EE670 - Wireless Communications

Python Assignment #1

Prajeeth Babu Kodru

22104071

Question:

1. Simulate AWGN and Rayleigh fading communication channels in PYTHON. Generate the BER curves vs SNR for QPSK detection in AWGN and Rayleigh Fading wireless channels in the SNR range required to achieve at least $BER = 10^{-6}$ for each system. Superpose the analytical curves over them. Submit the code and relevant plot for each system.

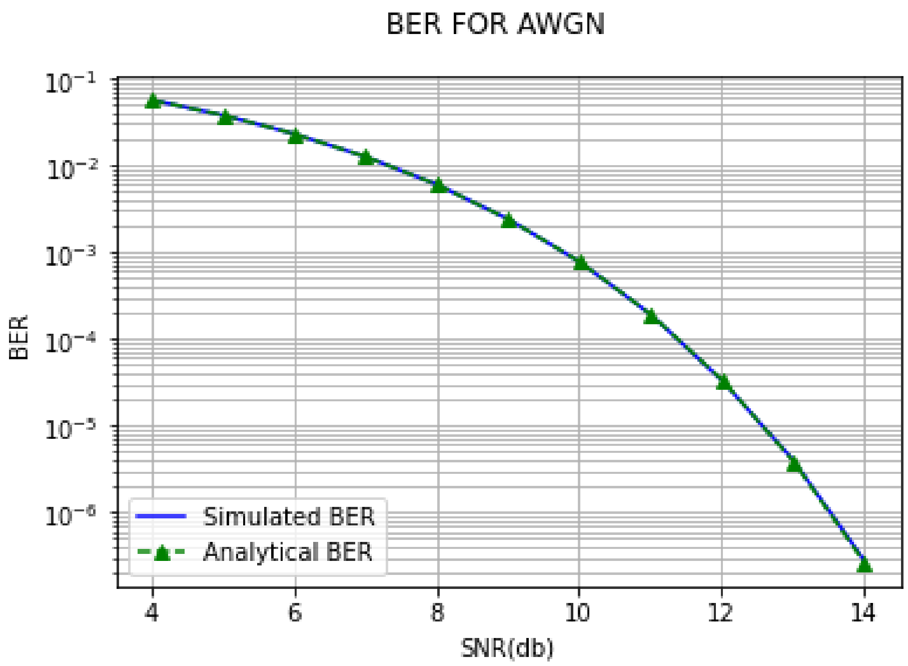
Solution:

Code for AWGN channel:

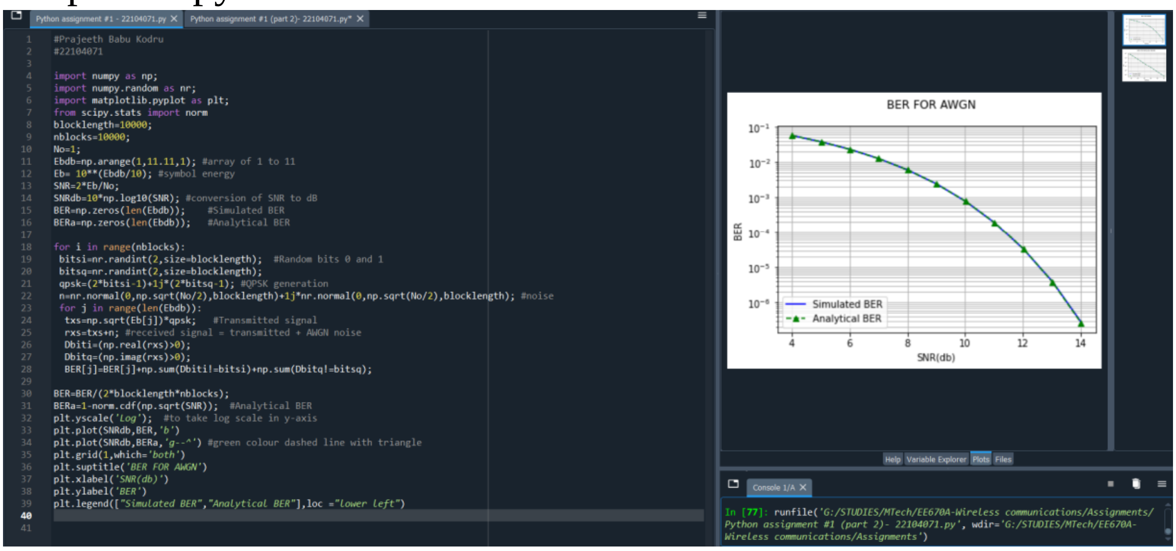
```
import numpy as np;
import numpy.random as nr;
import matplotlib.pyplot as plt;
from scipy.stats import norm
blocklength=10000;
nblocks=10000;
No=1;
Ebdb=np.arange(1,11.11,1); #array of 1 to 11
Eb= 10**(Ebdb/10); #symbol energy
SNR=2*Eb/No;
SNRdb=10*np.log10(SNR); #conversion of SNR to dB
BER=np.zeros(len(Ebdb));    #Simulated BER
BERa=np.zeros(len(Ebdb));   #Analytical BER

for i in range(nblocks):
 bitsi=nr.randint(2,size=blocklength);  #Random bits 0 and 1
 bitsq=nr.randint(2,size=blocklength);
 qpsk=(2*bitsi-1)+1j*(2*bitsq-1); #QPSK generation

n=nr.normal(0,np.sqrt(No/2),blocklength)+1j*nr.normal(0,np.sqrt(No/2),blocklength); #noise
 for j in range(len(Ebdb)):
  txs=np.sqrt(Eb[j])*qpsk;   #Transmitted signal
  rxs=txs+n; #received signal = transmitted + AWGN noise
  Dbiti=(np.real(rxs)>0);
  Dbitq=(np.imag(rxs)>0);
  BER[j]=BER[j]+np.sum(Dbiti!=bitsi)+np.sum(Dbitq!=bitsq);

BER=BER/(2*blocklength*nblocks);
```

```
BERa=1-norm.cdf(np.sqrt(SNR));  #Analytical BER
plt.yscale('Log');  #to take log scale in y-axis
plt.plot(SNRdb,BER,'b')
plt.plot(SNRdb,BERa,'g--^') #green colour dashed line with triangle
plt.grid(1,which='both')
plt.suptitle('BER FOR AWGN')
plt.xlabel('SNR(db)')
plt.ylabel('BER')
plt.legend(["Simulated BER","Analytical BER"],loc ="lower left")
```

Output for AWGN channel:



Output in spyder

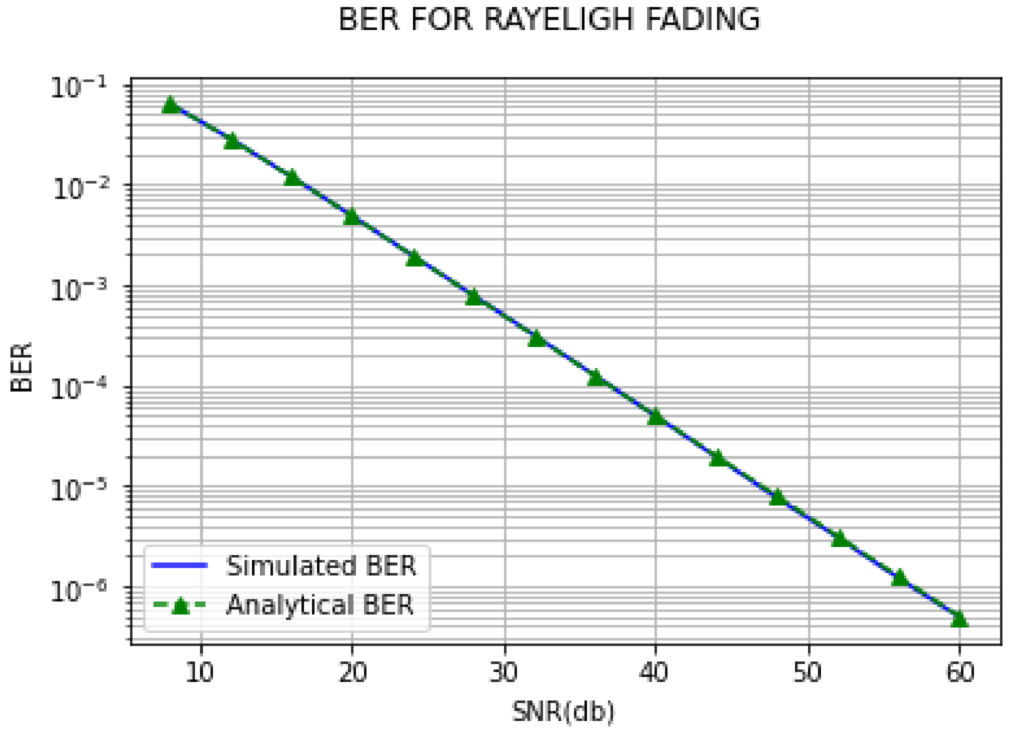


Code for Rayleigh fading channel:

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

blocklength=10000;
nblocks=10000;
No=1;
Ebdb=np.arange(5,61,4);
Eb= 10**(Ebdb/10);
SNR=2*Eb/No;
SNRdb=10*np.log10(SNR);
BER=np.zeros(len(Ebdb));    #Simulated BER initialization
BERa=np.zeros(len(Ebdb));   #Analytical BER initialization
for blk in range(nblocks):
 bitsi=nr.randint(2,size=blocklength);  #Random values 0,1
 bitsq=nr.randint(2,size=blocklength);
 qpsk=(2*bitsi-1)+1j*(2*bitsq-1);   #QPSK generation

n=nr.normal(0,np.sqrt(No/2),blocklength)+1j*nr.normal(0,np.sqrt(No/2),blockle
ngth);

h=nr.normal(0,np.sqrt(No/2),blocklength)+1j*nr.normal(0,np.sqrt(No/2),blockle
ngth);
 for k in range(len(Ebdb)):
  t_s=np.sqrt(Eb[k])*qpsk;  #transmitted signal
  r_s=h*t_s+n;  #received signal=Transmitted signal + noise
  eqsy=1/h*r_s;
  Rbiti=(np.real(eqsy)>0);
  Ibitq=(np.imag(eqsy)>0);
  BER[k]=BER[k]+np.sum(Rbiti!=bitsi)+np.sum(Ibitq!=bitsq);

BER=BER/(2*blocklength*nblocks);
BERa=0.5*(1-np.sqrt(SNR/(2+SNR)));  #Average BER
plt.yscale('Log');
plt.plot(SNRdb,BER,'b')
plt.plot(SNRdb,BERa,'g--^')
plt.grid(1,which='both')
```

```
plt.suptitle('BER FOR RAYELIGH FADING')
plt.xlabel('SNR(db)')
plt.ylabel('BER')
plt.legend(["Simulated BER","Analytical BER"],loc ="lower left")
```

Output for Rayleigh fading channel:



Output in spyder:

`