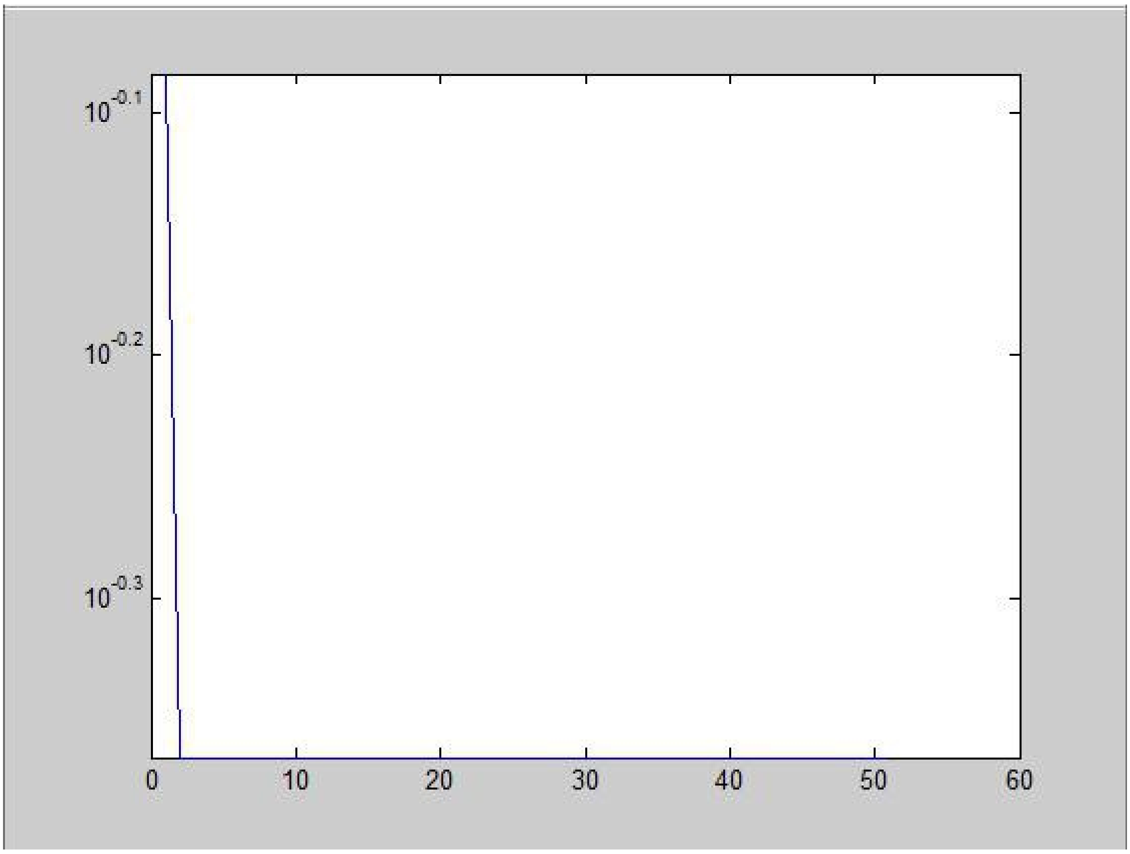
Problem -1

- In addition to the all files given , I have created another class named PSK and other 3 functions named BER,transmit & receive respectively.
- In main file named BasebandAWGN_ExamP1 , I declared some new variables like ebn0db , varrn etc which are used for calculation of BER and I considered 51 values of ebn0 for the same.
- Binary sequence is also adjusted as to make the matrix dimensions same.
- NumSybs term is rounded off to the nearest whole number.
- Changes in the first for loop in the file BasebandAWGN_ExamP1 are made , as the used modulation is 8-PSK. Different 8 cases are considered.
- `yawgnbaseforloop` variable is defined to store the 51 values of noise level.
- Constructor PSK is called. This class has all the variables defined required in functions transmit, receive and BER.
- Transmit function is called.
  - Binary stream is convolved with R-cosine filter.
  - Size of the matrix is adjusted to 4000.
  - Signal is added with noise and stored in a variable named data.
  - Data again convolved with R-cosine pulse wave which is used as a Matched Filter.
  - 4000 bits of data is considered and its absolute value is found out.
  - Using Maximum Likelihood Detection Criteria, decision is taken whether transmitted bit is 0 or 1. Databits are returned in parameter z.
- This data is passed to the another function called receiver.
  - Comparison between received bits and binary sequence is done. 3 bits at a time are considered using 8-PSK detector.
  - Compared values are returned in main file and name of the variable , where I have stored the received bits is rdata.
- BER function is called from the main file.
  - Logic of transmit and receiver function is combined here. Once we get the compared values as in above case,number of errors are determined by counting number of 1's in a sequence.
  - Using a loop, total number of errors is calculated.
  - BER is calculated by diving total number of errors with actual transmitted bits.
  - Curve for BER vs EbN0 is plot.



MATLAB CODE :

BasebandAWGN_ExamP1P File :

```
%%%%%%%%%%%%%%%%%%%%%%%%%
%BasebandGen Script  EE5183
%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
clear classes;
clear all;
clc;
dbstop if error  % stop if an error occurs
clear all  %  clears work space between runs

rcrolloff=1.0;   %rolloff Factor for nyquist pulse
%symbolrate=320e3;  %desired symbol rate
symbolrate=64e3;  %desired symbol rate
%NumSamplesTxPulse=200; %Warning the program can modify this value (pulse
length)!
NumSamplesTxPulse=64; %Warning the program can modify this value (pulse
length)!
SampleLaunchPeriod=63;  % period in samples between symbol launches
EbNodB=10;
ebn0db=(1:1:51);   % array is considered to plot agaist BER
receivedbits=[];
ebn0dblin=10.^(ebn0db/10);
```

```
varrn=(1/2)./(ebn0dblin);                         % array of noise created for
plotting BER graph
%txfinal=obj.txpulse;
%NumBitsPerSymbol=2;  %2-Tupple bit QPSK
NumBitsPerSymbol=3;  %3-Tupple bit PSK
%NumberSourceBits=40;
NumberSourceBits = 4000; %information source
%NumberSourceBits = 500; %information source
Es=1;  % energy per symbol
EbNoLin=10^(EbNodB/10);
%vn=(1/2)/EbNoLin;

% fc is set below: carrier frequency
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s = RandStream('mt19937ar','Seed',0);  %Do not modify this line
%disp(s);
savedState=s.State;  %Do not modify this line
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
binary_sequence=randi(s,[0,1],1, NumberSourceBits);
binseqpattern=binary_sequence(1:51);                     % 10 bits of pattern
are generated for testing purpose ( 51 bits )
%sigplusnoisearray=binseqpattern+varrn;                   % 10 bit signal added
with noise
NumSymbs1=length(binary_sequence)/NumBitsPerSymbol;  %Number of symbols
NumSymbs=round(NumSymbs1);                % rounded of as the above division
giving ans in decimals
b=zeros(1, NumSymbs);
inputdata=[];
for k=1:NumSymbs
    %n-tupple: (e.g. 2-tupple b1,b0 --> 2*b1+b0 mapping to complex coeff.)
    switch binary_sequence((k-
1)*NumBitsPerSymbol+1:k*NumBitsPerSymbol)*[4,2,1]';
        % QPSK Mapping Defined
        case 0
            b(k)=exp(1i*((2*pi/8)*0));
        case 2
            b(k)=exp(1i*((2*pi/8)*1));
        case 3
            b(k)=exp(1i*((2*pi/8)*2));
        case 7
            b(k)=exp(1i*((2*pi/8)*3));
        case 6
            b(k)=exp(1i*((2*pi/8)*4));
        case 4
            b(k)=exp(1i*((2*pi/8)*5));
        case 5
            b(k)=exp(1i*((2*pi/8)*6));
        case 1
            b(k)=exp(1i*((2*pi/8)*7));
    end
end
%%%%%%%%%
%%%%%%%%%%%%%%%%%
p.rcrolloff=rcrolloff;
p.symbolrate=symbolrate;
p.NumSamplesTxPulse=NumSamplesTxPulse;
p.SampleLaunchPeriod=SampleLaunchPeriod;
                          %%%%%%%%%%%%%%%%%%%%%%% added recently
a=BasebandGenNew(p,Es,b);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
upsamp=1;
y=resample(a.basebandsig, upsamp, 1);
vn=(1/2)/EbNoLin;
txpulse_ts2=resample(a.txpulse, upsamp, 1);
Ts2=a.Ts/upsamp;
yawgnbase=y+sqrt(vn/2)*randn(size(y))+1i*sqrt(vn/2)*randn(size(y));
txpulsefinal=a.txpulse;
for t=1:1:51
    yawgnbaseforloop=y+sqrt(varrn(t)/2)*randn(size(y))
+1i*sqrt(varrn(t)/2)*randn(size(y));
end
noisearray=yawgnbaseforloop;
noisearrayfinal=noisearray(1:51);                %noise to be added in loop
with parameters 51
yawgnbasefinal=yawgnbase(1:4000);
%yawgnbasefinal=yawgnbase(1:NumberSourceBits);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  q.yawgnbasefinal=yawgnbasefinal;
 %q.vn=vn;
 q.binary_sequence=binary_sequence;
 q.yawgnbase=yawgnbase;
 q.ebn0db=ebn0db;
 q.y=y;
 q.binseqpattern=binseqpattern;
 q.varrn=varrn;
 q.receivedbits=receivedbits;
 q.noisearrayfinal=noisearrayfinal;
 q.txpulsefinal=txpulsefinal;
%a=BasebandGenNew(p,Es,b);
r=PSK(p,b,q);

%%%%%%%%%%%%%%%%%%%%%%%%%%

% upsamp=1;
% y=resample(a.basebandsig, upsamp, 1);
%
% txpulse_ts2=resample(a.txpulse, upsamp, 1);
% Ts2=a.Ts/upsamp;
z=[];
z=transmit(r);                          % data transmission with AWGN Noise
disp(z);

%%%%%%%%%%%%%%%%  Modifications for Receiver Function %%%%%%%%%%%%%%%%%%%%
rdata=receiver(r,z);                    % transmitted data with AWGN noise
added is input to the receiver
disp(rdata);
p=0;
for k=1:1:4000
    if rdata(k)==1
        p=p+1;
    end
end
disp(p);                       % no of 1's in a matrix = no. of errors
errorcal=[];
errorcal=BER(r);
length(errorcal);

%q.z=z;
%rdata=PSK(z);
%receiveddata=receiver(r);
%FOR BASEBAND MODE USE the AWGN Channel
```

```
% EbNoLin=10^(EbNodB/10);
% vn=(1/2)/EbNoLin;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Modifications for BER function%%%%%%%%%%%%%%%%%%%%%%%%%%%%




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%yawgnbase=y+sqrt(vn/2)*randn(size(y))+1i*sqrt(vn/2)*randn(size(y));

figure(6);
xax1=Ts2*(0:length(yawgnbase)-1);
subplot(2,1,1), plot(xax1, real(yawgnbase));
xlabel('sample time (sec)');
ylabel('Real');
titletext=strcat({'Baseband Signal in AWGN, EbNo='},{num2str(EbNodB)},...
    {'dB, SamplingFrequency= '}, {num2str(1/Ts2)});
title(titletext);
subplot(2,1,2), plot(xax1, imag(yawgnbase));
xlabel('sample time (sec)');
ylabel('Imag');


figure(7);
fs=1/Ts2;
NumPointsFFT=2048;
xax = ([0:NumPointsFFT-1]/NumPointsFFT)*fs;
yaxdB=20*log10(abs(fft(yawgnbase,NumPointsFFT)));
plot([xax-xax(end)/2],
[yaxdB(NumPointsFFT/2+1:NumPointsFFT),yaxdB(1:NumPointsFFT/2)], 'b');
titletext=strcat({'FFT of QPSK Baseband Signal: Rsymb='},
{num2str(a.symbolrate)},{' Rolloff= '}, {num2str(a.rcrolloff)},...
    {' SamplingRate= '}, {num2str(fs)});
xtext='Frequency in Hz';
ytext='Baseband Signal Power in dB';
xlabel(xtext);
ylabel(ytext);
title(titletext);

x=1:1:1326;
  figure(10);
  semilogy(ebn0db,errorcal);
%%%%%%%%%%%%%Add a Method BasebandDemod to Class BasebandGenNew%%%%%%%%%
%We can alternatively add a function to the Basebandmodel if desired


% %ccc=BasebandDemod(arg1, arg2, ...argN);
% %numerrors=sum(xor(ccc.brecov, binary_sequence))
```

PSK Class :

```
classdef PSK < handle
    %UNTITLED2 Summary of this class goes here
```

```
    %   Detailed explanation goes here

    properties
        rcrolloff=[];
        symbolrate=[];
        NumSamplesTxPulse=[];
        SampleLaunchPeriod=[];
        b=[];
        noisearrayfinal=[];
       % inputdata=[];
      %  trans;
         ebn0db=[];
         yawgnbase=[];
         yawgnbasefinal=[];
         binary_sequence=[];
         y=[];
        % z=[];
        binseqpattern=[];
        varrn=[];
        receivedbits=[];
        txpulsefinal=[];
    end

    methods
        function obj1=PSK(varargin)
            for k=1:nargin
                switch k
                    case 1
                        obj1.rcrolloff=varargin{1}.rcrolloff;
                        obj1.symbolrate=varargin{1}.symbolrate;

obj1.NumSamplesTxPulse=varargin{1}.NumSamplesTxPulse;

obj1.SampleLaunchPeriod=varargin{1}.SampleLaunchPeriod;

                    case 2
                        obj1.b=varargin{2};
                    case 3
                        obj1.yawgnbase=varargin{3}.yawgnbase;
                        obj1.yawgnbasefinal=varargin{3}.yawgnbasefinal;
                        obj1.binary_sequence=varargin{3}.binary_sequence;
                        obj1.y=varargin{3}.y;
                        obj1.binseqpattern=varargin{3}.binseqpattern;
                        obj1.varrn=varargin{3}.varrn;
                        obj1.ebn0db=varargin{3}.ebn0db;
                        obj1.receivedbits=varargin{3}.receivedbits;
                        obj1.noisearrayfinal=varargin{3}.noisearrayfinal;
                        obj1.txpulsefinal=varargin{3}.txpulsefinal;
                end
            end
        end
    end

end
```

Transmit Function :

```
function varargout = transmit( obj1 )
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%mixeddata=conv(obj1.binary_sequence,a.txpulse);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
convdata=conv(obj1.binary_sequence,obj1.txpulsefinal);
convdatafinal=convdata(1:4000);
%data=obj1.binary_sequence+obj1.yawgnbasefinal;           % data multiplied
with AWGN Noise & value of the matrix returned in variable z
data=convdatafinal+obj1.yawgnbasefinal;           % data multiplied with
AWGN Noise & value of the matrix returned in variable z
matcheddata=conv(data,obj1.txpulsefinal);
matcheddatafinal=matcheddata(1:4000);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

magdata=zeros(1,4000);
 for k=1:4000
    magdata(k)=abs(matcheddatafinal(k));
 end
 databits=zeros(1,4000);
 for l=1:4000
     if magdata(l)>1                          % max likelihood detection is
used to determine 0 or 1
         databits(l)=1;
         l=l+1;
     else
         databits(l)=0;
         l=l+1;
     end
 end
%varargout{1}=databits;                        % detected o/p is passed to
varb z
varargout{1}=databits;
end
```

Receiver Function :

```
function varargout = receiver(obj1, z )
% %UNTITLED Summary of this function goes here
% %   Detailed explanation goes here
 receiveddata=z;
 compare=zeros(1,4000);
 %noiseremoval=receiveddata-obj1.yawgnbasefinal;     % Noise removed from
the received data
       for k=1:1333
           compare((k-1)*3+1:k*3)=xor(receiveddata((k-
1)*3+1:k*3),obj1.binary_sequence((k-1)*3+1:k*3));  % 8-ary PSK detector is
designed which compares 3 bits at a time
       end
```

```
 varargout{1}=compare;     % comparison between transmitted & received bits
(XOR o/p is found out) & passed in variable rdata
 a=obj1.symbolrate;

 end
```

BER Function :

```
function varargout = BER( obj1 )
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
noe2=[];
ber2=[];
%%%%%%%%%%%%%  Design for transmitter considering ebn0%%%%%%%%%%%%%%%%%%
   for ebn0db=1:1:51
sigplusnoisearray=obj1.binseqpattern+obj1.noisearrayfinal;
%sigplusnoisearray=obj1.binseqpattern+obj1.varrn;
r=sigplusnoisearray-(ebn0db/0.51);
magdata=zeros(1,51);
       for k=1:51
          %magdata(k)=abs(sigplusnoisearray(k));
          magdata(k)=abs(r(k));
       end
       databits=zeros(1,51);
             for l=1:51
                if magdata(l)>1                          % max likelihood
detection is used to determine 0 or 1
                databits(l)=1;
                l=l+1;
                else
                databits(l)=0;
                l=l+1;
                end
             end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

 %%%%%%%%%%%%%  Design for receiver considering ebn0 loop %%%%%%%%%%%%%%%%
 compare=zeros(1,4000);
   for k=1:17
       compare((k-1)*3+1:k*3)=xor(databits((k-
1)*3+1:k*3),obj1.binseqpattern((k-1)*3+1:k*3));  % 8-ary PSK detector is
designed which compares 3 bits at a time
   end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %%%%%%%%%%%%%%%% Bit Error Calculation%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   noe=0;
for k=1:1:4000
    if compare(k)==1
        noe=noe+1;
    end
end
noe2=[noe2,noe];
nod=51;
biterrorrate=noe2/nod;
%ber2=[ber2,ber];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    end
    ber2=[ber2,biterrorrate];
varargout{1}=ber2;
end
```