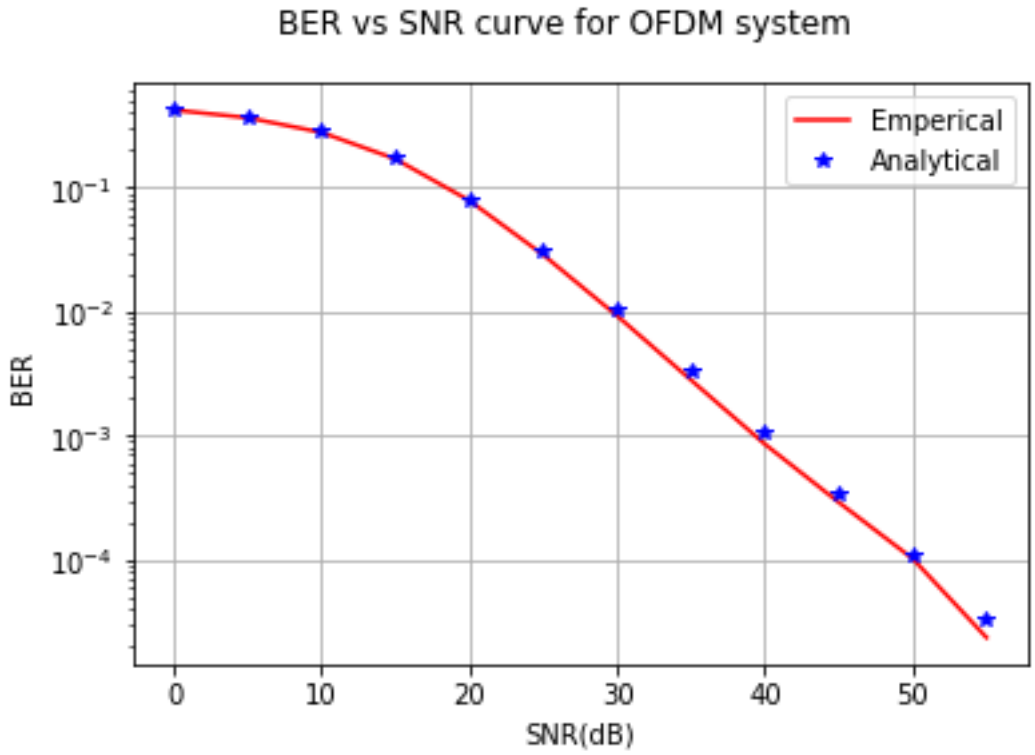
EE670 Python Assignment 3
M. Aravind
21104403

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

No = 1
SNRdB = np.arange(0, 60, 5)
SNR = 10**(SNRdB/10)
Eb = (SNR*No)/2
BER = np.zeros(len(SNRdB))
BERt = np.zeros(len(SNRdB))

nBlocks = 1000
N = 64                  # subcarriers
L_tilda = 10            # cyclic prefix Length
L = 3                   # channel taps

Overlap1 = np.zeros(L-1)
for blk in range(nBlocks):
    bitsI = nr.randint(2, size = N)
    bitsQ = nr.randint(2, size = N)
    QPSKsym = (2*bitsI - 1) + 1j*(2*bitsQ - 1)
    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')
    noise = nr.normal(0, np.sqrt(No/2), N+L_tilda+L-1) + nr.normal(0, np.sqrt(No/2), N+L_tilda+L-1)
    H = np.fft.fft(h_pad)
    for snr in range(len(SNRdB)):
        X = QPSKsym * np.sqrt(Eb[snr])      # frequency domain samples
        x = np.fft.ifft(X)                  #time domain samples
        CP = x[N-L_tilda:]                  #cyclic_prefix
        x_tx = np.concatenate((CP, x))      #cyclic prefix + samples
        y_rx = np.convolve(x_tx,h) + noise
        y_rx1= y_rx[0:N+L_tilda]
        y_rx2 = y_rx[N+L_tilda:]
        y_rx1[0:L-1] = y_rx1[0:L-1] + Overlap1
        Overlap1 = y_rx2                    # save overlap part
        #removal of CP
        ofdm_rx = y_rx1[L_tilda:]
        Y = np.fft.fft(ofdm_rx)
        Y_eq = Y/H
        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
SNR_eff = (L*SNR)/N
BERt = 0.5*(1-np.sqrt(SNR_eff/(2+SNR_eff)))

plt.yscale('log')
plt.plot(SNRdB, BER, 'r-', SNRdB, BERt, 'b*')
plt.grid(1)
plt.suptitle('BER vs SNR curve for OFDM system')
plt.xlabel('SNR(dB)')
plt.ylabel('BER')
plt.legend(['Emperical','Analytical'])

```

Output Plot:



Inference:

For an N Subcarrier OFDM system and L i.i.d. Rayleigh Fading Channel Taps of Unit Gain, we get effective SNR as: $SNR_{eff} = \frac{L}{N} SNR$

And BER for an OFDM System is given by: $BER = \frac{1}{2}\left(1 - \sqrt{\frac{SNR_{eff}}{2+SNR_{eff}}}\right)$

And BER for a SISO-OFDM System is given by: $BER_{SISO} = \frac{1}{2}\left(1 - \sqrt{\frac{\frac{LP}{NN_0}}{2+\frac{LP}{NN_0}}}\right)$

Above is Theoretical BER, and after superposing it with or Analytical BER from Simulated OFDM System in Python, We Found it is almost same with some minor deviations from calculated values.