PHASE SHIFT MODULATION

DS-BPSK

USING PYTHON

SUBMITTED BY

ANNA TOMS P J (6) | RAKHIL P R(38) | THUSHAR TOM(51) | SMRUTHY M(55)

Contents

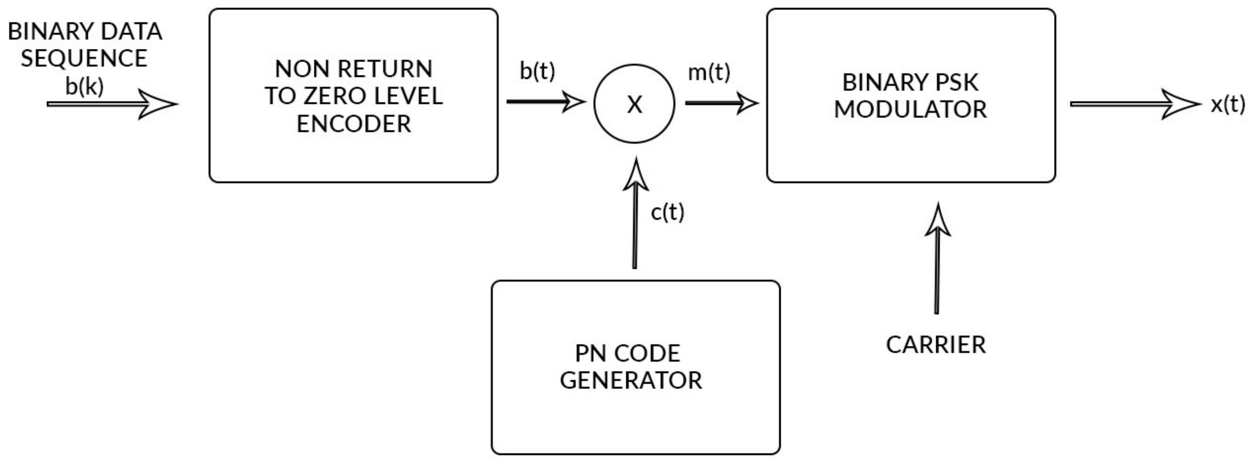
List of Figures

1 AIM :

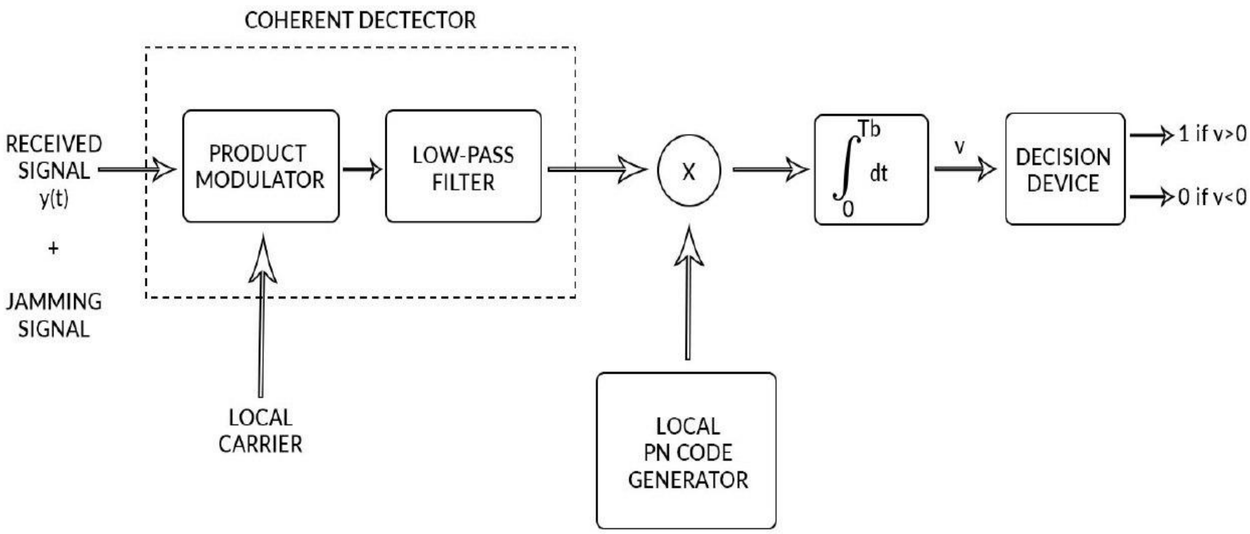
To simulate DS-BPSK system in python (Transmitter and Receiver with PN sequence and Jamming) and to plot SNR vs BER, Constellation of BPSK.

2 BLOCK DIAGRAM :

Transmitter



Receiver



3 EQUATIONS :

$$Non-Zero\ \ level\ \ encoder\ \ :\ \ 2 * Binary\ \ input\ \ sequence - 1$$
$$PN\ \ sequence\ \ :\ \ 1 + D6 + D7$$
$$Basis\ \ function\ \ of\ \ carrier = \sqrt{\frac{2}{Tb}}\cos(2\pi f_c t)$$
$$Probability\ \ of\ \ error\ \ :\ \frac{1}{2}erfc(\sqrt{\frac{Eb}{N0}})$$
$$BPSKsignal\ \ :\ x(t) = b(t)\sqrt{Eb}\sqrt{\frac{2}{Tb}}\cos(2\pi f_c t)$$

$$where\ \ b(t) : Bipolarsignal = \begin{cases} 1 & \text{for } 1 \\ -1 & \text{for } 0 \end{cases}$$

$$\sqrt{Eb} : Energy\ \ of\ \ one\ \ bit$$

4 PYTHON CODE :

```
from scipy import*
from pylab import*
import matplotlib.pyplot as plt
import array
import numpy as np
import random

#intialising binary sequence

n=50
bin_ip_seq = np.random.choice([0, 1], size=(n))
n=len(bin_ip_seq)
print ("The input binary sequence is : ")
print (bin_ip_seq)
print(n)
print("\r")

#passing it to non return to zero level encoder
NRZ_INP=np.zeros(n)

for i in range(0,n):
    NRZ_INP[i]= 2*bin_ip_seq[i] -1

#PN sequence Generator

A = np.array([randint(0,2),randint(0,2),randint(0,2),
randint(0,2),randint(0,2),randint(0,2),randint(0,2)])
print(A)
```

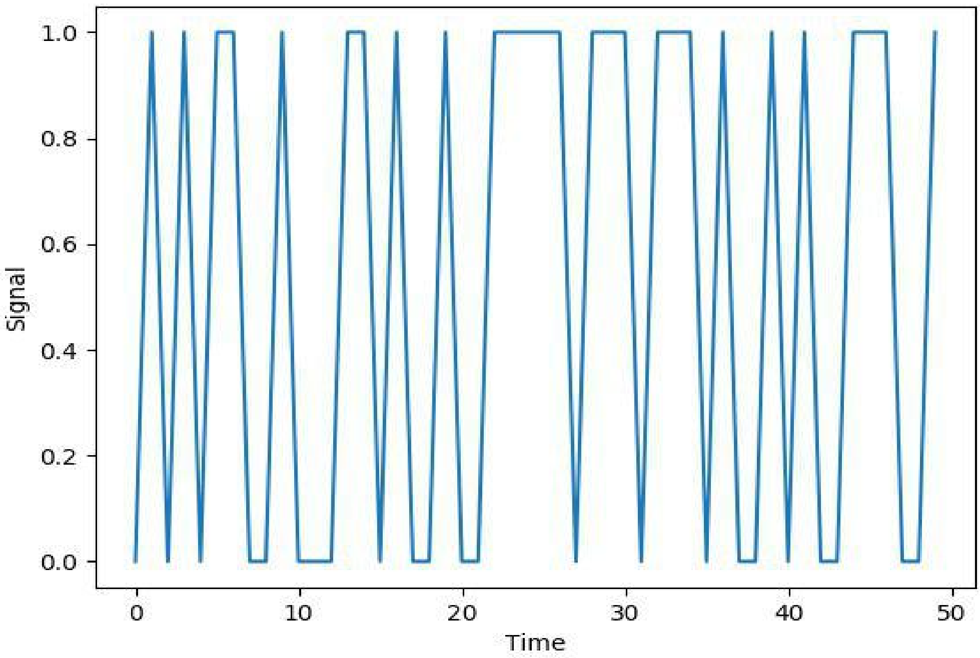
```
size = 2**len(A) - 1

PN = []

for i in range(size):
        PN.append(A[-1])
        A = [A[-1]^A[-2], A[0], A[1],A[2], A[3],A[4],A[5]]

for i in range(0,size):
    PN[i]= 2*PN[i] -1
print("NON ZERO TO RETURN PN \r")


#PN sequence Multiplier
z=[]

for i in range(0,n):
        for j in range(1):
                for k in range(0,size):
                                z.append(PN[k]*NRZ_INP[i])

N=len(z)

#BPSK Modulator

Tb=1
t=r_[0:Tb:0.1]
fc=1
carrier=sqrt(2*(Tb**-1))*sin(2*pi*fc*t)
M=len(carrier)
plt.plot(carrier)
show()
bpskarray=[]
for i in range(0,N):
  if z[i]>=0:
    bpskarray.append(carrier)

  else:
    bpskarray.append((-1*carrier))
bpsksignal=concatenate(bpskarray)
print("bpsksignal")
print(len(bpsksignal))
print(bpsksignal)



#Jamming signal
x=[]

x=rand(n*1270)
t1=r_[0:n*12.7:0.01]
Eb=.00001*var(carrier)*len(t1)
jamming=Eb*x*(sqrt(2*(Tb**-1))*(np.cos(2*np.pi*fc*t1)+1j*(np.sin(2*np.pi*fc*t1))))
figure(4)
plt.plot(jamming)
print("\n Jamming")
print (jamming)
jamvar=10**(-0.1*(jamming))
```

```
print("varience of carrier\n")
print(Eb)
snr=Eb*((jamvar)**-1)
print(snr)



snrdb=10*log10(snr)
print("\nsnrdb")
print(snrdb)
print(len(snrdb))



#Reception

receivedsignal=np.array(bpsksignal)+np.array(jamming)
receivearr=[]
print("\n Received signal")
print(receivedsignal)

ber=[]

for i in range(50):
        noise2=sqrt(i)*randn(len(bpsksignal))
        recsig=bpsksignal+noise2
        receivearr=[]

        for j in range(N):
                out = sum(carrier*recsig[j*M:(j+1)*M])

                #print(out)
                if out>0:
                        receivearr.append(1)
                else:
                        receivearr.append(-1)
        rbit=receivearr

        berate=sum(abs(np.array(z)-rbit))*(n**-1)
        print(berate)
        ber.append(berate)
print("\nDemodulated signal")
print(receivearr)
print("\r")
w=len(receivearr)

#PN Squence Multiplication
q=[]
x=0
for i in range(int(N/size)):
        for k in range(size):
                        q.append(PN[k]*receivearr[x])
                        x+=1

print("\nPN multiplied")
print (q)
print(len(q))

print("\r")
#Final array generation
```

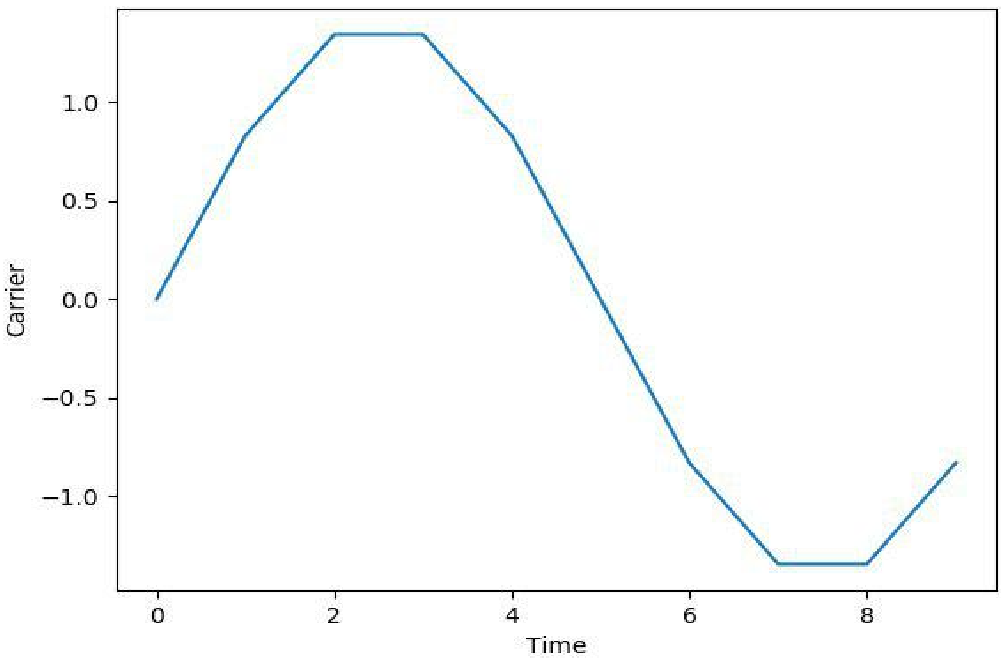
```
ber=[]
finalarr=[]
for i in range(0,len(q),size):
        if q[i]==-1:
                finalarr.append(0)
        else:
                finalarr.append(1)



print("\nBER")
print(ber)
print(len(ber))

print("\nFinal arary")
print(finalarr);
print(len(finalarr))
print("\nbinary input")
print(bin_ip_seq)


figure(1)
plt.plot(bpsksignal)
xlabel('Time')
ylabel('BPSK Signal')


figure(2)
plt.plot(t1,ber, 'bo',t1,ber,'k')
xlabel('SNR')
ylabel('BER')
plt.plot(snr[25000:25050],ber)
show()

#check whether input sequence in same as received seq
for i in range(n):
        if (finalarr[i]==bin_ip_seq[i]):
                c=1
        else:
                c=0
                break

if(c==1):
        print("\nyes")
else:
        print("NO")

print(len(snr))
print(len(jamming))
print(len(ber))
print(len(carrier))
print(len(q))


plt.show()
```

5 OBSERVATIONS :

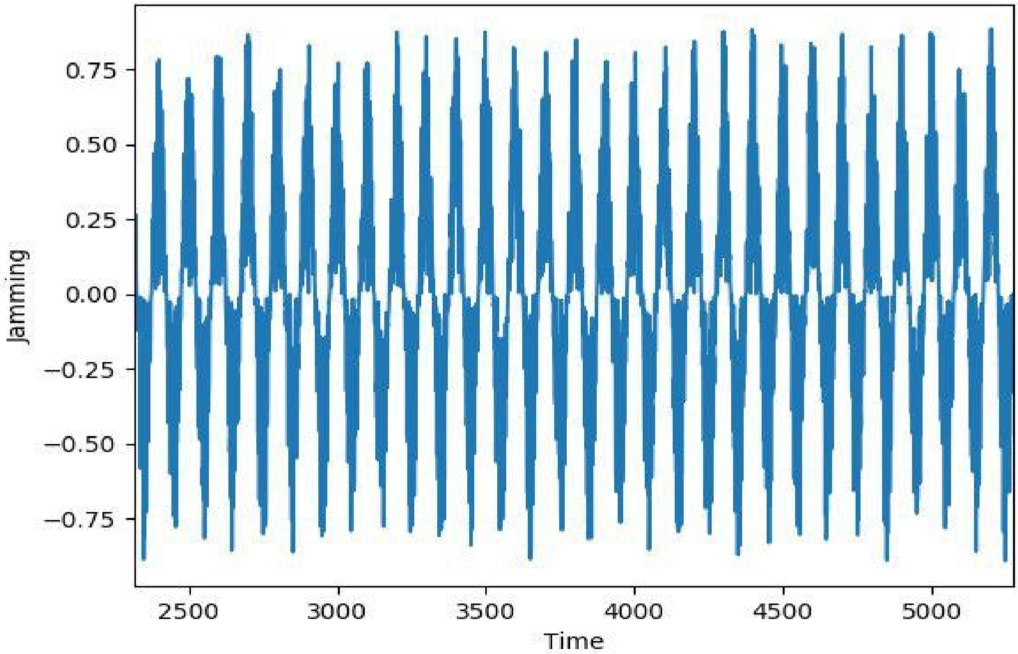
Input Signal



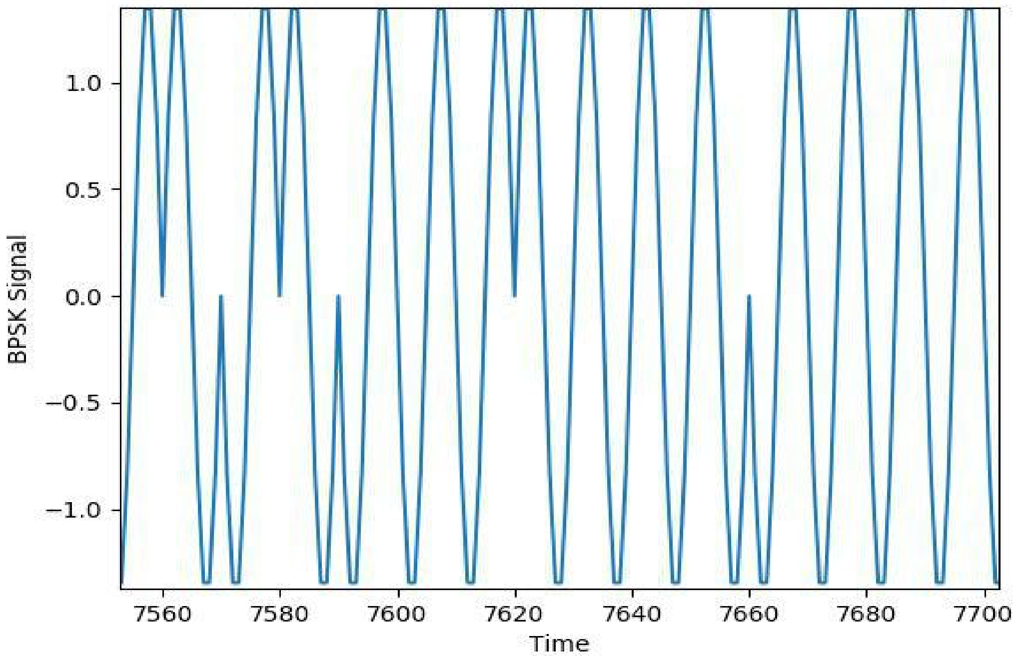
Carrier Signal



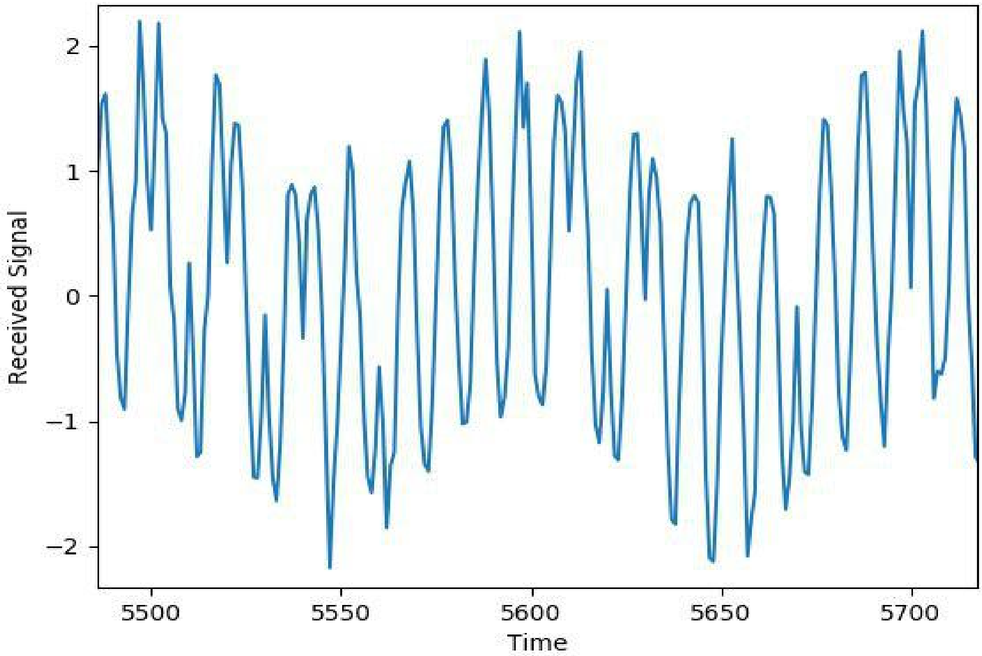
Jamming Signal



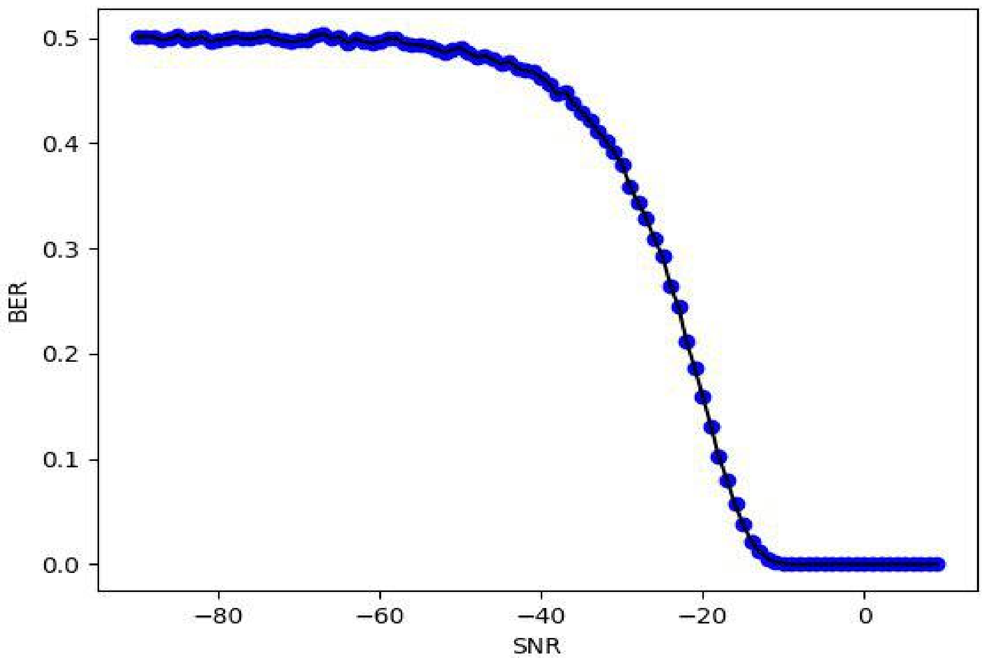
Bpsk Signal



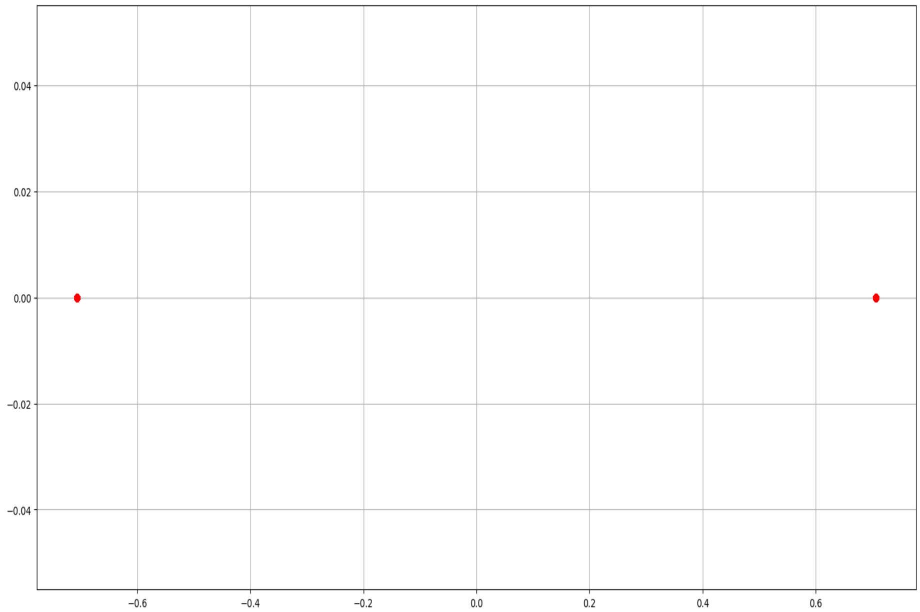
Received Signal



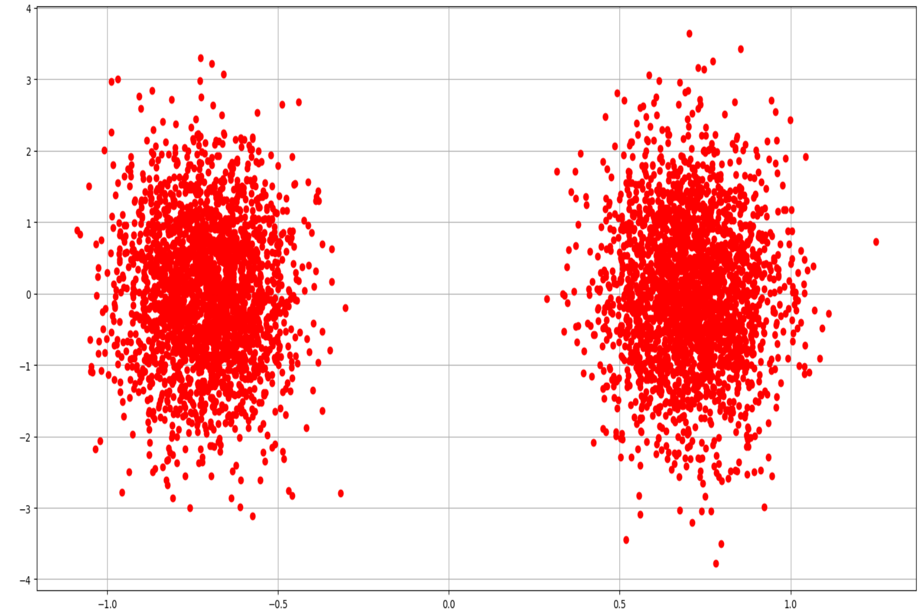
Bit Error Rate Vs Signal to Noise Ratio



Signal Constellation Without Jamming



Signal Constellation With Jamming



6 RESULT :

Simulated DS-BPSK system in python (Transmitter and Receiver with PN sequence and Jamming) and to plot SNR vs BER, Constellation of BPSK.