**EE-5283-002**

**Spread Spectrum Communication**

**&**

**GPS (Global Positioning System)**

**HomeWork #1**

**Submitted By**

**Name:**  Nimish Joshi

**Banner Id:**  @01482152

* **Homework Topic #1**
* **Probability Distribution & Curves:**

A probability distribution is a function or rule that assigns probabilities to each value of a random variable. The distribution may in some cases be listed. In other cases it is presented as a graph. A probability distribution can be graphed, and sometimes this helps to show us features of the distribution that were not apparent from just reading the list of probabilities. The random variable is plotted along the *x*-axis, and the corresponding probability is plotted along the *y* - axis.

Whereas a curve that describes the distribution of probability over the values of a random variable is called as Probability Curve.

**1). Program: bpsk.m**

% Program 3-1

% bpsk.m

%

% Simulation program to realize BPSK transmission system

% Programmed by H.Harada and T.Yamamura,

%

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Preparation part \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

sr=256000.0; % Symbol rate

ml=1; % Number of modulation levels

br=sr.\*ml; % Bit rate (=symbol rate in this case)

nd = 1000; % Number of symbols that simulates in each loop

ebn0=0;

while (ebn0 <= 12)

ebn0=ebn0+1;

IPOINT=8; % Number of oversamples

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Filter initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

irfn=21; % Number of filter taps

alfs=0.5; % Rolloff factor

[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1); %Transmitter filter coefficients

[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0); %Receiver filter coefficients

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START CALCULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

nloop=100; % Number of simulation loops

noe = 0; % Number of error data

nod = 0; % Number of transmitted data

for iii=1:nloop

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data generation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data=rand(1,nd)>0.5; % rand: built in function

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BPSK Modulation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data1=data.\*2-1; % 0,1 set to -1,1 set

[data2] = oversamp( data1, nd , IPOINT) ;

data3 = conv(data2,xh); % conv: built in function

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Attenuation Calculation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

spow=sum(data3.\*data3)/nd; % estimating signal power from the data sequence

attn=0.5\*spow\*sr/br\*10.^(-ebn0/10); % given E\_b/N\_o get the attenuation

attn=sqrt(attn);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fading channel \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Generated data are fed into a fading simulator

% In the case of BPSK, only Ich data are fed into fading counter

% [ifade,qfade]=sefade(data3,zeros(1,length(data3)),itau,dlvl,th1,n0,itnd1,now1,length(data3),tstp,fd,flat);

% Updata fading counter

%itnd1 = itnd1+ itnd0;

%\*\*\*\*\*\*\*\*\*\*\*\* Add White Gaussian Noise (AWGN) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

inoise=randn(1,length(data3)).\*attn; % randn: built in function

data4=data3+inoise;

data5=conv(data4,xh2); % conv: built in function

sampl=irfn\*IPOINT+1;

% be careful here, instead of the parameter IPOINT, 8 is used, IPOINT

% change could result in errors

data6 = data5(sampl:8:8\*nd+sampl-1);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BPSK Demodulation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

demodata=data6 > 0; % very simple demodulation

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Bit Error Rate (BER) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

noe2=sum(abs(data-demodata)); % sum: built in function

nod2=length(data); % length: built in function

noe=noe+noe2;

nod=nod+nod2;

fprintf('%d\t%e\n',iii,noe2/nod2);

end % for iii=1:nloop

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output result \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ber(ebn0) = noe/nod;

end

figure(1)

semilogy(ber,'r-\*');

xlabel('Eb/n0(dB)-time');

ylabel('BER:-ampliture');

title('Probability Curve for BPSK')

whitebg('c')

grid on;

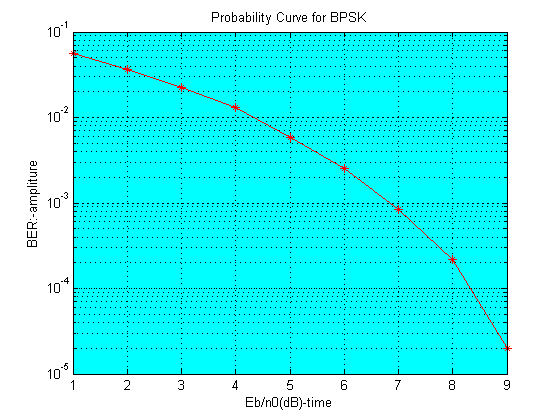
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);

fid = fopen('BERbpsk.dat','a');

fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);

fclose(fid);

%----------------END OF THE PROGRAM---------------------%



* Above is the code for the bpsk probability curve, in which we used to loop named ‘while’ loop and ‘for’ loop respectively.
* There is an assumption in the beginning of the program, that we oversample the receiving filter at the receiver.
* In order to get desire curve, several functions are used, like ‘*comb.m’,’ deplay.m’* etc.
* To determine the relation between Eb/N0 and BER, we keep Eb/N0 constant and will vary the other value.
* **What is EB/N0 ?**

Eb/No is the measure of signal to noise ratio for a digital communication system. It is measured at the input to the receiver and is used as the basic measure of how strong the signal is. Different forms of modulation -- BPSK, QPSK, QAM, etc. -- have different curves of theoretical bit error rates versus Eb/No as shown in Figure 1. These curves show the communications engineer the best performance that can be achieved across a digital link with a given amount of RF power.

**2). Program: Qpsk.m**

% Program 3-5

% qpsk.m

%

% Simulation program to realize QPSK transmission system

%

% Programmed by H.Harada and T.Yamamura

%

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Preparation part \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

sr=256000.0; % Symbol rate

ml=2; % ml:Number of modulation levels (BPSK:ml=1, QPSK:ml=2, 16QAM:ml=4)

br=sr .\* ml; % Bit rate

nd = 1000; % Number of symbols that simulates in each loop

ebn0=0; % Eb/N0

while (ebn0 <= 11)

ebn0=ebn0+1;

IPOINT=8; % Number of oversamples

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Filter initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

irfn=21; % Number of taps

alfs=0.5; % Rolloff factor

[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1); %Transmitter filter coefficients

[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0); %Receiver filter coefficients

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START CALCULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

nloop=100; % Number of simulation loops

noe = 0; % Number of error data

nod = 0; % Number of transmitted data

for iii=1:nloop

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data generation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data1=rand(1,nd\*ml)>0.5; % rand: built in function

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QPSK Modulation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ich,qch]=qpskmod(data1,1,nd,ml);

[ich1,qch1]= compoversamp(ich,qch,length(ich),IPOINT);

[ich2,qch2]= compconv(ich1,qch1,xh);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Attenuation Calculation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

spow=sum(ich2.\*ich2+qch2.\*qch2)/nd; % sum: built in function

attn=0.5\*spow\*sr/br\*10.^(-ebn0/10);

attn=sqrt(attn); % sqrt: built in function

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fading channel \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Generated data are fed into a fading simulator

% [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,itnd1,now1,length(ich2),tstp,fd,flat);

% Updata fading counter

%itnd1 = itnd1+ itnd0;

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Add White Gaussian Noise (AWGN) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ich3,qch3]= comb(ich2,qch2,attn);% add white gaussian noise

[ich4,qch4]= compconv(ich3,qch3,xh2);

syncpoint=irfn\*IPOINT+1;

ich5=ich4(syncpoint:IPOINT:length(ich4));

qch5=qch4(syncpoint:IPOINT:length(qch4));

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QPSK Demodulation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[demodata]=qpskdemod(ich5,qch5,1,nd,ml);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Bit Error Rate (BER) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

noe2=sum(abs(data1-demodata)); % sum: built in function

nod2=length(data1); % length: built in function

noe=noe+noe2;

nod=nod+nod2;

fprintf('%d\t%e\n',iii,noe2/nod2); % fprintf: built in function

end % for iii=1:nloop

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output result \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ber(ebn0) = noe/nod;

end

figure(1)

semilogy(ber,'r-\*');

xlabel('Eb/n0(dB)-time');

ylabel('BER:-ampliture');

title('Probability Curve for QPSK')

whitebg('w')

grid on;

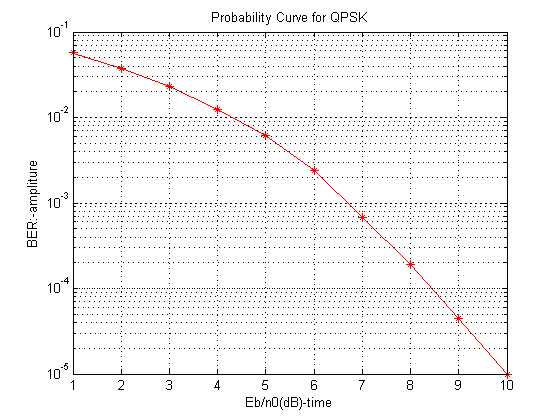
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod); % fprintf: built in function

fid = fopen('BERqpsk.dat','a');

fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod); % fprintf: built in function

fclose(fid);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end of file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

****

* **About Qpsk:** Two BPSK signals generates QPSK signals. We use two orthogonal career signals. Here from the above diagram we can see that simulation result is quite similar to the theoretical values

**3). Program: Oqpsk.m**

% Program 3-11

% oqpsk.m

%

% Simulation program to realize OQPSK transmission system

%

% Programmed by H.Harada and T.Yamamura

%

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Preparation part \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

sr=256000.0; % Symbol rate

ml=2; % ml:Number of modulation levels (BPSK:ml=1, QPSK:ml=2, 16QAM:ml=4)

br=sr.\*ml; % bit rate

nd = 1000; % Number of symbols that simulates in each loop

ebn0=0; % Eb/N0

while (ebn0 <= 12)

ebn0=ebn0+1;

IPOINT=8; % Number of oversamples

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Filter initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

irfn=21; % Number of taps

alfs=0.5; % Rolloff factor

[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1); %Transmitter filter coefficients

[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0); %Receiver filter coefficients

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START CALCULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

nloop=100; % Number of simulation loops

noe = 0; % Number of error data

nod = 0; % Number of transmitted data

for iii=1:nloop

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data generation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data1=rand(1,nd\*ml)>0.5; % rand: built in function

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OQPSK Modulation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ich,qch]=qpskmod(data1,1,nd,ml);

[ich1,qch1]=compoversamp(ich,qch,length(ich),IPOINT);

ich21=[ich1 zeros(1,IPOINT/2)];

qch21=[zeros(1,IPOINT/2) qch1]; % introducing half a symbol delay

[ich2, qch2]=compconv(ich21,qch21,xh);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Attenuation Calculation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

spow=sum(ich2.\*ich2+qch2.\*qch2)/nd; % sum: built in function

attn=0.5\*spow\*sr/br\*10.^(-ebn0/10);

attn=sqrt(attn); % sqrt: built in function

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fading channel \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Generated data are fed into a fading simulator

% [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,itnd1,now1,length(ich1),tstp,fd,flat);

% Updata fading counter

%itnd1 = itnd1+ itnd0;

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Add White Gaussian Noise (AWGN) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[ich3,qch3]= comb(ich2,qch2,attn);% add white gaussian noise

[ich4,qch4]= compconv(ich3,qch3,xh2);

syncpoint=irfn\*IPOINT+1;

ich5=ich4(syncpoint:IPOINT:length(ich4));

qch5=qch4(syncpoint+IPOINT/2:IPOINT:length(qch4));

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OQPSK Demodulation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[demodata]=qpskdemod(ich5,qch5,1,nd,ml);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Bit Error Rate (BER) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

noe2=sum(abs(data1-demodata)); % sum: built in function

nod2=length(data1); % length: built in function

noe=noe+noe2;

nod=nod+nod2;

fprintf('%d\t%e\n',iii,noe2/nod2); % fprintf: built in function

end % for iii=1:nloop

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output result \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ber(ebn0) = noe/nod;

end

figure(1)

semilogy(ber,'k-\*');

xlabel('Eb/n0(dB)-time');

ylabel('BER:-ampliture');

title('Probability Curve for OQPSK')

whitebg('y')

grid on;

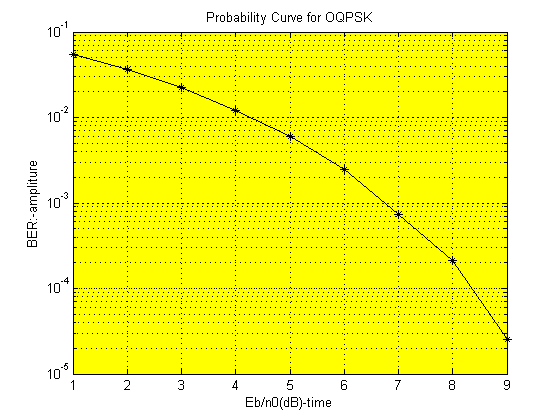
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod); % fprintf: built in function

fid = fopen('BERoqpsk.dat','a');

fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod); % fprintf: built in function

fclose(fid);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end of file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



* **Homework Topic #2**

\*Spectra of oversampled signal by computing FFT of the signal fragment

#Program:

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output result \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

figure(1)

plot(abs(fft(data2)))

title('Spectra of Oversampled signal using FFT')

whitebg('w')

grid on;

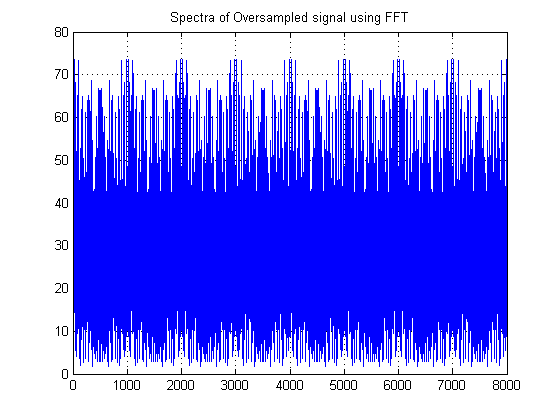
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);

fid = fopen('BERbpsk.dat','a');

fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);

fclose(fid);

%-------------------End OF THE PROGRAM-----------------%



**Figure: Spectra of Oversampled Signal**

* **Homework Topic #3**

\*Spectra Of the Signal\*

#Program:

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output result \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

figure(1)

plot(data4);

title('Spectra of the signal:- After noise added');

whitebg('w')

figure(2)

plot(data5);

title('Spectra of the signal:- After the receive filter');

whitebg('c')

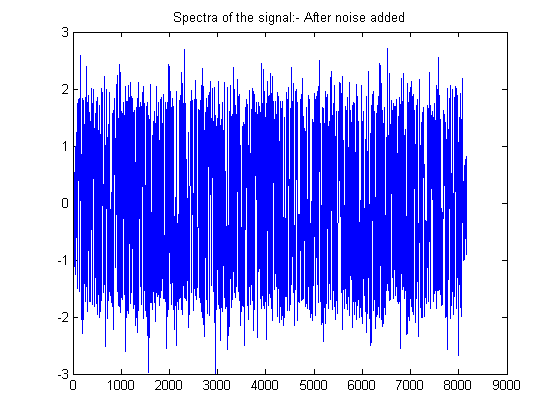
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);

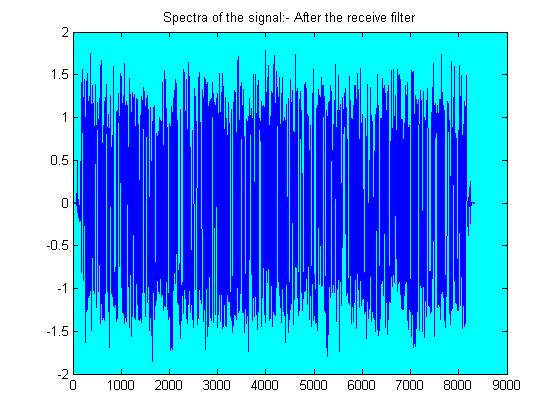
fid = fopen('BERbpsk.dat','a');

fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);

fclose(fid);

%-------------------End OF THE PROGRAM-----------------%





**Figure: Spectra Of the Signal 1)after noise added 2) after the receiver filter.**

* **Homework Topic #4**

\*Transmit and Receive Filter Shapes

#Program

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output result \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

figure(1)

plot(xh);

grid on;

title('transmitting filter shapes');

whitebg('y')

figure(2)

plot(xh2);

grid on;

title('receiving filter shapes');

whitebg('w')

fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);

fid = fopen('BERbpsk.dat','a');

fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);

fclose(fid);

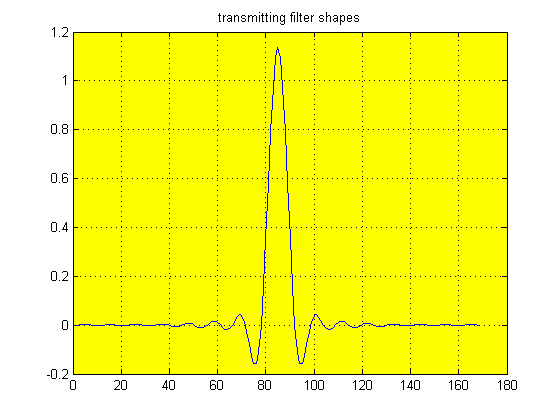
%-------------------End OF THE PROGRAM-----------------%

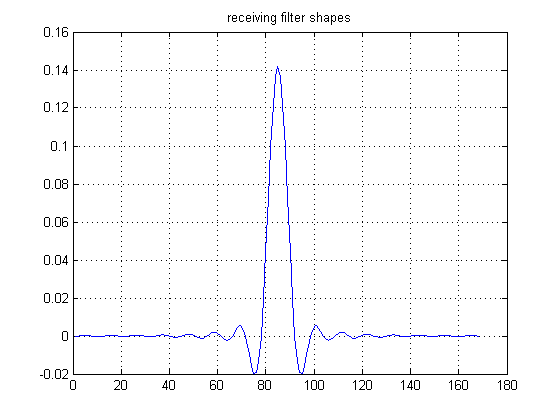
* **Pulse Shape Filter:**

Pulse shape filtering process isthe process of changing the waveform of transmitted pulses. Its purpose is to make the transmitted signal better suited to its purpose or the [communication channel](https://en.wikipedia.org/wiki/Communication_channel), typically by limiting the effective [bandwidth](https://en.wikipedia.org/wiki/Bandwidth_(signal_processing)) of the transmission. By filtering the transmitted pulses this way, the [inter-symbol interference](https://en.wikipedia.org/wiki/Intersymbol_interference) caused by the channel can be kept in control. In RF communication, pulse shaping is essential for making the signal fit in its frequency band.

* **Matched Filter:**

1. Matched filter is a theoretical frame work and not the name of a specific type of filter. It is an ideal filter which processes a received signal to minimize the effect of noise. Hence, it maximizes the signal to noise ratio (SNR) of the filtered signal.





**Figure: Transmit & Receive Filter Shapes**

* **Homework Topic #5**

\* Signal fragments after oversampling and after transmit filter\*

#Program:

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output result \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

figure(1)

plot(data2(1:100));

grid on;

title('Signal fragments after Oversampling Filter');

whitebg('y')

figure(2)

plot(xh(1:50));

grid on;

title('Signal Fragments after Transmit Filter');

whitebg('w')

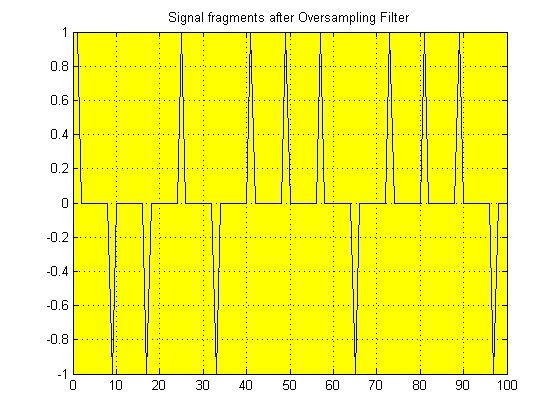
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);

fid = fopen('BERbpsk.dat','a');

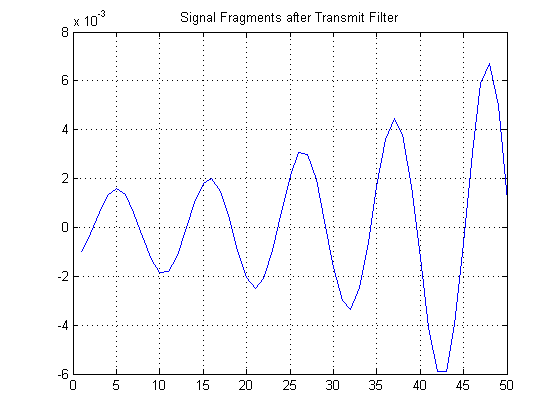
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);

fclose(fid);

%-------------------End OF THE PROGRAM-----------------%



**Figure: Signal fragments after Oversampling**



**Figure: Signal fragments after transmit filter**

* **Homework Topic: #6**

\*Fragments of Signals before and after Receive Filter\*

#Program:-

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output result \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

figure(1)

plot(data4(1:80));

grid on;

title('Fragment of the signal before the receiver filter');

whitebg('g')

figure(2)

plot(data5(1:70));

grid on;

title('Fragment of the signal after the receiver filter');

whitebg('c')

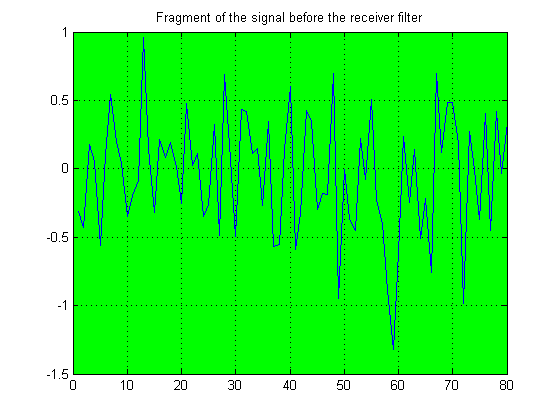
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);

fid = fopen('BERbpsk.dat','a');

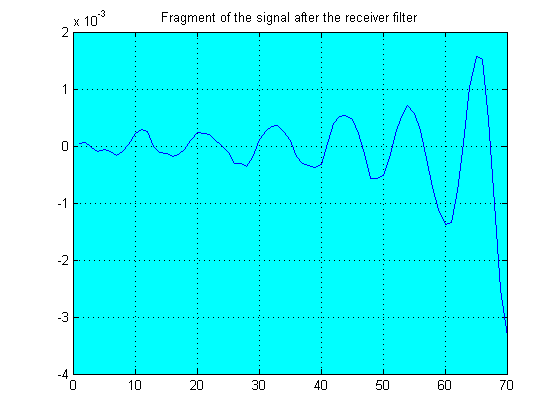
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);

fclose(fid);

%-------------------End OF THE PROGRAM-----------------%



-------------------------------------------------------------------------------------------------------------------



**Figure: Fragment of the Signal Before and after receive Filter**