Experiment No 4-

Bit error rate of Quadrature Phase Shift Keying (QPSK) (with and without Gray labelling) in Additive White Gaussian Noise (AWGN)

Kushagra Khatwani

190020021

- Matlab Initialization-

```
close all;
clear;
clc;
```

- Message Signal-

```
%msg signal
num = 20000;                                %No of bits in the message
const = exp(1i*2*pi.*[1,3,5,7]/8);              %Constellation
msg = randi([0,1],num,1);                   %Message
```

- (A) QPSK Modulation (with gray labelling)-

```
%QPSK modulation
%00 ->  1+j
%01 -> -1+j
%11 -> -1-j
%10 ->  1-j
QPSK=zeros(length(msg)/2,1);
%definition of the QPSK symbols using Gray coding.
for n=1:length(msg)/2
    tmp=msg(2*n-1);
    tmp1=msg(2*n);
    if (tmp==0)&(tmp1==0)
        QPSK(n)=exp(1i*pi/4);%45 degrees
    end
    if (tmp==0)&(tmp1==1)
        QPSK(n)=exp(1i*3*pi/4);%135 degrees
    end
    if (tmp==1)&(tmp1==1)
        QPSK(n)=exp(1i*5*pi/4);%225 degrees
    end
    if (tmp==1)&(tmp1==0)
```

```
        QPSK(n)=exp(1i*7*pi/4);%315 degrees
    end
end
```

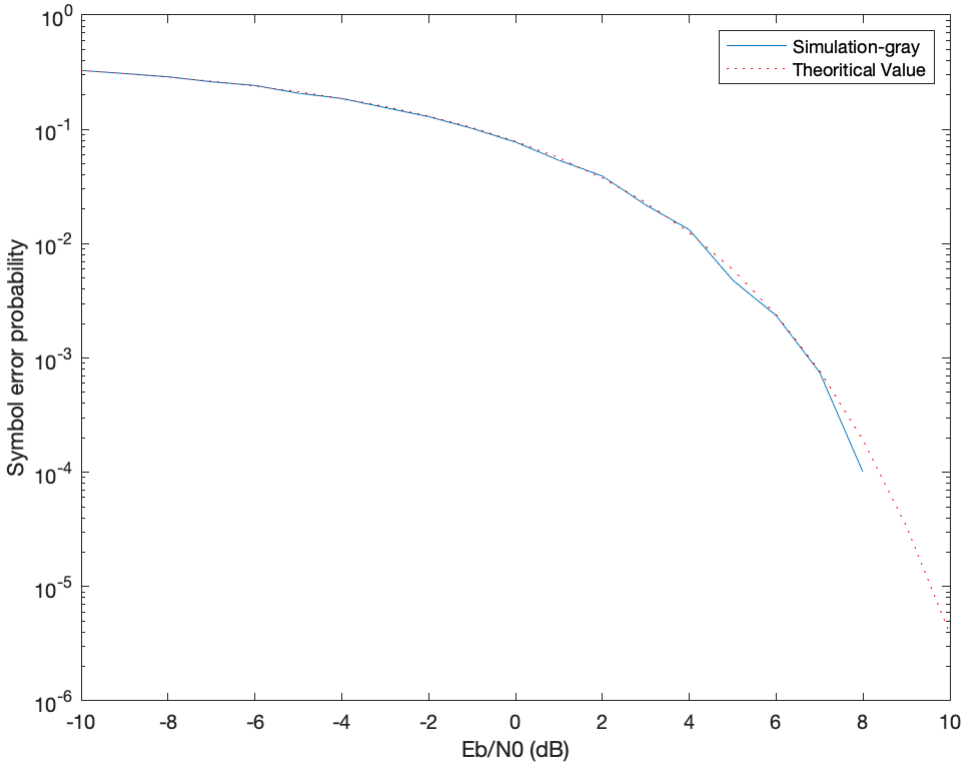
- Demodualtion and Bit Error Rate Calculation-

```
ebnodb = -10:10;     % SNR array varying in the range of -10 to +10 dB in steps of 1 dB
number_snrs = length(ebnodb);
perr_est = zeros(number_snrs,1);    % Declaring array having all zeroes

%Demodulation
for n=1:number_snrs
    sigma = sqrt(1/(4*(10^(ebnodb(n)/10))));
    w = sigma*randn(num/2,1)+1i*sigma*randn(num/2,1);  %Random 2D gaussian noise genera
    r = QPSK+w;            %Received signal
%Method for Demodulation(without using for loops)-
% creating array of  size (num/2,4) with
% each row as constellation array
arr1=ones(num/2,4).*const;
% creating array size (num,4) with
% each column as received signal array
arr2=[r,r,r,r];
% finding distance between
% constellation points and received signal
% idx stores index(1,2,3,4) of min of row of distance array
[dist,idx]=min(abs(arr1-arr2),[],2);
% using index to get demodulated signal bit array
% idx=1 -> 00
% idx=2 -> 01
% idx=3 -> 11
% idx=4 -> 10
demod=zeros(length(msg),1);
    for j=1:length(idx)
            if idx(j)==1
                demod(2*j-1:2*j)=[0,0];
            end
            if idx(j)==2
                demod(2*j-1:2*j)=[0,1];
            end
            if idx(j)==3
                demod(2*j-1:2*j)=[1,1];
            end
            if idx(j)==4
                demod(2*j-1:2*j)=[1,0];
            end
    end
% comparing demodulated and message bit array
% calculating Bit Error Rate for each SNR
perr_est(n)=sum(demod~=msg)/num;
end
```

- Plotting-

```
semilogy(ebnodb,perr_est); % plotting error from simulation
hold on;
%COMPARE WITH THEORETICAL VALUES USING EQUATION BASED ON Q-Func
snro = 10.^(ebnodb/10); % raw SNR values
perr_th = qfunc(sqrt(2*snro)); % theoretical error (with gray coding)
semilogy(ebnodb,perr_th,':r'); % plotting theoretical error
xlabel('Eb/N0 (dB)');
ylabel('Symbol error probability');
legend('Simulation-gray','Theoritical Value','Location','NorthEast');
hold off;
```



- (B) QPSK Modulation (without gray labelling)-

```
%QPSK modulation
%00 ->  1+j
%01 -> -1+j
%10 -> -1-j
%11 ->  1-j
QPSK=zeros(length(msg)/2,1);
%definition of the QPSK symbols using Gray coding.
for n=1:length(msg)/2
    tmp=msg(2*n-1);
    tmp1=msg(2*n);
```

```
    if (tmp==0)&(tmp1==0)
        QPSK(n)=exp(1i*pi/4);%45 degrees
    end
    if (tmp==0)&(tmp1==1)
        QPSK(n)=exp(1i*3*pi/4);%135 degrees
    end
    if (tmp==1)&(tmp1==0)
        QPSK(n)=exp(1i*5*pi/4);%225 degrees
    end
    if (tmp==1)&(tmp1==1)
        QPSK(n)=exp(1i*7*pi/4);%315 degrees
    end
end
```

- Demodualtion and Bit Error Rate Calculation-

```
ebnodb = -10:10;     % SNR array varying in the range of -10 to +10 dB in steps of 1 dB
number_snrs = length(ebnodb);
perr_est = zeros(number_snrs,1);    % Declaring array having all zeroes

%Demodulation
for n=1:number_snrs
    sigma = sqrt(1/(4*(10^(ebnodb(n)/10))));
    w = sigma*randn(num/2,1)+1i*sigma*randn(num/2,1);  %Random 2D gaussian noise genera
    r = QPSK+w;            %Received signal
%Method for Demodulation(without using for loops)-
% creating array of  size (num/2,4) with
% each row as constellation array
arr1=ones(num/2,4).*const;
% creating array size (num,4) with
% each column as received signal array
arr2=[r,r,r,r];
% finding distance between
% constellation points and received signal
% idx stores index(1,2,3,4) of min of row of distance array
[dist,idx]=min(abs(arr1-arr2),[],2);
% using index to get demodulated signal bit array
% idx=1 -> 00
% idx=2 -> 01
% idx=3 -> 11
% idx=4 -> 10
demod=zeros(length(msg),1);
    for j=1:length(idx)
            if idx(j)==1
                demod(2*j-1:2*j)=[0,0];
            end
            if idx(j)==2
                demod(2*j-1:2*j)=[0,1];
            end
            if idx(j)==3
                demod(2*j-1:2*j)=[1,0];
```

```
            end
            if idx(j)==4
                demod(2*j-1:2*j)=[1,1];
            end
    end
% comparing demodulated and message bit array
% calculating Bit Error Rate for each SNR
perr_est(n)=sum(demod~=msg)/num;
end
```

- Plotting-

```
semilogy(ebnodb,perr_est); % plotting error from simulation
hold on;
%COMPARE WITH THEORETICAL VALUES USING EQUATION BASED ON Q-Func
snro = 10.^(ebnodb/10); % raw SNR values
perr_th = 4*qfunc(sqrt(2*snro))/3; % theoretical error (without gray coding)
semilogy(ebnodb,perr_th,':r'); % plotting theoretical error
xlabel('Eb/N0 (dB)');
ylabel('Symbol error probability');
legend('Simulation-non gray','Theoritical Value','Location','NorthEast');
hold off;
```