EE 670A Wireless Communications

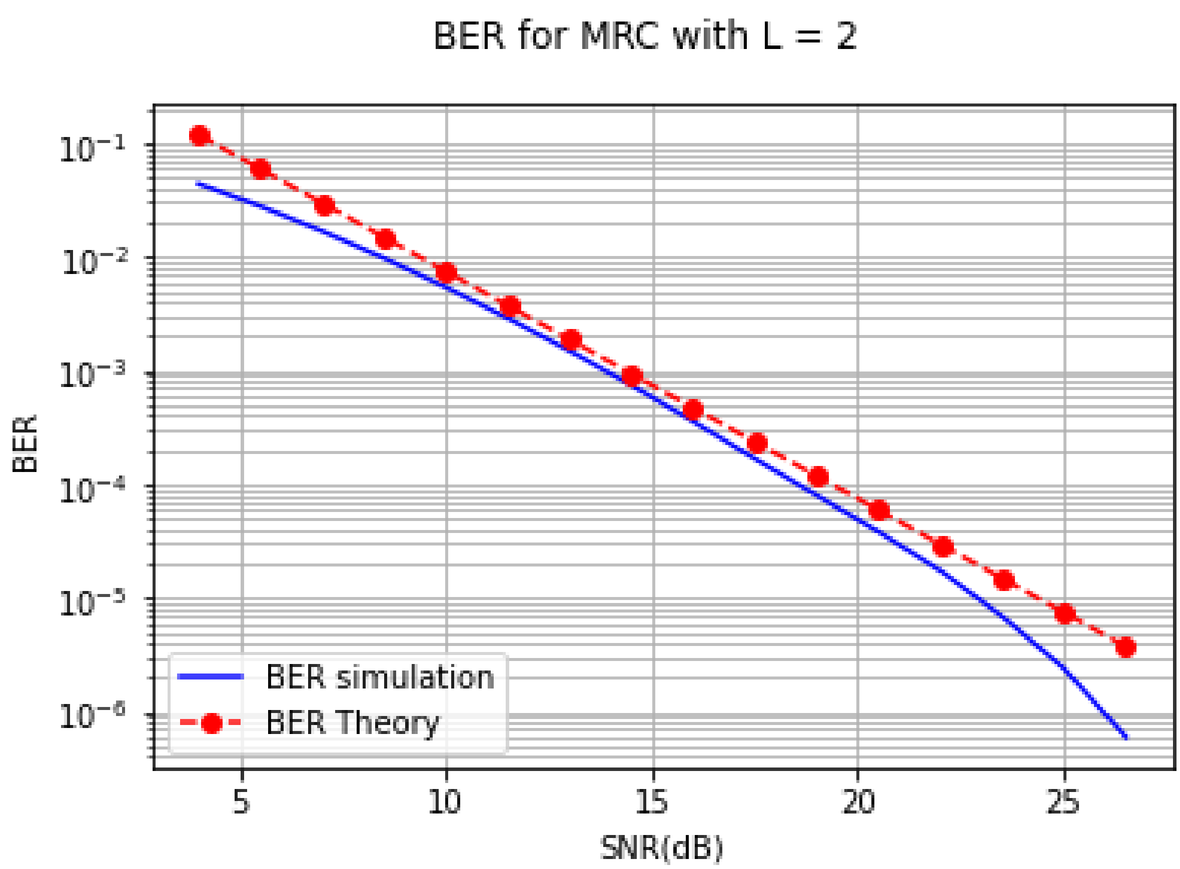
PYTHON Assignment #2

NAME :S Srikanth Reddy
RollNo :22104092

L=2 system:

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as nr
from scipy.special import comb

blockLength = 1000; "number of symbols per block"
nBlocks = 10000; "number of blocks"
No =1; "noise power"
L=2; "number of receive antennas"
Eb_dB = np.arange(1.0,25,1.5); "energy per bit in dB"
Eb = 10**(Eb_dB/10);
SNR = 2*Eb/No; "Signal-to-Noise power Ratio calculation"
SNR_dB = 10*np.log10(SNR); "conversion to dB"
BER = np.zeros(len(Eb_dB));
BERt = np.zeros(len(Eb_dB));

for blk in range(nBlocks):
    "Rayleigh fading channel coefficient with unit average power"
    h=(nr.normal(0.0,1.0,(L,1))+1j*nr.normal(0.0,1.0,(L,1)))/np.sqrt(2);
    "ZMSCG Noise with power No"
    noise =
nr.normal(0.0,np.sqrt(No/2),(L,blockLength))+1j*nr.normal(0.0,np.sqrt(No/2),(L,blockLength));
    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"

    for K in range(len(SNR_dB)):
        TxSym = np.sqrt(Eb[K])*Sym; "transmit symbols with amplitude scaling"
        RxSym = h*TxSym + noise; "output symbol reception across L receive antennas"
        MRCout = np.sum(np.conj(h)*RxSym,axis=0); "employing Maximal Ratio Combining"
        DecBitsI = (np.real(MRCout)>0); "decoded inphase bits"
        DecBitsQ = (np.imag(MRCout)>0); "decoded quadrature bits"
```

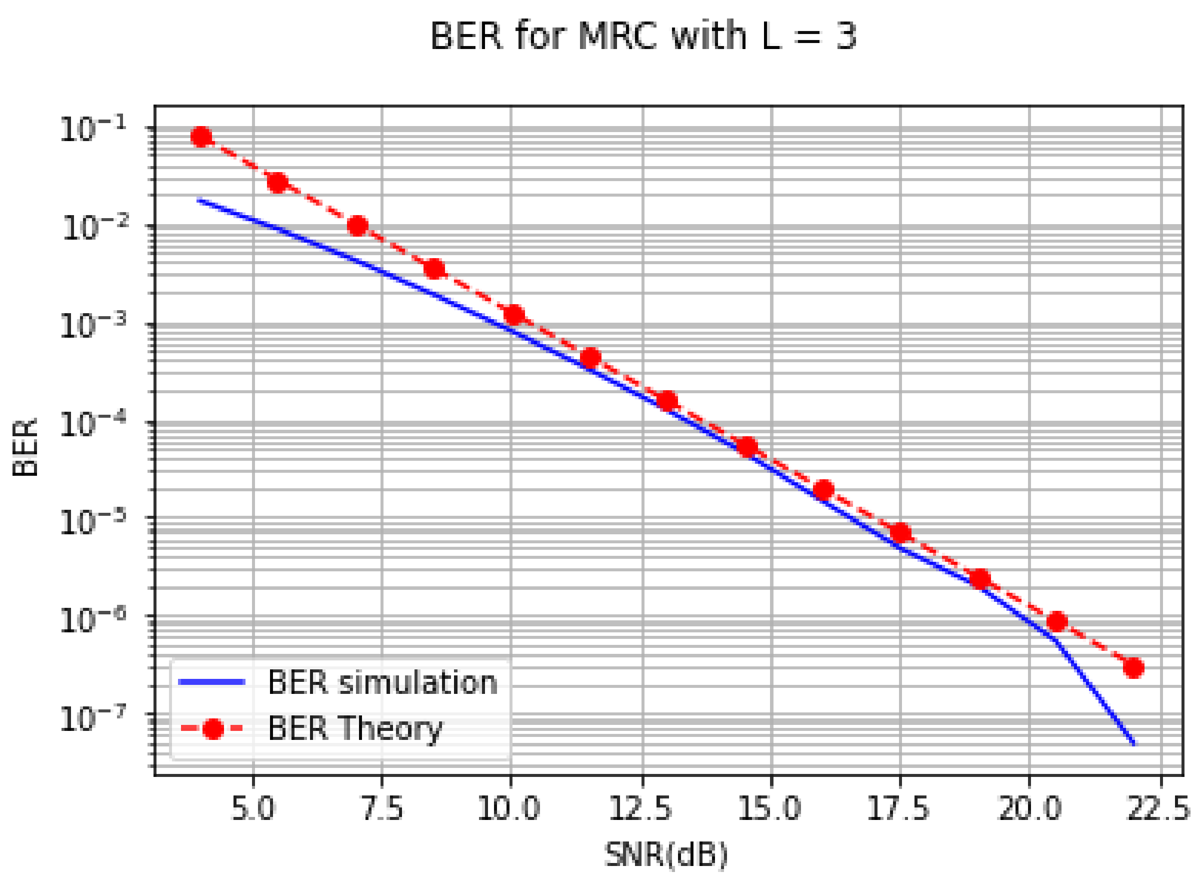
```
        "calculating total number of bits in error"
        BER[K] = BER[K] + np.sum(DecBitsI != BitsI) + np.sum(DecBitsQ!= BitsQ)

BER = BER/blockLength/2/nBlocks; "simulation BER"
BERt = comb(2*L-1,L-1)/2**L/SNR**L; "theoritical BER"
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 MRC with L = 2')
plt.xlabel('SNR(dB)')
plt.ylabel('BER')
plt.legend(["BER simulation","BER Theory"],loc="lower left")
```



L=3 system:

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as nr
from scipy.special import comb

blockLength = 1000; "number of symbols per block"
nBlocks = 10000; "number of blocks"
No =1; "noise power"
L=3; "number of receive antennas"
Eb_dB = np.arange(1.0,20,1.5); "energy per bit in dB"
Eb = 10**(Eb_dB/10);
SNR = 2*Eb/No; "Signal-to-Noise power Ratio calculation"
SNR_dB = 10*np.log10(SNR); "conversion to dB"
BER = np.zeros(len(Eb_dB));
BERt = np.zeros(len(Eb_dB));

for blk in range(nBlocks):
    "Rayleigh fading channel coefficient with unit average power"
    h=(nr.normal(0.0,1.0,(L,1))+1j*nr.normal(0.0,1.0,(L,1)))/np.sqrt(2);
    "ZMSCG Noise with power No"
    noise =
nr.normal(0.0,np.sqrt(No/2),(L,blockLength))+1j*nr.normal(0.0,np.sqrt(No/2),(L,blockLength));
    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"

    for K in range(len(SNR_dB)):
        TxSym = np.sqrt(Eb[K])*Sym; "transmit symbols with amplitude scaling"
        RxSym = h*TxSym + noise; "output symbol reception across L receive antennas"
        MRCout = np.sum(np.conj(h)*RxSym,axis=0); "employing Maximal Ratio Combining"
        DecBitsI = (np.real(MRCout)>0); "decoded inphase bits"
        DecBitsQ = (np.imag(MRCout)>0); "decoded quadrature bits"
```

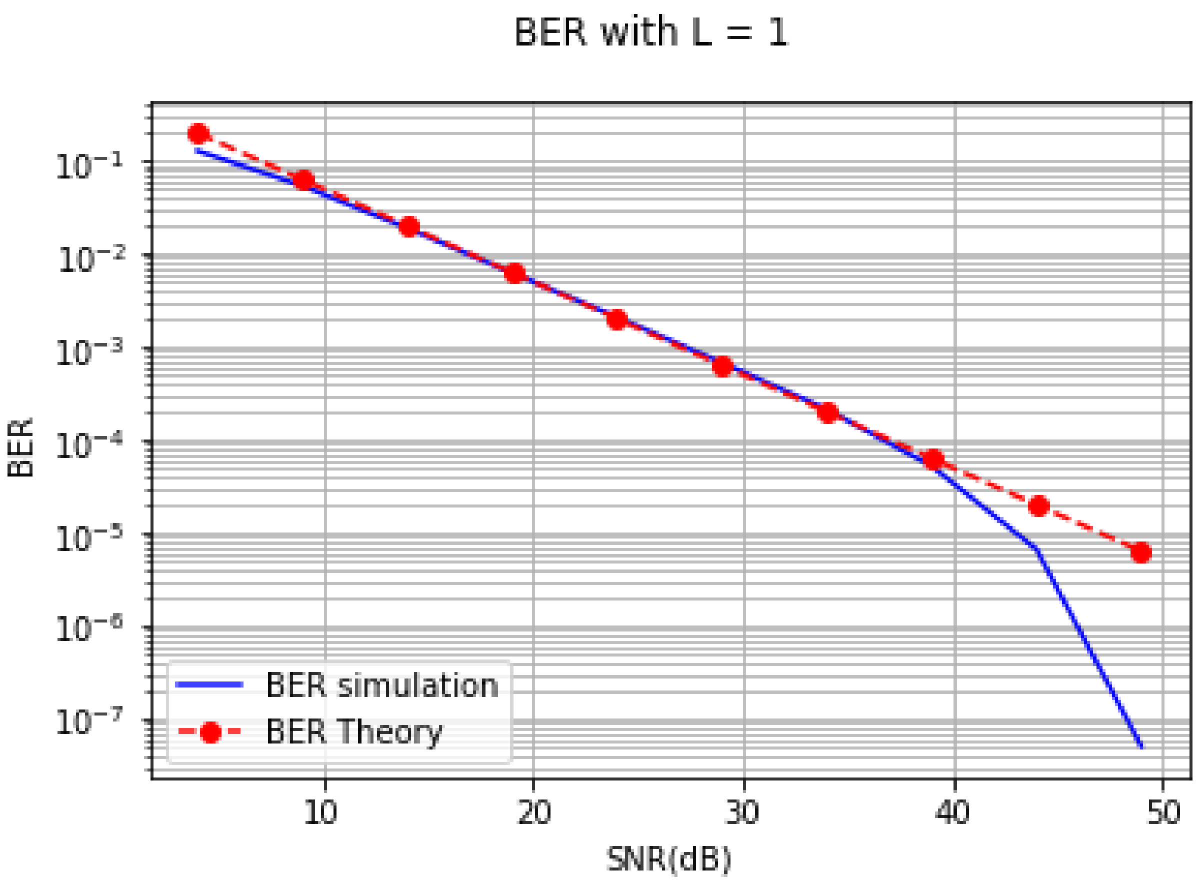
```
        "calculating total number of bits in error"
        BER[K] = BER[K] + np.sum(DecBitsI != BitsI) + np.sum(DecBitsQ!= BitsQ)

BER = BER/blockLength/2/nBlocks; "simulation BER"
BERt = comb(2*L-1,L-1)/2**L/SNR**L; "theoritical BER"
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 MRC with L = 3')
plt.xlabel('SNR(dB)')
plt.ylabel('BER')
plt.legend(["BER simulation","BER Theory"],loc="lower left")
```



Comparision with L=1 system:

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as nr
from scipy.special import comb

blockLength = 1000; "number of symbols per block"
nBlocks = 10000; "number of blocks"
No =1; "noise power"
L=1; "number of receive antennas"
Eb_dB = np.arange(1.0,50,5); "energy per bit in dB"
Eb = 10**(Eb_dB/10);
SNR = 2*Eb/No; "Signal-to-Noise power Ratio calculation"
SNR_dB = 10*np.log10(SNR); "conversion to dB"
BER = np.zeros(len(Eb_dB));
BERt = np.zeros(len(Eb_dB));

for blk in range(nBlocks):
    "Rayleigh fading channel coefficient with unit average power"
    h=(nr.normal(0.0,1.0,(L,1))+1j*nr.normal(0.0,1.0,(L,1)))/np.sqrt(2);
    "ZMSCG Noise with power No"
    noise =
nr.normal(0.0,np.sqrt(No/2),(L,blockLength))+1j*nr.normal(0.0,np.sqrt(No/2),(L,blockLength));
    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"

    for K in range(len(SNR_dB)):
        TxSym = np.sqrt(Eb[K])*Sym; "transmit symbols with amplitude scaling"
        RxSym = h*TxSym + noise; "output symbol reception across L receive antennas"
        MRCout = np.sum(np.conj(h)*RxSym,axis=0); "employing Maximal Ratio Combining"
        DecBitsI = (np.real(MRCout)>0); "decoded inphase bits"
        DecBitsQ = (np.imag(MRCout)>0); "decoded quadrature bits"
```

```
        "calculating total number of bits in error"
        BER[K] = BER[K] + np.sum(DecBitsI != BitsI) + np.sum(DecBitsQ!= BitsQ)

BER = BER/blockLength/2/nBlocks; "simulation BER"
BERt = comb(2*L-1,L-1)/2**L/SNR**L; "theoritical BER"
plt.yscale('log')
plt.plot(SNR_dB,BER,'b-')
plt.plot(SNR_dB,BERt,'r--o')
plt.grid(1,which = 'both')
plt.suptitle('BER with L = 1')
plt.xlabel('SNR(dB)')
plt.ylabel('BER')
plt.legend(["BER simulation","BER Theory"],loc="lower left")
```



Observation:

As we increase the number of receive antennas (L) and apply Maximal Ratio Combining (MRC) the diversity order increases, thus reducing the BER for a particular value of SNR.

In other words, to obtain same BER, SNR required will decrease when increasing L.