EE 670A Wireless Communications

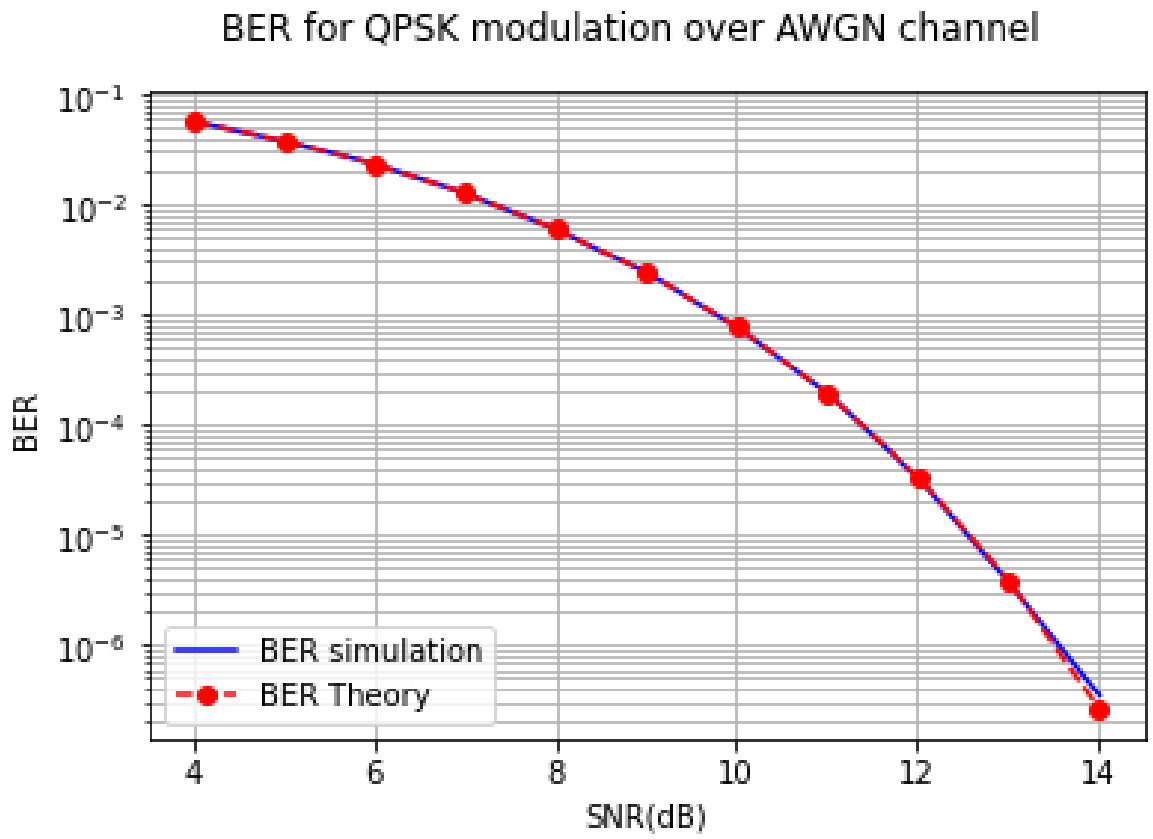
PYTHON Assignment #1

NAME :S Srikanth Reddy
RollNo :22104092

AWGN Channel:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Aug  31 08:50:46 2022

@author: S Srikanth Reddy
"""

import numpy as np
import matplotlib.pyplot as plt
import numpy.random as nr
from scipy.stats import norm as nrm

blockLength = 10000;
nBlocks = 1000;
No =1; "Noise power  = No/2, consider No=1"
EbdB = np.arange(1.0,11.1,1); "sequence of energy per bit in dB"
Eb = 10**(EbdB/10); "conversion to normal scale"
SNR = 2*Eb/No; "SNR calculation, SNR = (energy per bit)/(noise power)"
SNRdB = 10*np.log10(SNR); "conversion to dB"
BER = np.zeros(len(EbdB));
BERt = np.zeros(len(EbdB));

for blk in range(nBlocks):
  BitsI = nr.randint(2,size=blockLength); "generating inphase bits in random"
  BitsQ = nr.randint(2,size=blockLength); "generating quadrature bits in random"
  Sym = (2*BitsI-1)+1j*(2*BitsQ-1); "QPSK modulation"
  noise=nr.normal(0,np.sqrt(No/2),blockLength)+1j*nr.normal(0,np.sqrt(No/2),blockLength);
"noise is complex Gaussian where its real part as well as imaginary part are also Gaussian with
mean 0 and variance No/2"
  for K in range(len(EbdB)):
    TxSym = np.sqrt(Eb[K])*Sym;
    RxSym = TxSym + noise; "system model for AWGN channel"
    DecBitsI = (np.real(RxSym)>0);
    DecBitsQ = (np.imag(RxSym)>0);
```

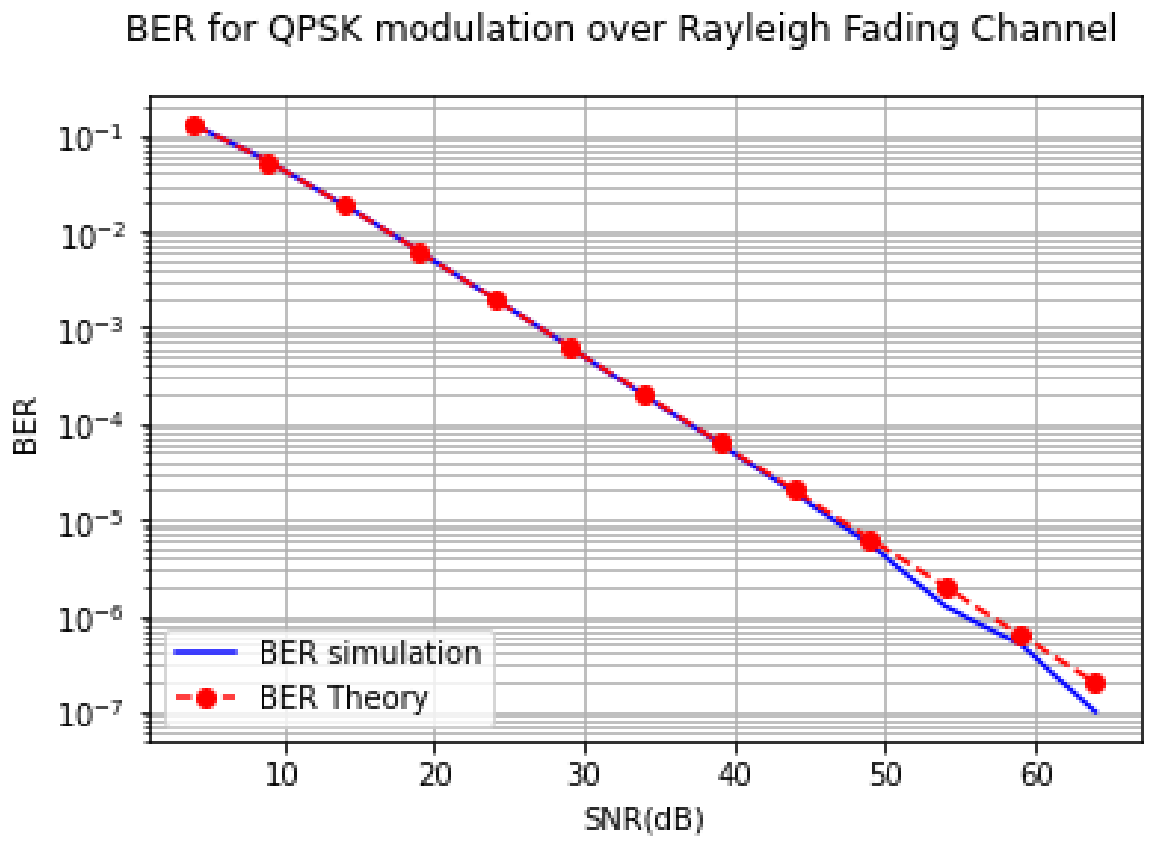
```
    BER[K] = BER[K] + np.sum(DecBitsI != BitsI) + np.sum(DecBitsQ != BitsQ)


BER = BER/blockLength/2/nBlocks; "calculating BER for the system"
BERt = 1- nrm.cdf(np.sqrt(SNR)); "calculating theoretical BER"

plt.yscale('log')
plt.plot(SNRdB,BER,'b-')
plt.plot(SNRdB,BERt,'r--o')
plt.grid(1,which = 'both')
plt.suptitle('BER for QPSK modulation over AWGN channel')
plt.xlabel('SNR(dB)')
plt.ylabel('BER')
plt.legend(["BER simulation","BER Theory"],loc="lower left")
```



Rayleigh Fading channel:

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as nr

blockLength = 10000;
nBlocks = 1000;
No =1;
Eb_dB = np.arange(1.0,66,5);
Eb = 10**(Eb_dB/10);
SNR = 2*Eb/No;
SNR_dB = 10*np.log10(SNR);
BER = np.zeros(len(Eb_dB));
BERt = np.zeros(len(Eb_dB));

for blk in range(nBlocks):
  BitsI = nr.randint(2,size=blockLength);
  BitsQ = nr.randint(2,size=blockLength);
  Sym = (2*BitsI-1)+1j*(2*BitsQ-1);
  noise = nr.normal(0,np.sqrt(No/2),blockLength)+1j*nr.normal(0,np.sqrt(No/2),blockLength);
  h=nr.normal(0,np.sqrt(1/2),blockLength)+1j*nr.normal(0,np.sqrt(1/2),blockLength);
"defining Rayleigh Fading channel coefficient"

  for K in range(len(Eb_dB)):
    TxSym = np.sqrt(Eb[K])*Sym;
    RxSym = h*TxSym + noise; "system model for Rayleigh Fading channel"
    EqSym = 1/h*RxSym;
    DecBitsI = (np.real(EqSym)>0);
    DecBitsQ = (np.imag(EqSym)>0);
    BER[K] = BER[K] + np.sum(DecBitsI != BitsI) + np.sum(DecBitsQ!= BitsQ)

BER = BER/blockLength/2/nBlocks;
BERt = 1/2*(1-np.sqrt(SNR/(2+SNR)));
```

```
plt.yscale('log')
plt.plot(SNR_dB,BER,'b-')
plt.plot(SNR_dB,BERt,'r--o')
plt.grid(1,which = 'both')
plt.suptitle('BER for QPSK modulation over Rayleigh Fading Channel ')
plt.xlabel('SNR(dB)')
plt.ylabel('BER')
plt.legend(["BER simulation","BER Theory"],loc="lower left")
```



Observation:

For both AWGN and Rayleigh Fading channels BER has been simulated vs SNR and plotted as shown. Also the theoretical BER vs SNR plot has been superposed on top. We can see that the simulation is very close to the theoretical curve and deviates slightly at high SNRs.