EE670 - Wireless Communications

Python Assignment #2

Prajeeth Babu Kodru

22104071

Question:

Simulate **multiple antenna Rayleigh fading** communication systems with $L = 2,3$ antennas in PYTHON. Generate the BER curves vs SNR for QPSK detection in the SNR range required to achieve at least $BER = 10^{-6}$ for each system. Superpose the analytical curves over them and compare with $L = 1$ antenna system. Submit the code and relevant plot for each system.

Solution:

Code:

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

blockLength = 10000; # Number of symbols per block
nBlocks = 10000; # Number of blocks
EbdB = np.arange(1.0,18.1,1.5); # Energy per bit in dB
Eb = 10**(EbdB/10); # Energy per bit Eb
No = 1; # Total noise power No
SNR = 2*Eb/No; # Signal-to-noise power ratio
SNRdB = 10*np.log10(Eb/No); # SNR values in dB
BER1 = np.zeros(len(EbdB)); # Bit error rate (BER) values
BER2 = np.zeros(len(EbdB));
BER3 = np.zeros(len(EbdB));
BERt1 = np.zeros(len(EbdB)); # Analytical values of BER from formula
BERt2 = np.zeros(len(EbdB));
BERt3 = np.zeros(len(EbdB));

#L=1
for blk in range(nBlocks):
    #Rayleigh fading channel coefficient with avg power unity
    h=nr.normal(0,np.sqrt(1/2))+1j*nr.normal(0,np.sqrt(1/2));
    #Complex gaussian noise with power No

noise=nr.normal(0,np.sqrt(No/2),blockLength)+1j*nr.normal(0,np.sqrt(No/2),(b
lockLength));
    BitsI=nr.randint(2,size=blockLength); #Bits for I channel
    BitsQ=nr.randint(2,size=blockLength); #Bits for Q channel
```

```
    Sym=(2*BitsI-1)+1j*(2*BitsQ-1); #Complex QPSK symbols

    for K in range(len(SNRdB)):
        #Tx operation
        TxSym=np.sqrt(Eb[K])*Sym; #Transmit symbols with power scaling

        #Multi-antenna channel and reception
        RxSym=h*TxSym+noise; #Output symbols across L antennas

        #Rx for MRC
        MRCout=1/h*RxSym;
        DecBitsI=(np.real(MRCout)>0); #Decoded bits for I channel
        DecBitsQ=(np.imag(MRCout)>0); #Decoded bits for Q channel
        #Evaluating total number of bit errors

BER1[K]=BER1[K]+np.sum(DecBitsI!=BitsI)+np.sum(DecBitsQ!=BitsQ);

BER1 = BER1/blockLength/nBlocks/2;# Evaluating BER from simulation
BERt1 = 0.5*(1-np.sqrt(SNR/(2+SNR)));  # Evaluating BER from formula

#for L=2
L=2;
for blk in range(nBlocks):
    #Rayleigh fading channel coefficient with avg power unity
    h1=(nr.normal(0,1,(L,1))+1j*nr.normal(0,1,(L,1)))/np.sqrt(2);
    h2=(nr.normal(0,1,(L,1))+1j*nr.normal(0,1,(L,1)))/np.sqrt(2);
    #Complex gaussian noise with power No

noise1=(nr.normal(0,np.sqrt(No/2),(L,blockLength))+1j*nr.normal(0,np.sqrt(No
/2),(L,blockLength)));

noise2=(nr.normal(0,np.sqrt(No/2),(L,blockLength))+1j*nr.normal(0,np.sqrt(No
/2),(L,blockLength)));
    BitsI=nr.randint(2,size=blockLength); #Bits for I channel
    BitsQ=nr.randint(2,size=blockLength); #Bits for Q channel
    Sym=(2*BitsI-1)+1j*(2*BitsQ-1); #Complex QPSK symbols

    w1 = h1/(np.abs(h1)**2 + np.abs(h2)**2);
    w2 = h2/(np.abs(h1)**2 + np.abs(h2)**2);
    for K in range(len(SNRdB)):
```

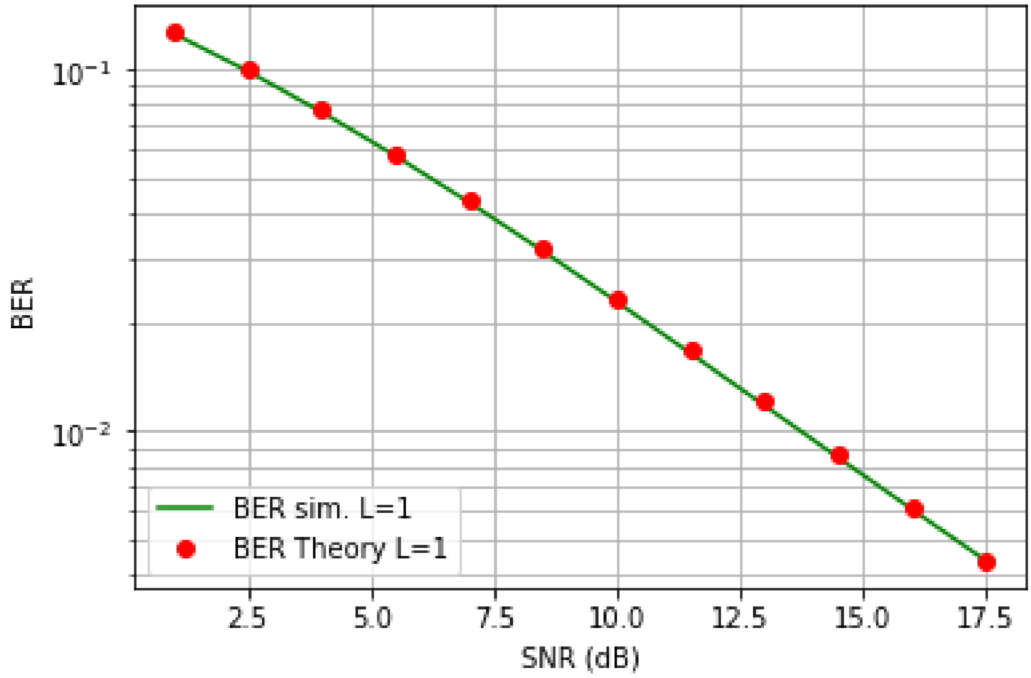
```
        #Tx operation
        TxSym=np.sqrt(Eb[K])*Sym; #Transmit symbols with power scaling

        #Multi-antenna channel and reception
        RxSym1=h1*TxSym+noise1; #Output symbols across L antennas
        RxSym2=h2*TxSym+noise2;

        #Rx for MRC
        MRCout=np.conj(w1)*RxSym1+np.conj(w2)*RxSym2;
        DecBitsI=(np.real(MRCout)>0); #Decoded bits for I channel
        DecBitsQ=(np.imag(MRCout)>0); #Decoded bits for Q channel
        #Evaluating total number of bit errors

BER2[K]=BER2[K]+np.sum(DecBitsI!=BitsI)+np.sum(DecBitsQ!=BitsQ);

BER2 = BER2/blockLength/nBlocks/2;
BERt2 = comb(2*L-1, L)/2**L/SNR**L;

#for L=3
L=3;
for blk in range(nBlocks):
    #Rayleigh fading channel coefficient with avg power unity
    h1=(nr.normal(0,1,(L,1))+1j*nr.normal(0,1,(L,1)))/np.sqrt(2);
    h2=(nr.normal(0,1,(L,1))+1j*nr.normal(0,1,(L,1)))/np.sqrt(2);
    h3=(nr.normal(0,1,(L,1))+1j*nr.normal(0,1,(L,1)))/np.sqrt(2);
    #Complex gaussian noise with power No

noise1=(nr.normal(0,np.sqrt(No/2),(L,blockLength))+1j*nr.normal(0,np.sqrt(No
/2),(L,blockLength)));

noise2=(nr.normal(0,np.sqrt(No/2),(L,blockLength))+1j*nr.normal(0,np.sqrt(No
/2),(L,blockLength)));

noise3=(nr.normal(0,np.sqrt(No/2),(L,blockLength))+1j*nr.normal(0,np.sqrt(No
/2),(L,blockLength)));

    BitsI=nr.randint(2,size=blockLength); #Bits for I channel
    BitsQ=nr.randint(2,size=blockLength); #Bits for Q channel
    Sym=(2*BitsI-1)+1j*(2*BitsQ-1); #Complex QPSK symbols
```

```
    w1 = h1/(np.abs(h1)**2 + np.abs(h2)**2+ np.abs(h3)**2);
    w2 = h2/(np.abs(h1)**2 + np.abs(h2)**2+ np.abs(h3)**2);
    w3 = h3/(np.abs(h1)**2 + np.abs(h2)**2+ np.abs(h3)**2);

    for K in range(len(SNRdB)):
        #Tx operation
        TxSym=np.sqrt(Eb[K])*Sym; #Transmit symbols with power scaling

        #Multi-antenna channel and reception
        RxSym1=h1*TxSym+noise1; #Output symbols across L antennas
        RxSym2=h2*TxSym+noise2;
        RxSym3=h3*TxSym+noise3;
        #Rx for MRC

MRCout=np.conj(w1)*RxSym1+np.conj(w2)*RxSym2+np.conj(w3)*RxSym3;
        DecBitsI=(np.real(MRCout)>0); #Decoded bits for I channel
        DecBitsQ=(np.imag(MRCout)>0); #Decoded bits for Q channel
        #Evaluating total number of bit errors

BER3[K]=BER3[K]+np.sum(DecBitsI!=BitsI)+np.sum(DecBitsQ!=BitsQ);


BER3 = BER3/blockLength/nBlocks/2;
BERt3 = comb(2*L-1, L)/2**L/SNR**L;

# Plotting the bit error rate from Simulation and formula
plt.figure(1)
plt.yscale('log')
plt.plot(SNRdB, BER1,'g-')
plt.plot(SNRdB, BERt1,'ro')
plt.grid(1,which='both')
plt.suptitle('BER for Rayleigh fading channel with L=1')
plt.legend(["BER sim. L=1", "BER Theory L=1"], loc ="lower left");
plt.xlabel('SNR (dB)')
plt.ylabel('BER')

plt.figure(2)
plt.yscale('log')
plt.plot(SNRdB, BER2,'b-')
plt.plot(SNRdB, BERt2,'yo')
```

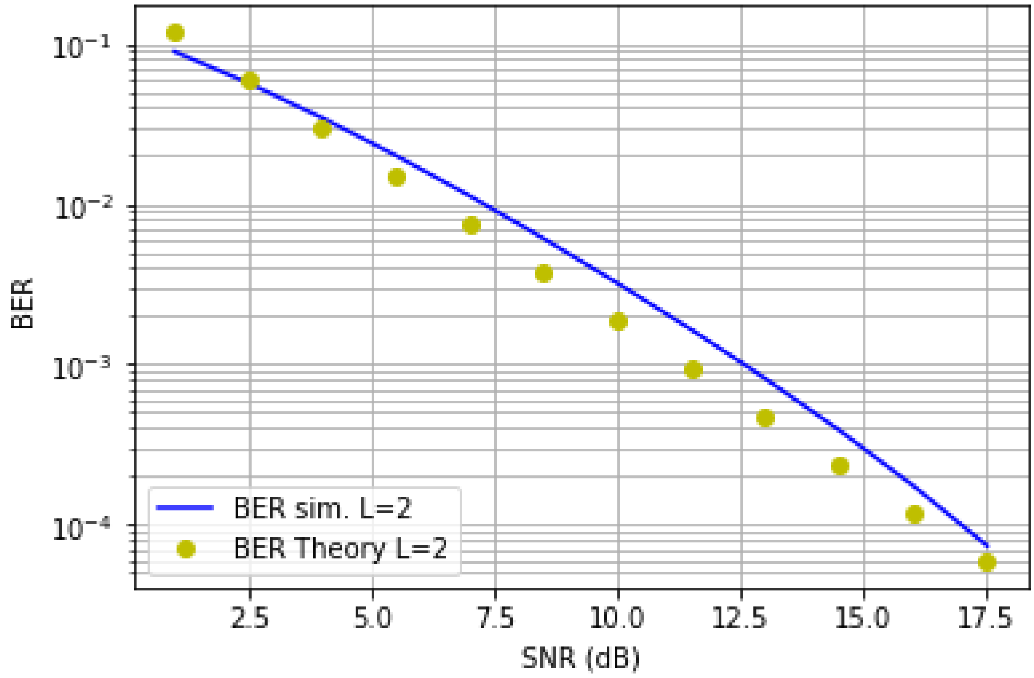
```
plt.grid(1,which='both')
plt.suptitle('BER for Rayleigh fading channel with L=2')
plt.legend(["BER sim. L=2", "BER Theory L=2"], loc ="lower left");
plt.xlabel('SNR (dB)')
plt.ylabel('BER')

plt.figure(3)
plt.yscale('log')
plt.plot(SNRdB, BER3,'y-')
plt.plot(SNRdB, BERt3,'ko')
plt.grid(1,which='both')
plt.suptitle('BER for Rayleigh fading channel with L=3')
plt.legend(["BER sim. L=3", "BER Theory L=3"], loc ="lower left");
plt.xlabel('SNR (dB)')
plt.ylabel('BER')

plt.figure(4)
plt.yscale('log')
plt.plot(SNRdB, BER1,'g-')
plt.plot(SNRdB, BERt1,'ro')
plt.plot(SNRdB, BER2,'b-')
plt.plot(SNRdB, BERt2,'yo')
plt.plot(SNRdB, BER3,'y-')
plt.plot(SNRdB, BERt3,'ko')
plt.grid(1,which='both')
plt.suptitle('BER for MRC')
plt.legend(["BER sim. L=1", "BER Theory L=3","BER sim. L=2", "BER
Theory L=2","BER sim. L=3", "BER Theory L=3"], loc ="lower left");
plt.xlabel('SNR (dB)')
plt.ylabel('BER')
```

Output

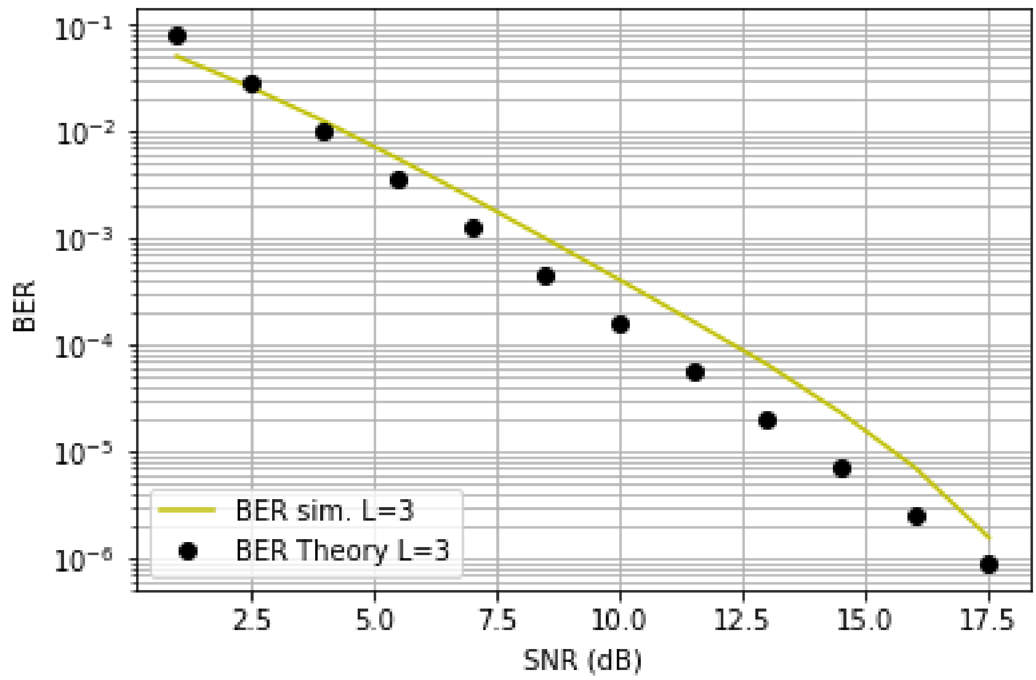
BER for Rayleigh fading channel with L=1



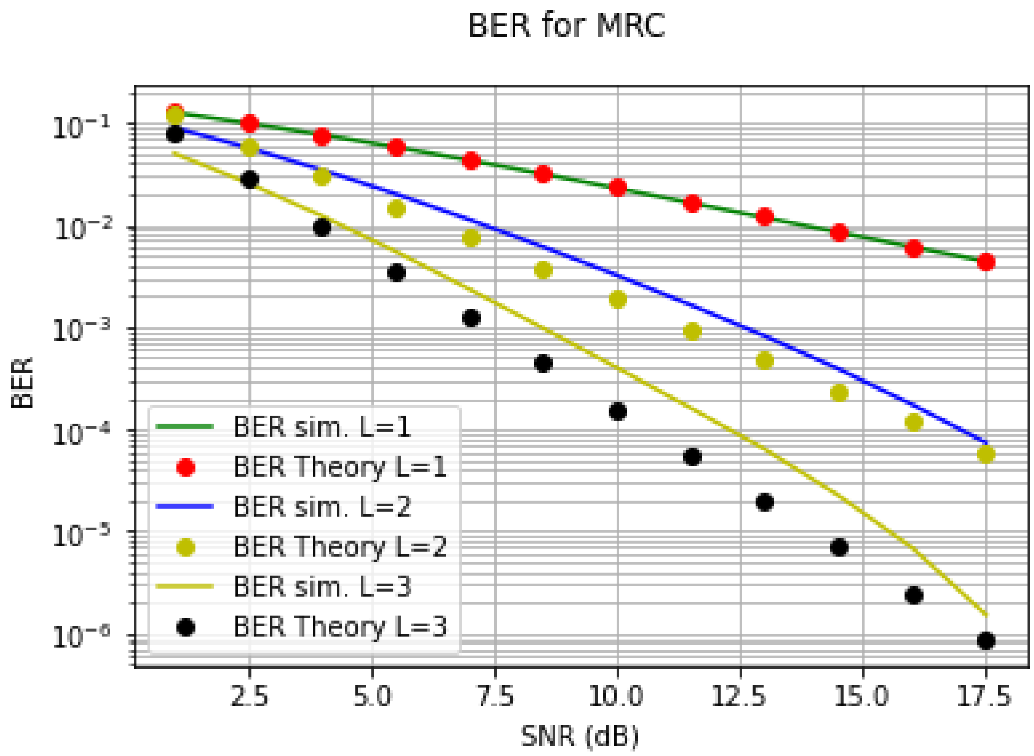
BER for Rayleigh fading channel with L=2



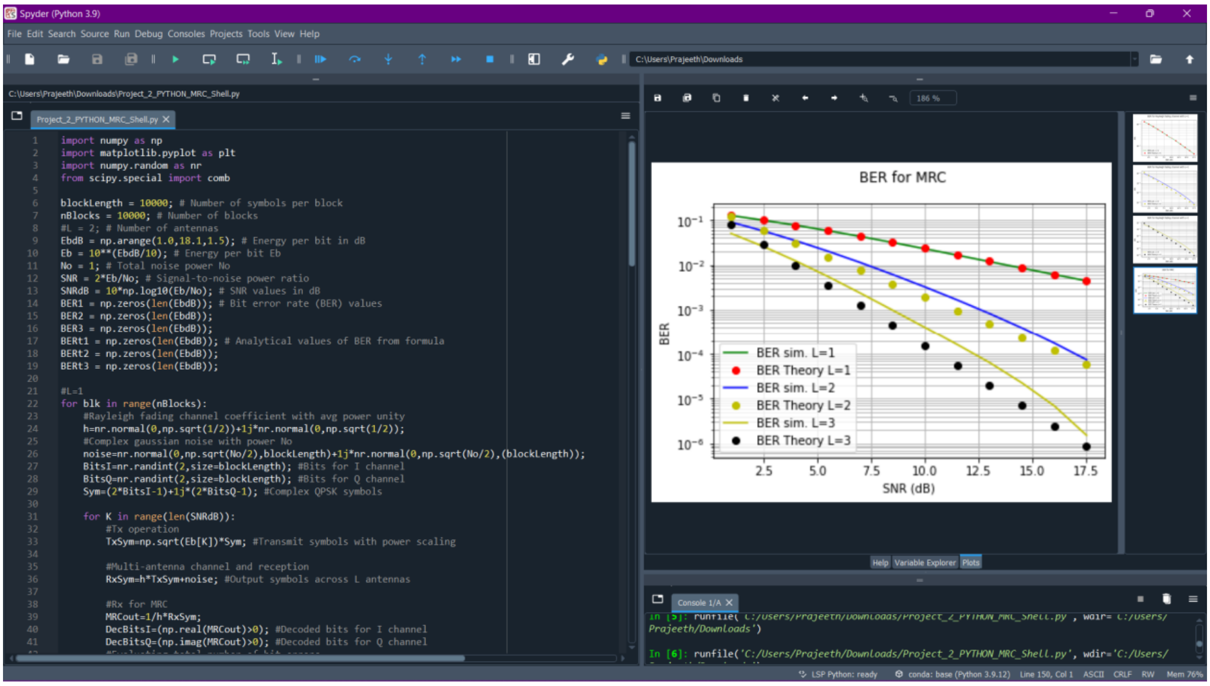
BER for Rayleigh fading channel with L=3



Comparing L=2, L=3 with L=1



Output in spyder



Conclusion

BER for L receiver antenna system with MRC with high SNR approximation is given by-

$$BER = {}^{2L-1}C_L \times \frac{1}{2^L} \times \frac{1}{SNR^L}$$

Therefore for L=1,2,3 the BER is given as

$$BER_{L=1} = \frac{1}{2SNR}$$

$$BER_{L=2} = \frac{3}{4}\frac{1}{SNR^2}$$

$$BER_{L=3} = \frac{5}{4}\frac{1}{SNR^3}$$

As we increase number of receive antennas (L), BER of the system decreases as given by

$$BER \propto \frac{1}{SNR^L}$$

So, by using Multiple Antenna System, We are able to reduce our BER.