Digital Communication Final Project

By Team No. 14

Name	ID
Ibrahim Sobhy Ibrahim	10
Ahmed Hassan AbdelKadar	14
Ahmed Samir Hassanen	19
Omar Adel Amr Fatoh	129
Omar AbdElaziz Fahmy	131
Yehia Hossam Ali Soliman	259

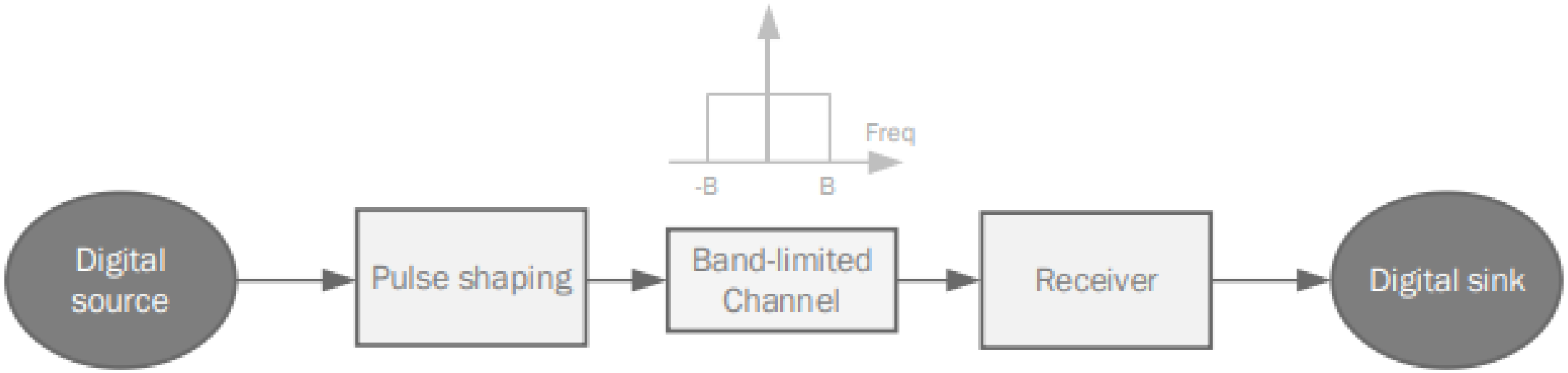


Figure 1 Communication system with a band-limited channel with bandwidth B.

Inter-Symbol Interference due to band-limited channels

In this part, we will investigate another common channel in digital communication systems, which is the band-limited channel. As the name suggests, the channel only allows a limited range of frequency components to pass, while blocking frequency components outside this range. We investigate the simplest of such channels: a channel that has a flat response in the allowable range. Figure 1 shows the system that we will consider

Square Pulse Generator

The first thing we need to show here is the effect of a band-limited flat channel on the square signal. we need to create a band-limited channel such as the one in Figure 1, with a band $\boldsymbol{B = 100\ kHz}$. Then generate a square pulse of duration $\boldsymbol{T = 2/B}$, pass it through the filter, and then look at the output. we need here to show the signal before and after the channel, both in time and in frequency

$$B = 100\,KHz$$

$$Tsq = \frac{2}{B} = \frac{2}{100K} = 20\mu s$$

$$Ns = \frac{Tsq}{Ts} = Tsq * Fs$$

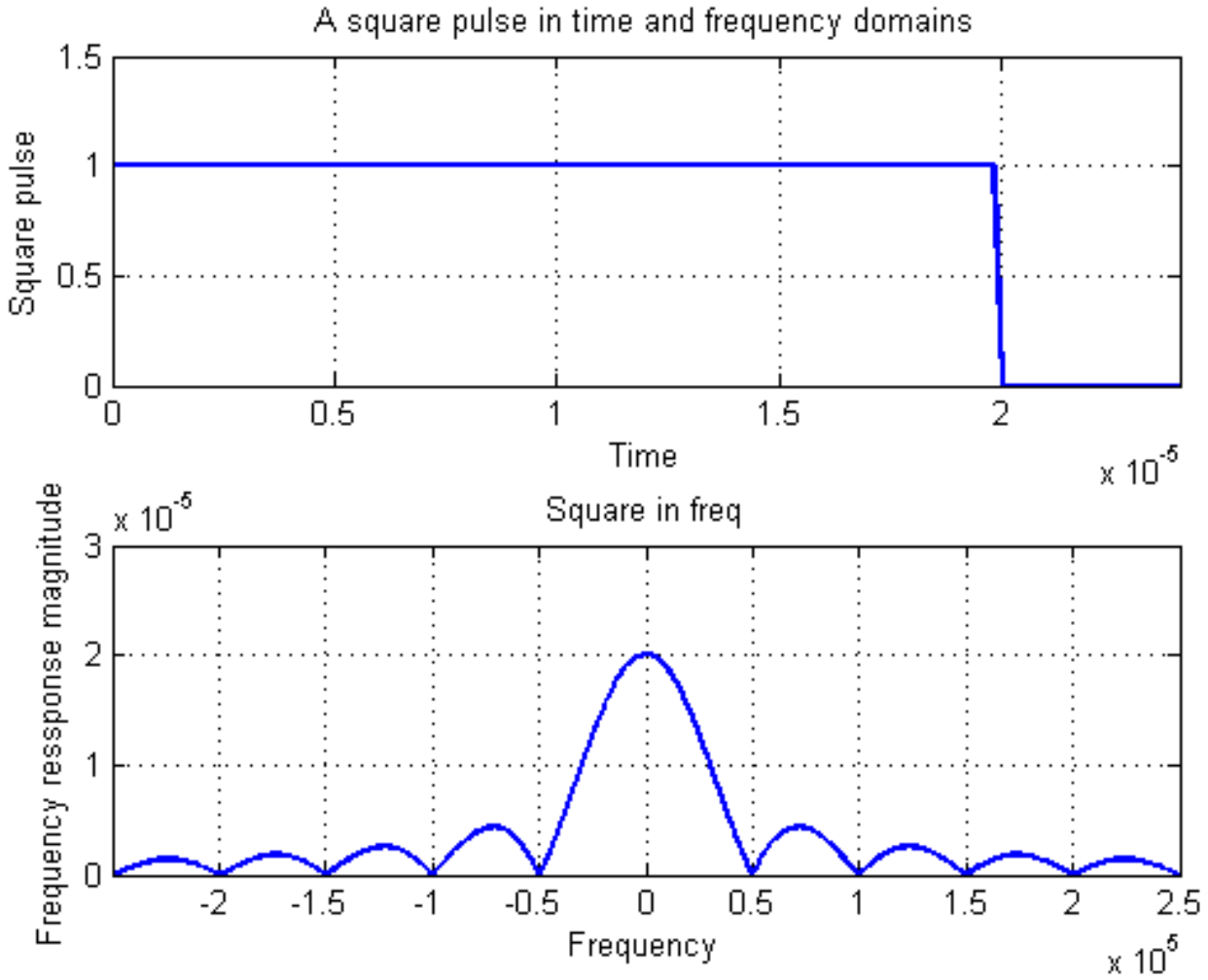
$$Fs = \frac{Ns}{Tsq} = \frac{Ns}{20\mu} = (50K)\,Ns$$

```
%%%%%%%%%%%part 1 %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fs = 5e6;                          % Sampling rate (samples per sec)
Ts = 1/fs;                         % Sampling time
N = 102400 - 1;                    % Total number of samples
t_axis = (0:N-1)*Ts;               % Time axis (the same during the entire experiment)
t_axis_sh = ((-(N-1)/2):((N-1)/2))*Ts;
f_axis = -fs/2:fs/N:fs/2-1/N;      % Frequency axis (the same during the entire experiment)

% Generate one square pulse with the following parameters
Energy_per_bit = 50.5;               % The total energy of all samples constituting the square pulse
B = 100*10^3;
T_sq = 2/B;
N_sq = round(T_sq/Ts);             %N_sq = 100

%generating one pulse
x_bits = 1;
x_square = GenerateSquarePulses(t_axis,T_sq,Energy_per_bit,fs,x_bits,1);
```

Square Pulse Generator Code

```
figure
subplot(2,1,1)

plot(t_axis,x_square,'linewidth',2)
grid on
xlim([0 T_sq*1.2])
xlabel('Time','linewidth',2)
ylabel('Square pulse','linewidth',2)

X_squareF = (1/fs)*abs(fftshift(fft(x_square)));


subplot(2,1,2)
plot(f_axis,abs(X_squareF),'linewidth',2)
title('Square in freq','linewidth',10)
grid on
xlim([-1/T_sq 1/T_sq]*5)
xlabel('Frequency','linewidth',2)
ylabel('Frequency ressponse magnitude','linewidth',2)
subplot(2,1,1)
title('A square pulse in time and frequency domains','linewidth',10)
```

One Square Pulse Output Code



Square Pulse Generation Output

```
function x_square = GenerateSquarePulses(t_axis,T_sq,E_bit,fs,x_bits,type)

Ts = 1/fs;
N = length(t_axis);


N_sq = round(T_sq/Ts);
one_square = zeros(1,N_sq);
if type == 1
    A = sqrt(2*E_bit / N_sq);
    one_square = A + one_square;
else
    alpha = 0.5;
    W = 1/T_sq;
    n = 102400 - 1;
    t_axis_sh = ((-(n-1)/2):((n-1)/2))*Ts;
    one_raised_cos = sinc(2*W*t_axis_sh).*cos(2*pi*alpha*W*t_axis_sh)./(1-(4*alpha*W*t_axis).^2);
end
x_square = zeros(1, N);

for i = 1:length(x_bits)
    if type == 1
        if x_bits(i) == 1
            x_square((i-1)*N_sq +1:i*N_sq) = x_square((i-1)*N_sq +1:i*N_sq) + one_square;
        end
    else
        if x_bits(i) == 1
            x_square = x_square + circshift(one_raised_cos' , N_sq*(i-1))';
        else
            x_square = x_square - circshift(one_raised_cos' , N_sq*(i-1))';
        end
    end
end

end
```

Generate Square Pulse Function

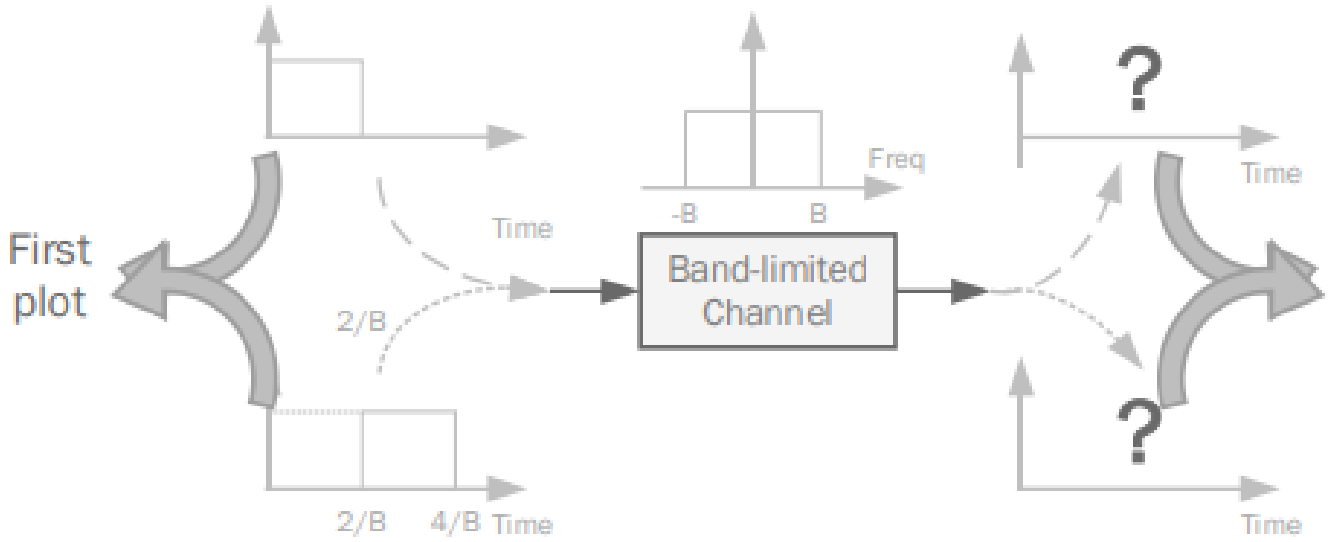


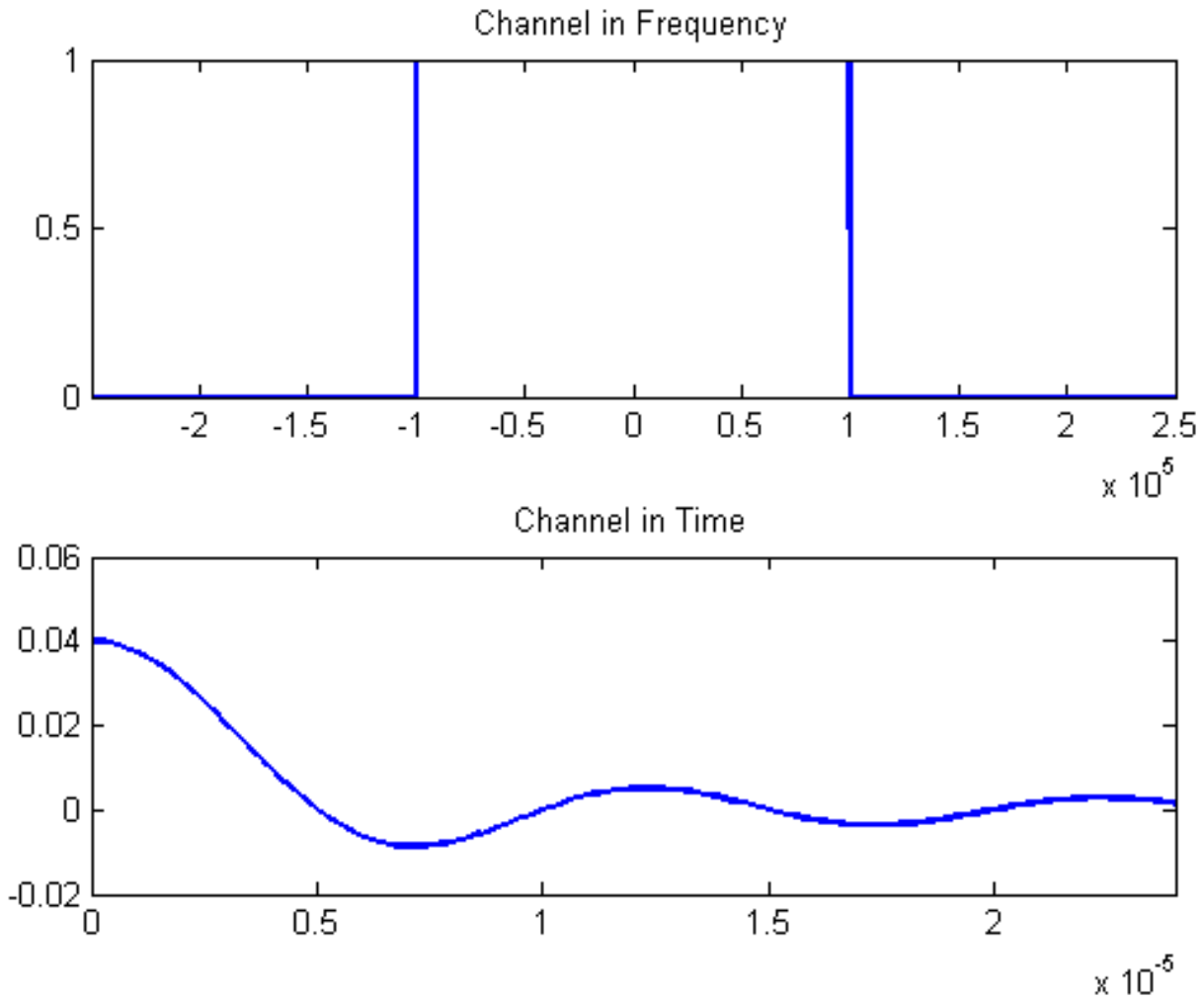
Figure 2 How to generate the two plots described above.

Generate two plots

The second thing we need to show is the effect of two consecutive square signals as they pass through the channel. Consider the same channel and the same square pulse duration as before. The plots we need to show here are in time domain only. Namely, 1. Show two plots of the first square pulse: one before it passes through the channel, and one after. 2. On top of the two previous plots, show similar plots for the second square pulse. Plot this pulse in the two plots using a different color, so that the shapes of the two pulses are distinguishable on the plots. 3. Note that we will have to pass the two squares separately, i.e., we cannot create the two pulses together in the same vector and pass that into the channel. If we do it this way, we won't be able to clearly distinguish the two pulses. The procedure that we need to follow to generate the plots required above are shown in Figure 2

```
%creating the channel


channelf = rectpuls(f_axis , 2*B);
channel = ifft(ifftshift(channelf));
channel2 = fft(fftshift(channelf));
figure
subplot(2,1,1)
plot(f_axis,abs(channelf),'linewidth',2)
title('Channel in Frequency','linewidth',10)
xlim([-1/T_sq 1/T_sq]*5)
subplot(2,1,2)
plot(t_axis,channel,'linewidth',2)
title('Channel in Time','linewidth',10)
xlim([0 T_sq*1.2])
```

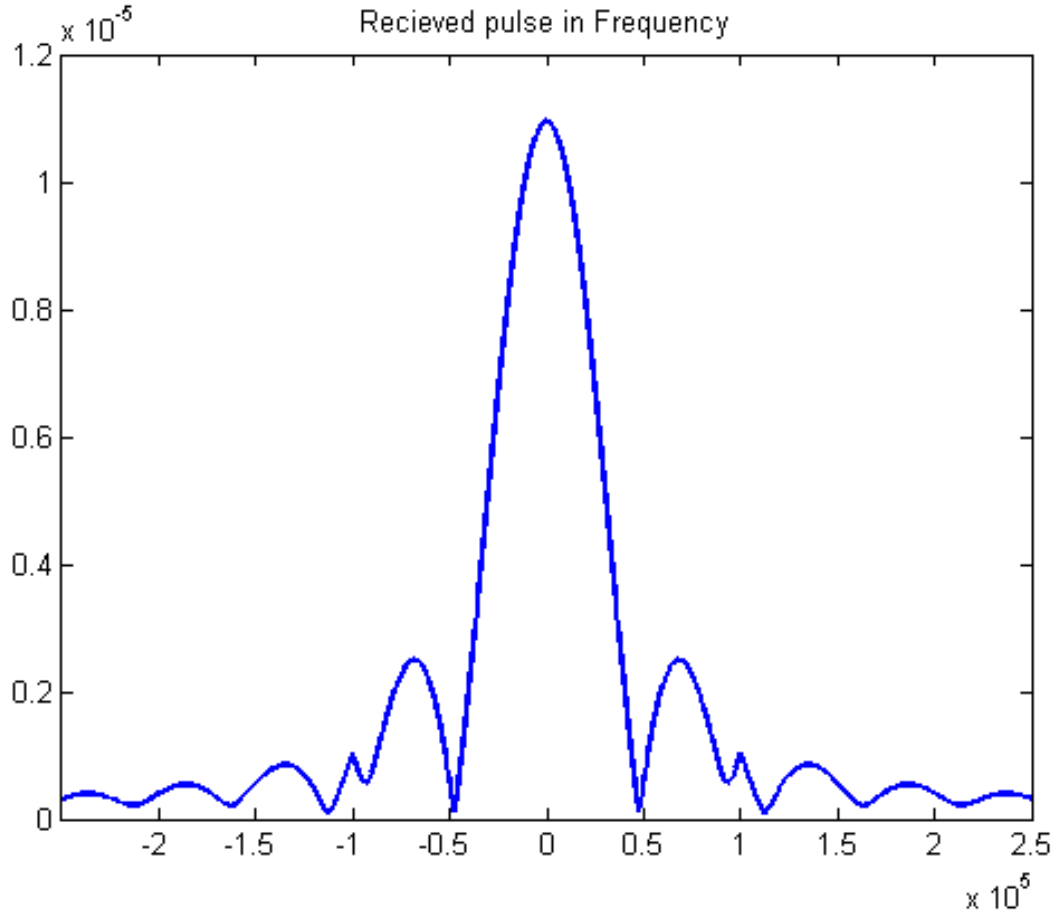
Channel Code

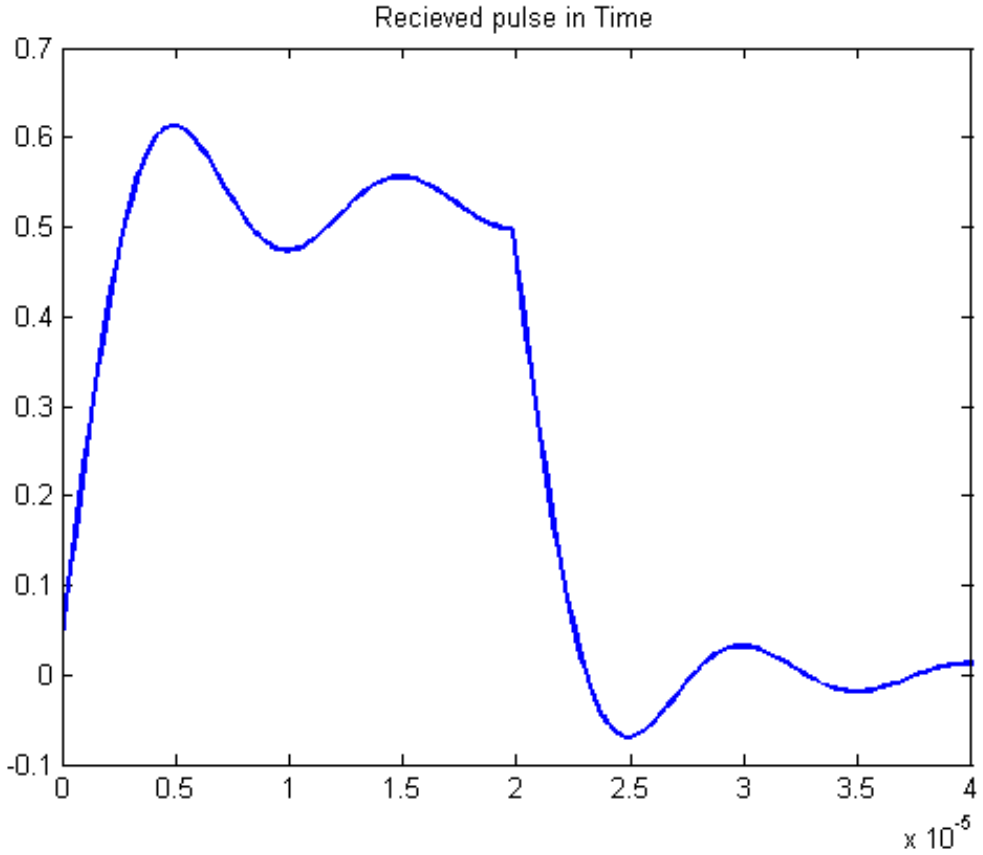


Channel Output

```
%passing the pulse to the channel


temp = conv(x_square , channel2);
y = temp(:,1:length(t_axis));
figure
grid on
plot(t_axis,y,'linewidth',2)
title('Recieved pulse in Time','linewidth',10)
xlim([0 T_sq*2])
yf = (1/fs)*abs(fftshift(fft(y)));
figure
plot(f_axis,yf,'linewidth',2)
title('Recieved pulse in Frequency','linewidth',10)
xlim([-1/T_sq 1/T_sq]*5)
```

Passing Pulse to Channel

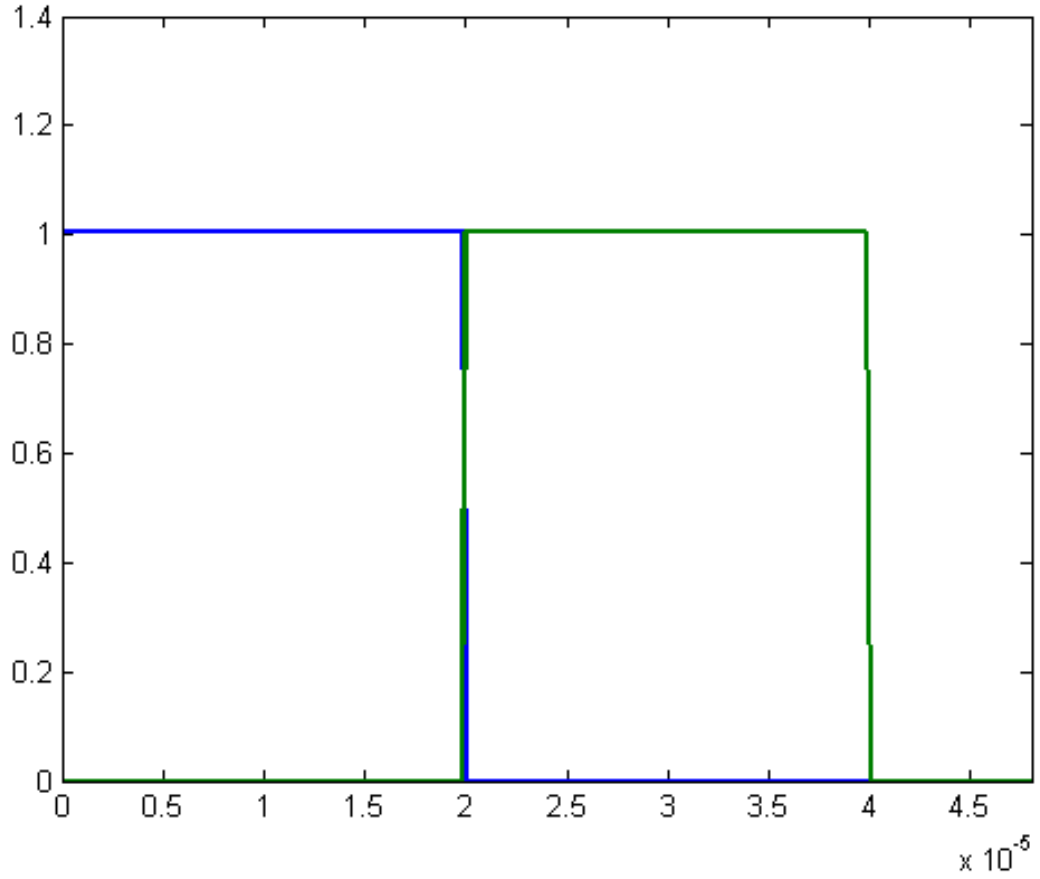


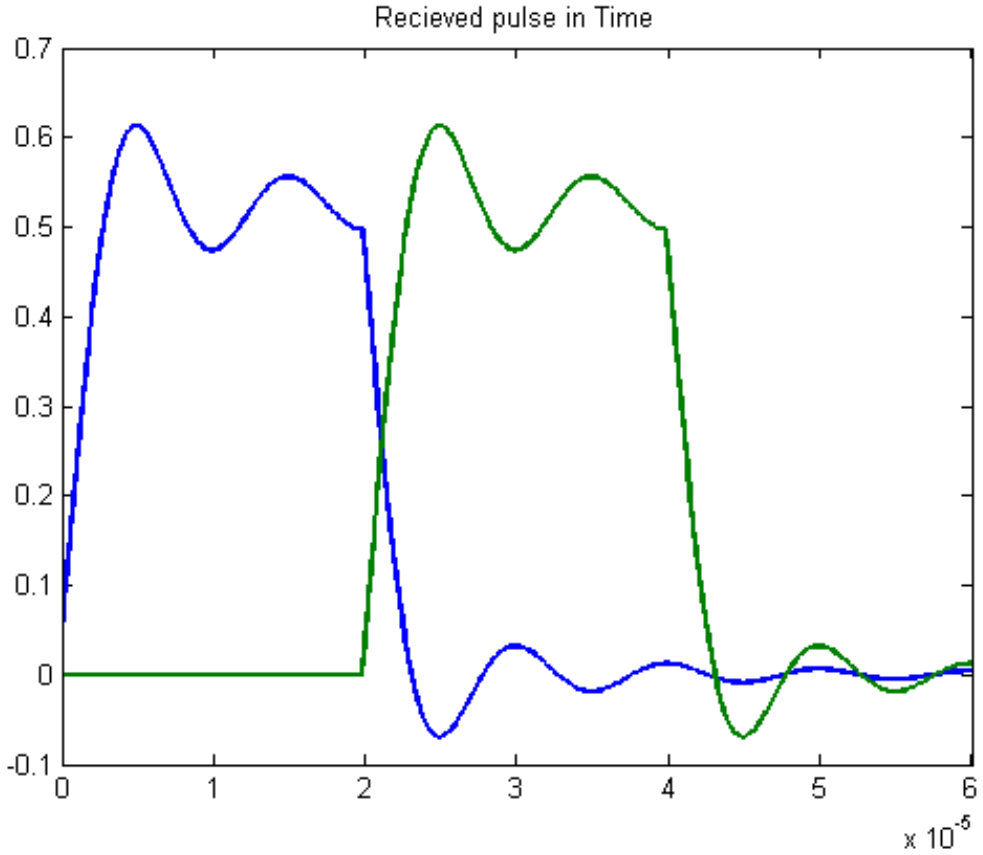


Passing Pulse to Channel

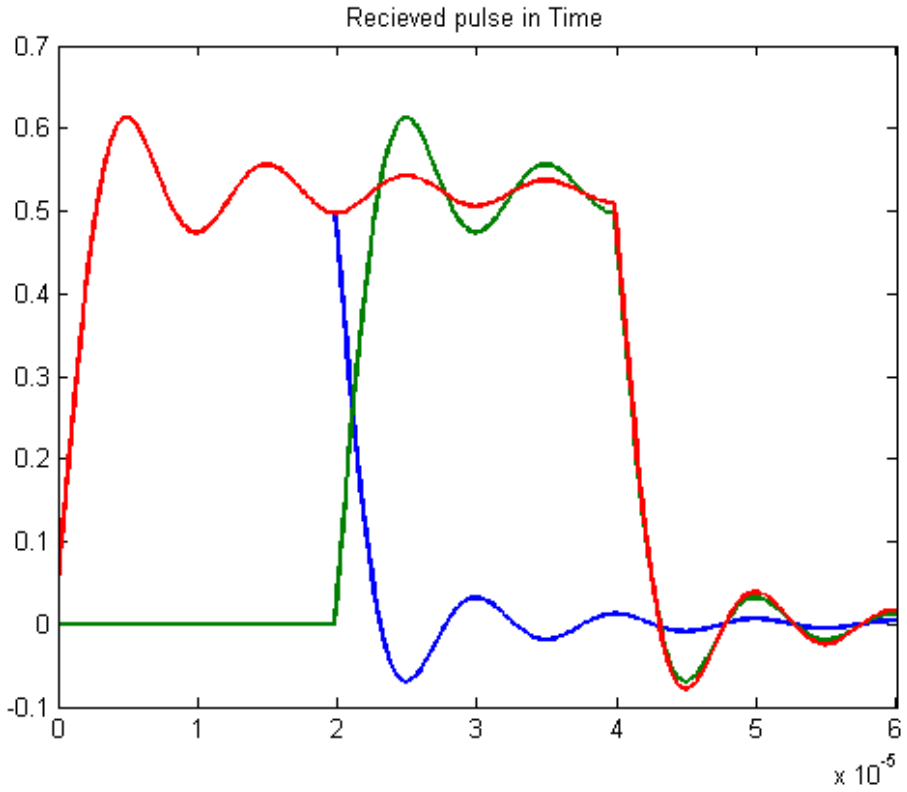
```
%creating two separate pulses

x_bits = 1;
x_square1 = GenerateSquarePulses(t_axis,T_sq,Energy_per_bit,fs,x_bits,1);

x_bits = [0 1];
x_square2 = GenerateSquarePulses(t_axis,T_sq,Energy_per_bit,fs,x_bits,1);
figure
plot(t_axis,x_square1,t_axis,x_square2 , 'linewidth',2)
xlim([0 T_sq*2.4])

%passing the two pulses to the channel

temp = conv(x_square1 , channel);
y1 = temp(:,1:length(t_axis));

temp = conv(x_square2 , channel);
y2 = temp(:,1:length(t_axis));

figure
grid on
plot(t_axis,y1,t_axis,y2,'linewidth',2)
title('Recieved pulse in Time','linewidth',10)
xlim([0 T_sq*3])
```

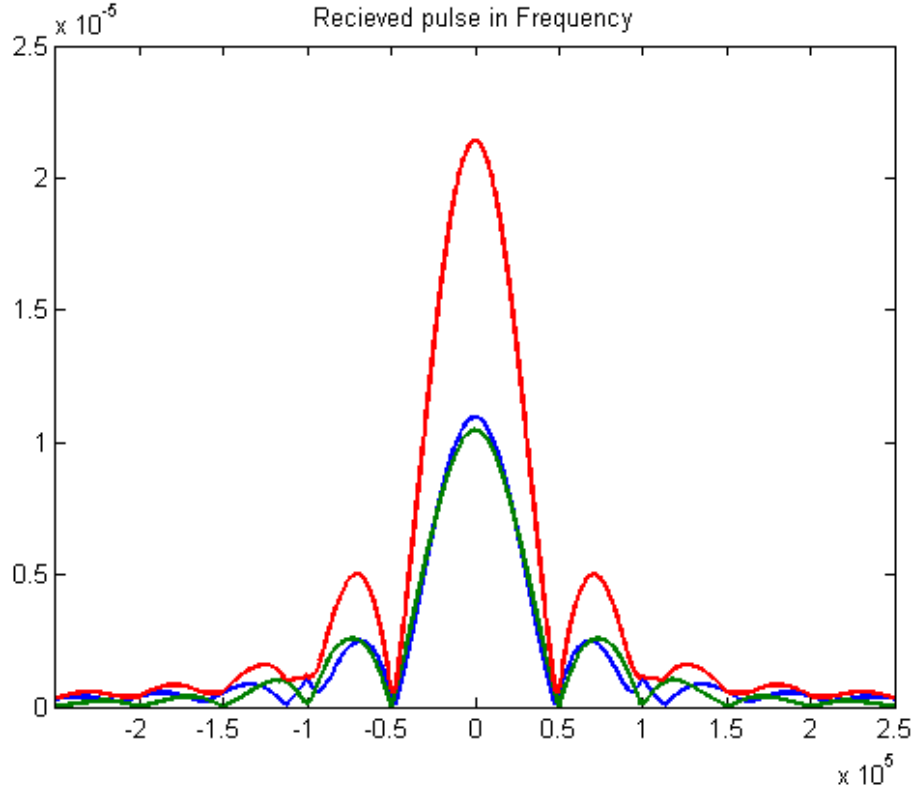
Generating Two separate Pulses





Generating Two separated Pulse





Inter-Symbol Interference (ISI)

Problem of ISI

As we can see, the square signal is no longer a square signal coming out of the channel. In fact, the shape of the square signal out of the channel has “leaked” outside of the duration $T = 2/B$ that was intended at the beginning. If there are multiple square signals after each other (one square signal for each bit), these leaked parts will interfere with the signals of other bits. This phenomenon is called Inter-Symbol Interference (ISI).

ISI can negatively affect the detection performance of multiple consecutive signals. To combat the effect of ISI in band-limited channels, one cannot use square pulses anymore. Instead, there are other pulse shapes that are better suited for such channels. We need to investigate such solutions.

So, we use raised cosine to solve the problem of ISI

Raised Cosine Filter

The raised-cosine filter is an implementation of a low-pass Nyquist filter.

It can be produced in terms :

A) Time Domain :

$$p(t) = \operatorname{sinc}(2Wt)\frac{\cos(2\pi\alpha Wt)}{1-16\alpha^2W^2t^2}$$

B) Frequency Domain:

$$P(f) = \begin{cases} \frac{1}{2W}, & 0 \le |f| < f_1 \\ \frac{1}{4W}\left\{1+\cos\left[\frac{\pi}{2W\alpha}(|f|-f_1)\right]\right\}, & f_1 \le |f| < 2W-f_1 \\ 0, & |f| \ge 2W-f_1 \end{cases}$$

Parameters of Raised Cosine

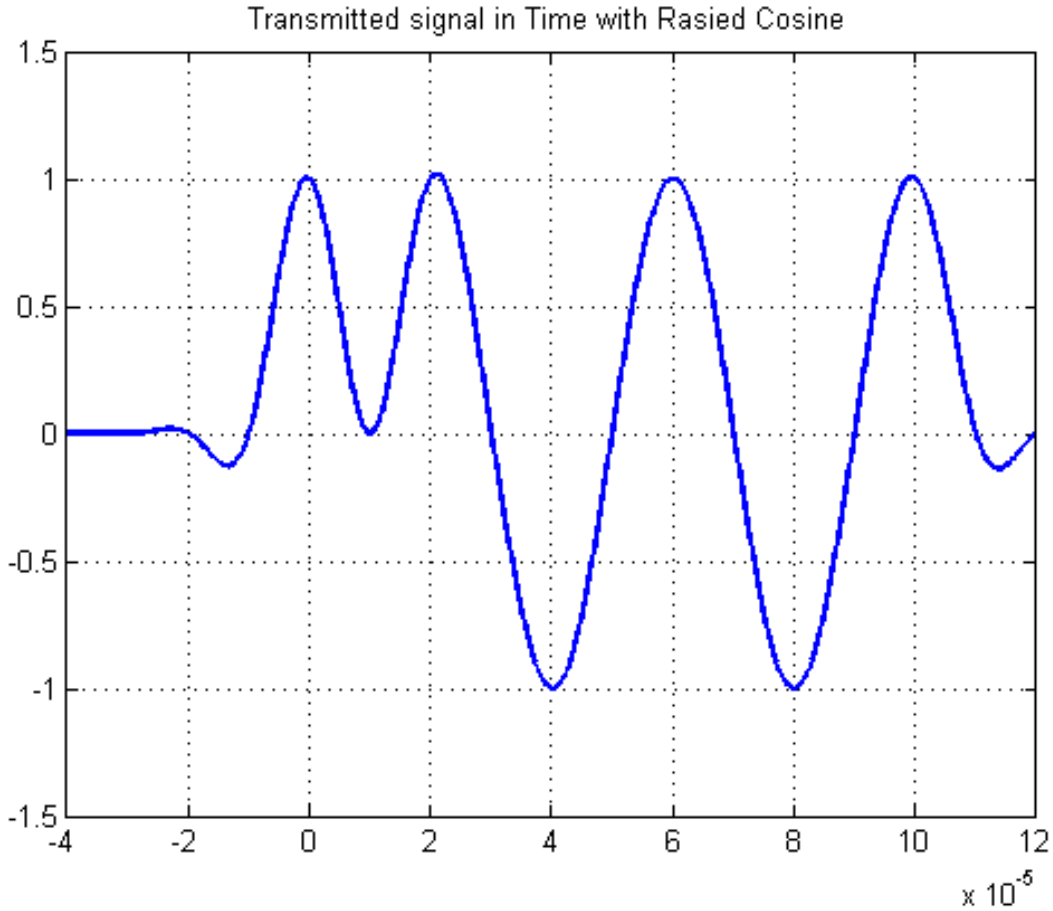
$$Bandwidth:\ W = \frac{Rb}{2} = 2Tb$$

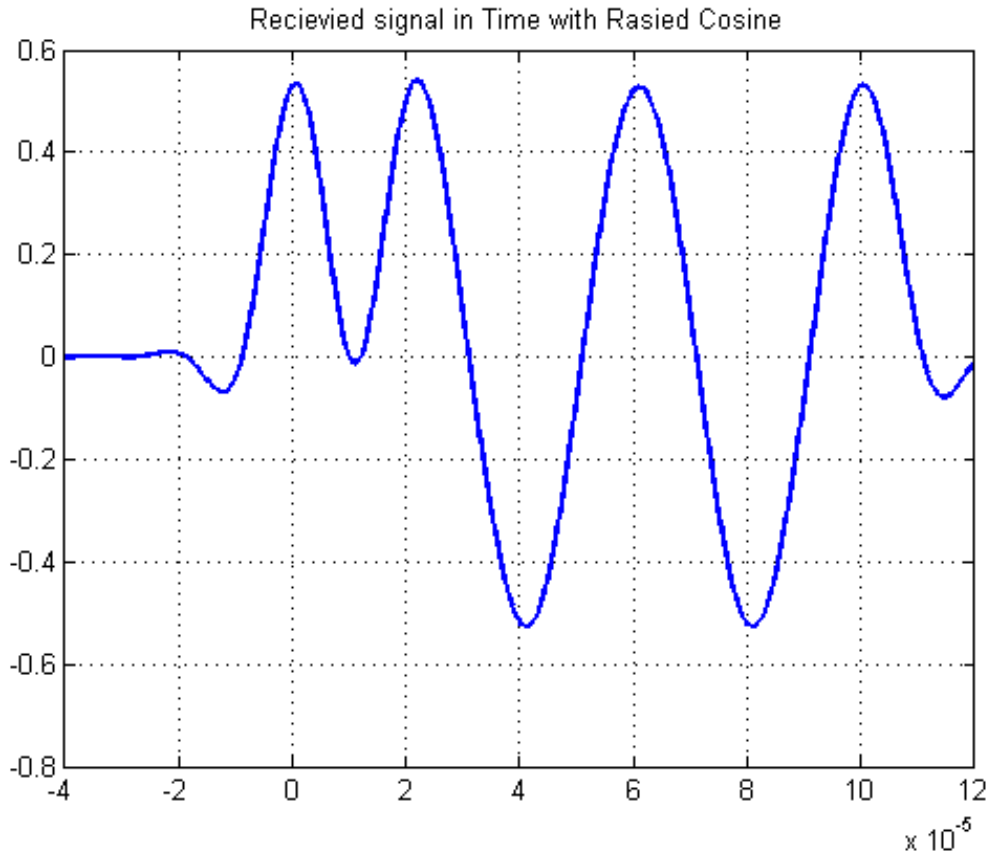
$$Roll\ of\ factor\ \alpha = 1 - \frac{f}{W}$$

The main parameter of a raised cosine filter is its roll-off factor, which indirectly specifies the bandwidth of the filter. Ideal raised cosine filters have an infinite number of taps. Therefore, practical raised cosine filters are windowed.

```
function x_square = GenerateSquarePulses(t_axis,T_sq,E_bit,fs,x_bits,type)

Ts = 1/fs;
N = length(t_axis);


N_sq = round(T_sq/Ts);
one_square = zeros(1,N_sq);
if type == 1
    A = sqrt(2*E_bit / N_sq);
    one_square = A + one_square;
else
    alpha = 0.5;
    W = 1/T_sq;
    n = 102400 - 1;
    t_axis_sh = ((-(n-1)/2):((n-1)/2))*Ts;
    one_raised_cos = sinc(2*W*t_axis_sh).*cos(2*pi*alpha*W*t_axis_sh)./(1-(4*alpha*W*t_axis).^2);
end
x_square = zeros(1, N);

for i = 1:length(x_bits)
    if type == 1
        if x_bits(i) == 1
            x_square((i-1)*N_sq +1:i*N_sq) = x_square((i-1)*N_sq +1:i*N_sq) + one_square;
        end
    else
        if x_bits(i) == 1
            x_square = x_square + circshift(one_raised_cos' , N_sq*(i-1))';
        else
            x_square = x_square - circshift(one_raised_cos' , N_sq*(i-1))';
        end
    end
end

end
```

Generate Square Pulse Function





Raised Cosine Filter Output

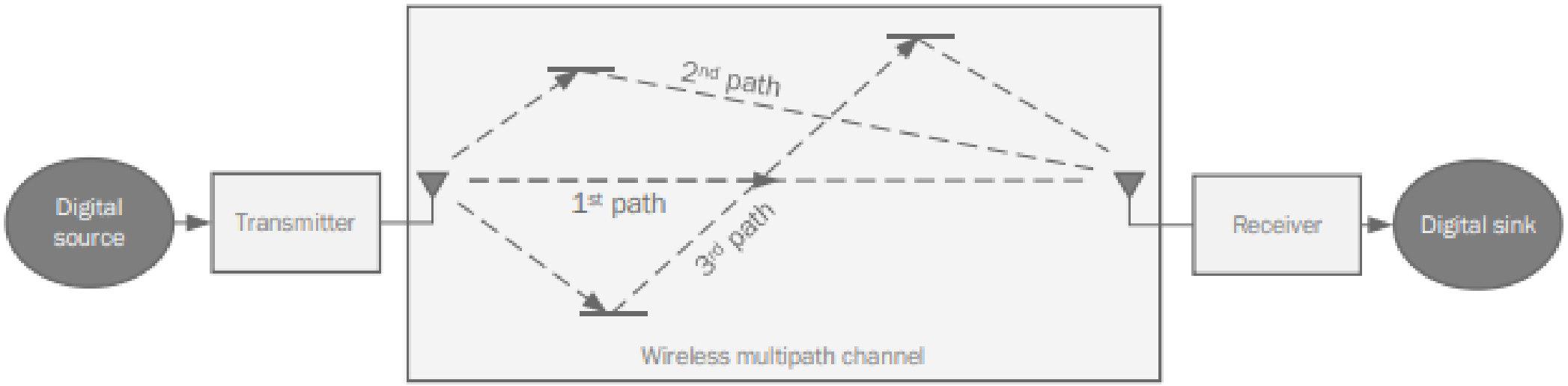


Figure 3 Signal propagation in a wireless multipath channel.

Inter-Symbol Interference due to multi-path channels

In this part, we consider another form of ISI that happens in channels that we face in wireless communication systems. The channel considered here is referred to as the multipath channel. To understand the effect of this channel, we refer to Figure 3.

Multi Path Channel

In wireless channels, signals are transmitted via electromagnetic waves which propagate through the air until it reaches the receiver. However, the nature of electromagnetic waves allow that multiple copies of the signal would travel around and reach the receiver at different times, such as is shown in Figure 3. This effect is what is known as the multipath effect. As is shown in the figure, a symbol transmitted by the transmitter would traverse multiple paths until it reaches the receiver. Therefore, the receiver is expected to receiver multiple copies of the same transmitted signal. Each of these copies would arrive at a different time (based on how long the path that the signal travelled through is) and with a different magnitude (based on how much attenuation that the signal suffered from during the transmission across the path).

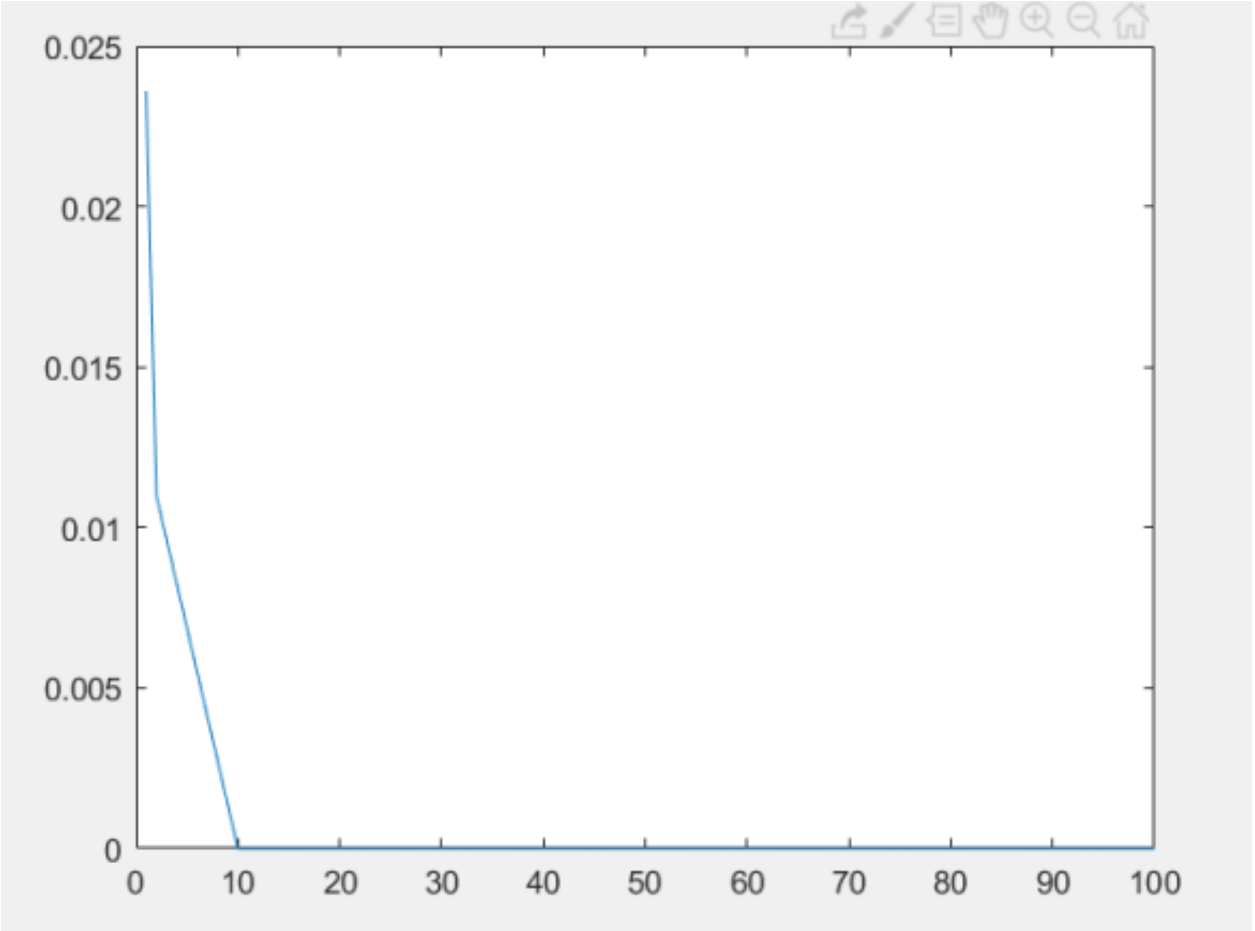
Part 2 Question

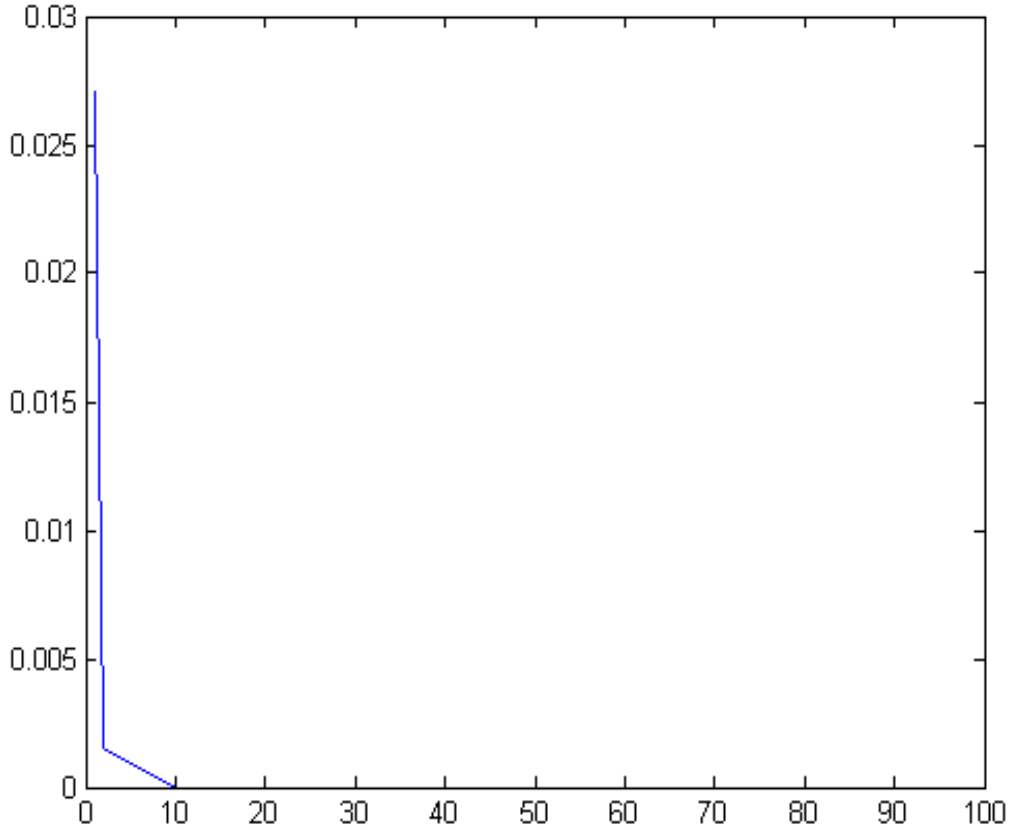
Knowing *Y*, *H*, and the statistics of the AWGN noise (i.e., mean and variance), what is the best way of estimating the transmitted symbols *X*?

The goal here is to give an answer to the previous question. Note that there are several techniques to solve this equation in the literature. We can investigate one or more of these solutions.

```
L=1000;
N=1;
h = randn(L,N) + 1i*randn(L,N);
power_profile = exp(-0.5*[0:L-1])';
power_profile = repmat(power_profile,1,N);
h = abs(h).*power_profile;

H = zeros(L,L);
last_ind = 1;
z = 1;
for n = 1:L
    for k = last_ind:-1:1
        H(n,z) = h(k);
        z = z +1;
    end
    z = 1;
    last_ind = last_ind + 1;
end
H_inv = inv(H);
BER_temp = [];
Eb = 1;
No = [1 0.5 0.1 0.05 0.01 0.005 0.001];
BER = [];
for nn = No
    N = sqrt(nn/2)*randn(L,1);
    for i = 1:10          %we will calculate the BER 10 times for each noise
        valid_vals = setdiff(-1:1, 0);
        x = valid_vals( randi(length(valid_vals), L, 1) ); %stream of bits transmitted
```

```
        y = H*x' + N;           %recieved

        x_rec = H_inv*y;        %recieved bits plus noise
        D = zeros(size(x_rec));
        for k = 1:L
            if x_rec(k) <= 0
                D(k)= -1 ;
            else
                D(k) = 1;
            end
        end

        n = 0;
        for k = 1:L
            if D(k) ~= x(k)
                n = n + 1;
            end
        end


        BER_temp = [BER_temp n/L];
    end
    BER = [BER mean(BER_temp)];
    BER_temp = [];
end

plot(Eb./No , BER)
xlim([0 100])
```





BER Calculations