Faculty of Enigeering

Cairo University

Digital Communications Project 2

Matched Filters, Correlators, ISI, and Raised cosine filters

Presented for **ELC 3070** MATLAB Project

Presented to:

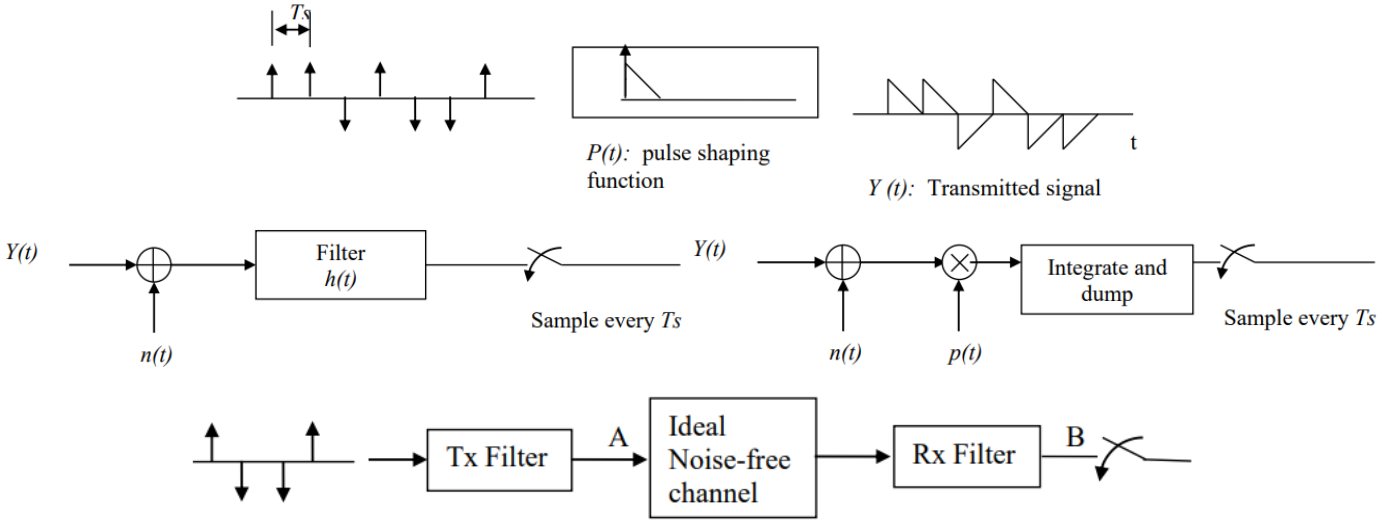
Dr. Mohamed Nafea

T.A: Mohamed Khaled

(3rd Year Electronics and Electrical Communication Engineers)	
مجدي أحمد عباس عبد الحميد الابرق	Sec: 3 / I.D: 9210899 / BN: 36
كريم ايمن محمد فخر الدين محمد علي عفيفي	Sec: 3 / I.D: 9210836 / BN: 26
مصطفى ابراهيم محمد ابراهيم	Sec: 4 / I.D: 9211158 / BN: 19
يحيى خالد عبد الفتاح محمد	Sec: 4 / I.D: 9211362 / BN: 40

Role of each member:

Each one of us created his own code, and in one meeting we came together on the best version by merging the 4 codes and wrote the documentation



REQUIREMENT 1

Matched Filters and Correlators in Noise Free Environment

The construction of the Matched filter and the Correlator within a free noise environment, Before Noise is Added. Initially, a pulse signal $p(t)$ with a symbol duration $T_s = 1$ second is generated and sampled 5 times with ($T_{Sampling} = 200$ milliseconds). Additionally, the pulse signal is normalized to achieve unity energy ($E_P = 1$). Ten random bits are then generated to form the impulse train. To enable convolution between the pulse signal $p(t)$ and the impulse train, ensuring that each symbol in the impulse train take five samples rather than one, upsampling is employed by inserting four zeros after each symbol in the impulse train. Applying Convolution then resultant signal as the output from the transmitter (T_x) output, subsequently passing through the channel to reach the Receiver, under the assumption of a noise-free environment.

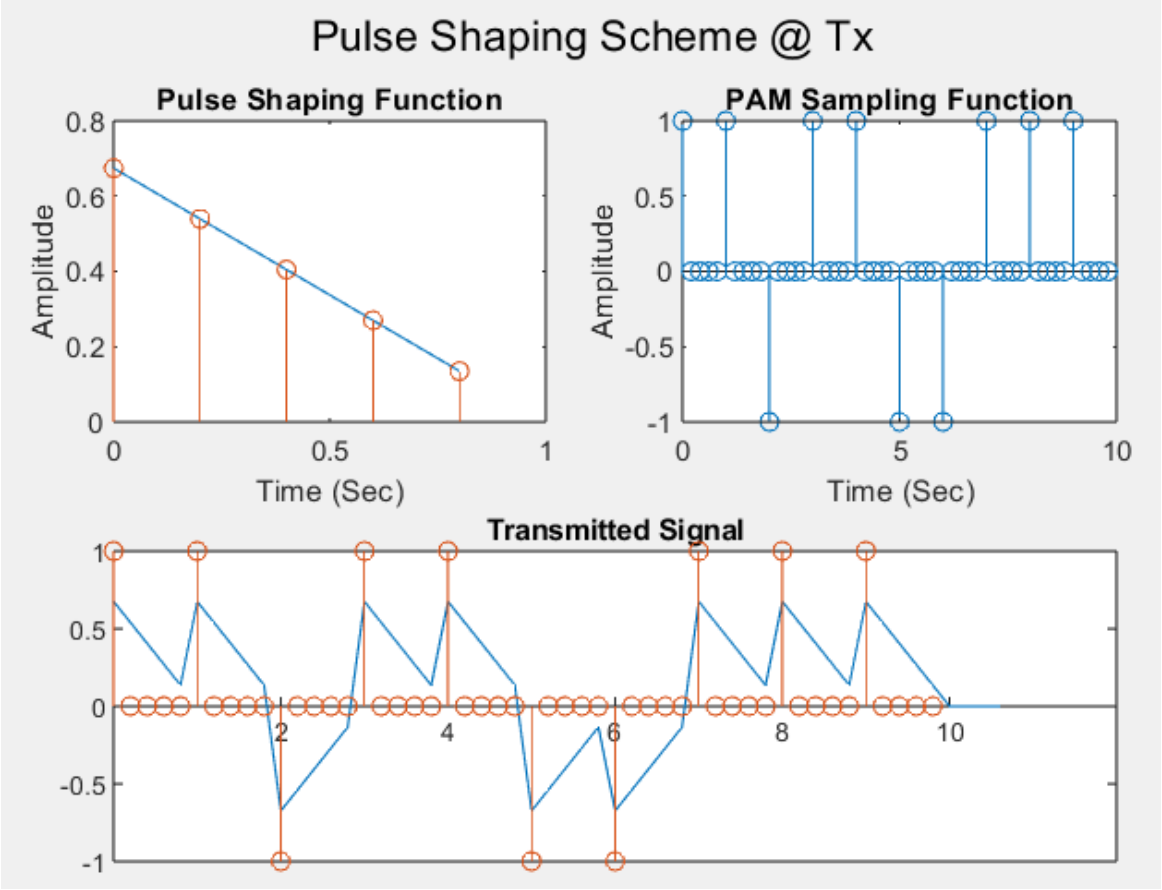


Figure 1: Pulse Shaping Scheme at Tx (Pulse Shaper, PAM Sampling & Tx Output)

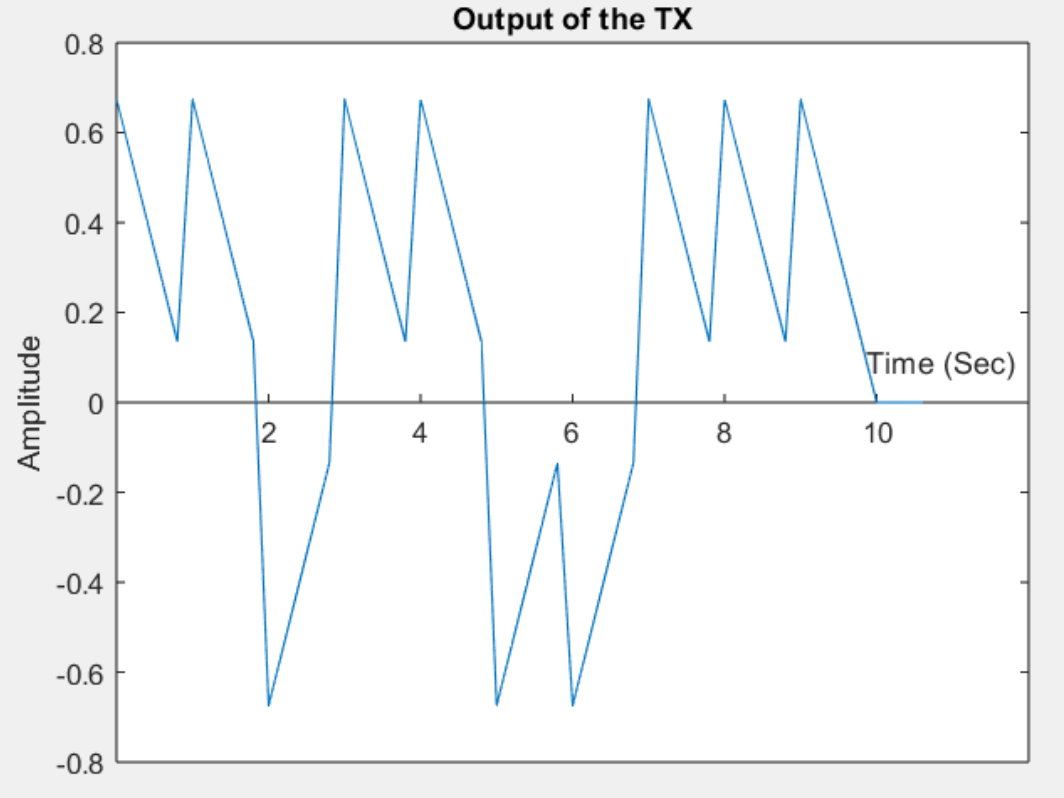


Figure 2: Output of the Tx

PART A

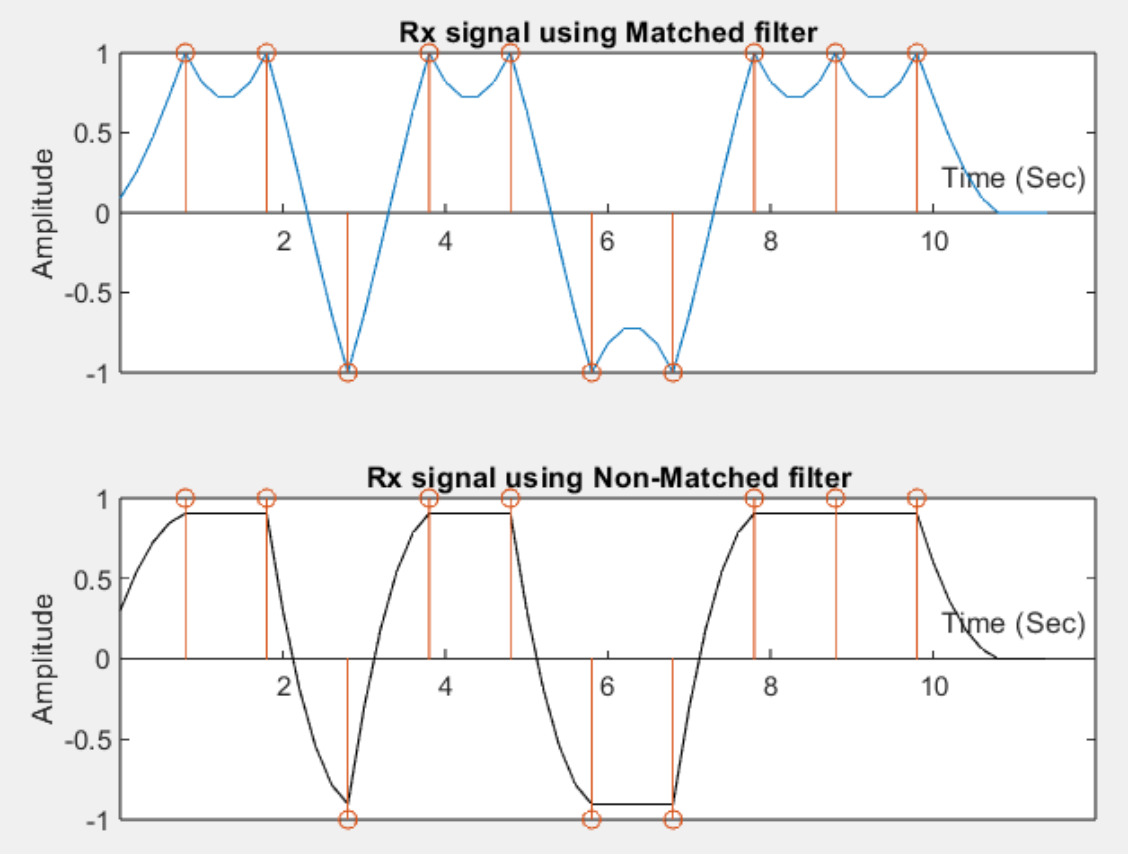


Figure 3: Rx Signal Using Matched and Non-Matched Filters along with sampling instants.

PROCEDURES

We need to compare the output of the Matched Filter with The Non-Matched hold filter (characterized by the same pulse period Ts).

However, to ensure a fair comparison between the two filters, it is necessary to normalize the impulse response beforehand.

To normalize any discrete signal, we initially compute its energy using the equation:

$$E_s = \sum_{n=-\infty}^{n=\infty} (x[n])^2 \ , \qquad X[n]_{normalized} = \frac{x[n]}{\sqrt{E_s}}$$

COMMENTS

As shown in Fig (3,4):

1) The matched filter has its peak amplitude conformed with the sampling instant which reflects preserving its full energy content since it is matched to the Rx filter. Hence, The SNR is maximized for such case.
2) On the other hand, the hold filter output does not have the full amplitude at the sampling instant → The SNR is degraded.

PART B

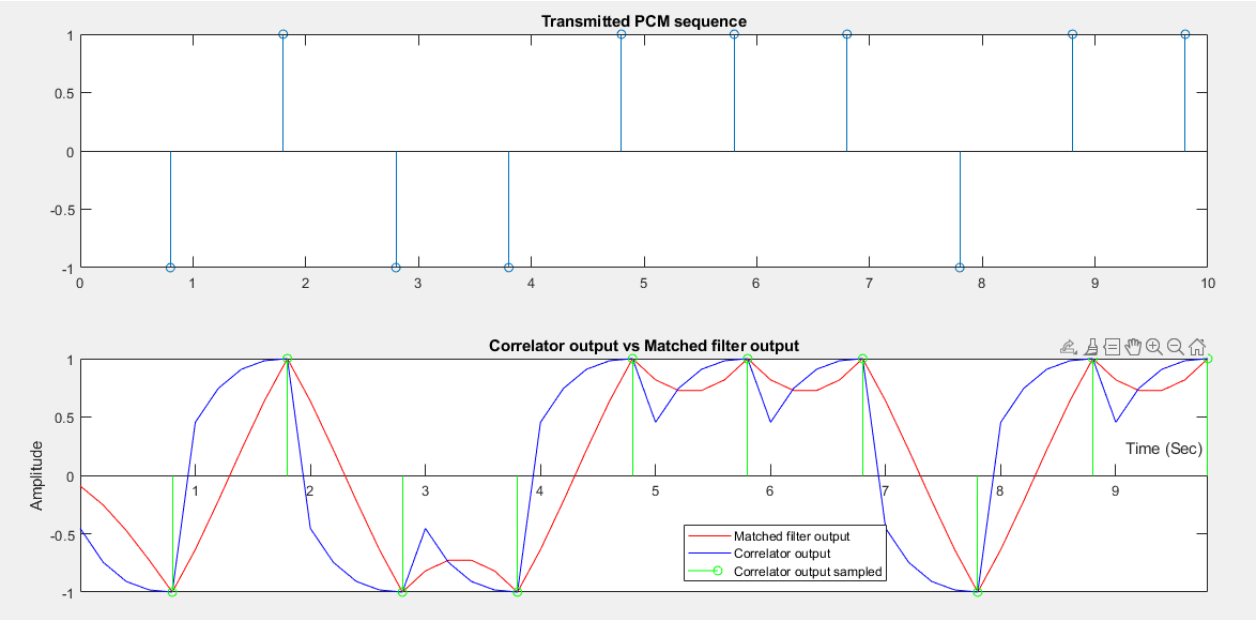
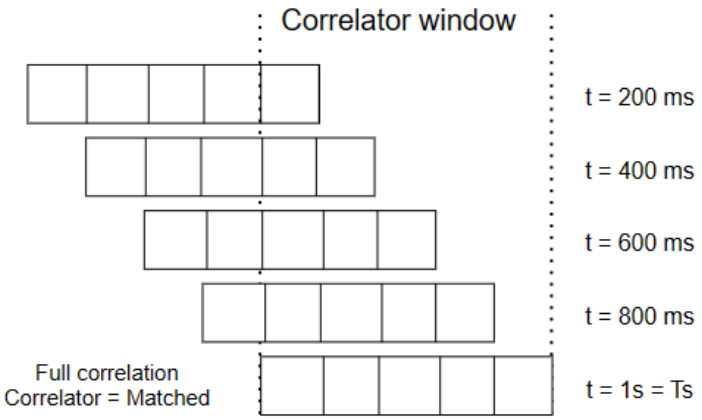


Figure 4: Correlator Output vs Matched Filter Output

COMMENT

As shown in Fig (4), An alternative way to show the receiver model in the Digital systems is the concept of the "Correlator" which mainly runs on the concept of "Integrate & dump". It is shown that the correlator output adapts to the matched filter output & matches it @ the sampling instant since it is the only instant that the correlator window is fully correlated with pulse-shaping function as elaborated below.



REQUIREMENT 2

Noise Analysis

In this section, we need to discuss the difference between the matched output & the non-matched output with respect to the noise margin & the potential of error in receiving the bits.

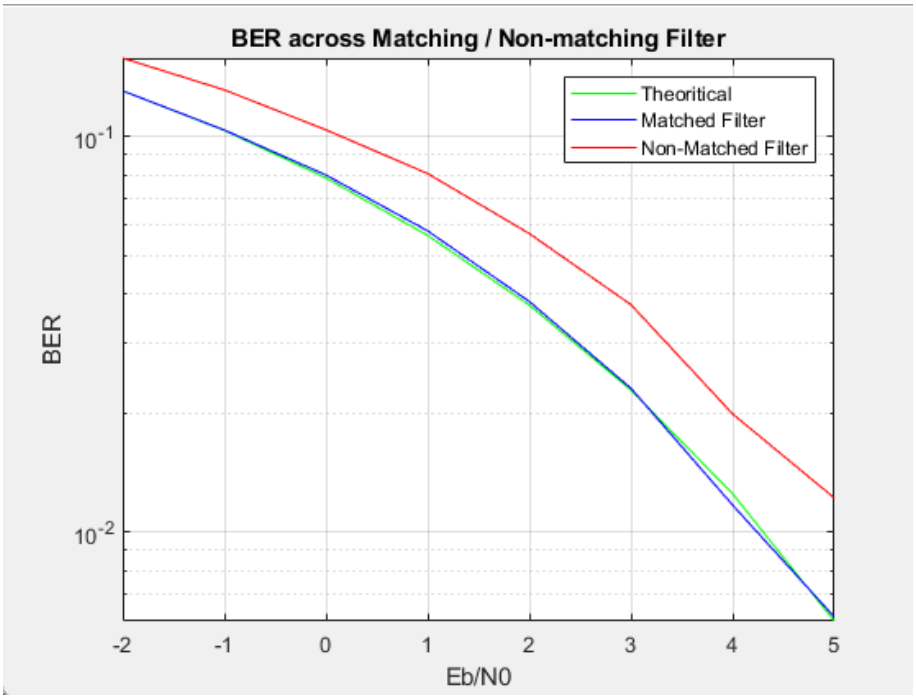


Figure 5: BER Across Matching / Non - Matching Filter

COMMENTS

As per discussed in the previous section, the output of the matched filter preserves its energy content with respect to the output of the non-matched filter which highlights the fact that the noise energy required to change the bit constellation for the matched is higher than that of the non-matched output.

Consequently, for the same bit energy, the probability of error of the receiver estimator is minimized for the matched output case.

SUMMARY

From sections (1/2), we can conclude that using a matched filter maximizes SNR at sampling instants for optimal detection performance, ideal for coherent detection with accurate signal knowledge. Its impulse response matches the pulse shape of the transmitted signal.

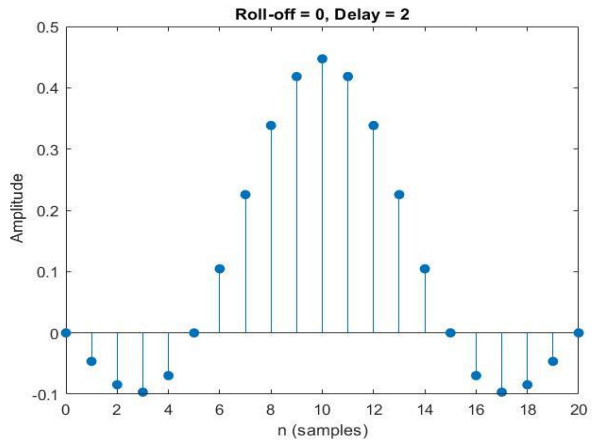
In contrast, non-matched filters may offer simpler implementation and reduced sensitivity to timing errors but result in lower SNR and detection performance.

REQUIREMENT 3

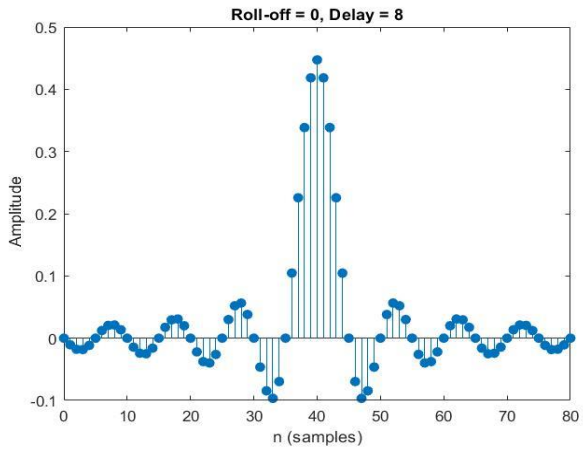
ISI and Raised Cosine

Transmitter Raised Cosine Filters

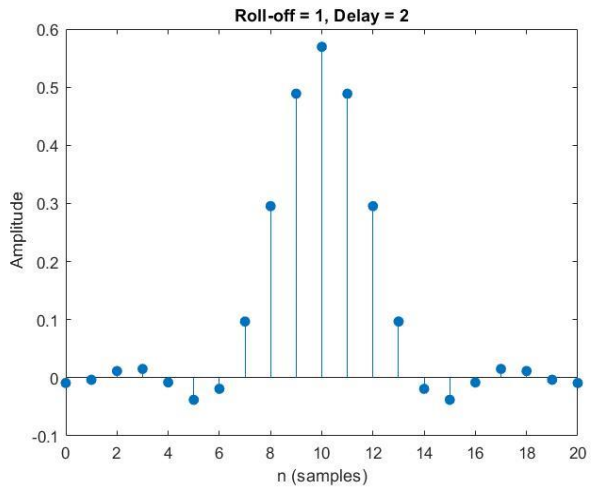
I. At Roll of = 0 and Delay = 2



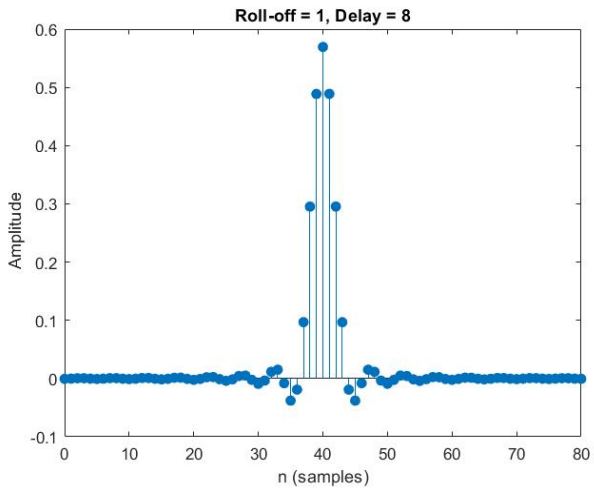
II. At Roll of = 0 and Delay = 8



III. At Roll of = 1 and Delay = 2



IV. At Roll of = 1 and Delay = 8



Eye diagram at transmitter and after Receiver

I. At Roll of = 0 and Delay = 2

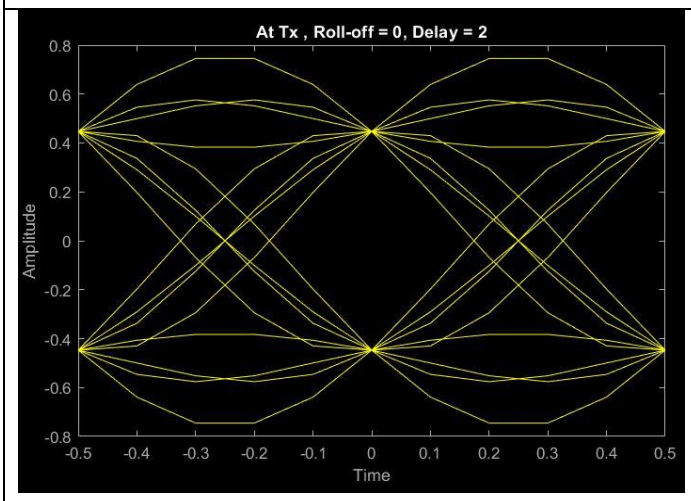


Figure 6: At Point A (Tx): Roll - off = 0, Delay = 2

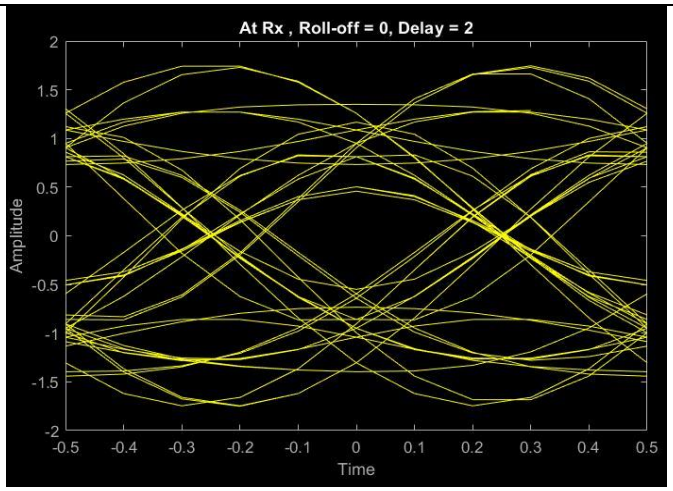


Figure 7: At Point A (Rx): Roll - off = 0, Delay = 2

II. At Roll of = 0 and Delay = 8

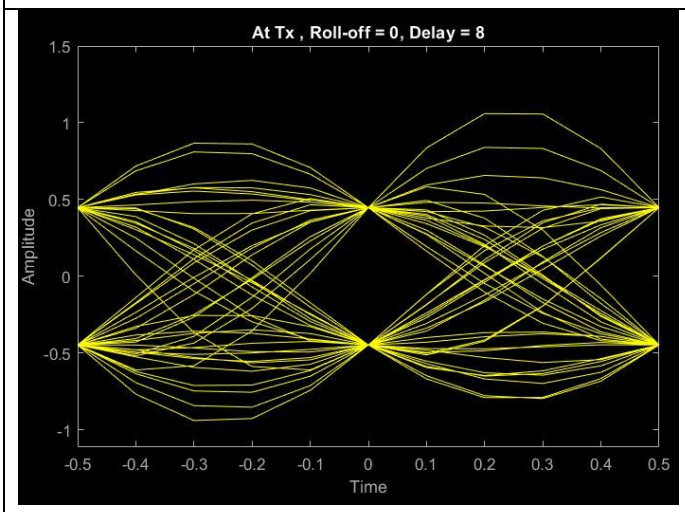


Figure 8: At Point A (Tx): Roll - off = 0, Delay = 8

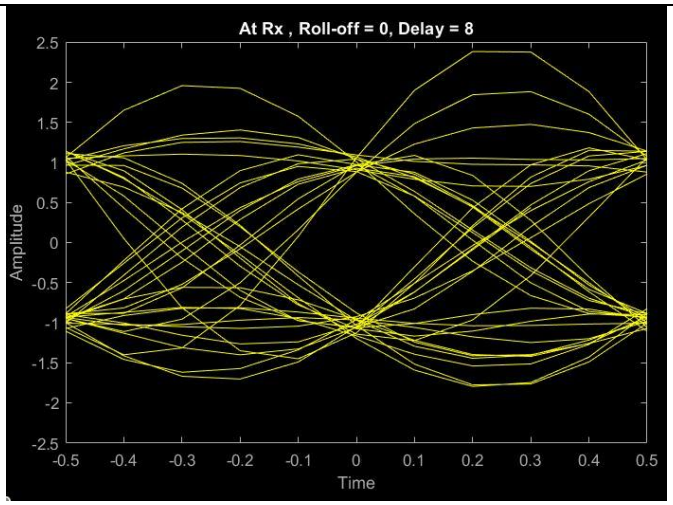


Figure 9: At Point A (Rx): Roll - off = 0, Delay = 8

III. At Roll of = 1 and Delay = 2

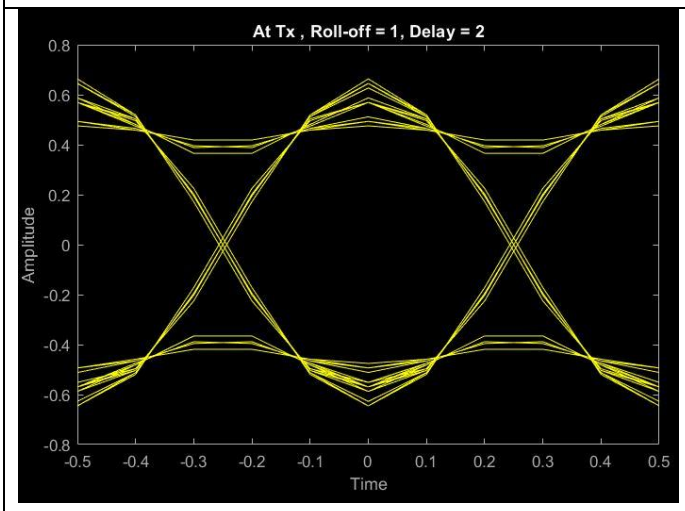


Figure 10: At Point A (Tx): Roll - off = 1, Delay = 2

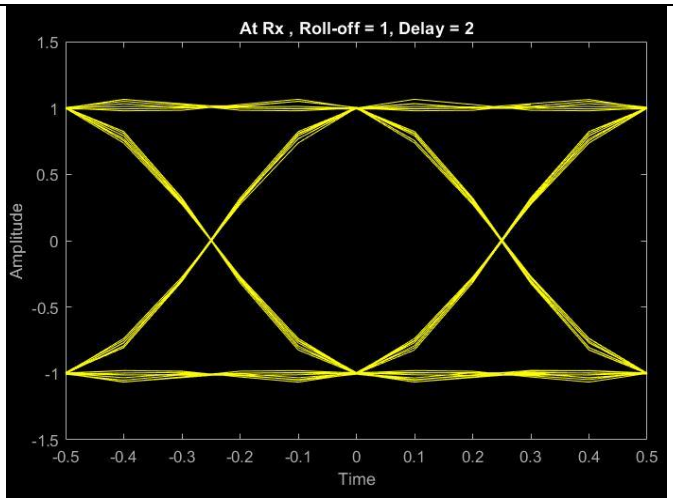


Figure 11: At Point A (Rx): Roll - off = 1, Delay = 2

IV. At Roll of = 1 and Delay = 8

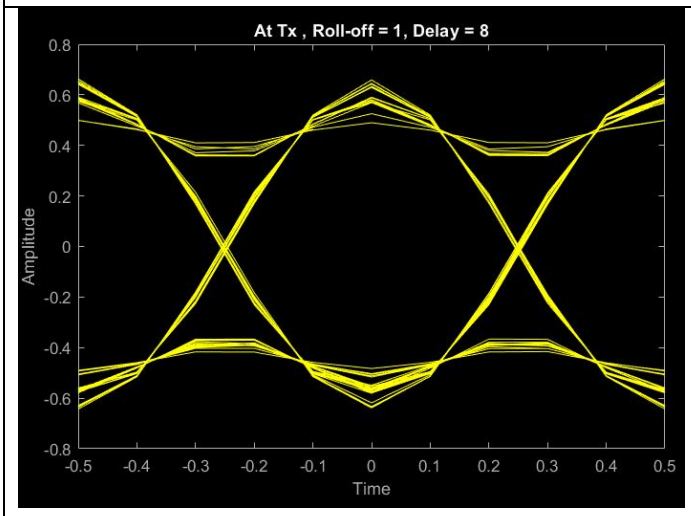


Figure 12: At Point A (Tx): Roll - off = 1, Delay = 8

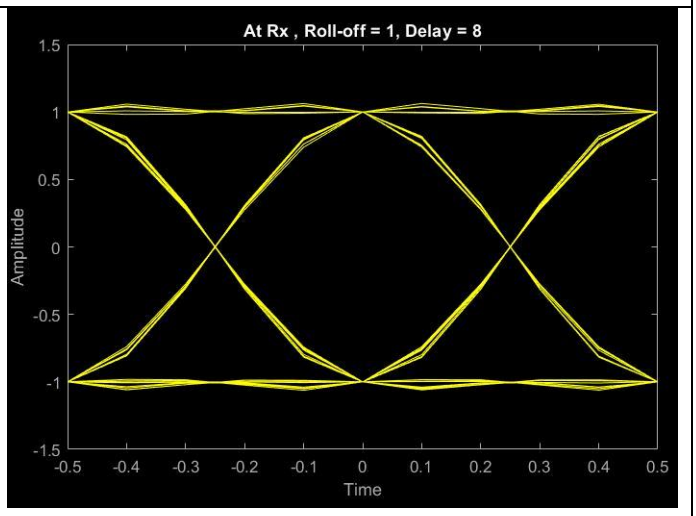


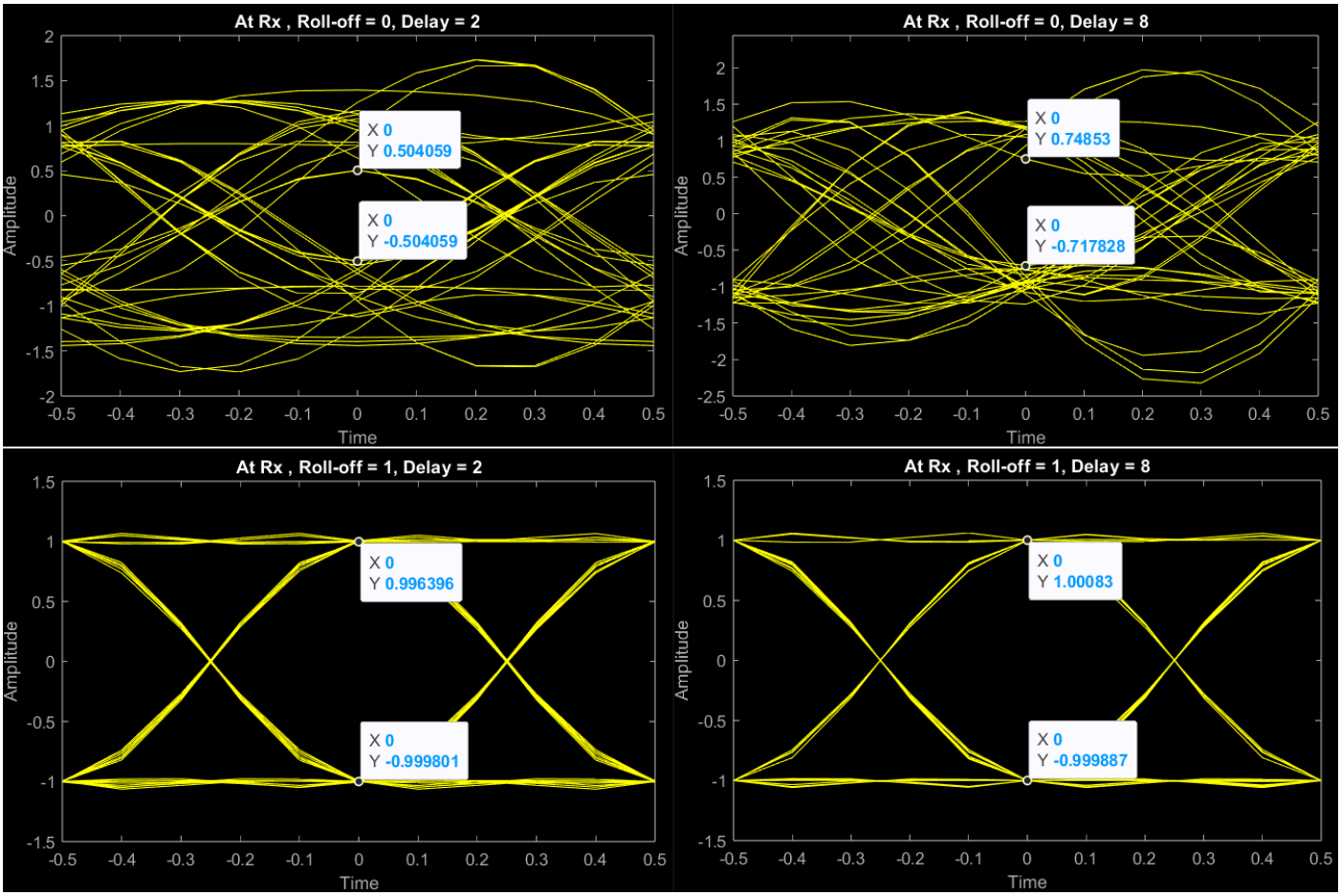
Figure 13: At Point A (Rx): Roll - off = 1, Delay = 8

POSTULATION

As the roll-off factor rises, indicating an increase in the utilized bandwidth, the signal contracts in the time domain, reducing inter-symbol interference (ISI) and resulting in a wider eye opening, which enhances sampling accuracy.

Conversely, with an increase in delay, reflecting the duration of the raised cosine filter window used in finite impulse response (FIR) filtering, the ripples from the raised cosine filter also increase, leading to heightened ISI and less noise margin. Concerning the sampling instant and the eye opening, achieving reliable communication necessitates aligning the sampling instant at the center of the eye to minimize the likelihood of sampling errors. As the eye remains relatively open for an extended period around the optimal sampling time, the system becomes more tolerant of timing errors, thus enhancing immunity to synchronization errors.

COMMENTS



As long as the span of the raised cosine scheme increases without ripples of high amplitudes @ the sampling instants, the noise margin increases which in turn corresponds to a wider eye opening as shown in the figures above It is worth noting that there is not a straight forward correlation between the span & the eye opening but the usual use-case is described in the (postulation section)

Index

Full MATLAB Code

```
close all;
clear;
clc
%% control flags
pulse_width = 1; % Ts = 1 s
pulse_samples = 5 ; % No. of samples to represent the shaping pulse signal
imp_tr_length = 10; %no of the impulses in the sampling
Spacing = pulse_width/pulse_samples; % To de − normalize the time index y[n] = y[nTs]
%% generate the transmitted signal
pulse = [5 4 3 2 1]/sqrt(55); %generate the unit energy pulse − shaping signal
imp_tr = randi([0 1],1, imp_tr_length); % generate the random logic bit − stream
imp_tr_mapped = 2 ∗ imp_tr − 1; % map logic 1 into (+1) & logic 0 into (−1)
imp_tr_upsampled = upsample(imp_tr_mapped, pulse_samples);
%create the sampling period by upsampling (Ts = 1s , Tp = 20 ms −−> 5 samples)
imp_tr_upsampled = imp_tr_upsampled(1: end − 4) ;
%to elimiate the spacing after the last pulse
tx_out = conv(pulse, imp_tr_upsampled); % produce the tranmitted signal

%% plot the pulse signal and its PAM
figure();
subplot(2,2,1);
pulse_x = (0: length(pulse) − 1).∗ Spacing;
plot(pulse_x, pulse);
xlim([0 1])
xlabel("Time (Sec)");
ylabel("Amplitude")
hold on;
stem(pulse_x, pulse);
title("Pulse Shaping Function");
subplot(2,2,2);
imp_x = (0: length(imp_tr_upsampled) − 1).∗ Spacing;
stem(imp_x, imp_tr_upsampled);
xlabel("Time (Sec)");
ylabel("Amplitude")
title("PAM Sampling Function");
subplot(2,2, [3 4]);
tx_out_x = (0: length(tx_out) − 1).∗ Spacing;
plot(tx_out_x, tx_out);
hold on;
stem(imp_x, imp_tr_upsampled);
set(gca, "XAxisLocation", 'origin');
title("Transmitted Signal");
sgtitle("Pulse Shaping Scheme @ Tx");
```

```
figure;
plot(tx_out_x, tx_out);
set(gca, 'XAxisLocation', 'origin');
xlabel("Time (Sec)");
ylabel("Amplitude")
title("Output of the TX");

%% Matched Filter
tr_x = (1: imp_tr_length) − Spacing;
MF_filter_TF = fliplr(pulse);
MF_filter_out = conv(tx_out, MF_filter_TF);
figure()
subplot(2,1,1);
MF_x = (0: length(MF_filter_out) − 1).∗ Spacing;
plot(MF_x, MF_filter_out);
hold on;
stem(tr_x, imp_tr_mapped);
set(gca, "XAxisLocation", 'origin');
title("Rx signal using Matched filter");
xlabel("Time (Sec)");
ylabel("Amplitude")
%% Non − matched Filter
NMF_filter_TF = ones(1, pulse_samples)/sqrt(length(pulse));
NMF_filter_out = conv(tx_out, NMF_filter_TF);
subplot(2,1,2);
plot(MF_x, NMF_filter_out, 'k');
hold on;
stem(tr_x, imp_tr_mapped);
set(gca, "XAxisLocation", 'origin');
title("Rx signal using Non − Matched filter");
xlabel("Time (Sec)");
ylabel("Amplitude")

%% Matching filter using correlator
corr_out = correlate(tx_out, pulse); %p(t) ∗ p(t)
% To show the correlator & the matched filter on the same plot
figure()
subplot(2,1,1);
stem(tr_x, imp_tr_mapped);
title("Transmitted PCM sequence");
subplot(2,1,2);
plot(MF_x, MF_filter_out, 'r');
hold on;
plot(MF_x(1: end − 4), corr_out, 'b');
hold on;
stem(tr_x, corr_out(5: 5: end), 'g');
title("Correlator output vs Matched filter output")
```

```
xlabel("Time (Sec)");
ylabel("Amplitude")
set(gca,'XAxisLocation','origin');
xlim([0 tr_x(end)]);
legend("Matched filter output","Correlator output ","Correlator output sampled",'Location','best');

%% Noise Analysis
% Generate the binary sequency with numerous bits to introduce the Pe as BER
imp_tr_length = 10000;
imp_tr = randi([0 1],1,imp_tr_length);  % generate the random logic bit - stream
imp_tr_mapped = 2*imp_tr - 1;  % map logic 1 into (+1) & logic 0 into (-1)
imp_tr_upsampled = upsample(imp_tr_mapped,pulse_samples);
%create the sampling period by upsampling (Ts = 1s , Tp = 20 ms --> 5 samples)
imp_tr_upsampled = imp_tr_upsampled(1:end-4) ;
%to elimiate the spacing after the last pulse
tx_out = conv(pulse,imp_tr_upsampled);  % produce the tranmitted signal
% Generate the noise
AWGN = randn(1,length(tx_out));
% ~N(0,1) Scaled the variance by scaling the standard deviation
% create 2 arrays to get the practical BER for each case
k = 1;
BER_MF_practical = zeros(1,8);
BER_NMF_practical = zeros(1,8);
BER_theoritical = zeros(1,8);
Eb = 1;
for i = logspace(-0.2,0.5,8)
%Since we are using unit - energy p(t). Hence, 1/N0 E [-2,5]dB
  N0 = Eb/i;
  noise = AWGN * sqrt(N0/2);
% A new signal due to the addition of the AWGN Noise @ the reciever "Z(t)"
z = noise + tx_out;
% Passing the signal (z(t)) to the matched/non - matched filter
MF_filter_at_Rx = correlate(z,pulse);
NMF_filter_at_Rx = correlate(z,NMF_filter_TF);
% Sampling the output @ the reciever
MF_filter_at_Rx = receiver_sampler(MF_filter_at_Rx,pulse_samples);
NMF_filter_at_Rx = receiver_sampler(NMF_filter_at_Rx,pulse_samples);

            % Reciever Decision
%Sample the filter output then take a threshold @ 0 :
% If > 0 --> Hence, the tranmitted symbol is p(t) --> 1
% If < 0 --> Hence , the transmitted symbol is -p(t) --> 0
MF_Rx_decision = decision_block(MF_filter_at_Rx,0);
NMF_Rx_decision = decision_block(NMF_filter_at_Rx,0);
% Calculate the BER by just counting the mismatches
% between the actual PCM & decision upon the constellation
 % We can compare directly with the mapping bec it is deterministic
check_MF = (MF_Rx_decision ~= imp_tr_mapped);
```

```
check_NMF = (NMF_Rx_decision ~= imp_tr_mapped);
BER_MF_practical(k) = length(check_MF(check_MF == 1))/imp_tr_length;
BER_NMF_practical(k) = length(check_NMF(check_NMF == 1))/imp_tr_length;
BER_theoritical(k) = 0.5 * erfc(sqrt(i));
k = k + 1;
end

% plot the BER for Matched/Unmatched/Theoritical filters on semi - log axis
figure();
BER_x = -2:5;
semilogy(BER_x, BER_theoritical, 'g');
hold on;
grid on ;
semilogy(BER_x, BER_MF_practical, 'b');
hold on;
semilogy(BER_x, BER_NMF_practical, 'r');
legend("Theoritical", "Matched Filter",...
    "Non - Matched Filter");
xlabel("Eb/N0");
ylabel("BER");
title("BER across Matching / Non - matching Filter");

%% ISI & Raised Cosine
imp_tr_length = 100;
imp_tr = randi([0 1],1, imp_tr_length);  % generate the random logic bit - stream
imp_tr_mapped = 2 * imp_tr - 1; % map logic 1 into (+1) & logic 0 into (-1)
imp_tr_upsampled = upsample(imp_tr_mapped, pulse_samples);
roll_off = [0 1];
delay = [2 8];
for i = 1:2
  for j = 1:2
    srrc_tx = rcosine(pulse_width, pulse_samples, 'sqrt', roll_off(i), delay(j));
    %     srrc_tx = rcosdesign(roll_off(i), delay(j), pulse_samples, 'sqrt');
tx_out = conv(imp_tr_upsampled, srrc_tx, 'valid'); %To exclude the last zeros from the convolution
    % Assume the channel is white & noise - free
    % Suppose using matching srrc filter
    srrc_Rx = fliplr(srrc_tx);
    figure
    impz(srrc_tx);
    title(['Roll - off = ', num2str(roll_off(i)), ', Delay = ', num2str(delay(j))]);
    % Knowing that the Raised Cosine versions are symmetric -> Srrc_Rx = Srrc_Tx
    Rx_in = conv(tx_out, srrc_Rx, 'valid');
    eyediagram(tx_out, 2 * pulse_samples);
    title(['At Tx , Roll - off = ', num2str(roll_off(i)), ', Delay = ', num2str(delay(j))]);
    eyediagram(Rx_in(1: end - 6),2 * pulse_samples);
    title(['At Rx , Roll - off = ', num2str(roll_off(i)), ', Delay = ', num2str(delay(j))]);
  end
end
```

```
function Z_signal = correlate (Y_signal, X_signal)
Z_signal = zeros(1, length(Y_signal));
for i = 1: length(X_signal): length(Y_signal)
corr_window = zeros(1, length(X_signal));
  for j = 0: length(corr_window) − 1
  corr_window(j + 1) = Y_signal(i + j);
  Z_signal(i + j) = sum(corr_window.∗ X_signal);
  end
end
end

function z = receiver_sampler (x, n)
% x − −−> output of the filter , n − −−> Sampling period
z = zeros(1, floor(length(x)/n));
for i = 1: length(z)
 z(i) = x(n ∗ i);
end
end
function Y_Signal = decision_block(X_Signal , threshold)
  Y_Signal = zeros (1, length (X_Signal));
  Y_Signal(X_Signal > threshold) = 1;
  Y_Signal(X_Signal <= threshold) = −1;
end
```

The END Thank You