Communication Assignment 2

Ziad Sherif Muhammed
Sec:1 Code: 9202586

Abdelrahman Muhammed Hamza
Sec:1 Code: 9202793

Part ||

2) **Code**

```
num_of_bits = 6;
samples_per_bit = 4;

binary_data = randi([0 1], 1, num_of_bits);
g  = binaryDataSampled(binary_data, num_of_bits, samples_per_bit);

% Plot random bits
t = 1:1:num_of_bits;
figure;
stem(t,binary_data,'o');
ylabel('Shaped pulses');
title('Binary Data')

% Plot random bits sampled
t = 0:1/samples_per_bit:num_of_bits;
figure;
stem(t,g,'o');
ylabel('Shaped pulses');
title('Pulse Data sampled at rate 4')

h1 = rectFilter(samples_per_bit);
h2 = holdFilter(samples_per_bit);
h3 = linearFilter(samples_per_bit);

% Plot 3 filters h1, h2 and h3
t = 0:1/samples_per_bit:1;
figure;
subplot(3,1,1);
stem(t,h1);
ylabel('h1');
title('matched filter')
subplot(3,1,2);
stem(t,h2);
ylabel('h2');
title('hold filter')
subplot(3,1,3);
```

```
stem(t,h3);
ylabel('h3');
title('linear filter')


% Add noise to the transmitted signal
E = PowerSignal(g);
snr = 10;
N0 = E / (10^(snr / 10));
r = awgn(g, snr, 'measured');

% apply 3 filters with input data with noise
%y1 = conv(r, h1,'same');
%y2 = conv(r, h2,'same');
%y3 = conv(r, h3,'same');
y1 = conv(r, h1);
y2 = conv(r, h2);
y3 = conv(r, h3);
y1(y1>1)=1;
y1(y1<-1)=-1;
y2(y2>1)=1;
y2(y2<-1)=-1;
y3(y3>1)=1;
y3(y3<-1)=-1;
g_decoded = DecodeSignal(g,0,num_of_bits,samples_per_bit);
y1_decoded = DecodeSignal(y1,0,num_of_bits,samples_per_bit);
y2_decoded = DecodeSignal(y2,0,num_of_bits,samples_per_bit);
y3_decoded = DecodeSignal(y3,0,num_of_bits,samples_per_bit);

BER1 = BitErrorRate(binary_data,y1_decoded);
BER2 = BitErrorRate(binary_data,y2_decoded);
BER3 = BitErrorRate(binary_data,y3_decoded);

BER1_theo = BERTheoritcal(N0);
BER2_theo = BERTheoritcal(N0);
BER3_theo = BERTheoritcal(4*N0/3);

fprintf("BER1_theo at snr = 10: %f \n", BER1_theo );
fprintf("BER2_theo at snr = 10: %f \n", BER2_theo );
fprintf("BER3_theo at snr = 10: %f \n", BER3_theo );
```

```
% Plot output of 3 received filters y1, y2 and y3

t = 0:1/samples_per_bit:num_of_bits+1;
t_decoded = 1:1:num_of_bits;
% plot output of matched filter
figure;
subplot(3,1,1);
hold on;
stem(t ,y1, 'ro');
y1_decoded(y1_decoded==0) = -1; % plot -1 instead of zero
stem(t_decoded,y1_decoded, 'bo');
plot(t ,y1,'g');
ylabel('matched filter');
title('matched filter output')
legend('output matched filter','output matched filter sampled','output matched filter
continous');

% plot output of holf filter
subplot(3,1,2);
hold on;
stem(t ,y2, 'ro');
y2_decoded(y2_decoded==0) = -1;
stem(t_decoded,y2_decoded, 'bo'); % plot -1 instead of zero
plot(t ,y2,'g');
ylabel('hold filter');
title('hold filter output')
legend('output hold filter','output hold filter sampled','output hold filter continous');

% plot output of linear filter
subplot(3,1,3);
hold on;
stem(t ,y3, 'ro');
y3_decoded(y3_decoded==0) = -1;
stem(t_decoded,y3_decoded, 'bo');
plot(t ,y3,'g');
ylabel('linear filter');
title('linear filter output')
legend('output linear filter','output linear filter sampled','output linear filter continous');
```

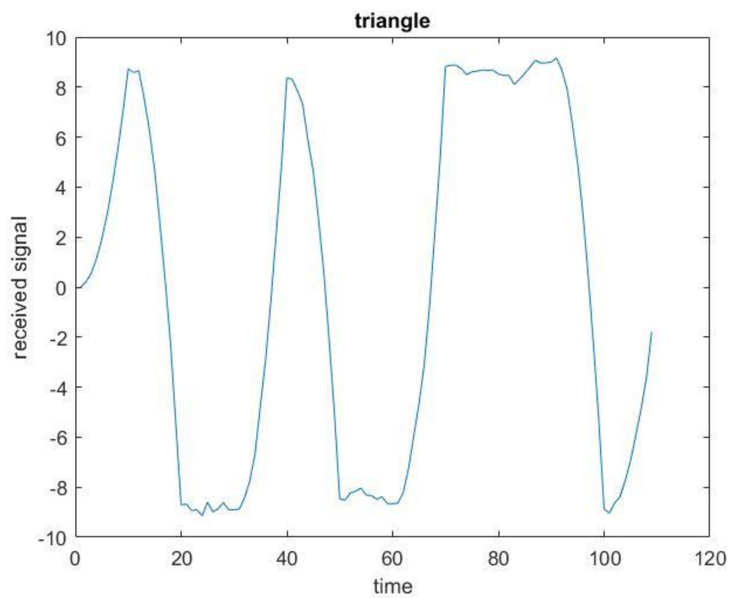
```
%----------------------------------- loop on different snr
num_of_bits = 50000;
samples_per_bit = 10;

binary_data = randi([0 1], 1, num_of_bits);
g  = binaryDataSampled(binary_data, num_of_bits, samples_per_bit);

% create 3 filters h1, h2 and h3
h1 = rectFilter(samples_per_bit);
h2 = holdFilter(samples_per_bit);
h3 = linearFilter(samples_per_bit);

snr = -10:1:20;
BER1_practical = zeros(1,length(snr));
BER1_theoritcal = zeros(1,length(snr));
BER2_practical = zeros(1,length(snr));
BER2_theoritcal = zeros(1,length(snr));
BER3_practical = zeros(1,length(snr));
BER3_theoritcal = zeros(1,length(snr));
for i = 1 : length(snr)
   E = PowerSignal(g);
   N0 = E / (10^(snr(i) / 10));
   r = awgn(g, snr(i), 'measured');
   y1 = conv(r, h1);
   y2 = conv(r, h2);
   y3 = conv(r, h3);
   y1_decoded = DecodeSignal(y1,0,num_of_bits,samples_per_bit);
   y2_decoded = DecodeSignal(y2,0,num_of_bits,samples_per_bit);
   y3_decoded = DecodeSignal(y3,0,num_of_bits,samples_per_bit);
   BER1_practical(i) = BitErrorRate(binary_data,y1_decoded);
   BER2_practical(i) = BitErrorRate(binary_data,y2_decoded);
   BER3_practical(i) = BitErrorRate(binary_data,y3_decoded);
   BER1_theoritcal(i) = BERTheoritcal(N0);
   BER2_theoritcal(i) = BERTheoritcal(N0);
   BER3_theoritcal(i) = BERTheoritcal(4*N0/3); % so that erfc be sqrt(3) / 2*sqrt(No)
end

% plot BER practical and theoritcal of 3 different systems

figure;
hold on;
```

```
semilogy(snr, BER1_practical, 'LineWidth', 1);
semilogy(snr, BER1_theoritcal, 'LineWidth', 1);
semilogy(snr, BER2_practical, 'LineWidth', 1);
semilogy(snr, BER2_theoritcal, 'LineWidth', 1);
semilogy(snr, BER3_practical, 'LineWidth', 1);
semilogy(snr, BER3_theoritcal, 'LineWidth', 1);
grid on;

legend('practical matched ', 'theoritcal matched', 'practical hold','theoritcal
hold','practical linear','theoritcal linear');
xlabel('E/No');
ylabel('BER');
title('BER');

%-------------------------------- Needed functions ---------------------------------
function p = PowerSignal(input)
   p = mean((input).^2 );
end

function ouput = DecodeSignal(input,threshold,num_of_bits,samples_per_bit)
   ouput = zeros(1,num_of_bits);
   for i = 1 : num_of_bits
      if input((i)*samples_per_bit ) > threshold
         ouput(i) =  1;
      else
         ouput(i) =  0;
      end
   end
end

function BER = BitErrorRate(input,output)
   BER = 0;
   for i = 1 : length(input)
      if input(i) ~= output(i)
         BER = BER + 1;
      end
   end
   BER= BER/length(input);
end
function BER = BERTheoritcal(N0)
```

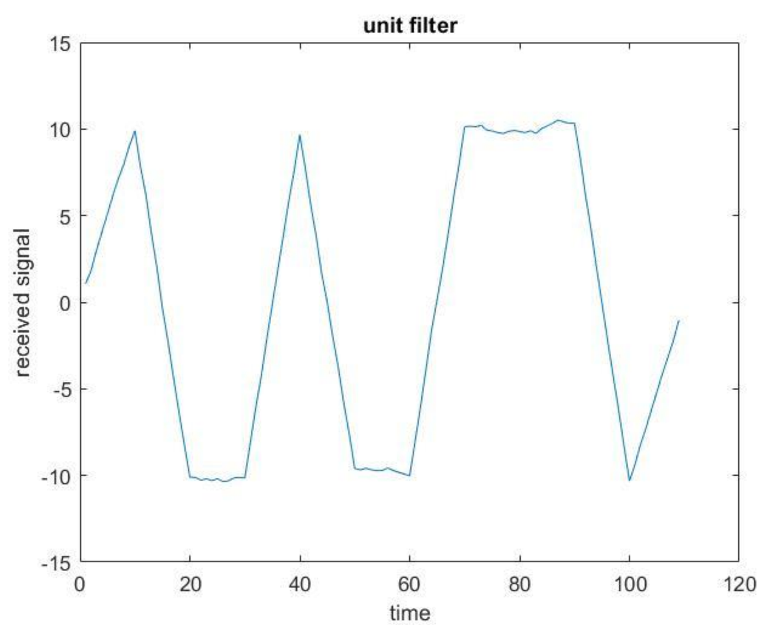
```
    BER = 0.5 * erfc(1/((N0)^0.5));
end

function filter = rectFilter(samples_per_bit)
    filter = ones(1, samples_per_bit + 1);
end
function filter = holdFilter(samples_per_bit)
    filter = zeros(1, samples_per_bit + 1);
    filter(round(samples_per_bit/2)+1) = 1;
end
function filter = linearFilter(samples_per_bit)
    filter = zeros(1, samples_per_bit+1);
    for i = 2:samples_per_bit + 1
        filter(i) = filter(i-1) + sqrt(3)/samples_per_bit;
    end
end
function g = binaryDataSampled(binary_data,num_of_bits,samples_per_bit)
    g = zeros(1, num_of_bits * samples_per_bit+1);
    t = 0:1/samples_per_bit:num_of_bits;
    for i = 1:num_of_bits
        if binary_data(i) == 1
            g((i-1)*(samples_per_bit) + 1:i*samples_per_bit) = 1;
        else
            g((i-1)*(samples_per_bit) + 1:i*samples_per_bit) = -1;
        end
    end

end
```

3) **Plots**

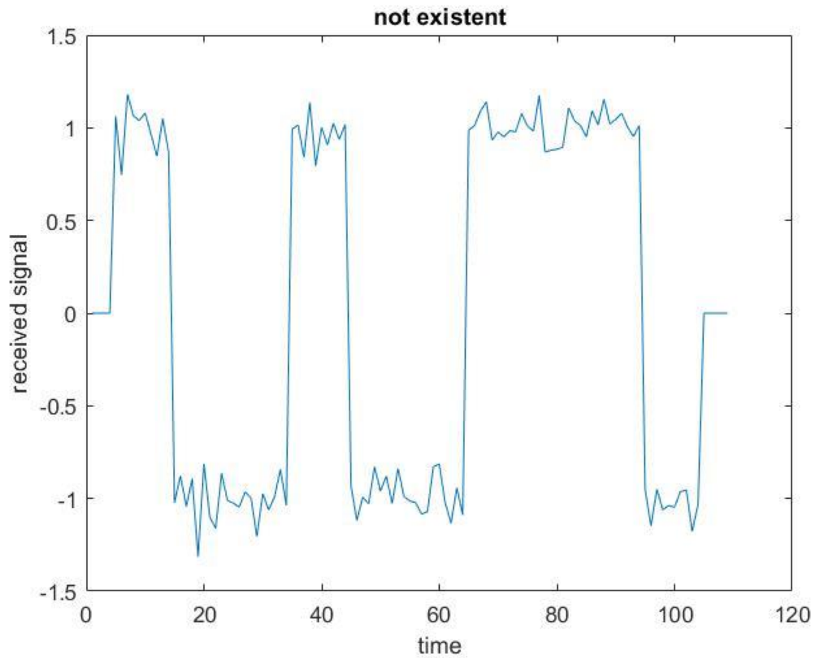
1. Linear



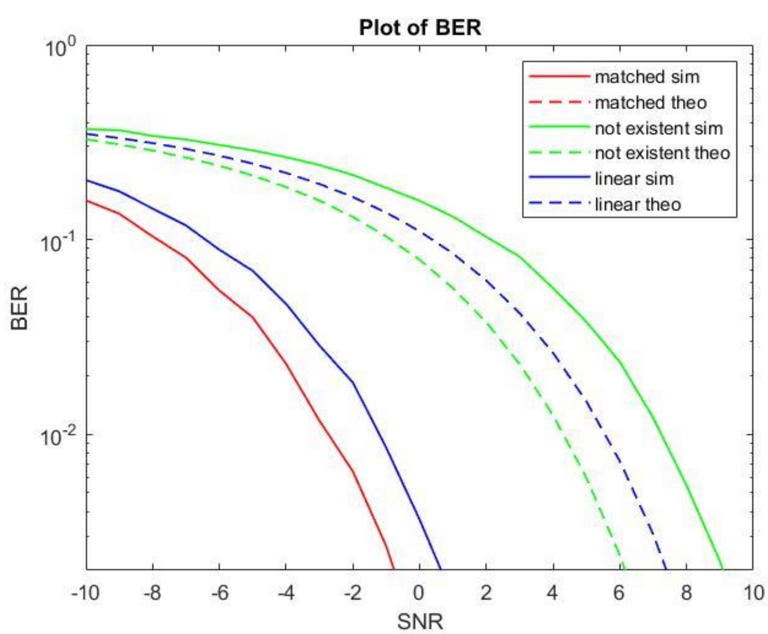
2. Matched



3. Pulse



4) BER Plot



5) BER is decreasing as a function E/No. due to:

1. Increasing the ratio of signal energy per bit (E) to power spectral density of noise (No), which is denoted as **E/No**, leads to a **decrease** in No and const (E). This decrease in No reduces the standard deviation (sigma) of **the added noise** since sigma = sqrt (No/2), which means that the added AWGN involves less variation and corresponds to smaller values **close to zero**. Consequently, which **it doesn`t affect the input signal** that much, therefore **BER decreases**.

2. In the theoretical expression for bit error rate (BER), the Q function is involved and it is known to be a decreasing function. The argument of the Q function is a * sqrt(E/No), where a is a constant. Since the Q function is decreasing, as the argument a * sqrt(E/No) increases, the Q function decreases and hence, the BER decreases.

   As the E/No increases, the argument of the Q function, a * sqrt(E/No), also increases due to the increasing nature of the square root function. Therefore, the Q function decreases and the BER decreases as well. Hence, the BER is a decreasing function of E/No.

6) The matched filter case is the one with lowest BER since it uses a filter matched to the pulse to **minimize** the probability of error. So, to achieve this, it equivalently **maximizes** the peak pulse SNR at the sampling instant, which in turn minimizes the probability of error.