15ECE312

DIGITAL COMMUNICATION

BER ANALYSIS OF QAM AND CORRESPONDING CONSTELLATIONS

Objective:

To take an arbitrary binary sequence and produces the corresponding M_QAM sequence modulated by both amplitude and phase. To find the Bit error rate of different M_QAM.

Theory :

Bit Error Rate (BER):

- As data is transmitted some of the bits may not be received correctly. The more bits that are incorrect, the more the signal will be affected.

- It's important to know what portion of the bits are in error so you can determine how much margin the system has before failure.

Sent Bits	1101101101
Received Bits	1100101101

error

$$BER = \frac{\#\text{ of Wrong Bits}}{\#\text{ of Total Bits}} = \frac{1}{10} = 0.1$$

- A bit error rate is defined as the rate at which errors occur in a transmission system.

- The bit error probability p, is the expectation value of the bit error ratio. The bit error ratio can be considered as an approximate estimate of the bit error probability. This estimate is accurate for a long-time interval and a high number of bit errors.

- **BER is has no unit, expressed as a percentage.**

- BER is normally displayed in Scientific Notation. The more negative the exponent, the better the BER.
- Better than 1.0E-6 is needed after the FEC for the system to operate.

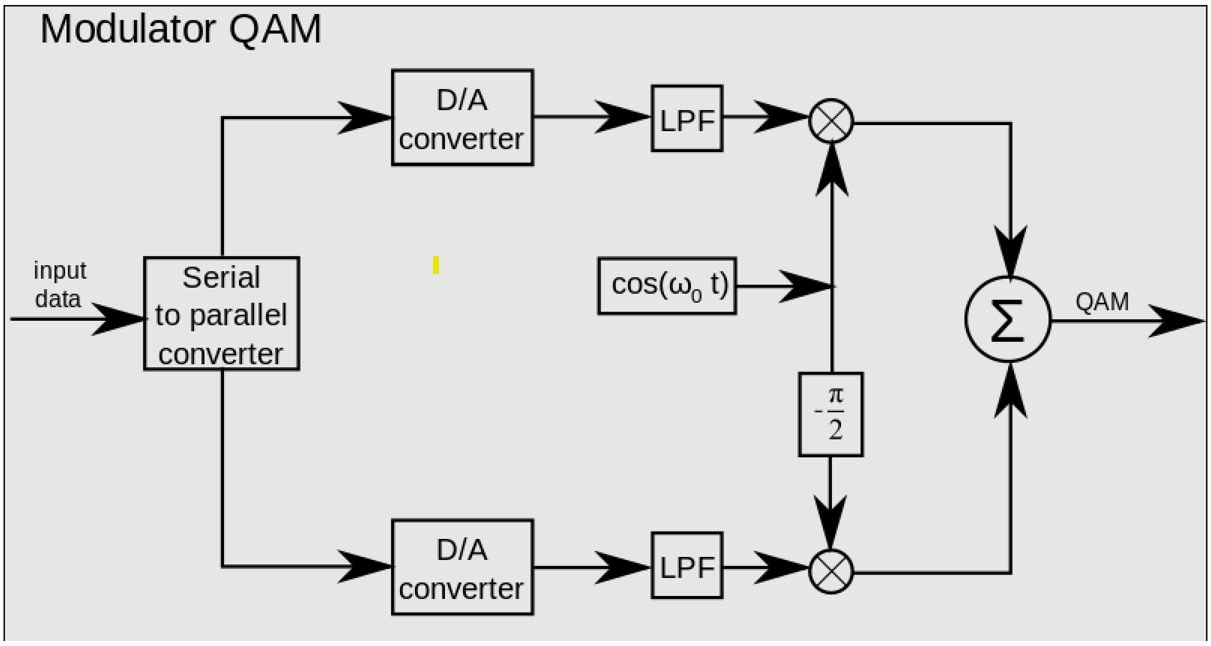
Factors affecting Bit Error Rate :

- ✓ Interference
- ✓ Increase transmitter power
- ✓ Lower order modulation
- ✓ Reduce bandwidth

QUADRATURE AMPLITUDE MODULATION

Quadrature Amplitude Modulation or QAM is a form of modulation which is widely used for modulating data signals onto a carrier used for radio communications. It is widely used because it offers advantages over other forms of data modulation such as PSK, although many forms of data modulation operate alongside each other.

Quadrature Amplitude Modulation, QAM is a signal in which two carriers shifted in phase by 90 degrees are modulated and the resultant output consists of both amplitude and phase variations. In view of the fact that both amplitude and phase variations are present it may also be considered as a mixture of amplitude and phase.



Geometric Representation of QAM Signals

The geometric signal representation of the signals in terms of two-dimensional signal vectors of the form $s_m = \left(\sqrt{\mathcal{E}_s}A_{mc}, \sqrt{\mathcal{E}_s}A_{ms}\right), \quad m = 1, 2, \ldots, M,$

and the orthogonal basis functions are

$$\psi_1(t) = \sqrt{\frac{1}{\mathscr{E}_s}} g_T(t) \cos 2\pi f_c t, \quad \psi_2(t) = -\sqrt{\frac{1}{\mathscr{E}_s}} g_T(t) \sin 2\pi f_c t.$$

For M = 4 rectangular QAM and M = 4 PSK are identical signal constellations.

The average transmitted energy for those signal constellations is simply the sum of the average energies in each of the quadrature carriers, the average energy per symbol is given as

$$\mathscr{E}_{av} = \frac{1}{M} \sum_{i=1}^{M} \| s_i \|^2 .$$

Probability of error for QAM:

We begin with QAM signal sets that have M = 4 points. The first is a four-phase modulated signal and the second is a QAM signal with two amplitude levels, labelled as sqrt(E1) and sqrt(E2) and four phases. Because the probability of error is dominated by the minimum distance between pairs of signal points. we impose the condition that dmin = 2sqrt(Es)for both signal constellations.

For the two-amplitude four-phase QAM, we place the points on circles of radii sqrt(E1) and sqrt(E2) Thus, with the constraint that dmin = 2sqrt(E), the average energy is

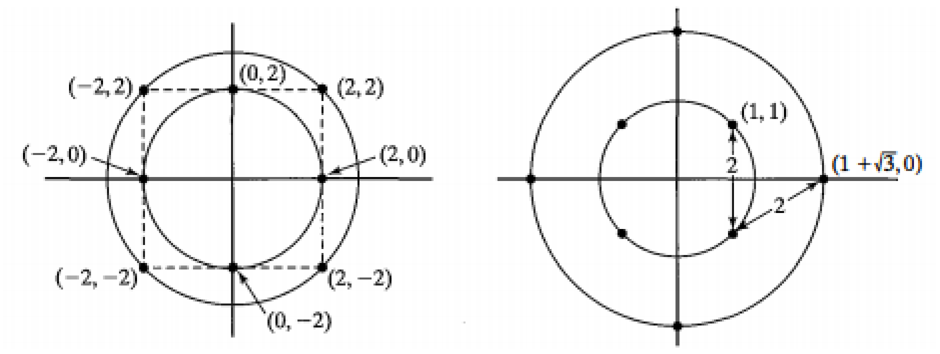
$$\mathscr{E}_{\text{av}} = \frac{1}{4}\left[2(3)\mathscr{E}_s + 2\mathscr{E}_s\right] = 2\mathscr{E}_s,$$

which is the same average energy as the M = 4-phase signal constellation. Hence, for all practical purposes, the error-rate performance of the two signal sets is the same. In other words, the two-amplitude QAM signal set has no advantage over M = 4-phase modulation. Next, consider M = 8 QAM. In this case, there are many possible signal constellations. The coordinates (Amc, Ams) for each signal point, normalized by sqrt(Es), are given in the figure. Assuming that the signal points are equally probable, the average transmitted signal energy is

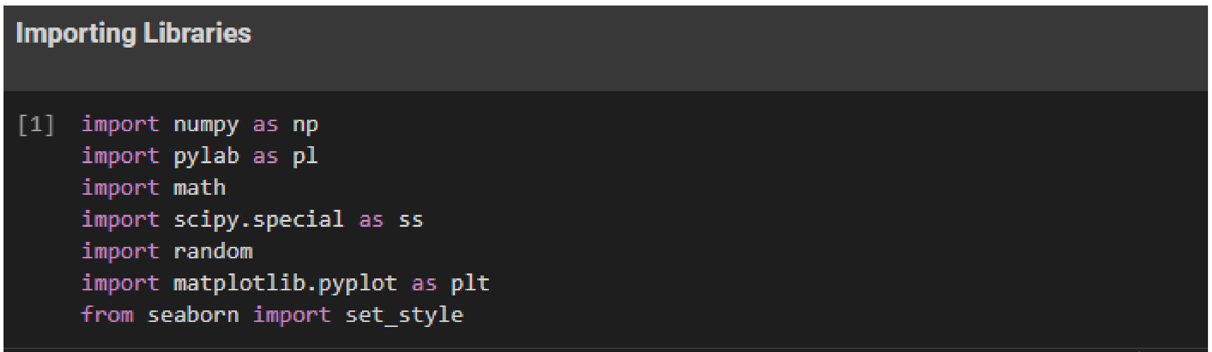
$$\mathscr{E}_{\text{av}} = \frac{\mathscr{E}_s}{M} \sum_{m=1}^{M} (a_{mc}^2 + a_{ms}^2),$$

where (amc, ams) are the normalized coordinates of the signal points.

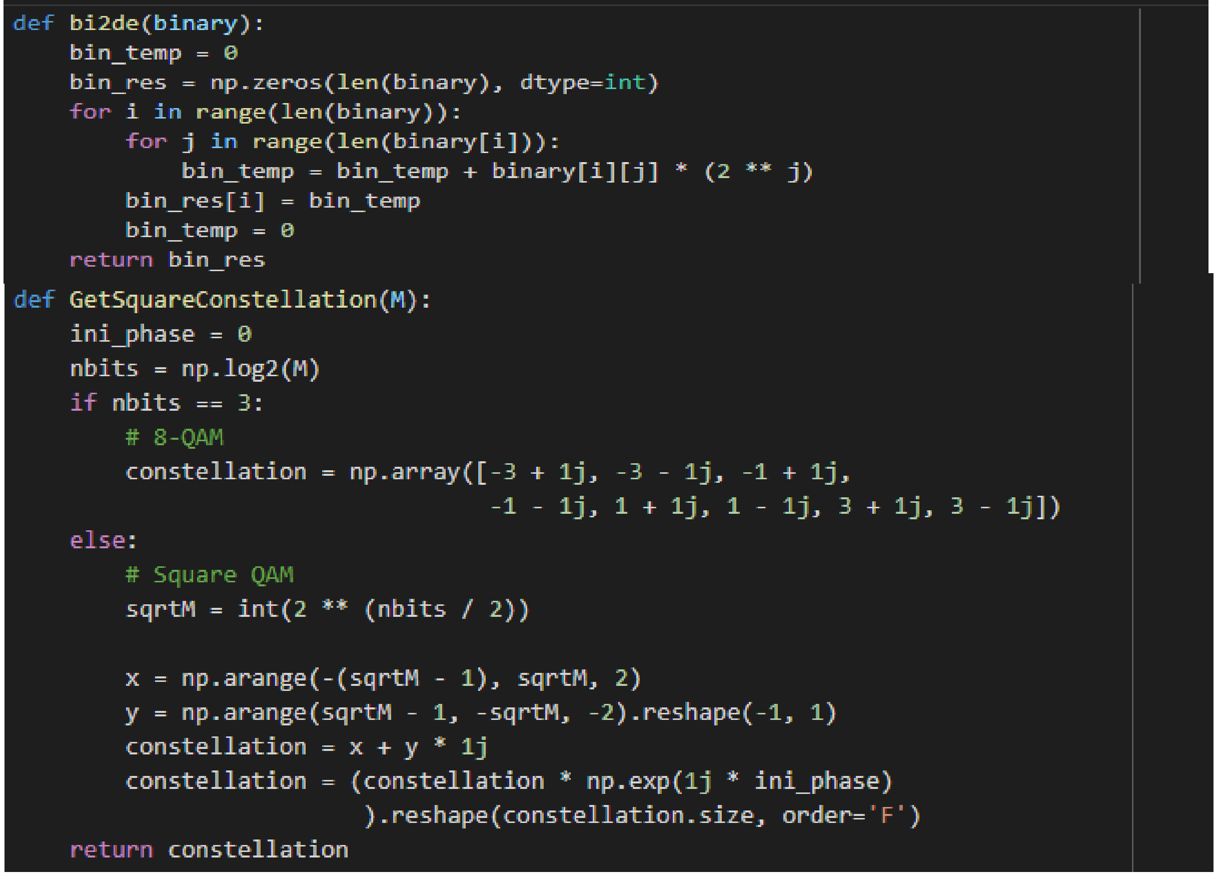
$$\begin{aligned} P_M &\le 1 - \left[1 - 2Q\left(\sqrt{\frac{3\mathscr{E}_{\text{av}}}{(M-1)N_0}}\right)\right]^2 \\ &\le 4Q\left(\sqrt{\frac{3k\mathscr{E}_{\text{bav}}}{(M-1)N_0}}\right) \end{aligned}$$

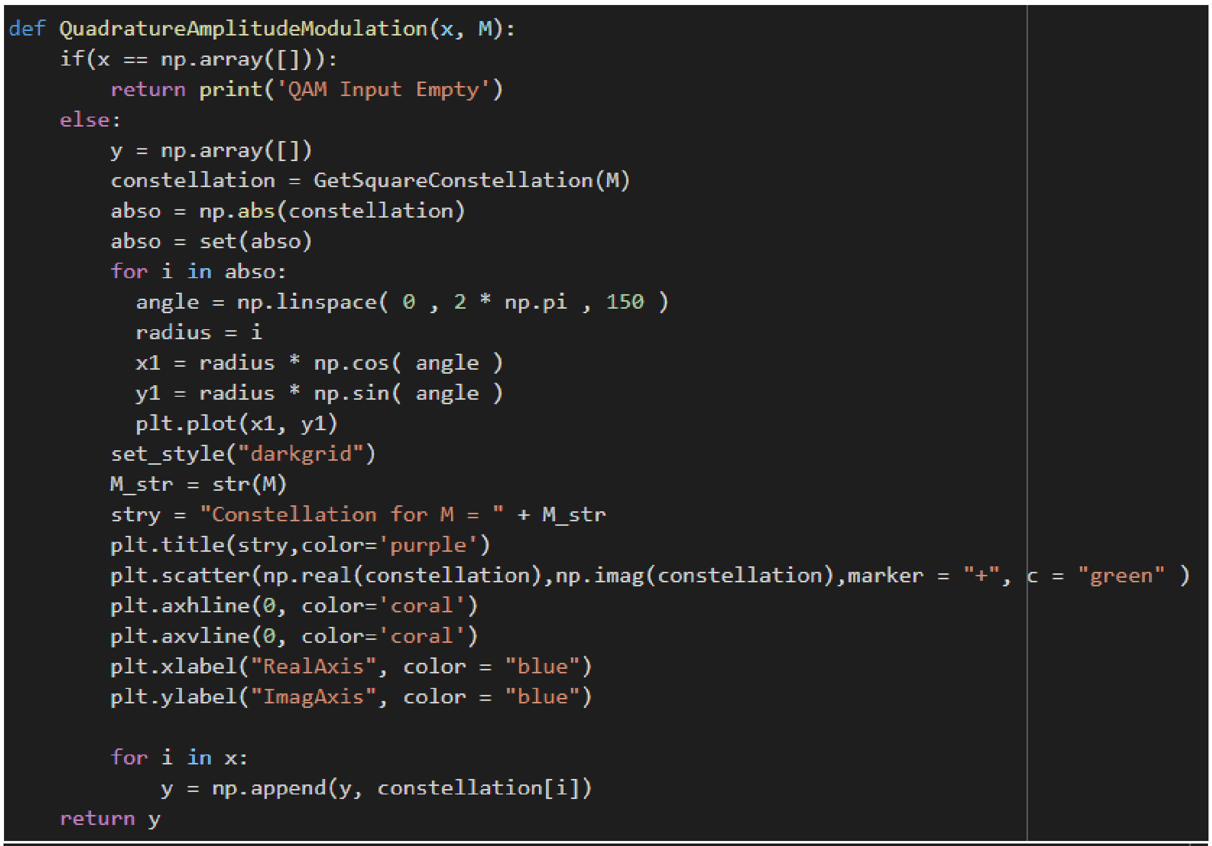


CODE:

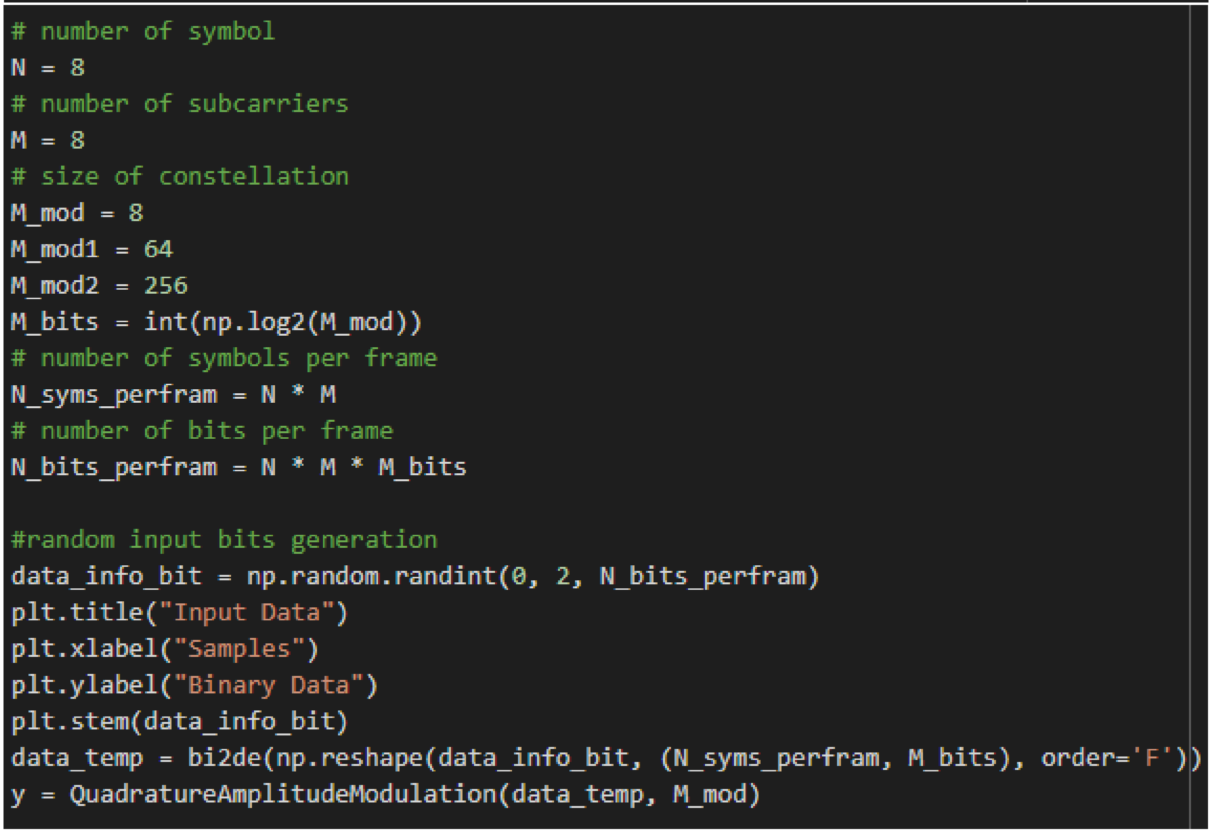

Importing Libraries

```
[1]  import numpy as np
     import pylab as pl
     import math
     import scipy.special as ss
     import random
     import matplotlib.pyplot as plt
     from seaborn import set_style
```

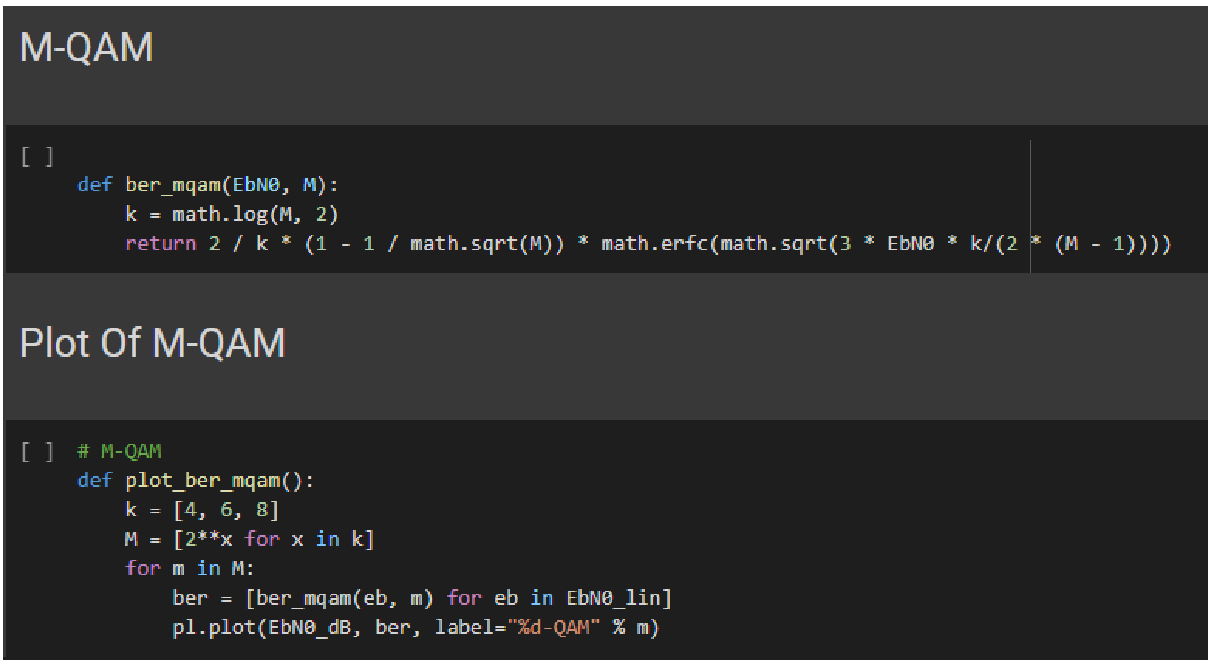

```
def bi2de(binary):
    bin_temp = 0
    bin_res = np.zeros(len(binary), dtype=int)
    for i in range(len(binary)):
        for j in range(len(binary[i])):
            bin_temp = bin_temp + binary[i][j] * (2 ** j)
        bin_res[i] = bin_temp
        bin_temp = 0
    return bin_res
def GetSquareConstellation(M):
    ini_phase = 0
    nbits = np.log2(M)
    if nbits == 3:
        # 8-QAM
        constellation = np.array([-3 + 1j, -3 - 1j, -1 + 1j,
                                  -1 - 1j, 1 + 1j, 1 - 1j, 3 + 1j, 3 - 1j])
    else:
        # Square QAM
        sqrtM = int(2 ** (nbits / 2))

        x = np.arange(-(sqrtM - 1), sqrtM, 2)
        y = np.arange(sqrtM - 1, -sqrtM, -2).reshape(-1, 1)
        constellation = x + y * 1j
        constellation = (constellation * np.exp(1j * ini_phase)
                     ).reshape(constellation.size, order='F')
    return constellation
```



```python
def QuadratureAmplitudeModulation(x, M):
    if(x == np.array([])):
        return print('QAM Input Empty')
    else:
        y = np.array([])
        constellation = GetSquareConstellation(M)
        abso = np.abs(constellation)
        abso = set(abso)
        for i in abso:
          angle = np.linspace( 0 , 2 * np.pi , 150 )
          radius = i
          x1 = radius * np.cos( angle )
          y1 = radius * np.sin( angle )
          plt.plot(x1, y1)
        set_style("darkgrid")
        M_str = str(M)
        stry = "Constellation for M = " + M_str
        plt.title(stry,color='purple')
        plt.scatter(np.real(constellation),np.imag(constellation),marker = "+", c = "green" )
        plt.axhline(0, color='coral')
        plt.axvline(0, color='coral')
        plt.xlabel("RealAxis", color = "blue")
        plt.ylabel("ImagAxis", color = "blue")

        for i in x:
            y = np.append(y, constellation[i])
    return y
```



```python
# number of symbol
N = 8
# number of subcarriers
M = 8
# size of constellation
M_mod = 8
M_mod1 = 64
M_mod2 = 256
M_bits = int(np.log2(M_mod))
# number of symbols per frame
N_syms_perfram = N * M
# number of bits per frame
N_bits_perfram = N * M * M_bits

#random input bits generation
data_info_bit = np.random.randint(0, 2, N_bits_perfram)
plt.title("Input Data")
plt.xlabel("Samples")
plt.ylabel("Binary Data")
plt.stem(data_info_bit)
data_temp = bi2de(np.reshape(data_info_bit, (N_syms_perfram, M_bits), order='F'))
y = QuadratureAmplitudeModulation(data_temp, M_mod)
```

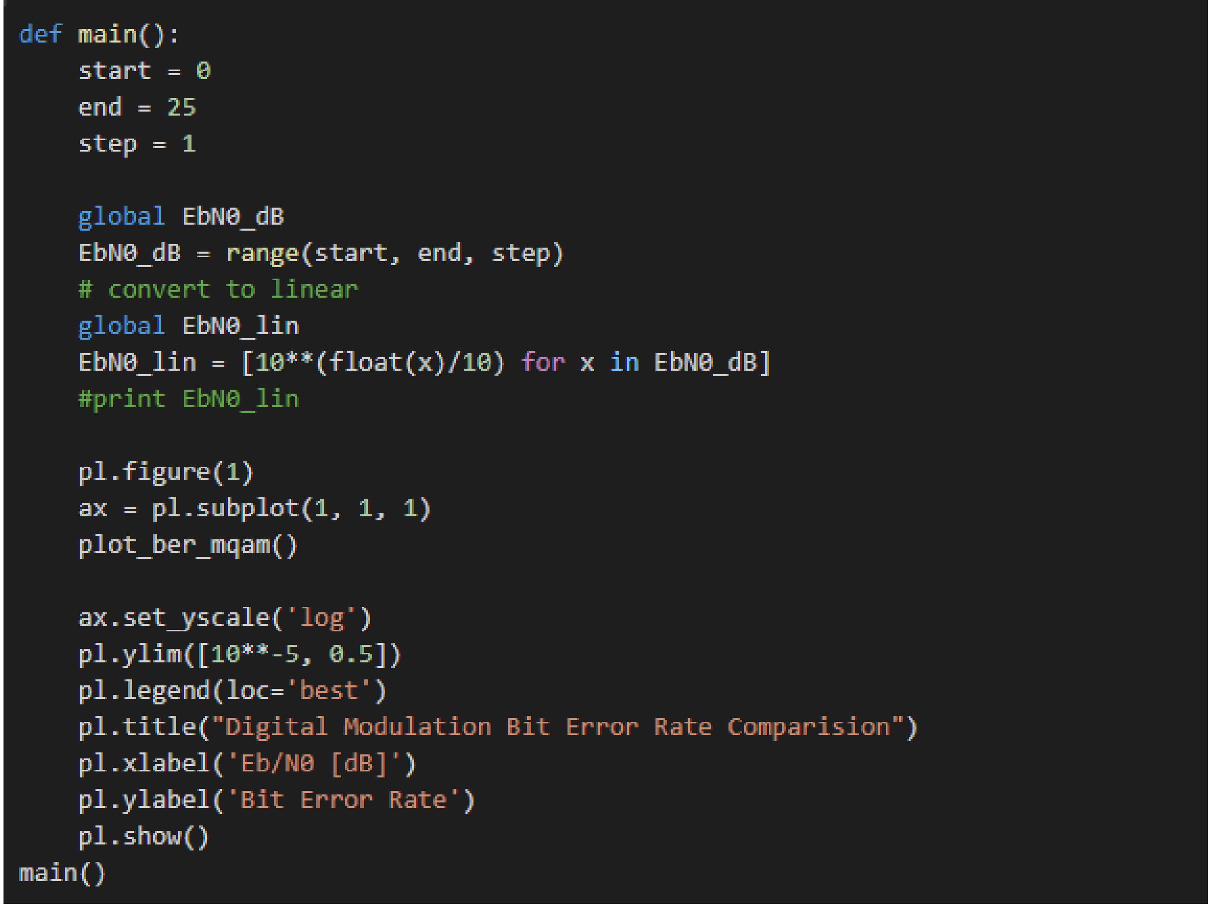


M-QAM

```
[ ]
    def ber_mqam(EbN0, M):
        k = math.log(M, 2)
        return 2 / k * (1 - 1 / math.sqrt(M)) * math.erfc(math.sqrt(3 * EbN0 * k/(2 * (M - 1))))
```

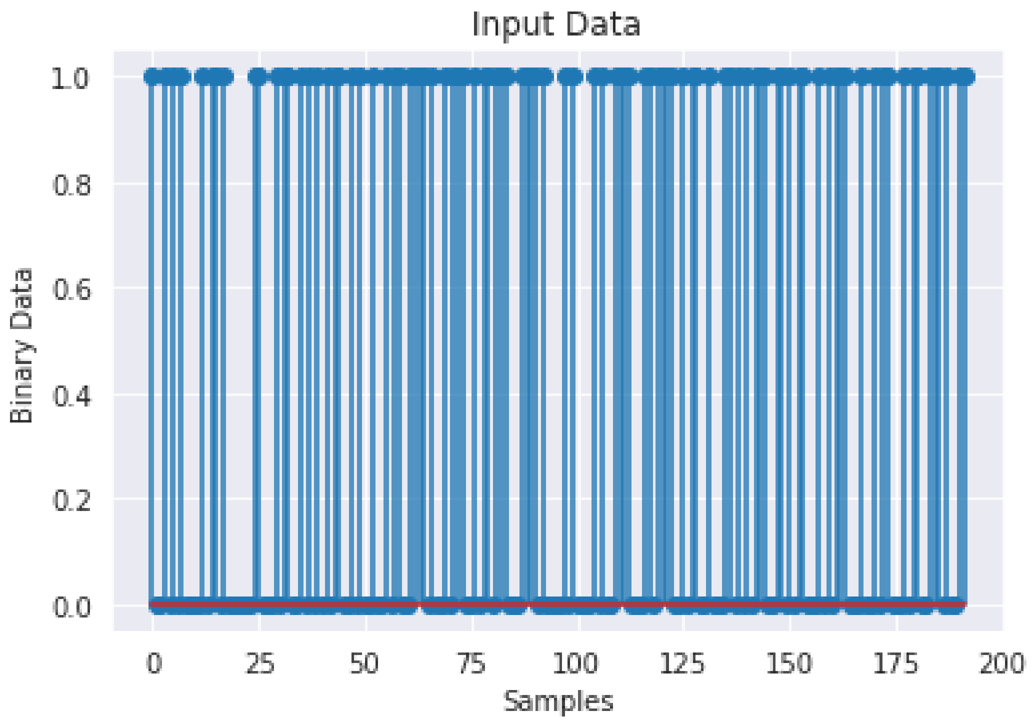
Plot Of M-QAM

```
[ ] # M-QAM
    def plot_ber_mqam():
        k = [4, 6, 8]
        M = [2**x for x in k]
        for m in M:
            ber = [ber_mqam(eb, m) for eb in EbN0_lin]
            pl.plot(EbN0_dB, ber, label="%d-QAM" % m)
```

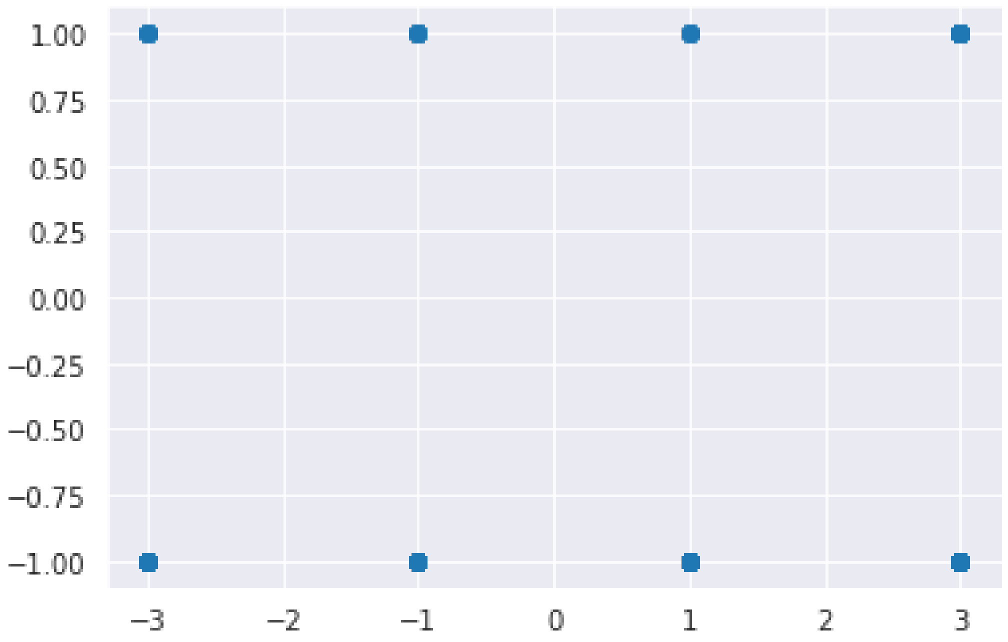


```
def main():
    start = 0
    end = 25
    step = 1

    global EbN0_dB
    EbN0_dB = range(start, end, step)
    # convert to linear
    global EbN0_lin
    EbN0_lin = [10**(float(x)/10) for x in EbN0_dB]
    #print EbN0_lin

    pl.figure(1)
    ax = pl.subplot(1, 1, 1)
    plot_ber_mqam()

    ax.set_yscale('log')
    pl.ylim([10**-5, 0.5])
    pl.legend(loc='best')
    pl.title("Digital Modulation Bit Error Rate Comparision")
    pl.xlabel('Eb/N0 [dB]')
    pl.ylabel('Bit Error Rate')
    pl.show()
main()
```

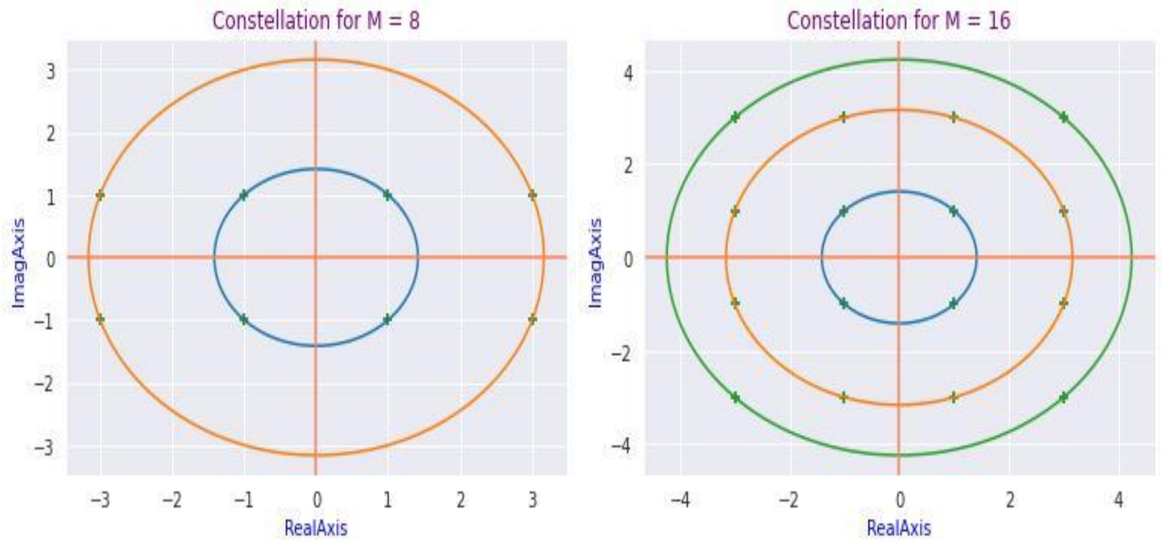
INPUT DATA



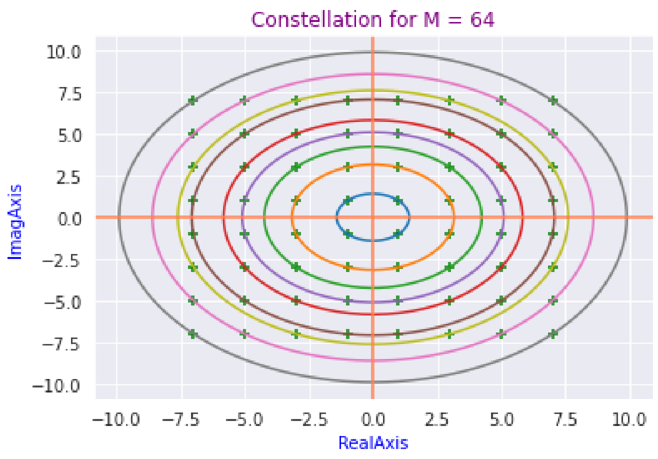
Corresponding 4_QAM plot:



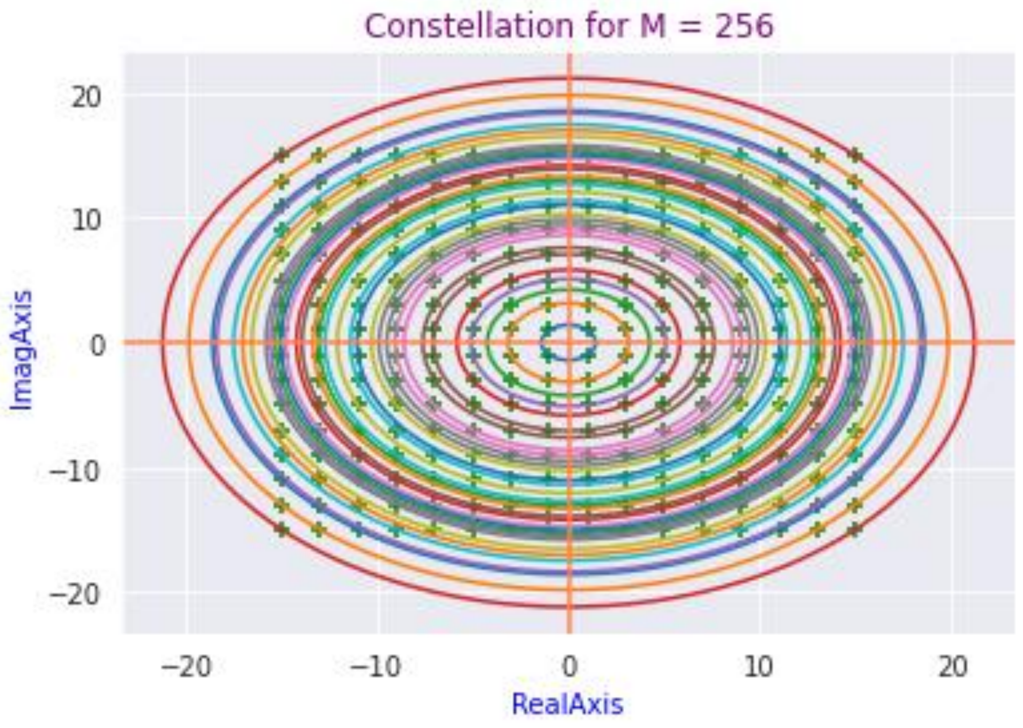
RESULT:



Constellation for M = 8
Constellation for M = 16
ImagAxis
RealAxis



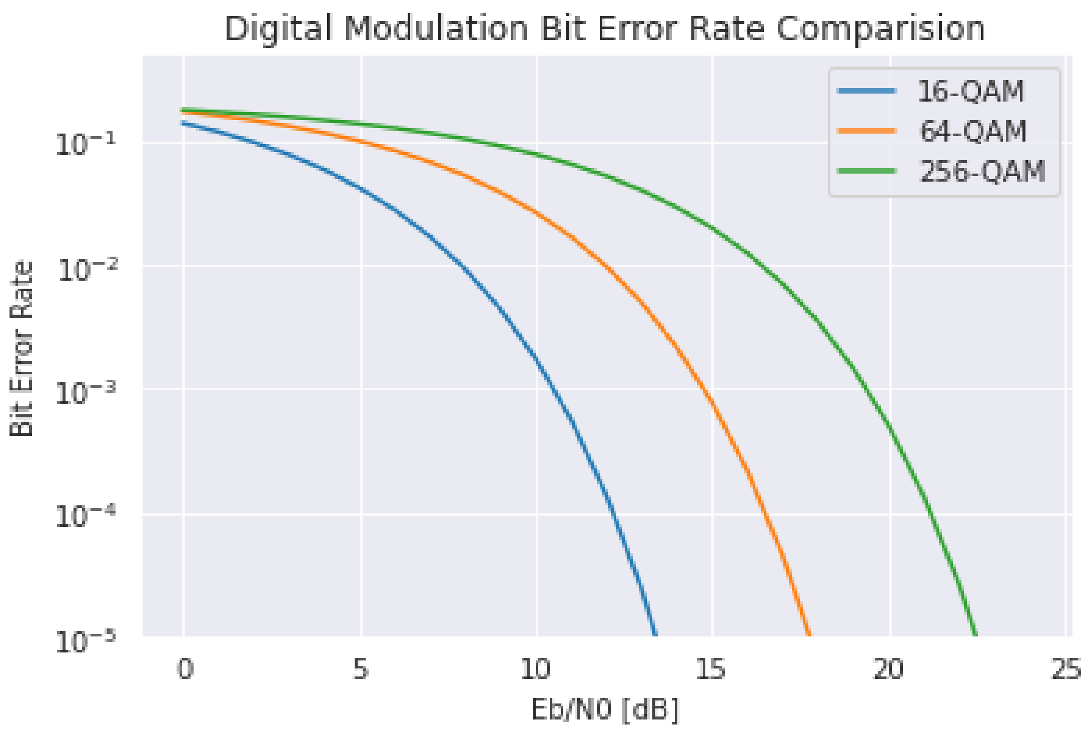
Constellation for M = 64
ImagAxis
RealAxis



Constellation for M = 256
ImagAxis
RealAxis

<u>Combined probability of Bit Error Rate</u> :

The probability of a symbol error plotted below is a function of the average SNR/bit



Inference :

Constellations corresponding to different M_QAM s are obtained by using amplitude and phase component of a carrier signal .As discussed in the theory the probability of error is inversely proportional to M and directly to avg(Eavg/N0) The signal is sliced and then can be inscribed into phase and amplitude components of the carrier which is what called as Quadrature , and the corresponding demodulator setup is required to demodulate it and combine amplitude and phase parts for a complete signal.

<u>References</u>:

- ✓ **FUNDAMENTALS OF COMMUNICATION SYSTEMS Second Edition John G. ProakisMasoudSalehi*
- ✓ *https://16qam-system.readthedocs.io/en/latest/index.html*
- ✓ *https://www.youtube.com/watch?v=vZQu4YDw*