Credit Hours System
ELCN316

Cairo University
Faculty of Engineering

MATLAB Simulation of Optimum Receiver for Detecting BPSK and BFSK Signals

Submitted by:

Marawan Kefah Fathi Mohamed Elbatel 1162419
Omar Essam-eldin Hassan Ibrahim 1162285

Submitted to:

Dr. Ahmed Khattab

Date: 10/01/2021

Abstract

Being modulated by a carrier wave of appropriate frequency, digital signals are transmitted in communication channels, bandpass in nature. Consequently, a receiver has to be wisely designed to distinguish the transmitted bits in the presence of noise with the least probability of error. This report first shows a sample pipeline visualization of a 100-bit random generated signal with a low SNR through the simulation. Then, it shows the same pipeline for a suitable SNR value for a more suitable number of bits 1000-bit. Finally, it gives analytical discussion for different signals scenarios of different number of transmitted bits in the MATLAB simulation of optimum receiver for detecting BPSK and BFSK signals.

TABLE OF CONTENTS

LIST OF FIGURES

1.Experiment Results

1.1 Signal Encoding & Modulation

The generated random sequence of binary bits is shown in Fig.1, irrespective to any modulating technique. These sequences are represented by the line code, Polar NRZ, rectangular pulses having amplitudes ±1, each lasting for a period of Tb. The effective bandwidth may be observed by plotting the PSD versus frequency shown in Fig.2, and taking the first null as the bandwidth, 0.0249 HZ. Then, the effective bandwidth can be computed by the equation,

$$Effective\ BW = \frac{R_b}{BW} = \frac{\frac{1}{T_b}}{0.0249} = \frac{\frac{1}{40}}{0.0249} = \frac{0.025}{0.0249} = 1.00401 \text{ bit per sec per HZ.}$$

We may also suggest that B.W of this sequence of rectangular pulses is the distance between the beginning of the signal to the first zero crossing. Thus, $Effective\ BW = \frac{R_b}{\frac{1}{T_b}} = \frac{R_b}{R_b} = 1$ bit per sec per HZ. The encoded signal also has a size of number of generated bits*time taken to send one bit.

,

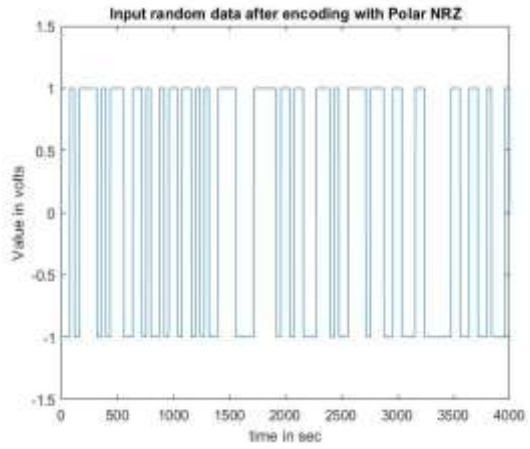

Fig 1. Polar NRZ Line encoding sample random input

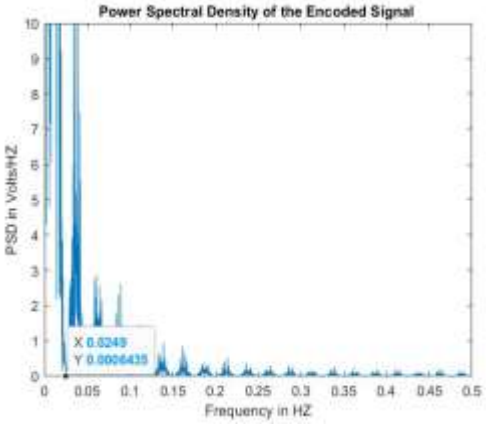

Fig 2. PSD of the Encoded Signal

This sequence of encoded pulses will be modulated using different carriers of amplitude A in the transmission either by FSK or by PSK to be sent via the assumed ideal-channel. A sample modulated signals for BPSK, BFSK are shown in Fig.3 and Fig 4 respectively.

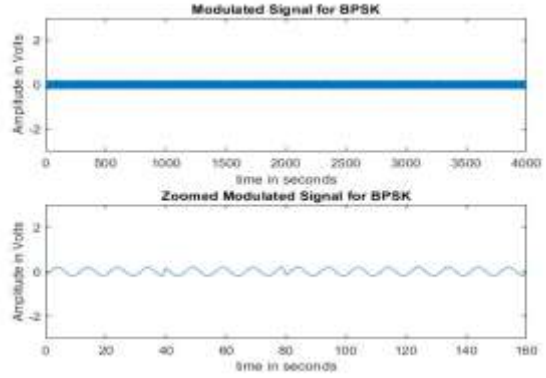

Fig 3. BPSK modulation of the encoded Signal

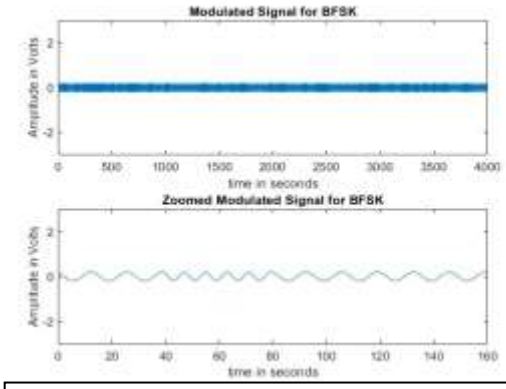

Fig 4. BFSK modulation of the encoded Signal

As clearly seen in the BPSK, a phase shift is widely observed at multiples of Tb to distinguish the sent signal. On the other side, the BFSK differentiate both signals by the frequency of the signal, the number of cycles it completes during the period Tb.

1.2 Noise in Channel

After successful modulation of the signal, we need to visualize the noise impact on that signal.

We only need to resemble the channel noise. This noise sequence has a two-sided power spectral density of $\frac{No}{2}$. In this project, we fix No = 2. Thus, the white and gaussian noise has a power spectral density $S_n(w) = \frac{N_o}{2} = \frac{2}{2} = 1$. The noise sequence is shown in Fig 5. the power of this noise sequence depends on the sampling period Ts, the smaller is Ts, the larger is the noise power.

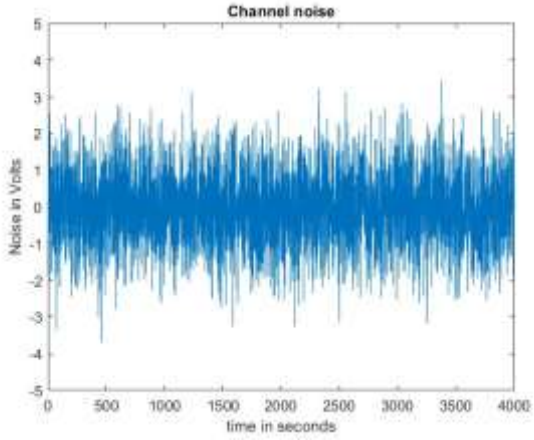

Fig 5. white and gaussian noise. $S_n(w) = 1$.

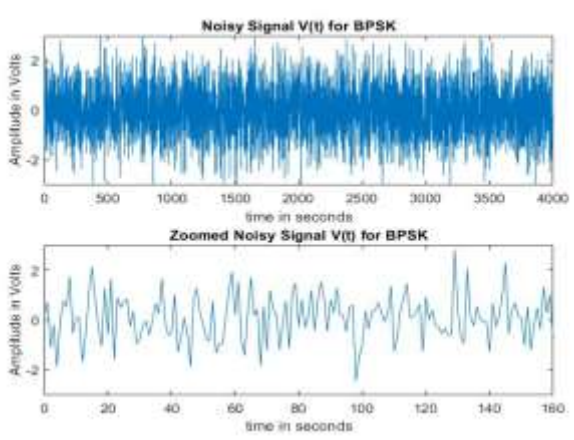

Fig 6. Modulated BPSK Signal with noise

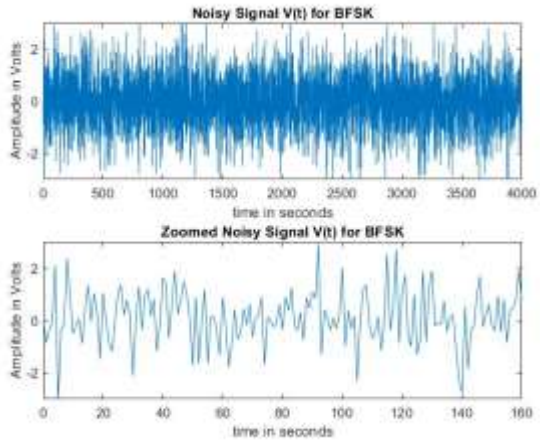

Fig 7. Modulated BFSK Signal with noise

With a fixed value of No = 2, for sample chosen SNRe of -4 dB (lowest in our simulation) using the value of A calculated. The modulated signals for both modulation techniques are shown in Fig.6 and Fig.7. As we used a low SNR for visualization, the noise impact can be seen on the plotted signals.

1.3 Receiver output & Behavior

Each of the noisy signal sequences is then passed through the matched filter respectively for each transmission method. the output of the matched-filter at the end of each Tb for each transmission scheme can be shown in Fig.8 and Fig.9.

For BPSK, one matched filter is used to distinguish the signals having the same frequency while different phases that may be resembled as +- in the threshold detector. Thus, a threshold of 0 can be used to detection and assignment of the received signal. On the other hand, two matched filters were needed for the BFSK with different frequencies, each of which designed to maximize the convolution output at Tb. Thus, at the threshold detector, it is sufficient to compare between both outputs before detection and assignment of the received signal.

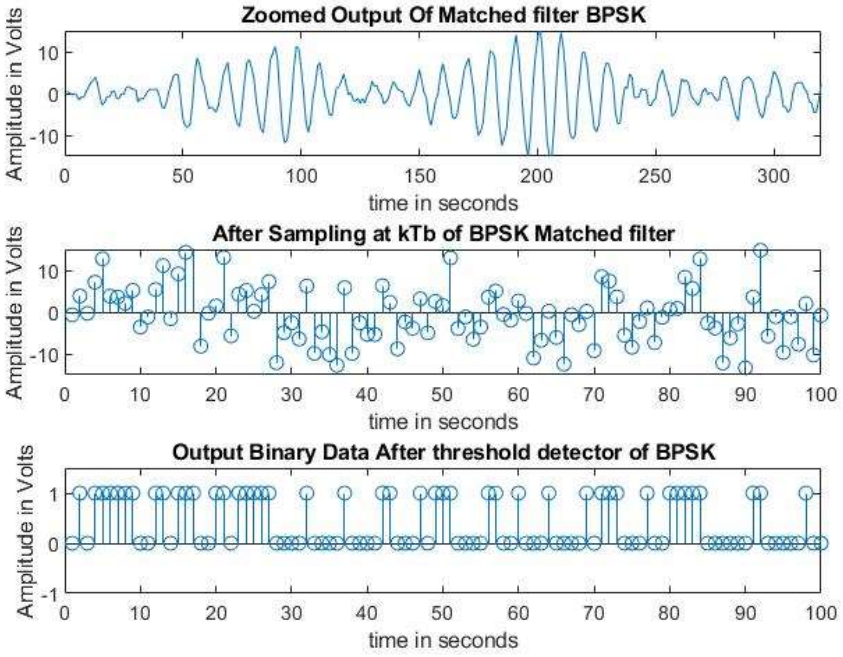

Fig 8. Receiver output BPSK Signal

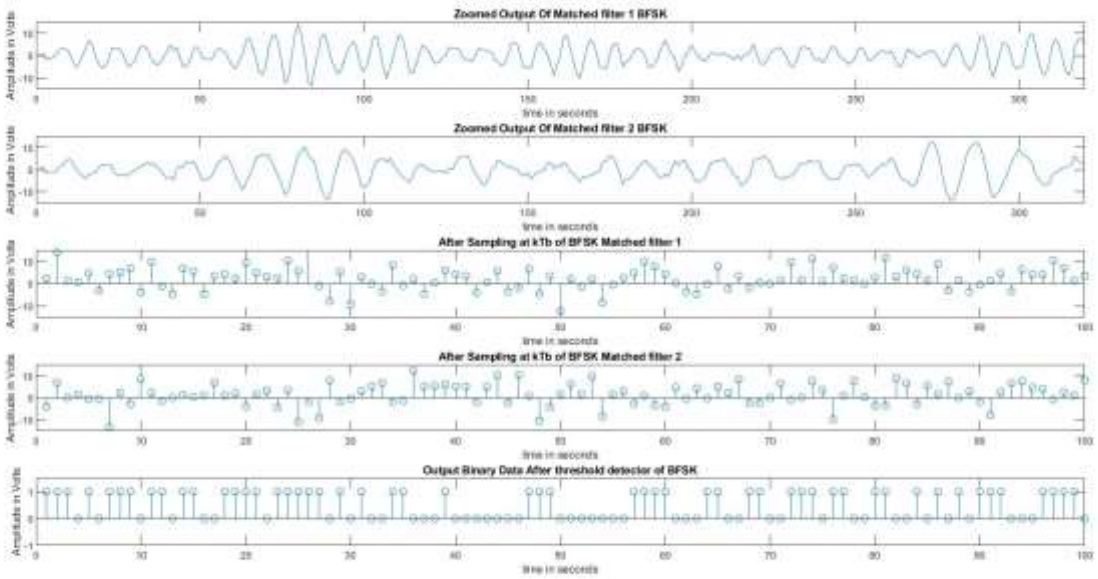

Fig 9. Receiver output BFSK Signal

The above visualization pipeline was shown for the lowest SNR=-4 and a low number of bits100 bits.

Fig.10 shows the pipeline for a suitable chosen SNR=3 for instance, for 1000 bits.
The bit error calculated for this setting scheme is

- $BERexp_{BPSK} = 0.029$
- $BERtheo_{BPSK} = 0.0229$

- $BERexp_{BFSK} = 0.063$
- $BERtheo_{BFSK} = 0.0789$

Since noise is a random process, repeating the same pipeline for 20 realizations of noise and calculating the average bit-error rate for the same settings.

- $AvgBERexp_{BPSK} = 0.0225$
- $AvgBERexp_{BFSK} = 0.0789$

- As noticed averaging the noise over the number of realizations gives a close estimate to the theoretical equation as it achieves a mean realization for the noise and the mean impact in our system.

- The average BER is closer to the theoretical than a single realization calculated BER. As a single realization exhibit the experiment on only one random noise rather than the mean..

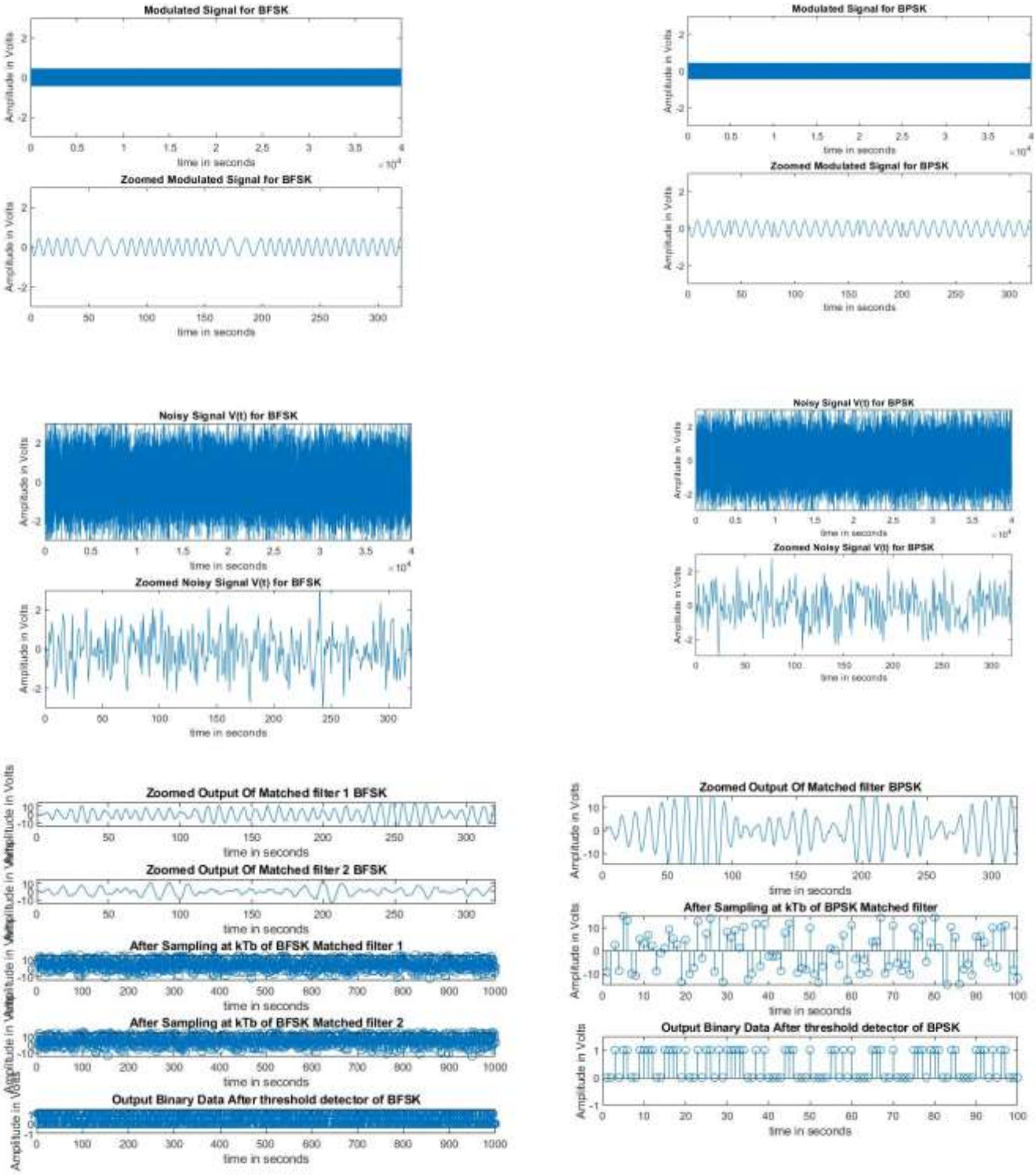

Fig. 10 Sender & Receiver Pipeline for SNR=3, number of bits=1000

1.4 Different Scenarios for transmitted signals

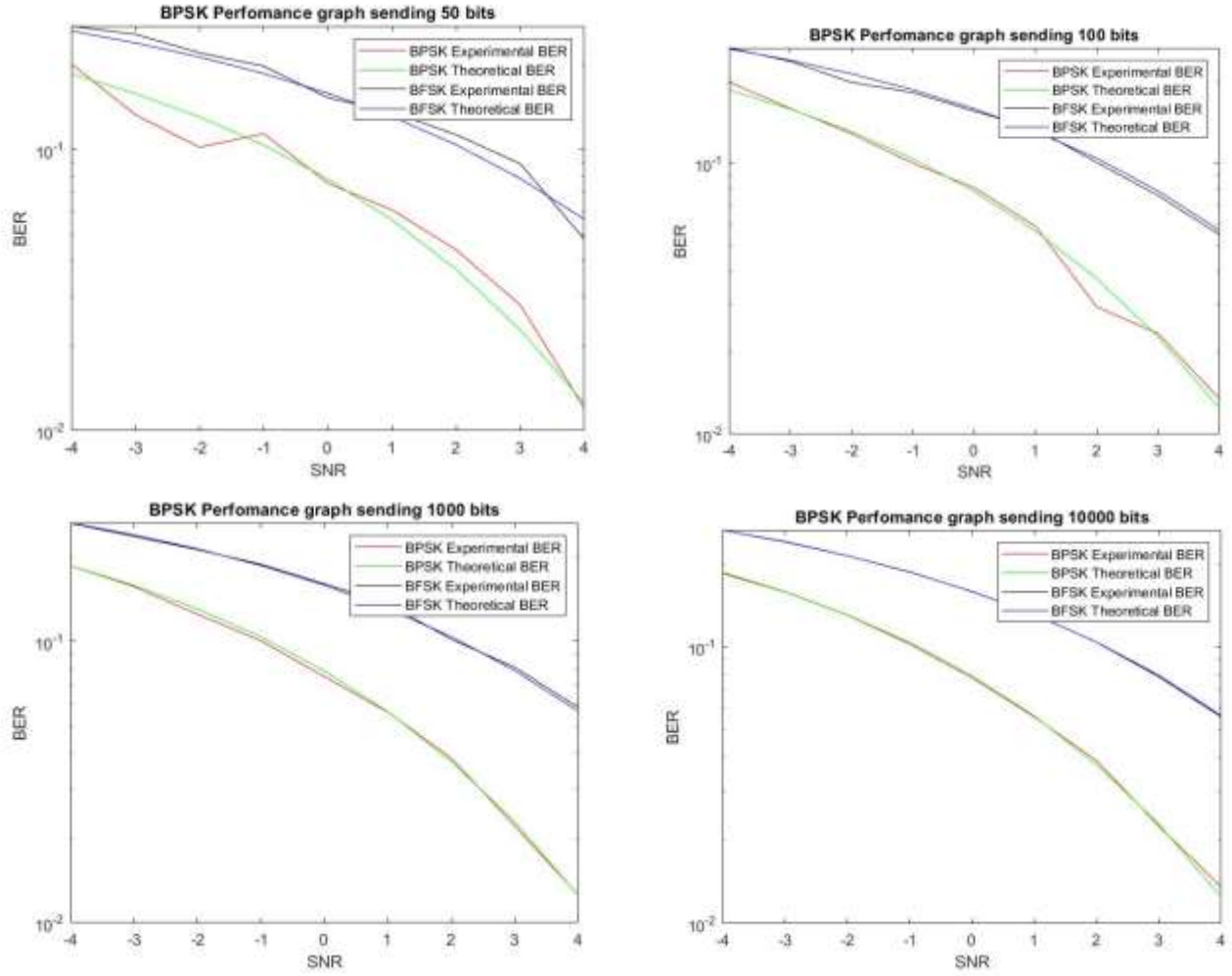

Fig 11. Different number of transmitted bits in range SNR= -4dB :4dB

As observed from Fig.11:

- Experimental calculation of BER in the simulation approaches and align with the theoretical calculations as we increase the number of transmitted bits.
- The BER of BPSK is lower than the BER of BFSK in all SNR cases from -4 dB to 4 dB.
- The experimental calculation of BER in our simulation match the theoratical calculation for both.
- As SNR (signal to noise ratio) increases, the total number of errornous bits decreases resulting in a decrease in the bit error rate.

Appendix

```
clc;
clear;
Ts=1;
N=40;
Ws=2*pi/Ts;
%we must sample at a substantially higher
%frequency than the bit frequency.
Tb=N*Ts;
% %plotting the encoded signal with PNRZ
% plot(time,encodedSignal)
% %xim([0 length(encodedSignal])
% ylim([-1.5 1.5])

SNRe=-4:1:4;
No=2;
realizationCount=20;
threshold=0;
numberOfInputBits=100000;
PerrorBPSKPexp=zeros(1,length(SNRe));
PerrorBPSKPtheo=zeros(1,length(SNRe));
PerrorBFSKPexp=zeros(1,length(SNRe));
PerrorBFSKPtheo=zeros(1,length(SNRe));
for i=1:length(SNRe)
   BpskBER=0;
   BfskBER=0;
   for j=1:realizationCount
      %Generating the input to our system TODO ask if 0,1 are equiprobable
      inputBinaryData=rand(1,numberOfInputBits)>0.5;
      %that this sequence of binaARY digits are represented by rectangular pulses having amplitudes ±1
      %==PolarRZ
      %TODO ask Dr if we can concat both with each other
      [encodedSignal,time]=pnrz(inputBinaryData,Ts,N);
       A=sqrt((10^(SNRe(i)/10))*2*(No/Tb));

%       figure(10);
%       plot(time,encodedSignal)
%       xlim([0 200])
%       ylim([-1.5 1.5])
%       xlabel('time in sec');
%       ylabel('Value in volts');
%       title('Input random data after encoding with Polar NRZ');
%
%       [PSD,F] = periodogram(encodedSignal,[],[],1);
%       figure(11);
%       plot(F,PSD);
%       title(''Power Spectral Density of the Encoded Signal '');
%       xlabel('Frequency in HZ');
%       ylabel('PSD in Volts/HZ');
%             xlim([0 10])
%       ylim([0 10])
```

```
% %         BPSK
      [outputBinaryDataBPSK]=BPSK(inputBinaryData,time,A,Ts,N,No,threshold,numberOfInputBits);
      BpskBER=BpskBER+sum(inputBinaryData~=outputBinaryDataBPSK)/numberOfInputBits;

      %BFSK
     [outputBinaryDataBFSK]=BFSK(inputBinaryData,time,A,Ts,N,No,threshold,numberOfInputBits);
      BfskBER=BfskBER+sum(inputBinaryData~=outputBinaryDataBFSK)/numberOfInputBits;
   end
   PerrorBPSKPexp(i)=BpskBER/realizationCount;
   PerrorBPSKPtheo(i)=0.5*erfc(sqrt((((A^2)*Tb)/2)/No));
   PerrorBFSKPexp(i)=BfskBER/realizationCount;
   PerrorBFSKPtheo(i)=0.5*erfc(sqrt((((A^2)*Tb)/2)/(No*2)));
end


figure(1);
semilogy(SNRe,PerrorBPSKPexp,'r');
hold on
semilogy(SNRe,PerrorBPSKPtheo,'g');
hold on
semilogy(SNRe,PerrorBFSKPexp,'k');
hold on
semilogy(SNRe,PerrorBFSKPtheo,'b');
title(['BPSK Perfomance graph sending ',num2str(numberOfInputBits),' bits']);
xlabel('SNR');
ylabel('BER');
legend('BPSK Experimental BER','BPSK Theoretical BER','BFSK Experimental BER','BFSK Theoretical BER');
hold off


% figure(2);
% semilogy(SNRe,PerrorBFSKPexp,'g');
% hold on
% semilogy(SNRe,PerrorBFSKPtheo,'b');
% title(['BFSK Perfomance graph sending',num2str(numberOfInputBits),' bits']);
% xlabel('SNR');
% ylabel('BER');
% legend('Experimental BER','Theoretical BER');
% hold off

-----------------------------------------------------------------------------------------------------------------------------
function [outputBinaryData] = BFSK(encodedSignal,time,A,Ts,N,No,threshold,numberOfInputBits)
%BFSK Summary of this function goes here
%   Detailed explanation goes here
Tb=Ts*N;
no=4;
n1=1;
W1 = (2*pi)*(no+n1)/Tb;
W2=  (2*pi)*(no-n1)/Tb;
[modulatedBFSKSignal,t]=BFSKModulator(encodedSignal,time,A,W1,W2,Ts,Tb);
% figure(4);
```

```
% plot(t,modulatedBFSKSignal)
% xlim([0 300])
% ylim([-0.5 0.5])
% title('the modulated BFSK signal');

%the channel is assumed to be ideal and the channel noise is assumed to be white and
%Gaussian with power spectral density No/2
[Vt,Vtime]=channel(modulatedBFSKSignal,t,No);
% figure(2);
% plot(Vtime,Vt)
% xlim([0 200])
% ylim([-5 5])
% title('the modulated BFSK signal with AWGN');

[outputBinaryData] = BFSKDemodulator(Vt,Vtime,W1,W2,Tb,threshold,numberOfInputBits);
end

-----------------------------------------------------------------------------------------------------------------------------
function [outputBinaryData] = BFSKDemodulator(Vt,t,W1,W2,Tb,~,numberOfInputBits)
%BFSKMODULATOR Summary of this function goes here
%   consists of the matched filter followed by sampler followed by
%   threshold detector

% figure(14);
% subplot(2,1,1)
% plot(t,Vt);
% title('Noisy Signal V(t) for BFSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% %        xlim([0 10])
% ylim([-3 3])
% subplot(2,1,2)
% plot(t,Vt);
% title('Zoomed Noisy Signal V(t) for BFSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% xlim([0 320])
% ylim([-3 3])

%Matched filter output
t=0:1:Tb-1;
Ht1=cos(W1*(Tb-t));
Ht2=cos(W2*(Tb-t));
Yt1=conv(Vt,Ht1);
Yt2=conv(Vt,Ht2);

% figure(4);
% plot(Yt);%
% title(''y(t) after matched filter'');
% xlabel('time)');
% ylabel('Volts');
% xlim([0 300]);
```

```
% ylim([-20 20]);

%sampling for each bit sent at time= Tb
sampledTime=Tb:Tb:Tb*numberOfInputBits;
sampledSignal1=Yt1(sampledTime);
sampledSignal2=Yt2(sampledTime);

%Threshold detector bigger than threshold =1 less than threshold =-
outputBinaryData=sampledSignal1>sampledSignal2;

% figure(18);
% subplot(5,1,1)
% plot(Yt1);
% title('Zoomed Output Of Matched filter 1 BFSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% xlim([0 320])
% ylim([-15 15])
%
% subplot(5,1,2)
% plot(Yt2);
% title('Zoomed Output Of Matched filter 2 BFSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% xlim([0 320])
% ylim([-15 15])
%
%
% subplot(5,1,3)
% stem(sampledSignal1);
% title('After Sampling at kTb of BFSK Matched filter 1');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% ylim([-15 15])
%
% subplot(5,1,4)
% stem(sampledSignal2);
% title('After Sampling at kTb of BFSK Matched filter 2');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% ylim([-15 15])
%
%
% subplot(5,1,5)
% stem(outputBinaryData);
% title('Output Binary Data After threshold detector of BFSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% ylim([-1 1.5])


end
```

```
---------------------------------------------------------------------------------------------------------------------------
function [modulatedSignalBFSK,time] = BFSKModulator(encodedSignal,t,A,W1,W2,Ts,Tb)
%BPSKMODULATOR Summary of this function goes here
%   Detailed explanation goes here


samplingTime=Ts:Ts:Tb;
j=1;
for i=1:length(encodedSignal)
   if (encodedSignal(i)==1)
      modulatedSignalBFSK(1,j:j+(Tb/Ts)-1)=A*cos(W1*samplingTime);
   else
      modulatedSignalBFSK(1,j:j+(Tb/Ts)-1)=A*cos(W2*samplingTime);
   end
   j=j+(Tb/Ts);
end

time=0:1:length(modulatedSignalBFSK)-1;

% figure(15);
% subplot(2,1,1)
% plot(time,modulatedSignalBFSK);
% title('Modulated Signal for BFSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% ylim([-3 3])
% %         xlim([0 10])
% subplot(2,1,2)
% plot(time,modulatedSignalBFSK);
% title('Zoomed Modulated Signal for BFSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% xlim([0 320])
% ylim([-3 3])
end
---------------------------------------------------------------------------------------------------------------------------
function [outputBinaryData] = BPSK(encodedSignal,time,A,Ts,N,No,threshold,numberOfInputBits)
%BPSK Summary of this function goes here
%   Detailed explanation goes here
Tb=Ts*N;
Wc = (4*2*pi)/Tb;
[modulatedBPSKSignal,t]=BPSKModulator(encodedSignal,time,A,Wc,Ts,Tb);
% figure(1);
% plot(t,modulatedBPSKSignal)
% xlim([0 200])
% ylim([-0.5 0.5])
% title('the modulated BPSK signal');

%the channel is assumed to be ideal and the channel noise is assumed to be white and
%Gaussian with power spectral density No/2
[Vt,Vtime]=channel(modulatedBPSKSignal,t,No);
```

```
% figure(2);
% plot(Vtime,Vt)
% xlim([0 200])
% ylim([-5 5])
% title('the modulated BPSK signal with AWGN');

[outputBinaryData] = BPSKDemodulator(Vt,Vtime,Wc,Tb,threshold,numberOfInputBits);
End
```

```
function [outputBinaryData] = BPSKDemodulator(Vt,t,Wc,Tb,threshold,numberOfInputBits)
%BPSKMODULATOR Summary of this function goes here
%   consists of the matched filter followed by sampler followed by
%   threshold detector

% figure(16);
% subplot(2,1,1)
% plot(t,Vt);
% title('Noisy Signal V(t) for BPSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% %        xlim([0 10])
% ylim([-3 3])
% subplot(2,1,2)
% plot(t,Vt);
% title('Zoomed Noisy Signal V(t) for BPSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% xlim([0 320])
% ylim([-3 3])

%Matched filter output
t=0:1:Tb-1;
Ht=cos(Wc*(Tb-t));
Yt=conv(Vt,Ht);

% figure(4);
% plot(Yt);%
% title(''y(t) after matched filter'');
% xlabel('time)');
% ylabel('Volts');
% xlim([0 300]);
% ylim([-20 20]);

%sampling for each bit sent at time= Tb
sampledTime=Tb:Tb:Tb*numberOfInputBits;
sampledSignal=Yt(sampledTime);

%Threshold detector bigger than threshold =1 less than threshold =-
outputBinaryData=sampledSignal>threshold;

% figure(17);
% subplot(3,1,1)
```

```
% plot(Yt);
% title('Zoomed Output Of Matched filter BPSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% xlim([0 320])
% ylim([-15 15])
%
% subplot(3,1,2)
% stem(sampledSignal);
% title('After Sampling at kTb of BPSK Matched filter');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% ylim([-15 15])
%
%
% subplot(3,1,3)
% stem(outputBinaryData);
% title('Output Binary Data After threshold detector of BPSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% ylim([-1 1.5])

end

-----------------------------------------------------------------------------------------------------------------------------------
function [modulatedSignalBPSK,time] = BPSKModulator(encodedSignal,t,A,Wc,Ts,Tb)
%BPSKMODULATOR Summary of this function goes here
%   Detailed explanation goes here

samplingTime=Ts:Ts:Tb;
j=1;
for i=1:length(encodedSignal)
   if (encodedSignal(i)==0)
      modulatedSignalBPSK(1,j:j+(Tb/Ts)-1)=-A*cos(Wc*samplingTime);
   else
      modulatedSignalBPSK(1,j:j+(Tb/Ts)-1)=A*cos(Wc*samplingTime);
   end
   j=j+(Tb/Ts);
end

time=0:1:length(modulatedSignalBPSK)-1;

% figure(13);
% subplot(2,1,1)
% plot(time,modulatedSignalBPSK);
% title('Modulated Signal for BPSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% ylim([-3 3])
% %        xlim([0 10])
% subplot(2,1,2)
% plot(time,modulatedSignalBPSK);
```

```
% title('Zoomed Modulated Signal for BPSK');
% xlabel('time in seconds');
% ylabel('Amplitude in Volts');
% xlim([0 320])
% ylim([-3 3])
end

---------------------------------------------------------------------------------------------------------------------------------
function [Vt,time] = channel(modulatedSignal,t,No)
%BPSK Summary of this function goes here
%   Detailed explanation goes here

%power specifies the power of noise in dBW.
Wt=wgn(1,length(modulatedSignal),10*log10(No/2));

% figure(12);
% plot(t,Wt);
% title('Channel noise');
% xlabel('time in seconds');
% ylabel('Nois in Volts');
%         xlim([0 10])
% ylim([-5 5])

%the channel is assumed to be ideal and the channel noise is assumed to be white and
%Gaussian with power spectral density No/2
Vt=modulatedSignal+Wt;
time=t;

end
---------------------------------------------------------------------------------------------------------------------------------
function [encodedSignal,time] = pnrz(inputBinaryData,Ts,N)
%PNRZ Summary of this function goes here
%   A function that encodes a binary bit using Polar NRZ

Tb=N*Ts;
Ws=2*pi*N/Tb;
%Sampling time
t=2*pi/Ws;
%% T will be equal to one here as N=40, Ts=1, Tb=40
time=0:t:length(inputBinaryData)*Tb;
encodedSignal = zeros(1,length(time));
for i =0:length(inputBinaryData)-1
   if inputBinaryData(i+1)== 0
      encodedSignal(i*Tb+1:(i+1)*Tb)=-1;
   else
      encodedSignal(i*Tb+1:(i+1)*Tb)=1;
   end

end
end
```