PROJECT REPORT

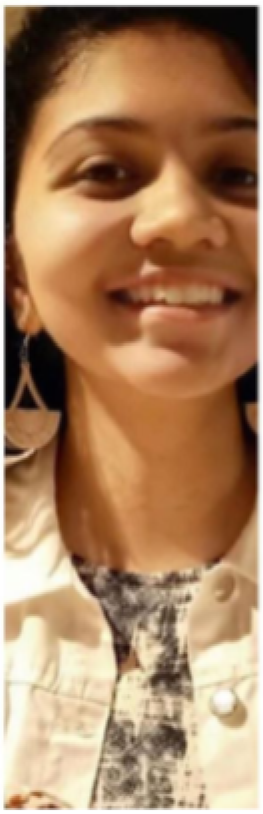
ECE211 - Analog and Digital Communications Theory

(Group 07)

Aanshi Patwari	**-**	**(AU1841004)**
Dipika Pawar	**-**	**(AU1841052)**
Miracle Rindani	**-**	**(AU1841017)**
Frency Chauhan	**-**	**(AU1841105)**

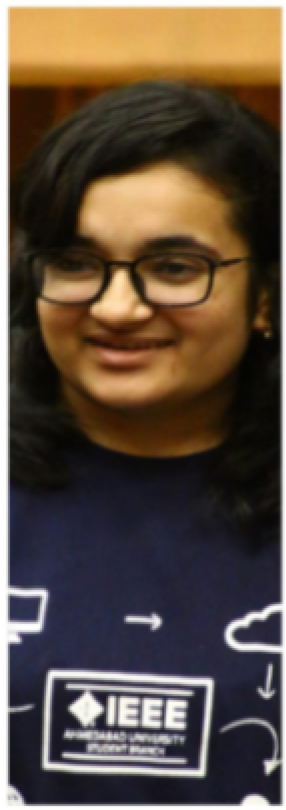
Under Guidance of Prof. Ashok Ranade,

Ahmedabad University, Navrangpura, Ahmedabad



AANSHI

DIPIKA



MIRACLE

FRENCY

BPSK AND QPSK SIMULATIONS

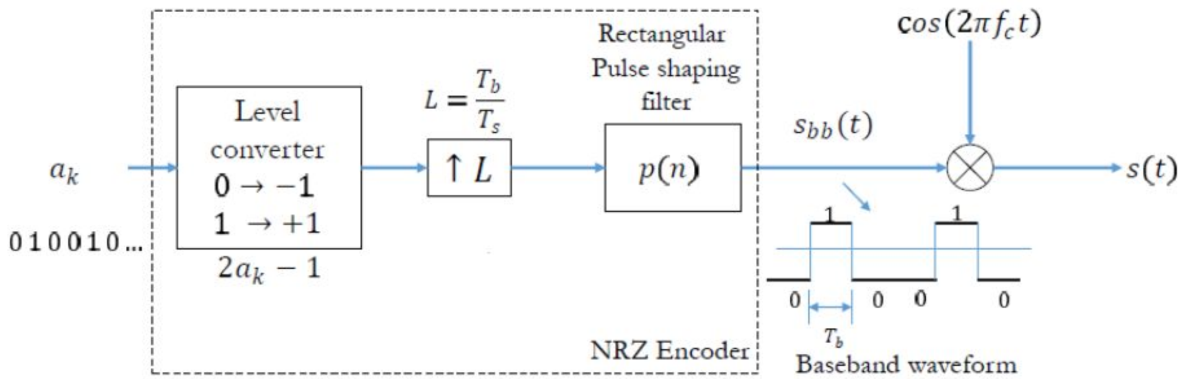
SYNOPSIS

- ## BPSK
- BPSK is the simplest method for encoding the data in two different phases.
- It can transmit one bit per symbol.It is also similar to the double side band suppressed carrier (DSBSC).
- It uses two phases each separated by a phase difference of 180 degrees.
- The modulator gives the output of 180 degree rotated phase modulated carrier wave which gets transmitted.
- The demodulator receives that wave and decodes it with some error rate to the original wave.

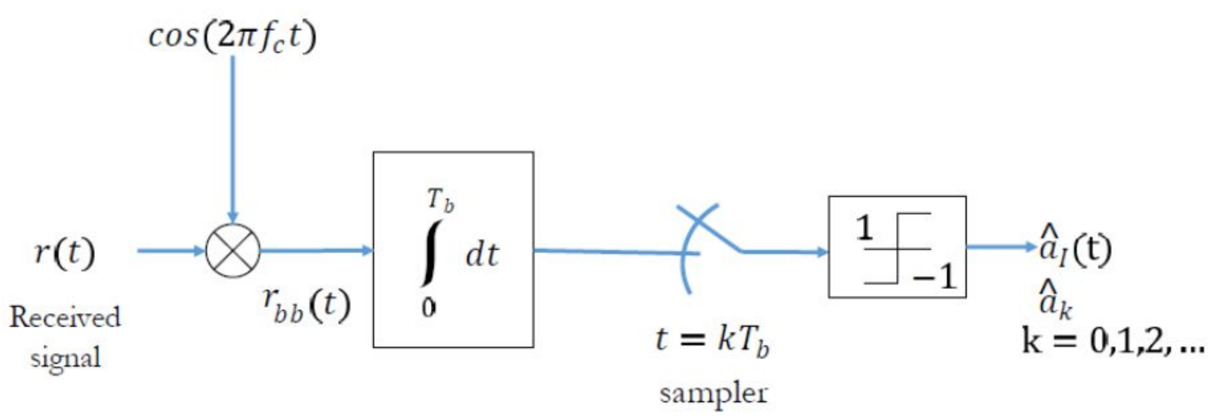
- ## QPSK
- QPSK, a variation of BPSK, encodes the data into four different phases.
- It can transmit 2 bits per symbol which inturn doubles the data rate using the same bandwidth or using the same data rate and halving the bandwidth.
- So, QPSK enables us to transmit at twice the data rate with the same bit error probability.it uses four phases separated by a phase difference of 90 degrees.

SYSTEM BLOCK DIAGRAM

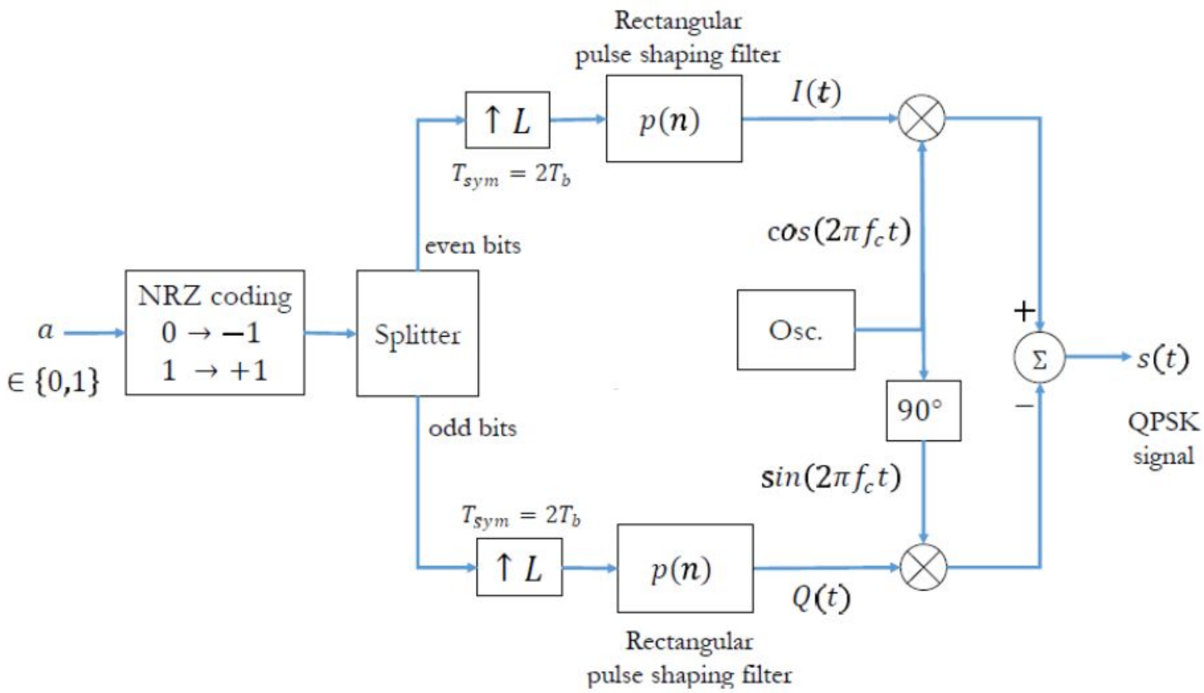
• BPSK Transmitter



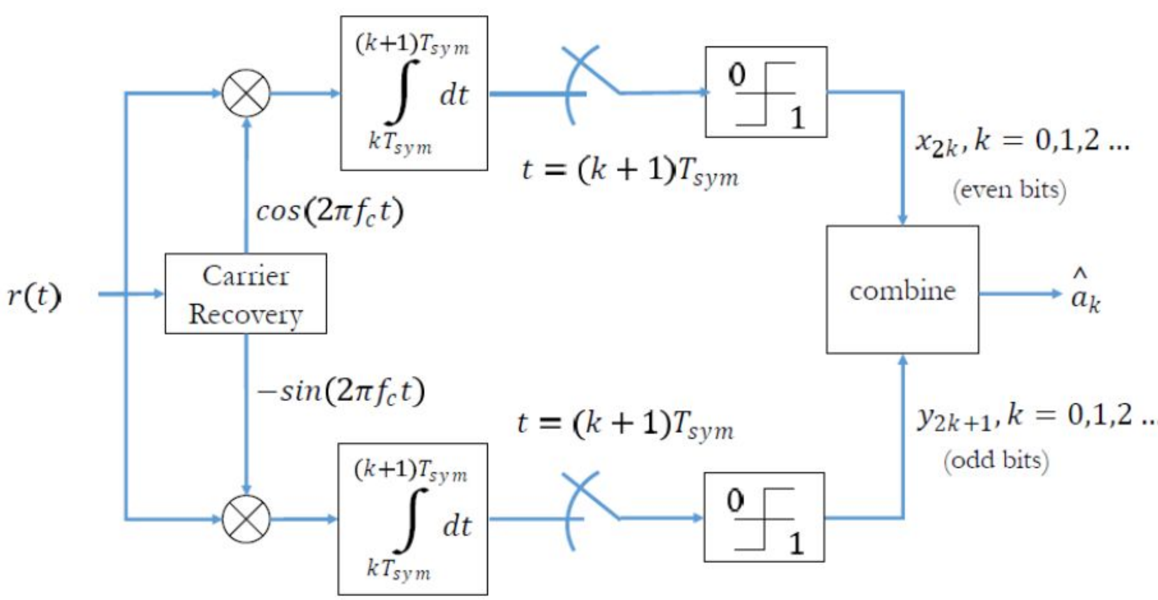
• BPSK Receiver



- **QPSK Transmitter**



- **QPSK Receiver**



SPECIFICATIONS

- All the simulations are done in **Scilab.**

- **BPSK**
- The binary message in the form of 0's and 1's are represented by two different phases: 0 and 180 degrees.
- For performing modulation, consider basic functions which are orthogonal to each other.
- For BPSK, there is only one basic function.The message bit is represented in terms of basic function.
- The BPSK signal is thus represented using 2 constellation points.

- Modulation is achieved by varying phases of basic function according to the form of the message bit.

  $s_1(t) = A_c\cos(2\pi f_c t), \quad 0 \leq t \leq T_b$ *for binary 1*

  $s_0(t) = A_c\cos(2\pi f_c t + \pi), \quad 0 \leq t \leq T_b$ *for binary 0*

  where A_c is the amplitude of the sinusoidal signal, f_c is the carrier frequency (Hz), t being the instantaneous time in seconds, T_b is the bit period in seconds.

- Then additive white gaussian noise is added to the signal and the signal is transmitted.
- The signal is received at the receiver and it is multiplied with carrier signal in the coherent receiver.

- Then the phase lock loop keeps check for the received signal to be in phase.
- Then the signal is integrated and passed through a decision block.
- Above a threshold value(Here threshold value is zero), the threshold detector takes the decision and returns the original waveform with some amount of bit error rate.

- **QPSK**
- Here the signal is considered to contain two message bits.
- So, there are four different phase states possible.

  $s(t) = A\cos(2\pi f_c t + \theta n)$, $\quad 0 \leq t \leq T_{sym}$, $n = 1,2,3,4$

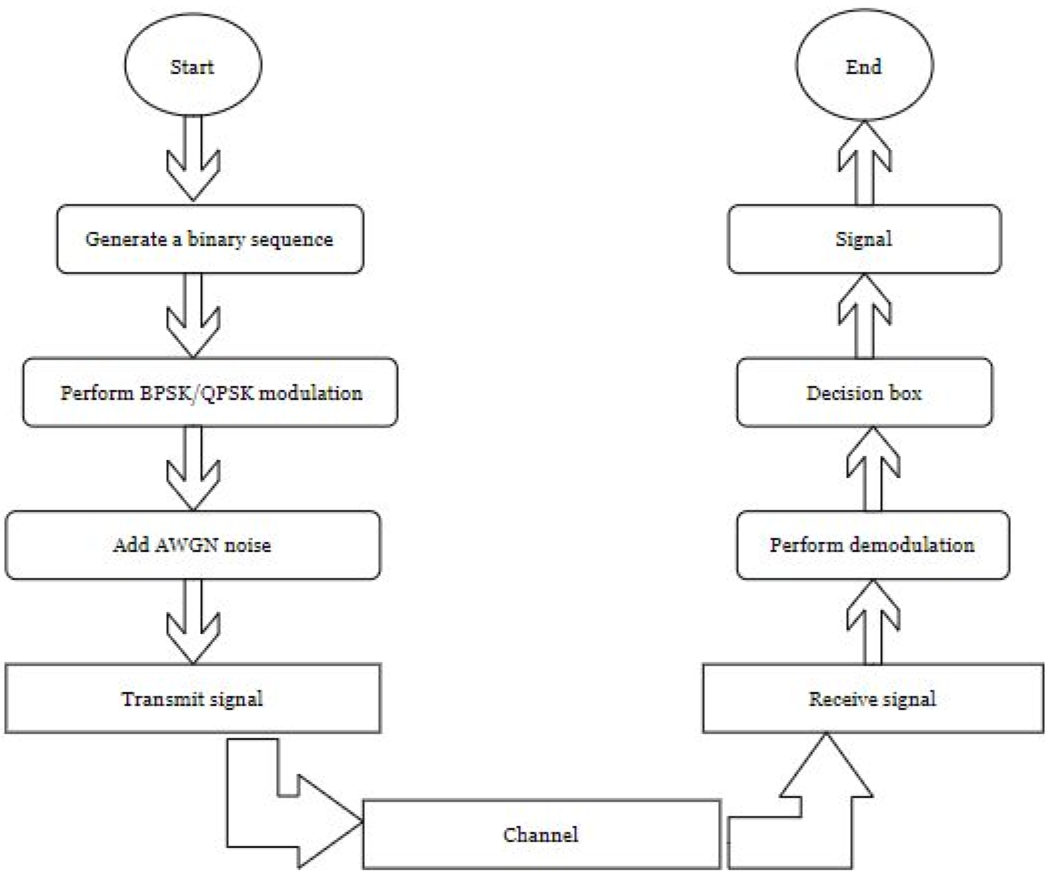
  where the signal phase is given by

  $\theta n = (2n - 1)\pi/4$

  Therefore, the four possible initial signal phases are $\pi/4$, $3\pi/4$, $5\pi/4$ *and* $7\pi/4$ *radians.*

- So, the signal is modulated using one of the four possible phase states.
- As there are 2 bits, there are 2 basic functions which are used for obtaining the modulated signal.
- Both the basic functions are orthogonal to each other.
- A splitter is used for splitting the odd and the even bits of the signal so that odd bits multiplied with one basic function and even bits with another one.
- Then noise is added to the signal and transmission of signal is done.
- After it is received the demodulation occurs through the process similar to that of BPSK.
- At the end the whole signal is combined to obtain back the original signal.

FLOWCHART



Start
Generate a binary sequence
Perform BPSK/QPSK modulation
Add AWGN noise
Transmit signal
Channel
Receive signal
Perform demodulation
Decision box
Signal
End

CODE

- **BPSK:**
- **Plotting the process of BPSK**

```
clear;

clf;


fm = 1;  //Modulating wave frequency


dt = 0.001;

T = 2;

t = 0:dt:T;



m = sin(2*%pi*fm*t);  //Modulating signal


//********************Level Converter Block****************//

N = length(t);

modulating_signal = zeros(1:N);

for i=1:N

    if m(i)>0 then

        modulating_signal(i) = 1;  //Modulating square wave

    else

        modulating_signal(i) = -1; //Modulating square wave

    end
```

```
end

fc = 10;  //Carrier wave frequency
Ac = 5;  //Carrier wave amplitude

carrier_signal = Ac*cos(2*%pi*fc*t);  //Carrier signal

phi = sqrt(2/T)*carrier_signal;  //Orthogonal component

psk = carrier_signal.*modulating_signal;  //Phase modulated wave
                                          // Transmitted Signal

//*****Addition of Gaussian Noise in Transmitted signal*****//

noiseVariance = 0.1;
noise = sqrt(noiseVariance)*rand(1,length(psk)); //Generation of noise
transmitted_signal = psk+noise; //Addition of noise to signal(Transmitted signal)

dm = (transmitted_signal).*phi; //Received Signal


//*********************Demodulation Block*******************//
demodulated_signal = zeros(1:N);
for i=1:N
    demodulated_signal(i) = (dm(i)*dt);//Integration of received signal
```

```
end

final_signal = zeros(1:N);
N = length(t);

//******************Decision Making Block******************//
for i = 1:N
    if demodulated_signal(i)>0 then
        final_signal(i) = 1;  //Output of decision block
    else
        final_signal(i) = -1;  //Output of decision block
    end
end

//**************Bit Error Probability Calculation*************//
count = 0;
for i=1:N
    if final_signal(i)==modulating_signal(i) then
        count = count;
    else
        count = count+1;
        end
end
disp(count);  //Wrong number of samples
disp(count/N);  //Displaying bit error probability = wrong
samples/total number of samples

//*************************Plots******************************//
```

```
//Modulating signal
subplot(321);
plot(t,modulating_signal);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Modulating signal", "fontsize", 4);

//Carrier signal
subplot(322);
plot(t,carrier_signal);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Carrier signal", "fontsize", 4);

//Modulated signal
subplot(323);
plot(t,psk);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("BPSK transmitted signal", "fontsize", 4);

//transmitted signal with noise
subplot(324);
plot(t,transmitted_signal);
```

```
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Signal with noise", "fontsize", 4);
```

//Received demodulated Signal

```
subplot(325);
plot(t,demodulated_signal);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Demodulated signal", "fontsize", 4);
```

//Received demodulated square signal

```
subplot(326);
plot(t,final_signal);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Demodulated signal from decision box", "fontsize", 4);
```

- **Plotting the Bit error rate of BPSK(different power levels)**

```
clear;
clf;

fm = 1;  //Modulating wave frequency

dt = 0.001;
T = 2;
t = 0:dt:T;

m = sin(2*%pi*fm*t);  //Modulating signal

//********************Level Converter Block*****************//
N = length(t);
modulating_signal = zeros(1:N);
for i =1:N
    if (m(i)>0) then
        modulating_signal(i) = 1;//Modulating square wave
    else
        modulating_signal(i) = -1;//Modulating square wave
    end
end

fc = 10;  //Carrier wave frequency
Acarrier = 1:0.5:15;

nle = length(Acarrier);
```

```
pb = zeros(1:nle);  //Bit error probability

eb = zeros(1:nle);  //Bit energy

//************Loop of Amplitudes***********//

for j=1:nle

    Ac = Acarrier(j);  //Carrier wave amplitude

    eb(j) = Ac*Ac/(2*T);


    carrier_signal = Ac*cos(2*%pi*fc*t);  //Carrier signal


    phi = sqrt(2/T)*carrier_signal;  //Phase shift


     psk = carrier_signal.*modulating_signal;  //Phase modulated
signal


//******Addition of Gaussian Noise in Transmitted signal******//

    noiseVariance = 0.1;

           //Generation  of noise

    noise = sqrt(noiseVariance)*rand(1,length(psk));

          //Addition of noise to signal(Transmitted signal)

    transmitted_signal = psk+noise;

    dm = (transmitted_signal).*phi; //Received Signal


//******************Demodulation Block*********************//

    demodulated_signal = zeros(1:N);

    for i=1:N

              //Demodulation of received signal

        demodulated_signal(i)=(dm(i)*dt);

    end
```

```
    final_signal = zeros(1:N);
    N = length(t);

//*********************Decision Block************************//
    for i=1:N
        if (demodulated_signal(i)>0) then
            final_signal(i) = 1;//Output of decision block
        else
            final_signal(i) = -1;//Output of decision block
        end
    end

//*******************Bit Error Probability******************//
    count = 0;
    for i=1:N
        if final_signal(i)==modulating_signal(i) then
            count = count;
        else
            count = count+1;
        end
    end

    pb(j)=count/N;
    disp(count/N); //Bit error probability value
end

//*****************************Plot**********************//
```

//Bit error probability at various energy levels

```
plot(eb,pb);

xgrid(3);

xlabel("Eb", "fontsize", 2);

ylabel("Bit error rate", "fontsize", 2);

title("Bit error probability curve for BPSK with noise","fontsize",4);
```

- **QPSK:**
- **Plotting the process of QPSK**

```
clear;

clf;

fm=1; //Modulating wave frequency

dt=0.001;

t=0:dt:2;

m=sin(2*%pi*fm*t); //Modulating wave


//********************Level Converter Block*****************//

N=length(t);

modulating_signal=zeros(1:N);

for i =1:N

    if(m(i)>0) then

        modulating_signal(i)=1; //Modulating square wave

    else

        modulating_signal(i)=-1; //Modulating square wave

    end

end


//******************Carrier wave generator**************//

fc=fm*10; //Carrier wave frequency

Ac=4.25;

oddSample=1:2:N;

evenSample=2:2:N;

carrier_odd=Ac*sin(2*%pi*fc*dt*oddSample); //Carrier wave 1 with
odd samples
```

```
carrier_even=Ac*cos(2*%pi*fc*dt*evenSample);  //Carrier  wave  2 with even samples

//***************Modulated signal generator****************//
modulated_signal=zeros(1:N);
for i=1:N
    if modulo(i,2)==0 then
       modulated_signal(i) = modulating_signal(i).*carrier_even(i/2);
    else
       modulated_signal(i) = modulating_signal(i).*carrier_odd(i/2+1);
    end
end

//******Addition of Gaussian Noise in Transmitted signal******//
r=rand(modulated_signal,"uniform");
hp=ffilt("hp",100,(fm)*dt);
filtered=filter(hp,1,r);
noiseVariance=0.1;
noise   =   sqrt(noiseVariance)*rand(1,length(modulated_signal)); //Generation of noise
qpsk=noise+modulated_signal; //Transmitted signal

//*********************Received signal*********************//
dm=zeros(1:N); //Received signal
for i=1:N
    if modulo(i,2)==0 then
        //disp(i);
```

```
        dm(i)=qpsk(i).*carrier_even(i/2);
    else
        dm(i)=qpsk(i).*carrier_odd(i/2+1);
    end
end

//*******************Demodulation Block******************//
demodulated_signal=zeros(1:N);
for i=1:N
    demodulated_signal(i)=(dm(i)*dt); //demodulated signal
end

//************************Decision Block*******************//
final_signal=zeros(1:N);
for i=1:N
  if (demodulated_signal(i)>0) then
        final_signal(i)=1; //Output of decision block
    else
        final_signal(i)=-1;//Output of decision block
    end
end

//*********************Bit Error Probability***************//
count=0;
for i=1:N
    if final_signal(i)==modulating_signal(i) then
        count=count;
```

```
    else
        count=count+1;
        end
end
disp(count);
disp(count/N); //Displaying the value of bit error probability

//**************************Plots*************************//
//Modulating signal
subplot(421);
plot(t,modulating_signal);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Modulating signal", "fontsize", 4);

//carrier wave 1
subplot(422);
plot(carrier_odd);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Odd carrier Signal", "fontsize", 4);

//carrier wave 2
subplot(423);
plot(carrier_even);
```

```
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Even carrier signal", "fontsize", 4);

//Modulated signal
subplot(424);
plot(modulated_signal);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Modulated signal", "fontsize", 4);

//Transmitted signal with noise
subplot(425);
plot(qpsk);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("signal with noise", "fontsize", 4);

//Received signal
subplot(426);
plot(dm);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
```

```
title("Received signal", "fontsize", 4);

//Demodulated Signal
subplot(427);
plot(demodulated_signal);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Demodulated signal", "fontsize", 4);

//Signal from decision box
subplot(428);
plot(final_signal);
xgrid(3);
xlabel("time","fontsize",2);
ylabel("amplitude","fontsize",2);
title("Final signal from decision box", "fontsize", 4);
```

- **Plotting the Bit error rate of QPSK(different power levels)**

```
clear;

clf;

fm=1; //Modulating wave frequency

dt=0.001;

t=0:dt:2;

T=2;

m=sin(2*%pi*fm*t); //Modulating wave

//********************Level Converter Block****************//

N=length(t);

modulating_signal=zeros(1:N);

for i =1:N

    if(m(i)>0) then

        modulating_signal(i)=1; //Modulating square wave

    else

        modulating_signal(i)=-1; //Modulating square wave

    end

end


//******************Carrier wave generator*****************//

fc=fm*10; //Carrier wave frequency

oddSample=1:2:N;

evenSample=2:2:N;

Acarrier=1:10;

nle=length(Acarrier);

pb=zeros(1:nle);

eb=zeros(1:nle);
```

```
//*****************Loop of amplitudes************************//
for j=1:nle
    Ac=Acarrier(j);
    eb(j)=Ac*Ac/(2*T);

carrier_odd=Ac*sin(2*%pi*fc*dt*oddSample); //Carrier wave 1
carrier_even=Ac*cos(2*%pi*fc*dt*evenSample); //Carrier wave 2

//****************Modulated signal generator*****************//
modulated_signal=zeros(1:N);
for i=1:N
    if modulo(i,2)==0 then
        //disp(i);

modulated_signal(i)=modulating_signal(i).*carrier_even(i/2);
    else

modulated_signal(i)=modulating_signal(i).*carrier_odd(i/2+1);
    end
end

//******Addition of Gaussian Noise in Transmitted signal******//
r=rand(modulated_signal,"uniform");
hp=ffilt("hp",100,(fm)*dt);
filtered=filter(hp,1,r);
noiseVariance=0.1;
```

```
noise   =   sqrt(noiseVariance)*rand(1,length(modulated_signal));
//Generation of noise

qpsk=noise+modulated_signal; //Transmitted signal


//********************Received signal***********************//

dm=zeros(1:N); //Received signal

for i=1:N

    if modulo(i,2)==0 then

        //disp(i);

        dm(i)=qpsk(i).*carrier_even(i/2);

    else

        dm(i)=qpsk(i).*carrier_odd(i/2+1);

    end

end

//********************Demodulation Block********************//

demodulated_signal=zeros(1:N);

for i=1:N

    demodulated_signal(i)=(dm(i)*dt);//demodulated signal

end


//********************Decision Block*******************//

final_signal=zeros(1:N);

for i=1:N

  if (demodulated_signal(i)>0) then

        final_signal(i)=1; //Output of decision block

    else

        final_signal(i)=-1; //Output of decision block

    end
```

```
end

//********************Bit Error Probability*****************//
    count=0;
    for i=1:N
        if final_signal(i)==modulating_signal(i) then
            count=count;
        else
            count=count+1;
            end
    end

    pb(j)=count/N;
    disp(pb(j)); //displaying the value of bit error probability
end

//*********************************Plot**********************//
//Bit error probability at various energy levels
plot(eb,pb);
xgrid(3);
xlabel("Eb", "fontsize", 2);
ylabel("Bit error rate", "fontsize", 2);
title("Bit error probability curve for QPSK with noise","fontsize",4);
```

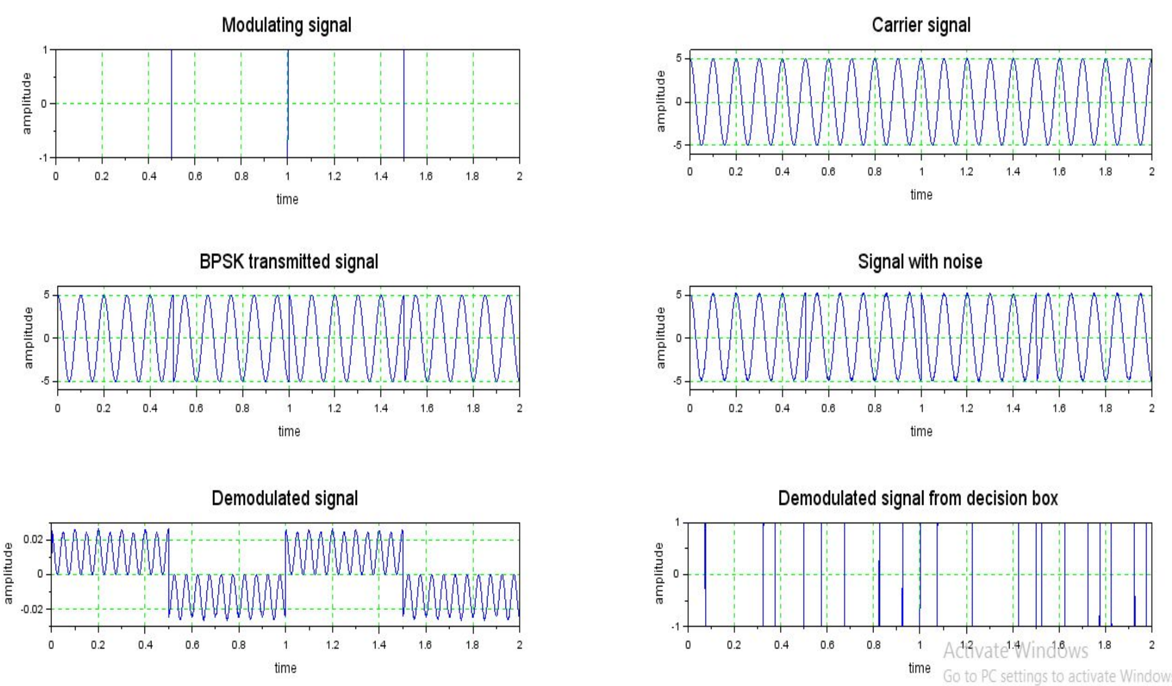
RESULTS

The following are the results generated for the BPSK and QPSK Simulations performed with their respective bit error probabilities:

From the graphs, we understand that the bit error probabilities for BPSK and QPSK are almost equal although QPSK is a more robust technique.

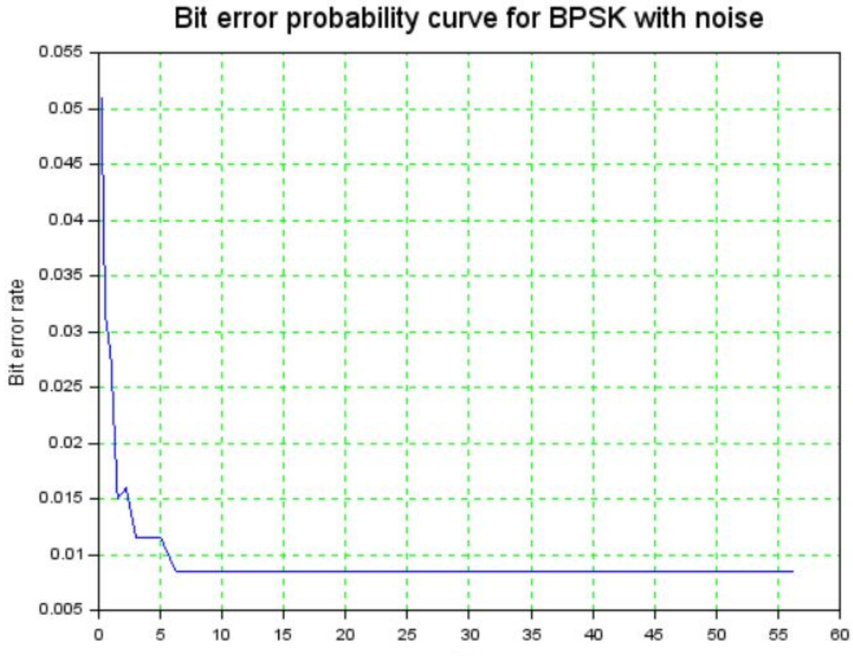
- ## BPSK

1) Graphs Generated: (with noise)

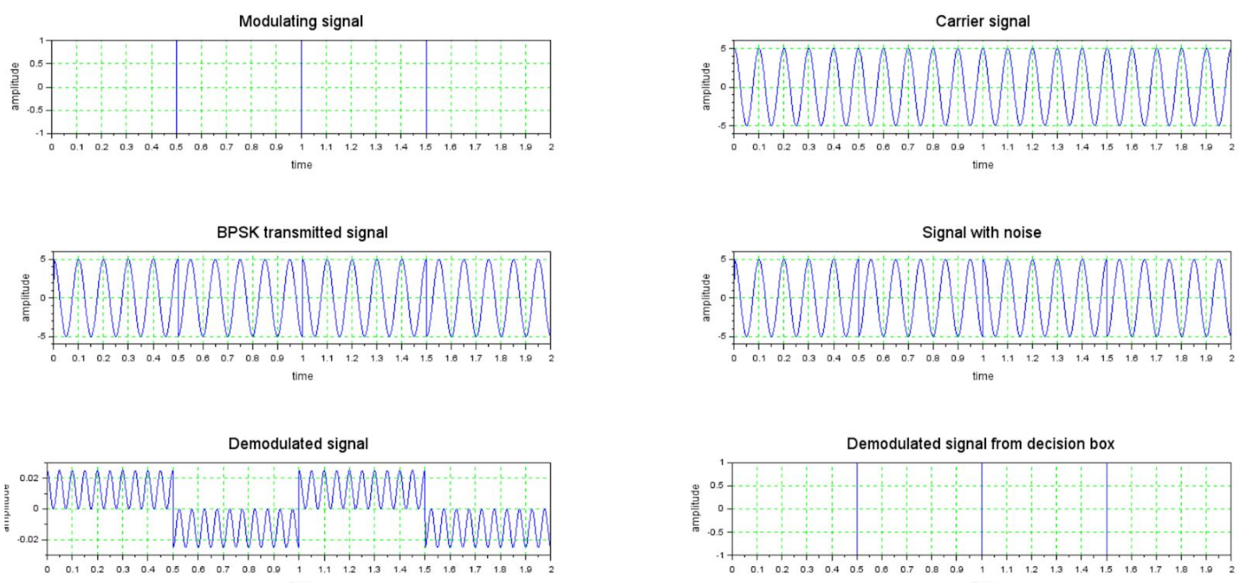


Bit error probability for BPSK, P_B = 0.0084958

2) Variation of bit error probability:

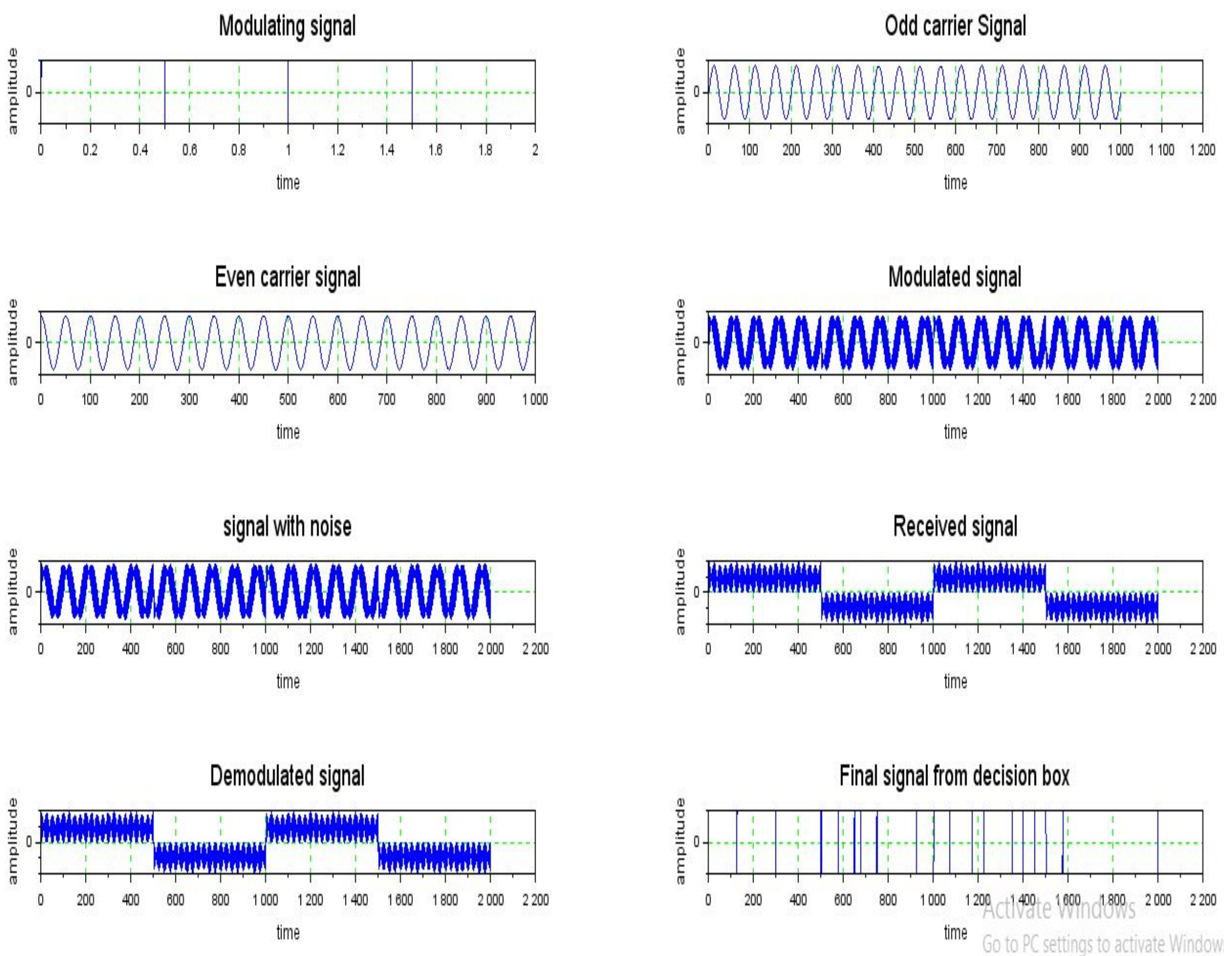


3) Without Noise: (Zero bit error probability)



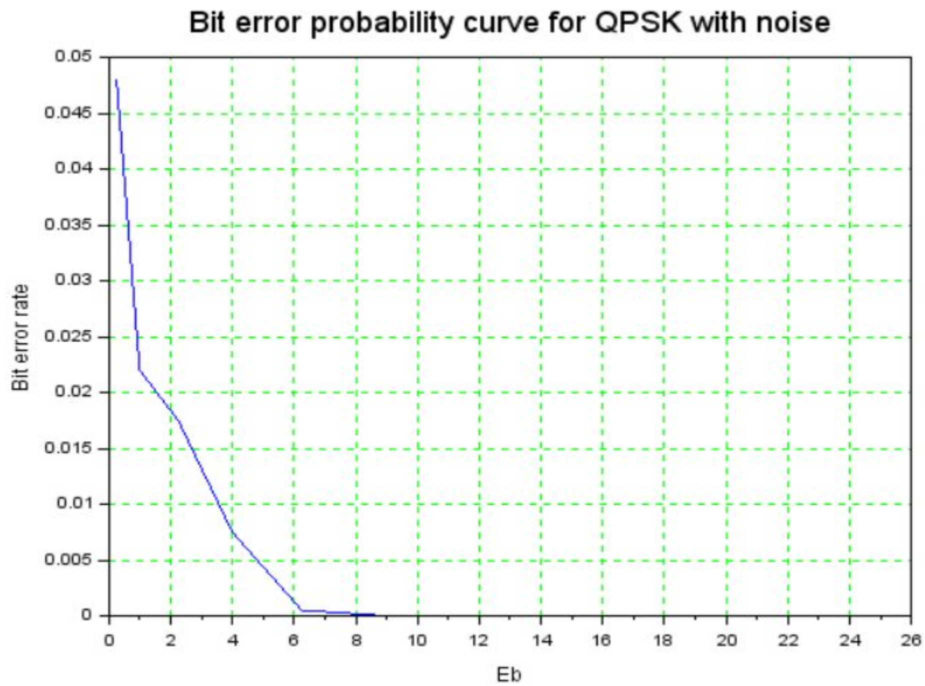
● **QPSK**

1) Graphs Generated:



Bit error probability for QPSK, P_B = 0.0079960

2) Variation of bit error probability:


Bit error probability curve for QPSK with noise
Bit error rate
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
2
4
6
8
10
12
14
16
18
20
22
24
26
Eb

APPLICATIONS

- Phase Shift Keying is widely used for Biometric purposes.
- It is also used in wireless communications like bluetooth and RFID.
- QPSK is used in various cellular wireless standards like GSM, LTE , 802.11 WLAN etc.
- It is also useful in optical communications and multi-channel WDM(Wavelength Division Multiplexer).

REFERENCES

1. https://www.gaussianwaves.com/2010/04/bpsk-modulation-and-demodulation-2/
2. https://www.gaussianwaves.com/2010/10/qpsk-modulation-and-demodulation-2/
3. https://www.tutorialspoint.com/digital_communication/digital_communication_phase_shift_keying.htm
4. https://www.tutorialspoint.com/digital_communication/digital_communication_quadrature_phase_shift_keying.htm
5. https://www.quora.com/What-are-all-the-differences-between-QPSK-and-BPSK