) = mean(BER_temp);%setting new BER
end
plot(snr,BER); %Plotting with snr
xlabel('SNR');
ylabel('BER');
title('BER curve against SNR');
```



Published with MATLAB® R2013a

Raised cosine filter rule:

Raised Cosine-Rolloff Nyquist Filtering

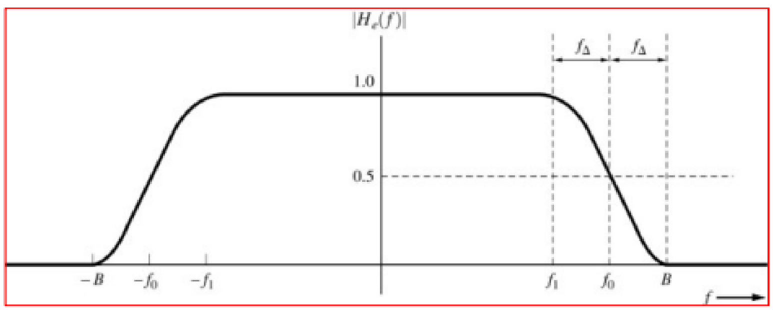
Find $H_e(f)$ such that $H_e(f) = \begin{cases} 1 & |f| < f_1 \\ \frac{1}{2}\left\{1 + \cos\left[\frac{\pi(|f| - f_1)}{2f_\Delta}\right]\right\} & f_1 < |f| < B \\ 0 & |f| > B \end{cases}$

B : absolute bandwidth, $f_\Delta = B - f_o$, $f_1 = f_o - f_\Delta$, f_o : 6 - dB bandwidth of filter

Rolloff factor : $r = \frac{f_\Delta}{f_o}$ $\qquad H_e(f) \leftrightarrow h_e(t) = 2f_o\left(\frac{\sin 2\pi f_o t}{2\pi f_o t}\right)\left[\frac{\cos 2\pi f_\Delta t}{1 - (4f_\Delta t)^2}\right]$

$$B = (1+r)f_o$$

$$D = 2f_o = \frac{2B}{1+r} = \textit{Baud Rate}$$



Part 1: Questions and Answers

- **Explain what is the mathematical criterion that ensures no ISI.**
  Nyquist pulse shaping using raised cosine filter with roll-off factor= 1

Part 2: Question and Answer

- **Knowing Y, H, and the statistics of the AWGN noise (i.e., mean and variance), what is the best way of estimating the transmitted symbols X?**

Calculating (X) by multiplying the inverse of (H) by (Y)

X = inv(H) * Y

Note: For better result estimation, we loop for getting the mean value of BER and concatenate all means