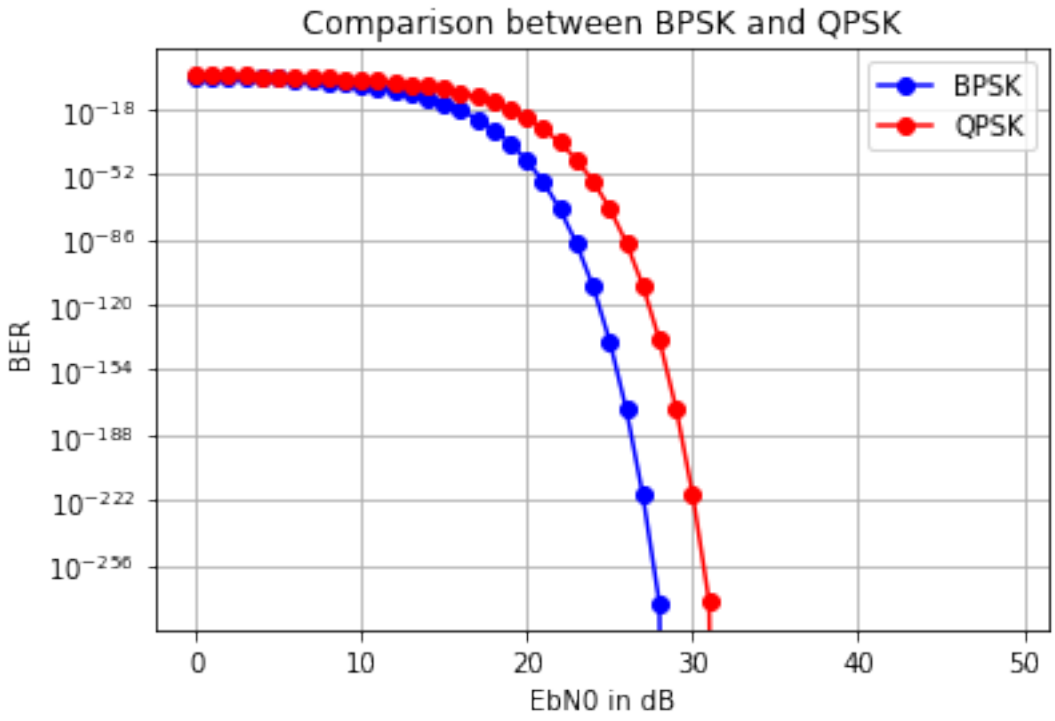
BER of BPSK and QPSK

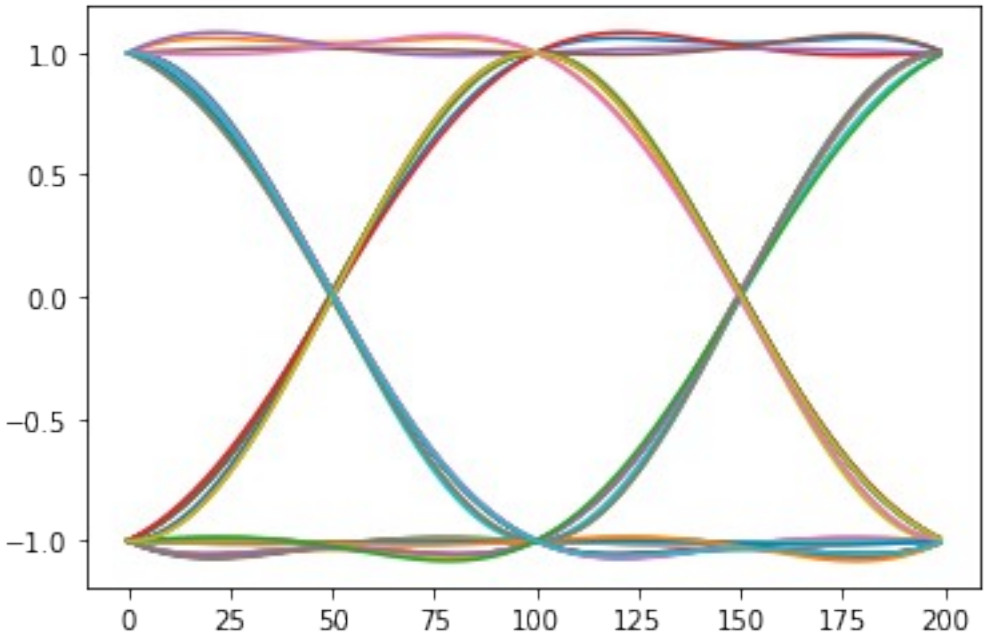
```
#Importing necessary libraries

import numpy as np
import matplotlib.pyplot as plt
from scipy import special as sp

EbN0dB_list = np.arange(0, 50)
EbN0 = [10**(EbN0dB/10) for EbN0dB in EbN0dB_list]
EbN0 = np.array(EbN0)
BER_BPSK = (1/2) * sp.erfc(np.sqrt(EbN0))
BER_QPSK =  sp.erfc(np.sqrt(EbN0/2))

plt.plot(EbN0dB_list, BER_BPSK, "bo-", label = "BPSK")
plt.plot(EbN0dB_list, BER_QPSK, "ro-", label = "QPSK")
plt.xscale("linear")
plt.yscale("log")
plt.title("Comparison between BPSK and QPSK")
plt.xlabel("EbN0 in dB")
plt.ylabel("BER")
plt.legend()
plt.grid()
plt.show()
```



Eye Diagram

```
#Importing necessary libraries

import numpy as np
import matplotlib.pyplot as plt

#Defining Raised Cosine functions

def raisedCosineFilter(time, beta, T):
    h = []
    for t in time:
        if (t == T/(2 * beta)) or (t == -T/(2 * beta)):
            val = (np.pi / (2 * T)) * np.sinc(1 / (2 * beta))
        else:
            val = (1/T) * np.sinc(t/T) * (np.cos(np.pi * beta * t/T) /
(1 - (2 * beta * t/T)**2))

        h.append(val)
    return np.array(h)

def get_RC_filter(beta, Ts):
    return lambda t : raisedCosineFilter(t, beta, Ts)

Ts = 1
Fs = 100

g = get_RC_filter(beta=0.95, Ts=Ts)
d = 2 * (np.random.randn(100) > 0.5) - 1
t = np.arange(0, (len(d))*Ts, 1/Fs)
x = sum(d[k] * g(t - k*Ts) for k in range(len(d)))

for i in range(int(len(x)/(2 * Fs * Ts))):
    plt.plot(x[i * Fs * Ts : (i * Ts + 2*Ts)*Fs])

print("Eye diagram")
plt.show()
```

```
Eye diagram
```



QPSK Simpler Code

```
#Importing necessary libraries

import numpy as np
import matplotlib.pyplot as plt
import scipy
from scipy import special

fm = 1
fc = 100
t = np.linspace(0,1,fc)

x0 = np.sin(2*np.pi*fm*t)
x1 = np.sin(2*(np.pi+np.pi/4)*fm*t)
x2 = np.sin(2*(np.pi+np.pi/2)*fm*t)
x3 = np.sin(2*(np.pi+3*np.pi/4)*fm*t)

inp = np.random.randint(0,4,10)

n = []

for i in inp:
    if i == 0:
         n = np.concatenate((n,x0))
    elif i ==1 :
        n = np.concatenate((n,x1))
    elif i == 2:
        n = np.concatenate((n,x2))
    elif i == 3:
```

```
        n = np.concatenate((n,x3))

bdl = np.arange(0,50)
b = [10**(bd/10) for bd in bdl]
b = np.array(b)
BER = special.erfc(np.sqrt(b/2))



plt.subplot(3,1,1)
plt.plot(inp)
plt.subplot(3,1,2)
plt.plot(n)
plt.subplot(3,1,3)
plt.plot(BER)
plt.yscale('log')
plt.show()
```