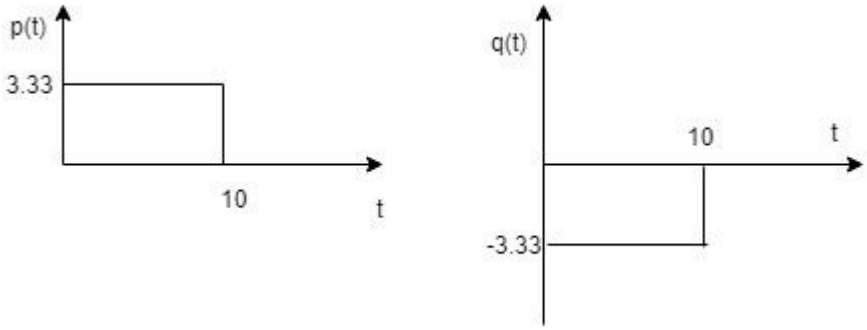
Coherent Communication Channel Simulator

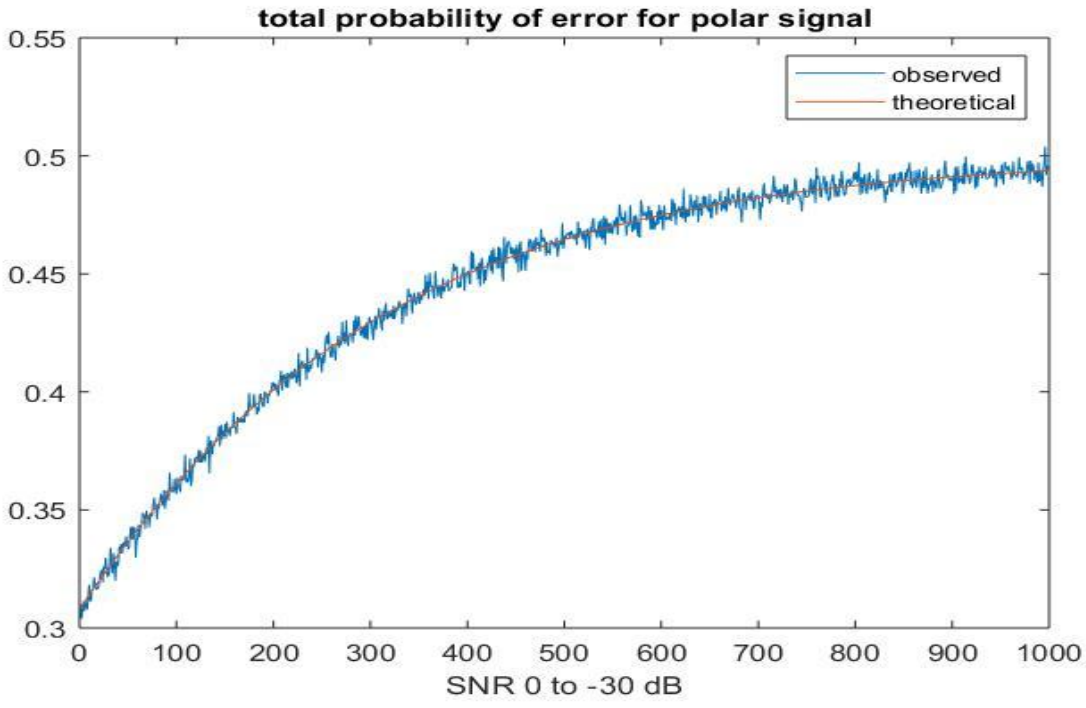
Simulation output:

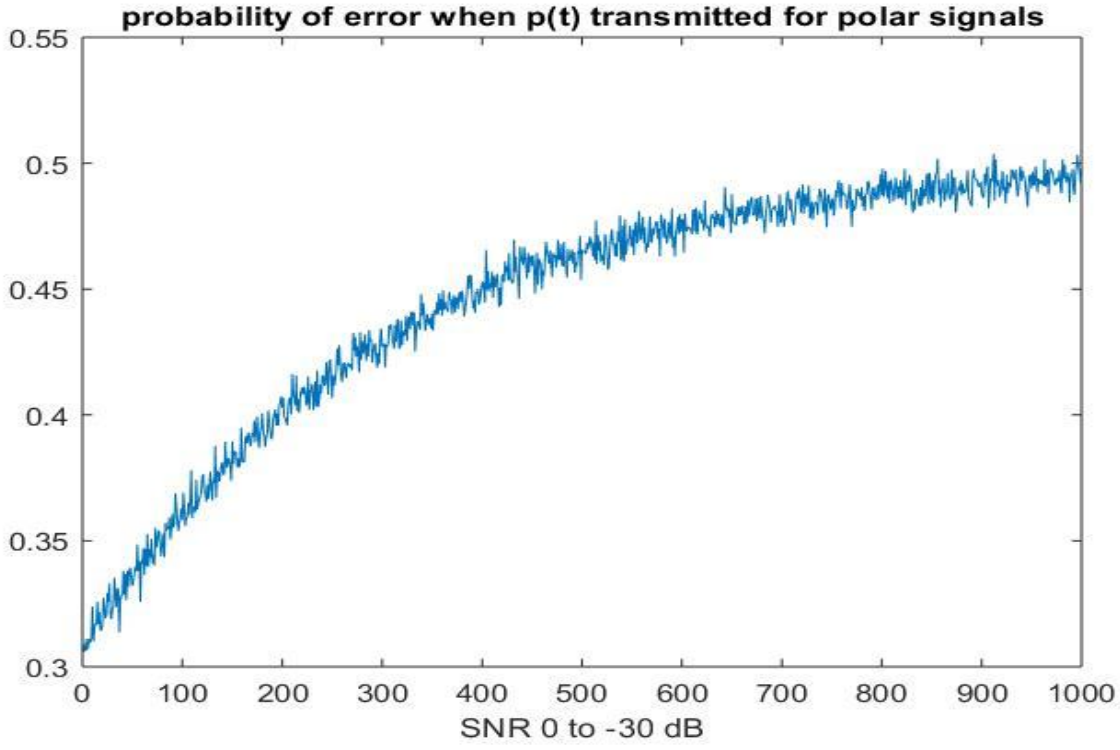
1. Observed probability of error and predicted probability of error graph.

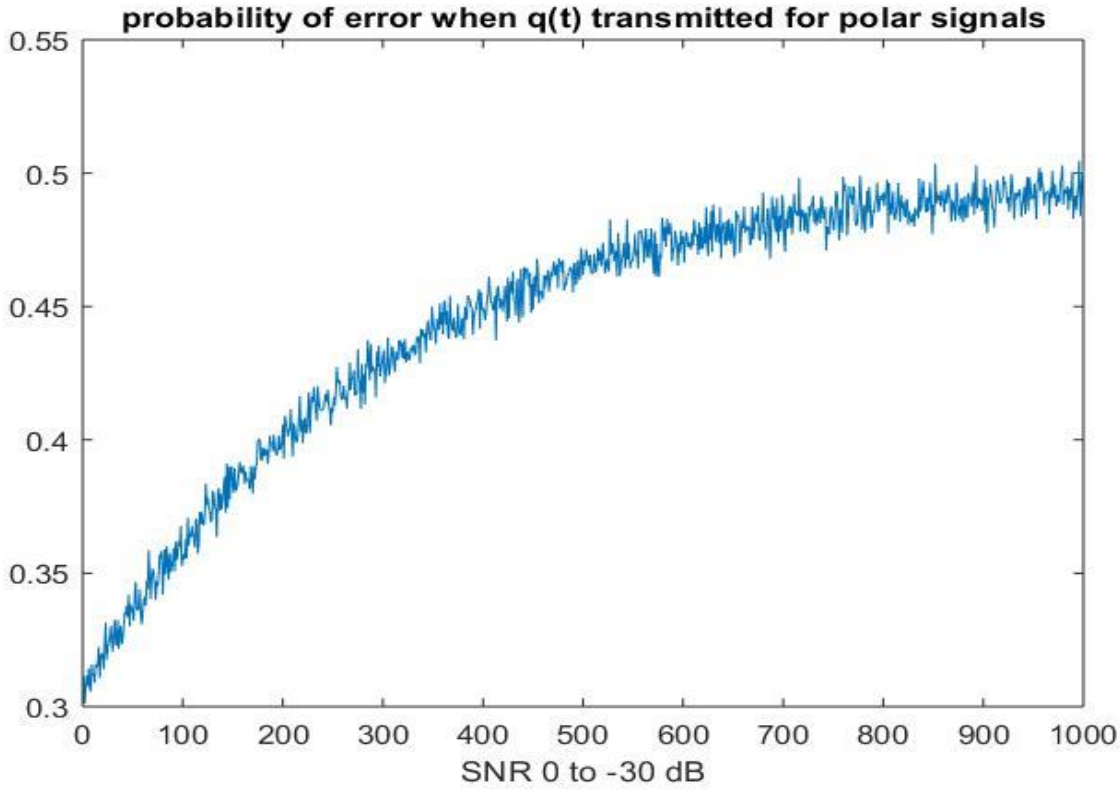
   Signals for this simulation are shown below-



Threshold is optimal and SNR is varied from 0 dB to -30 dB for this simulation .

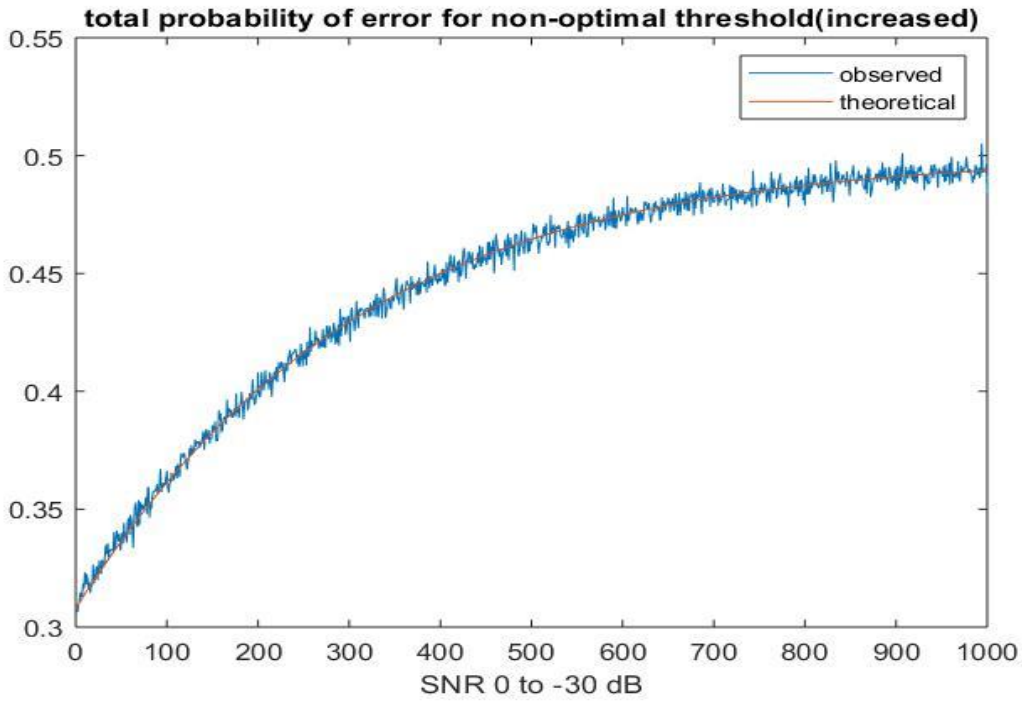


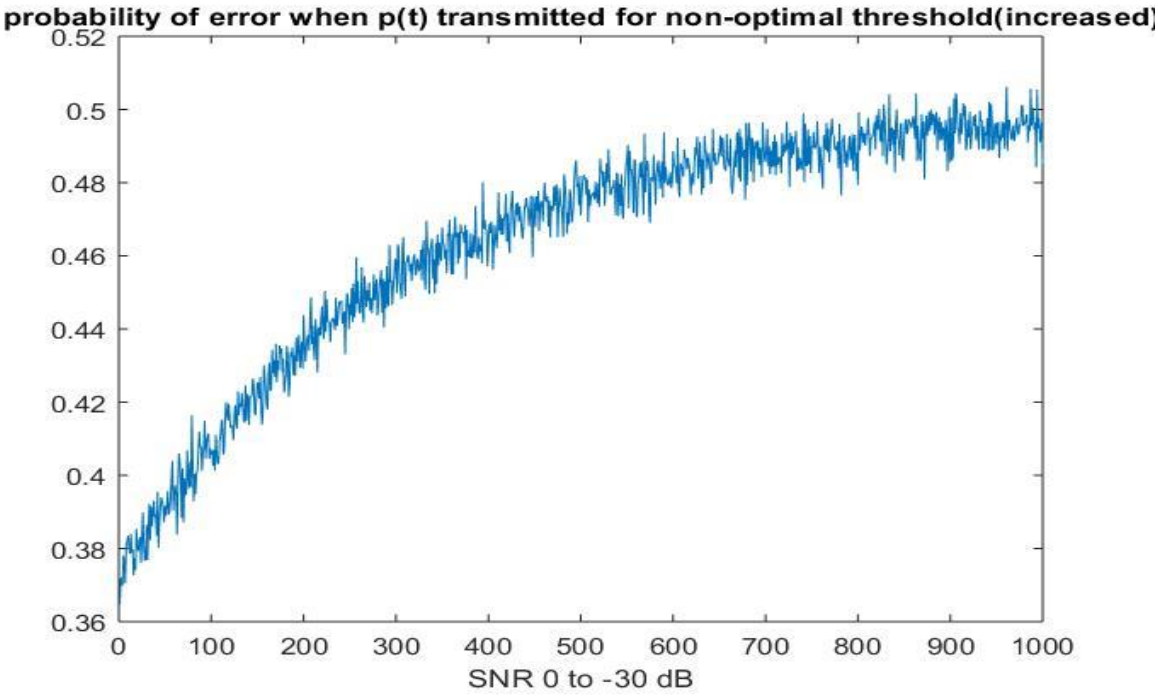

probability of error when p(t) transmitted for polar signals
0.55
0.5
0.45
0.4
0.35
0.3
0
100
200
300
400
500
600
700
800
900
1000
SNR 0 to -30 dB

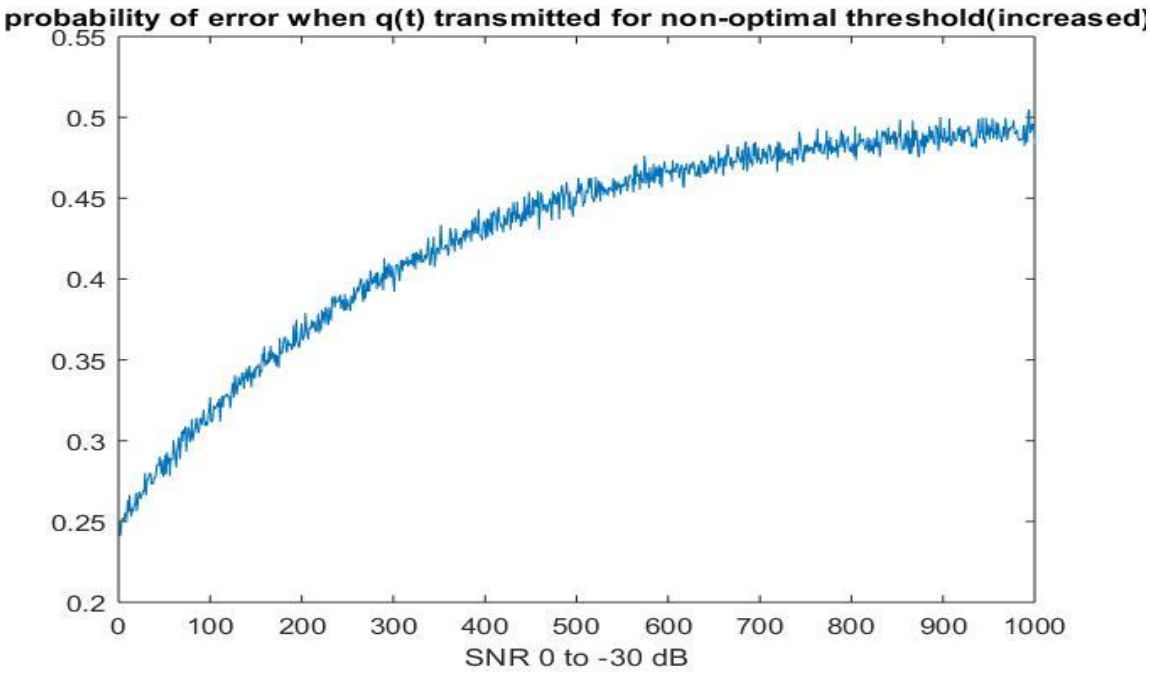

probability of error when q(t) transmitted for polar signals
0.55
0.5
0.45
0.4
0.35
0.3
0
100
200
300
400
500
600
700
800
900
1000
SNR 0 to -30 dB

2. Simulation for increased threshold(non-optimal) case.

Threshold = optimum threshold + 70

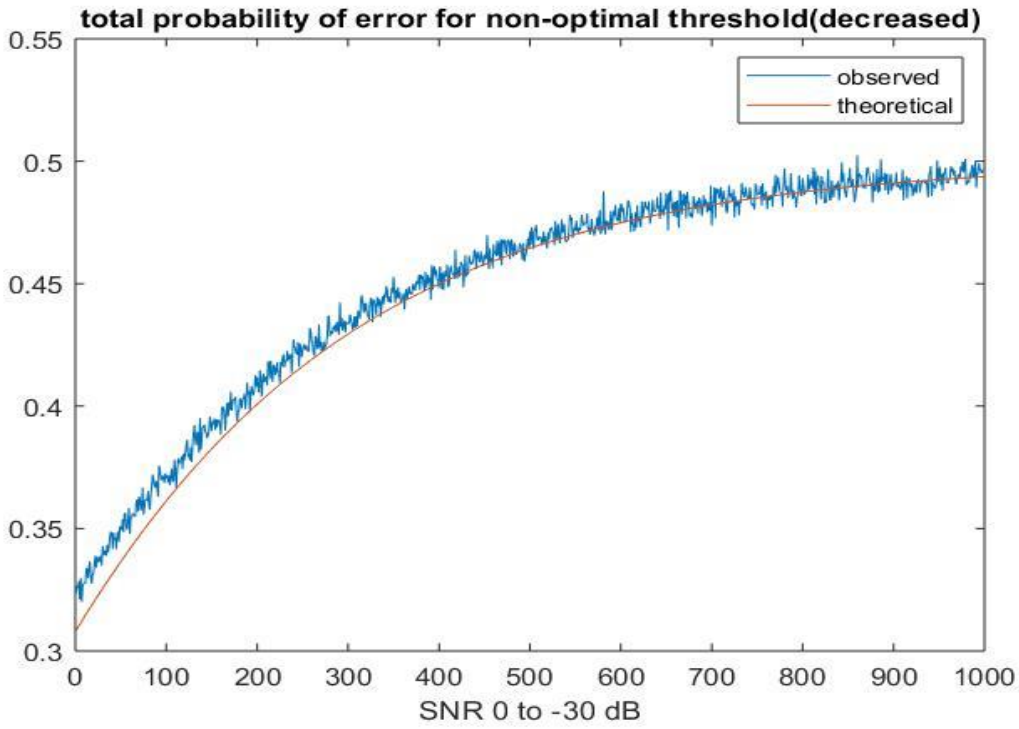


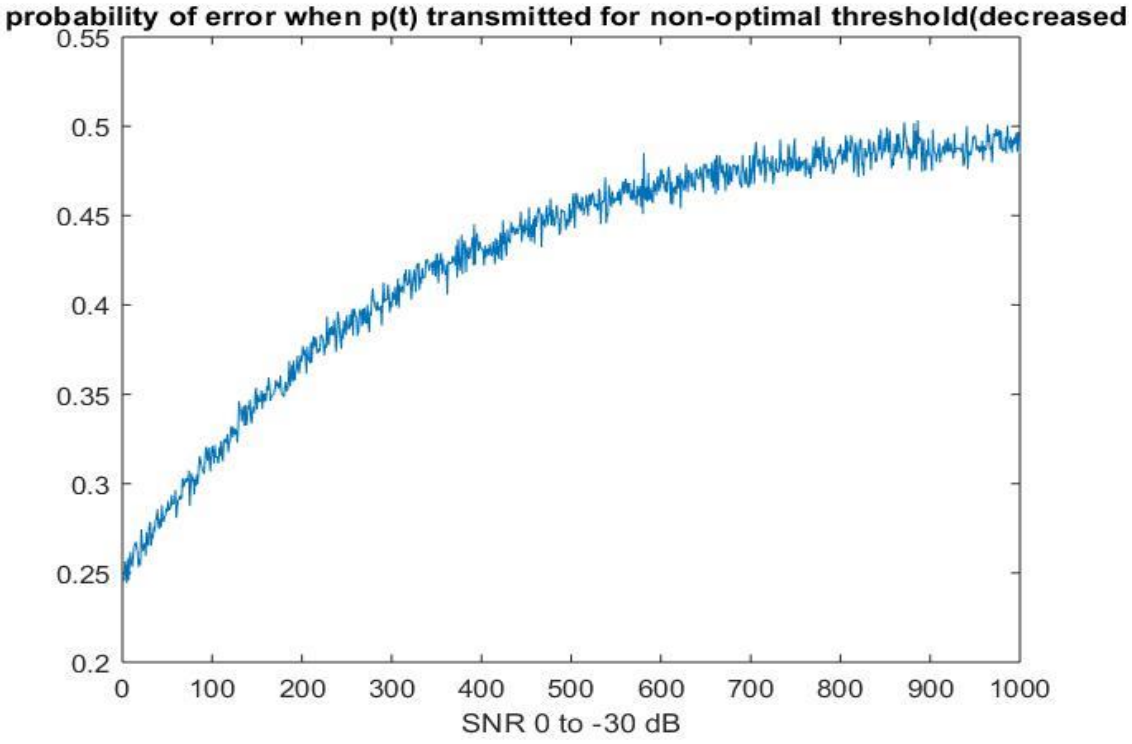


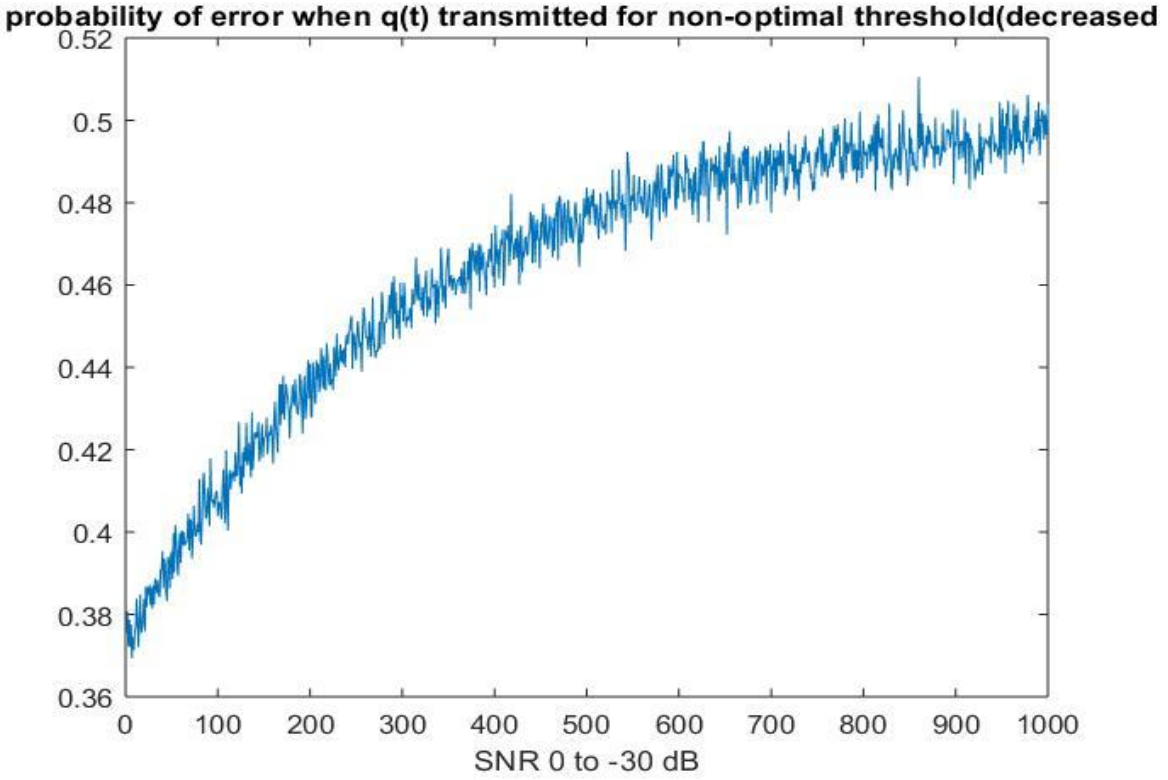


3. Simulation for decreased threshold(non-optimal) case .

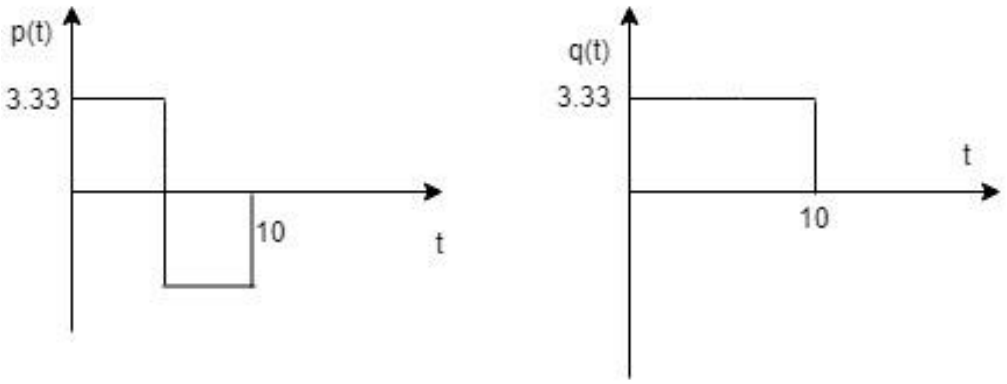
   Threshold = optimum threshold - 70

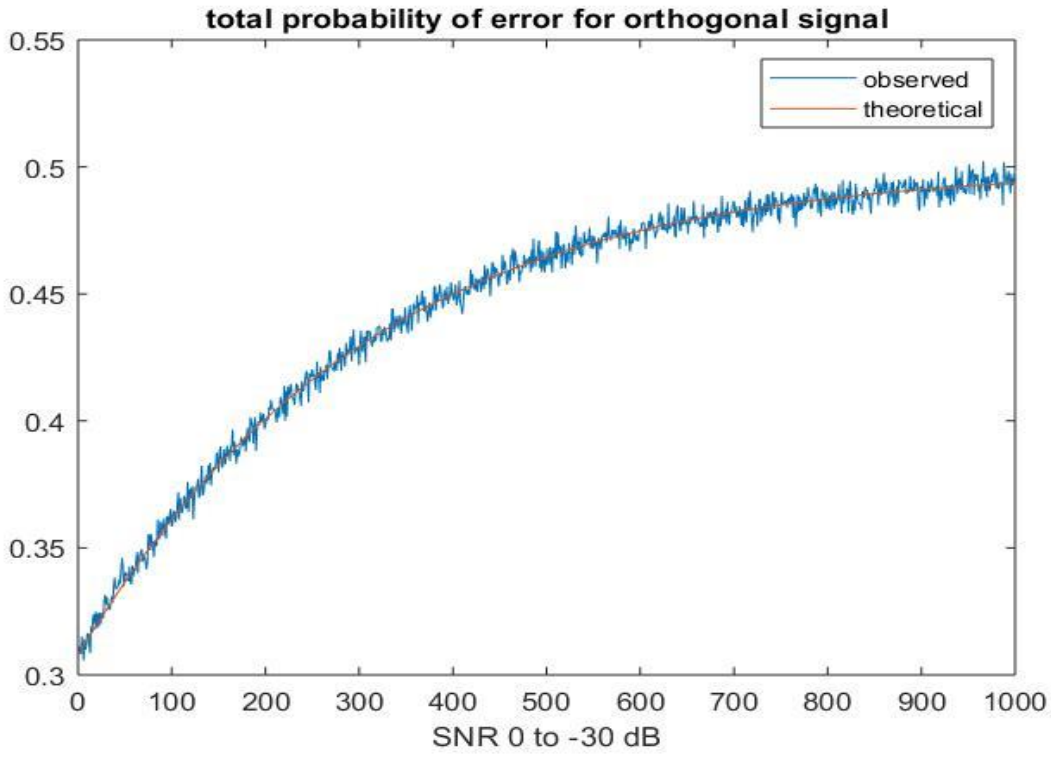


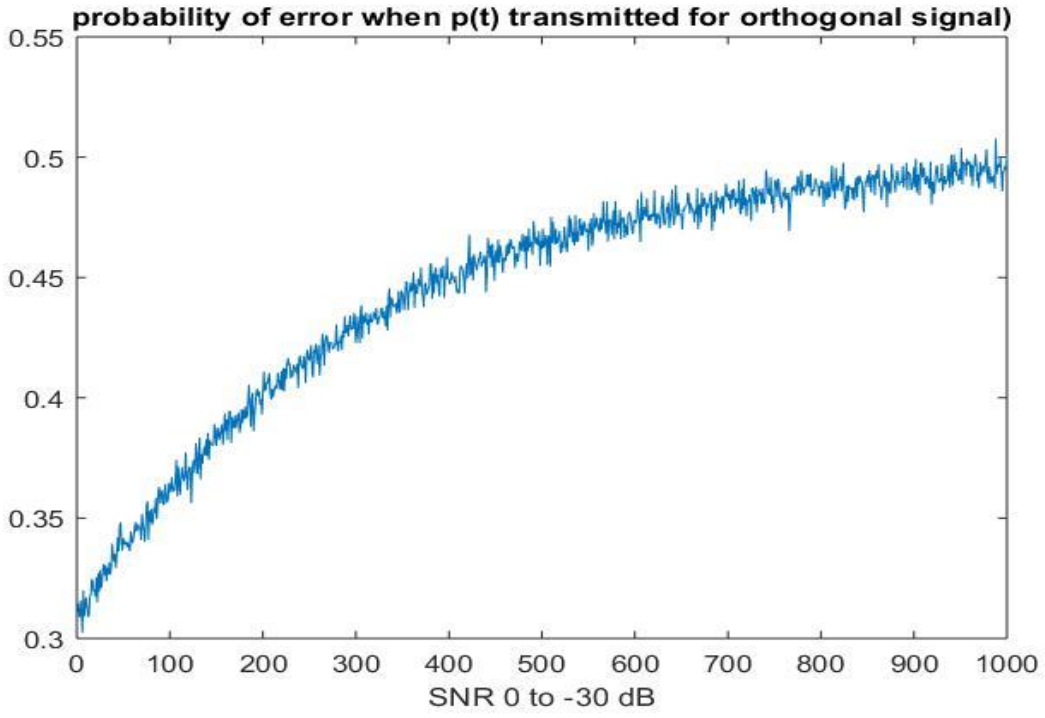

probability of error when p(t) transmitted for non-optimal threshold(decreased
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0
100
200
300
400
500
600
700
800
900
1000
SNR 0 to -30 dB

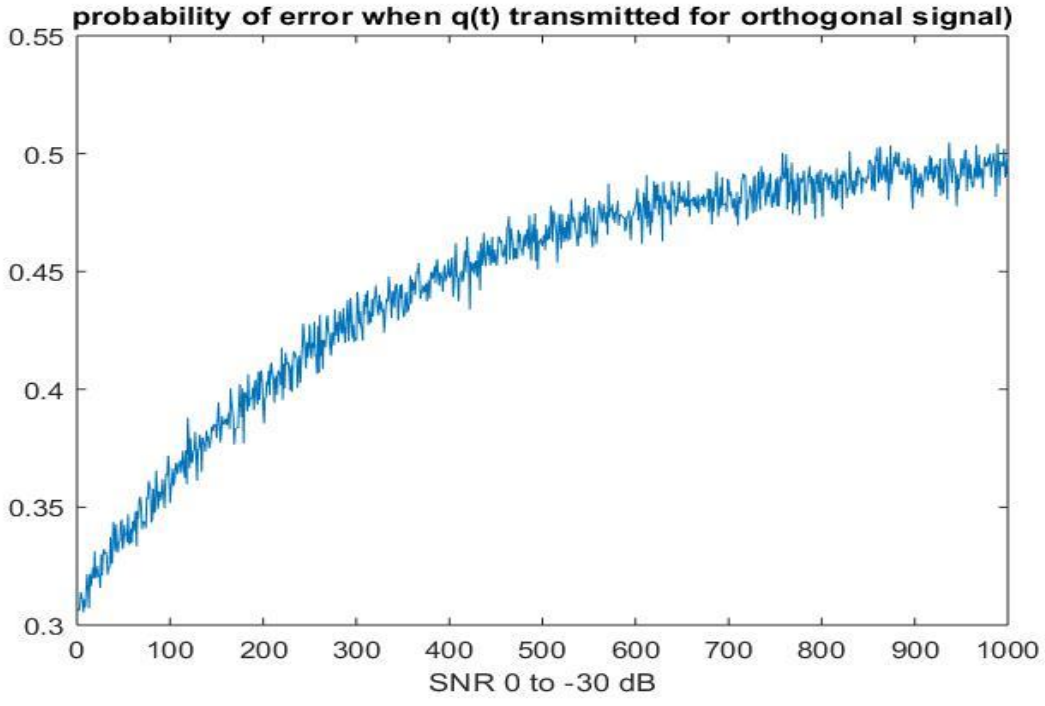

probability of error when q(t) transmitted for non-optimal threshold(decreased
0.52
0.5
0.48
0.46
0.44
0.42
0.4
0.38
0.36
0
100
200
300
400
500
600
700
800
900
1000
SNR 0 to -30 dB

4. For Homework3, problem 2 signals:
   Threshold = optimal threshold

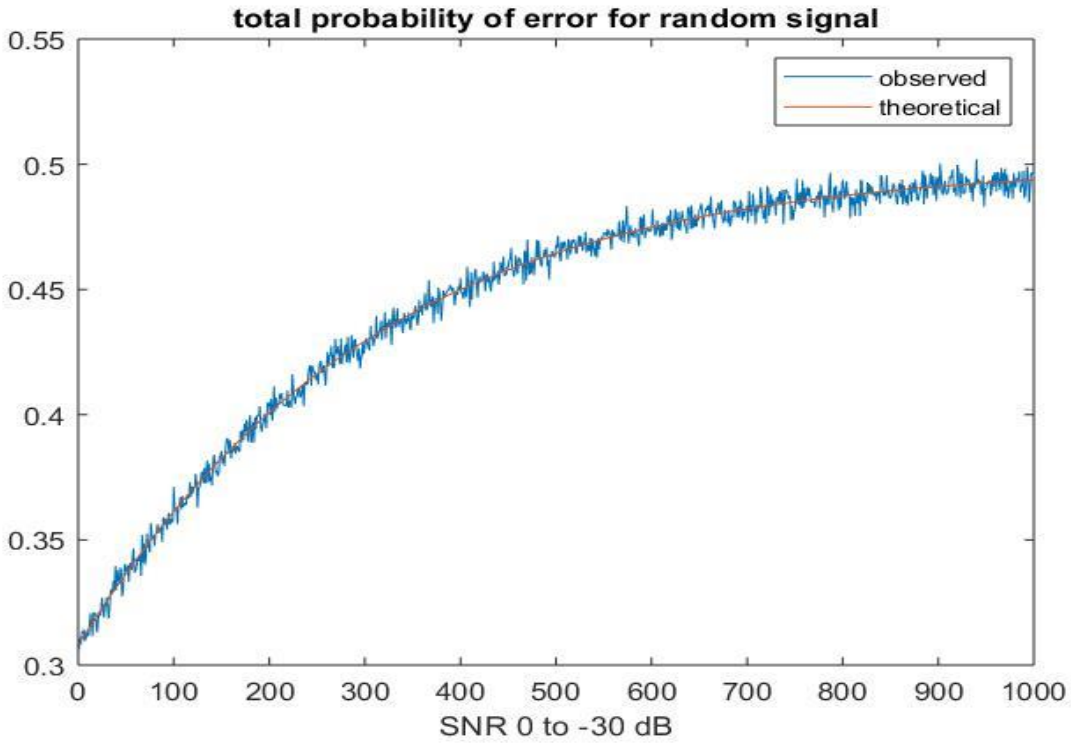


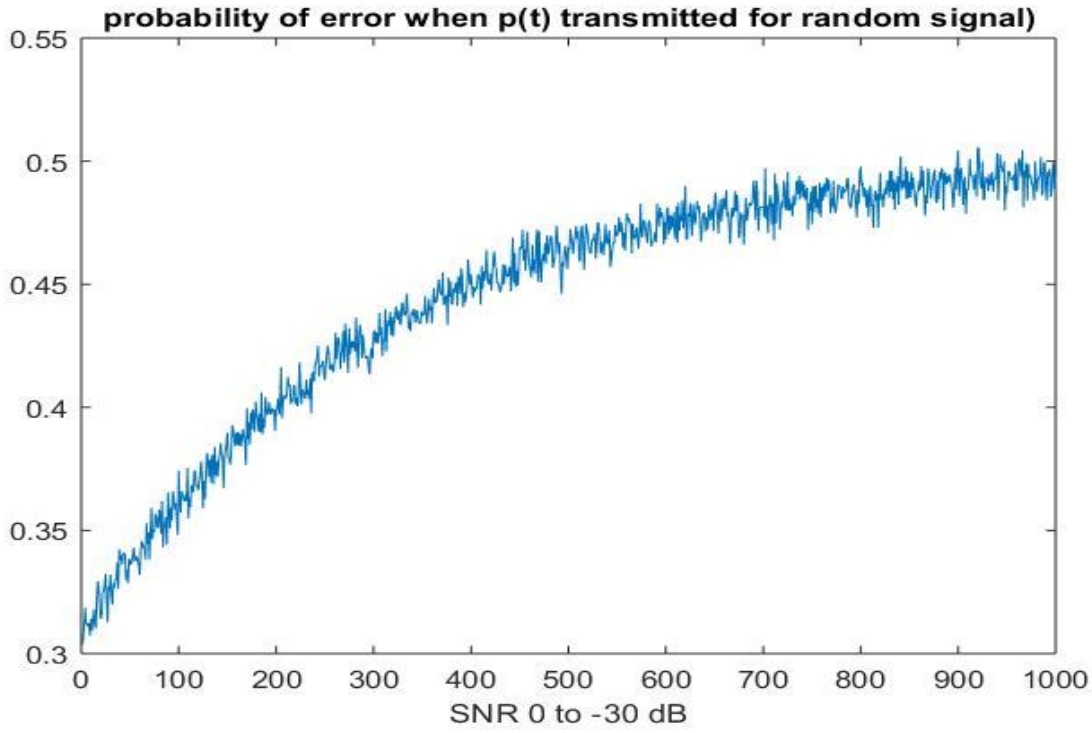


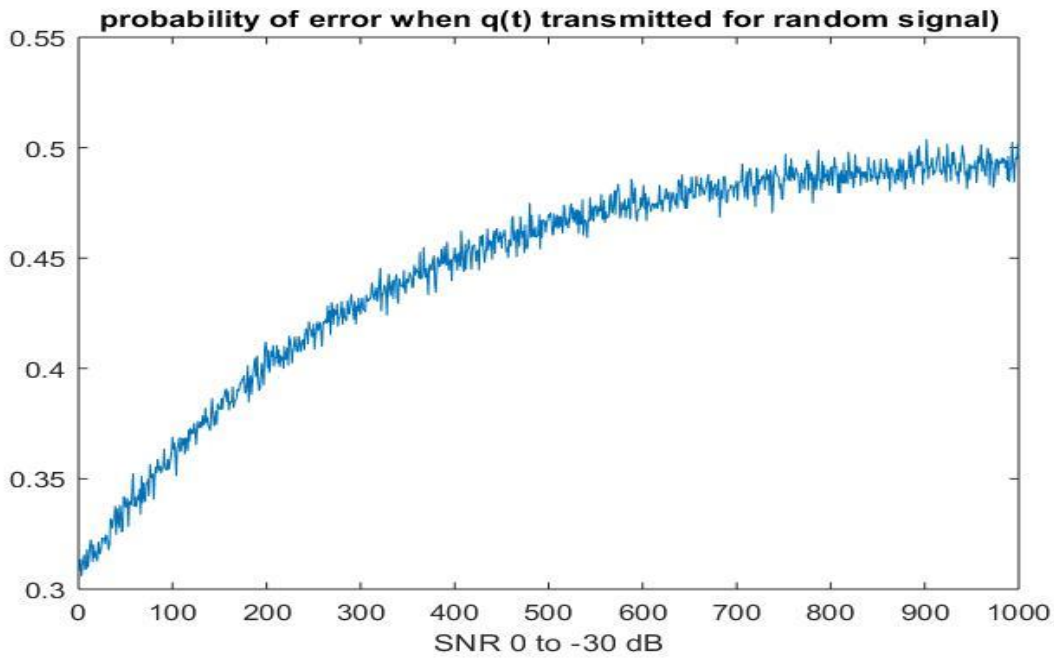

probability of error when p(t) transmitted for orthogonal signal)
0.55
0.5
0.45
0.4
0.35
0.3
0
100
200
300
400
500
600
700
800
900
1000
SNR 0 to -30 dB


probability of error when q(t) transmitted for orthogonal signal)
0.55
0.5
0.45
0.4
0.35
0.3
0
100
200
300
400
500
600
700
800
900
1000
SNR 0 to -30 dB

5. For random selected p(t) and q(t).
   Threshold = optimal threshold







MATLAB code:

```
clc;
clear;
samplesPerSymbol=10;
p_pulseEnergy = 100;
q_pulseEnergy = 100;
numSymbols = 20;
numSimulations = 1000;
noiseMean = 0;
transmittedSymbols=rand(1,numSymbols);
transmittedSymbols(transmittedSymbols>0.5)=1;
transmittedSymbols(transmittedSymbols<=0.5)=0;
%p_of_t=ones(1,samplesPerSymbol);
%q_of_t=-1*p_of_t;
%p_of_t = [1,1,1,1,1,-1,-1,-1,-1,-1];
%q_of_t = [1,1,1,1,1,1,1,1,1,1];
p_of_t=rand(1,samplesPerSymbol);
q_of_t=-rand(1,samplesPerSymbol);
p_of_t=p_of_t*(sqrt(p_pulseEnergy/(p_of_t*p_of_t')));
q_of_t=q_of_t*(sqrt(q_pulseEnergy/(q_of_t*q_of_t')));
idx1 = 1;
for snr_in_db = 0:-(30/(numSimulations-1)):-30
    idx2 = 1;
    for s = 1:1:(numSimulations)
```

```
for m = 1:1:samplesPerSymbol
ep(m) = (p_of_t(m))^2;
 eq(m) = (q_of_t(m))^2;
 epq(m) = (p_of_t(m))*(q_of_t(m));
end
 E_of_p = round(sum(ep));
 E_of_q = round(sum(eq));
 E_of_pq = round(sum(epq));
 optimal_threshold = (E_of_p  -  E_of_q)/2;
 noiseVariance(idx1) = round((E_of_p +  E_of_q - (2*
E_of_pq))/(10^((snr_in_db)/10)));
 theo_err_p(idx1) = qfunc(sqrt((E_of_p +  E_of_q - (2*
E_of_pq))/(4*noiseVariance(idx1))));
 threshold = optimal_threshold;
 %threshold = optimal_threshold + 70;
 %threshold = optimal_threshold - 70;
tx_signal = zeros(1,200);
j = 1;
for i= 1:1:20
    if transmittedSymbols(i) == 1;
        tx_signal(j:j+9) = p_of_t;
    else
        tx_signal(j:j+9) = q_of_t;
    end
    j = j+10;
end
noise_signal = noiseMean + sqrt(noiseVariance(idx1)) *
randn(1,200);
received_signal = tx_signal + noise_signal ;
filter =  fliplr(p_of_t) ;
filter1 = fliplr(q_of_t) ;
idx = 1;
op = zeros(1,20);
for k = 1:10:191
    y1 = conv(received_signal(k:k+9),filter);
    y2 = conv(received_signal(k:k+9),filter1);
    fil_op = y1(10)-y2(10);
    if fil_op > (threshold)
        op(idx) = 1;
    else
        op(idx) = 0;
    end
    idx = idx +1;
end
```

```
%figure(2),stem(op);
tx0_rx1 = 0;
tx1_rx0 = 0;
num_of_q = 0;
num_of_p = 0;
for l = 1:1:20
    if  transmittedSymbols(l) == 1
        if op(l) == 0
            tx1_rx0 = tx1_rx0 + 1;
        end
        num_of_p = num_of_p + 1;
    else
        if op(l) == 1
            tx0_rx1 = tx0_rx1 + 1;
        end
         num_of_q = num_of_q + 1;
    end
end
p_err_p_of_t = tx1_rx0/num_of_p ;
p_err(idx2) = p_err_p_of_t;
perr_p(idx1) = sum(p_err)/numSimulations;
p_err_q_of_t = tx0_rx1/num_of_q ;
q_err(idx2) = p_err_q_of_t;
perr_q(idx1) = sum(q_err)/numSimulations;
total_bit_error = (tx1_rx0 + tx0_rx1)/ numSymbols;
t_err(idx2) = total_bit_error;
perr_total(idx1) = sum(t_err)/numSimulations ;
idx2 = idx2 + 1;
    end
idx1 = idx1 +1;
end
figure(1),plot(perr_total),title('total probability of
error for random signal');
hold on;
plot(theo_err_p);
legend('observed','theoretical');
xlabel(' SNR 0 to -30 dB ');
figure(2),plot(perr_q),title('probability of error when
q(t) transmitted for random signal)');
xlabel(' SNR 0 to -30 dB ');
figure(3),plot(perr_p),title('probability of error when
p(t) transmitted for random signal)');
xlabel(' SNR 0 to -30 dB ');
```