

EE670 - Wireless Communications



Python Assignment #3

Prajeeth Babu Kodru

22104071

Question:

Simulate an OFDM wireless system in PYTHON with $N = 64$ subcarriers for a channel with $L = 3$ i.i.d. Rayleigh fading unit gain channel taps. Generate the BER curves vs dB SNR for QPSK symbols loaded over all the subcarriers and also superimpose the plots obtained via the corresponding analytical expression derived in class lectures. Choose the SNR range so as to obtain BER values up to 10^{-4} .

Solution:

Code:

```
import numpy as np;
import numpy.random as nr;
import matplotlib.pyplot as plt;
blocklength=64; #subcarriers
nblocks=10000;
No=1;
Ebdb=np.arange(1,55,5);
Eb= 10**((Ebdb/10));
SNR=2*Eb/No;
SNRdb=10*np.log10(SNR);
BER=np.zeros(len(Ebdb));
BERt=np.zeros(len(Ebdb));
L=3;
L_t=4;

for i in range(nblocks):
    bitsi=nr.randint(2,size=blocklength); #Bits for I channel
    bitsq=nr.randint(2,size=blocklength); #Bits for Q channel
    symbol=(2*bitsi-1)+1j*(2*bitsq-1); #Complex QPSK symbols
    n_v=nr.normal(0,np.sqrt(No/2),blocklength+L-
1+L_t)+1j*nr.normal(0,np.sqrt(No/2),blocklength+L-1+L_t);
    #Rayleigh fading channel coefficient with avg power unity
    h=nr.normal(0,np.sqrt(1/2),L)+1j*nr.normal(0,np.sqrt(1/2),L);
    H=np.fft.fft(h,blocklength);
    for j in range(len(Ebdb)):
        xs=np.sqrt(Eb[j])*symbol; #Transmit symbols with power scaling
        TxSym=np.fft.ifft(xs);
        cp=TxSym[blocklength-L_t:];
```

```

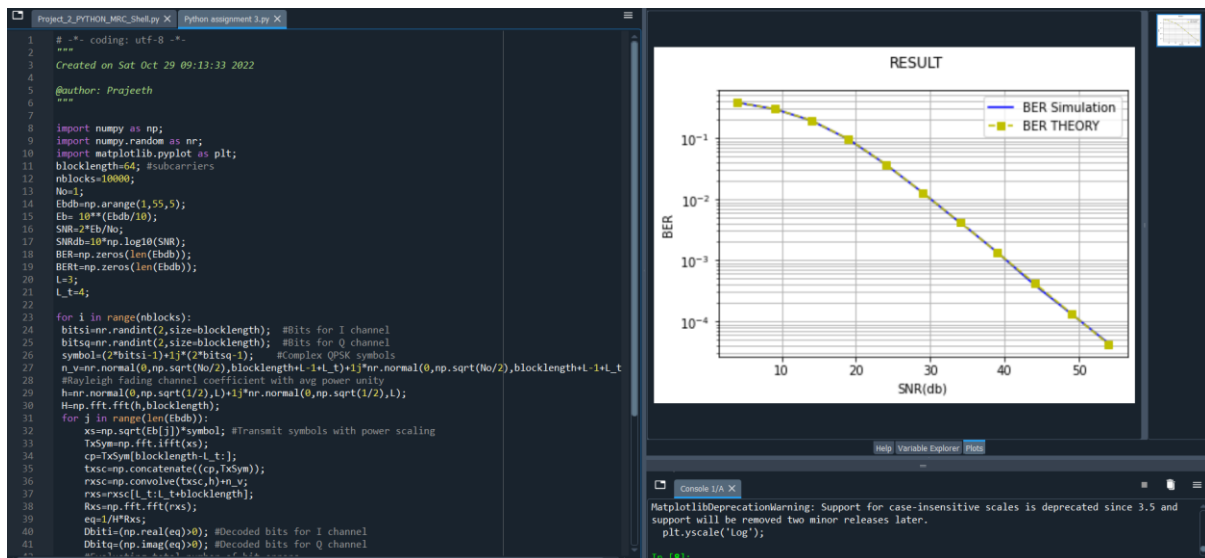
txsc=np.concatenate((cp,TxSym));
rxsc=np.convolve(txsc,h)+n_v;
rxs=rxsc[L_t:L_t+blocklength];
Rxs=np.fft.fft(rxs);
eq=1/H*Rxs;
Dbiti=(np.real(eq)>0); #Decoded bits for I channel
Dbitq=(np.imag(eq)>0); #Decoded bits for Q channel
#Evaluating total number of bit errors
BER[j]=BER[j]+np.sum(Dbiti!=bitsi)+np.sum(Dbitq!=bitsq);

BER=BER/blocklength/2/nblocks; #Evaluating BER from simulation
E_SNR = (L*SNR)/blocklength;
BERt = 0.5*(1-np.sqrt(E_SNR/(2+E_SNR))) # Evaluating BER from formula
plt.yscale('Log');
plt.plot(SNRdb,BER,'b')
plt.plot(SNRdb,BERt,'y--s')
plt.grid(1,which='both')
plt.suptitle('RESULT')
plt.xlabel('SNR(dB)')
plt.ylabel('BER')
plt.legend(["BER Simulation","BER THEORY"]);

```

Output

Output in spyder



RESULT

