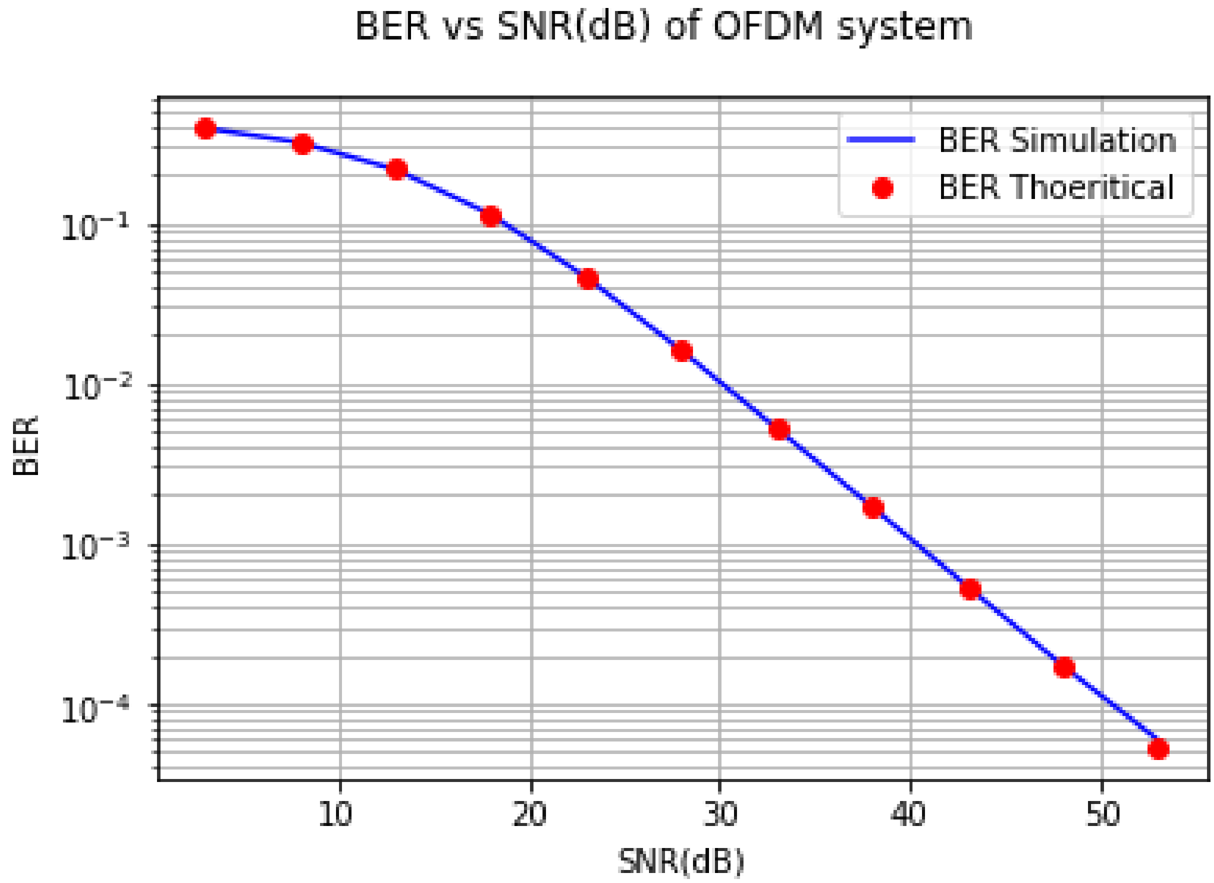
EE 670A Wireless Communications

PYTHON Assignment #3

NAME :S Srikanth Reddy
RollNo :22104092

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 27 18:19:09 2022

@author: SRIKANTH

Please be considerate while executing this code because it takes 3to4 minutes.
"""

import time
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as nr
start = time.time();
No =1; "noise power"
EbdB = np.arange(0,52,5); "energy per bit in dB"
Eb = 10**(EbdB/10);
SNR = 2*Eb/No; "Signal-to-Noise power Ratio calculation"
SNRdB = 10*np.log10(SNR); "conversion to dB"
BER = np.zeros(len(SNRdB));
BERt = np.zeros(len(SNRdB));
nBlocks = 100000; “number of blocks”
N = 64; " number of subcarriers"
L_tilde = 6; “number of cyclic prefix samples”
L = 3; “number of channel taps”
lBlckl = np.zeros(L-1);
for blk in range(nBlocks):
 bitsI = nr.randint(2,size = N);
 bitsQ = nr.randint(2,size = N);
 Sys = (2*bitsI - 1) + 1j*(2*bitsQ - 1);
"Rayleigh fading channel coefficient with unit average power"
h = nr.normal(0, np.sqrt(1/2), L) + 1j * nr.normal(0, np.sqrt(1/2), L);
 h_pad = np.pad(h, (0,N-L),'constant'); “zero-padded channel taps”
"ZMSCG Noise with power No"
noise = nr.normal(0, np.sqrt(No/2), N+L_tilde+L-1) + nr.normal(0,np.sqrt(No/2), N+L_tilde+L-1);
 H = np.fft.fft(h_pad); “N-point FFT”
 for snr in range(len(SNRdB)):
```

```
X = Sys * np.sqrt(Eb[snr]); “symbols with amplitude scaling”
x = np.fft.ifft(X); “N-point IFFT”
CP = x[N-L_tilde:]; “taking the last L_tilde number of samples for cyclic prefix”
x_tx = np.concatenate((CP, x)); “addition of Cyclic Prefix samples in the beginning”

“system model”
y_rx = np.convolve(x_tx,h) + noise;
y_rx1= y_rx[0:N+L_tilde];
y_rx2 = y_rx[N+L_tilde:];
y_rx1[0:L-1] = y_rx1[0:L-1] + IBlckI;
IBlckI = y_rx2;
ofdm_rx = y_rx1[L_tilde:];
Y = np.fft.fft(ofdm_rx);
Y_eq = Y/H; “recovering X(k) with single tap equalizer”
X_decI = (np.real(Y_eq)>0);
X_decQ = (np.imag(Y_eq)>0);
BER[snr] = BER[snr] + np.sum(X_decI!=bitsI) + np.sum(X_decQ!=bitsQ);

BER = BER/N/2/nBlocks;  "simulated BER"
SNR_eff = (L*SNR)/N;
BERt = 0.5*(1-np.sqrt(SNR_eff/(2+SNR_eff))); "theoritical BER"
plt.yscale('log')
plt.plot(SNRdB, BER, 'b-', SNRdB, BERt, 'ro')
plt.grid(1, which = 'both')
plt.suptitle('BER vs SNR(dB) of OFDM system')
plt.xlabel('SNR(dB)')
plt.ylabel('BER')
plt.legend(['BER Simulation','BER Thoeritical'])
end = time.time();
print("exec time:",(end-start),"seconds");
```



Conclusion:

Simulated BER vs SNR(dB) is almost similar to theoretical BER vs SNR(dB) curves for OFDM system.

BER of 10^{-4} occurs approximately at SNR = 50.3dB