EXPERIMENT NUMBER - 7

EXPERIMENT NAME - PROBABILITY OF ERROR (PAM)

DATE - 01/12/2022, THURSDAY

* AIM:

To find probability of errors in Pulse Amplitude Modulation (PAM) and plot the graphs.

* SOFTWARE REQUIRED:

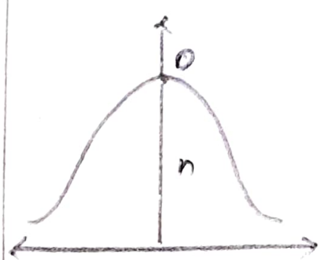
1. Anaconda3 2021.11 (Python 3.9.7 64-bit)
2. Spyder 5.1.5 Integrated Development Environment (IDE)

* THEORY:

Pulse Amplitude Modulation (PAM) are also called as antipodal signals and they can be represented using one basis function. When you transmit either '0' or '1', it is called as binary PAM.

Bit	Transmitted	$r(t)$	$y(t)$ [demod]
0	$\sqrt{E_b}\,\psi(t)$	$\sqrt{E_b}\,\psi(t) + n(t)$	$\sqrt{E_b} + n_1$
1	$-\sqrt{E_b}\,\psi(t)$	$-\sqrt{E_b}\,\psi(t) + n(t)$	$-\sqrt{E_b} + n_1$

It follows Gaussian distribution and it is random since, it is Gaussian for noise too, $E(n) = 0$.



Mean, median, mode lies at same point and it is symmetric.

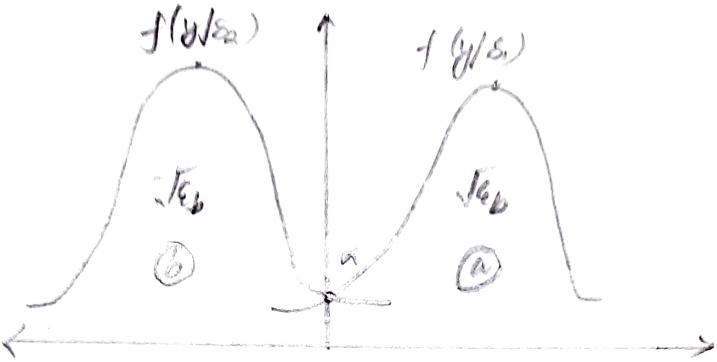
$$\text{PDF of Gaussian} = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-(y-\sqrt{E_b})^2 / 2N_0/2}$$

[Transmitted bit = 0
For PAM, $\mu = \sqrt{E_b}$]

If $\mu = -\sqrt{\varepsilon_b}$.

$$PDF = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-(-\sqrt{\varepsilon_b}))^2 / 2N_0/2}$$

[Transmitted bit = 1]



Region a: $f(y/s_1)$ dominating
Region b: $f(y/s_2)$ dominating

Possible Errors -

→ Value that belongs to $f(y/s_2)$ in $f(y/s_1)$
→ Value that belongs to $f(y/s_1)$ in $f(y/s_2)$

$P(\alpha)$ = Tx S_1, favouring S_2 + Tx S_2, favouring S_1

$$P(\alpha) = P(S_1) \int_{-\infty}^{\alpha} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-(-\sqrt{\varepsilon_b}))^2 / 2N_0/2}$$

$$+$$

$$P(S_2) \int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-\sqrt{\varepsilon_b})^2 / 2N_0/2}$$

Differentiating w.r.t α, we get -

$$P(S_1) f\left(\frac{\alpha}{S_1}\right) - P(S_2) f\left(\frac{\alpha}{S_2}\right) = 0$$

$$\frac{P(S_1)}{P(S_2)} = \frac{f(\alpha/S_2)}{f(\alpha/S_1)}$$

Solving by substituting the values, we get -

$$\frac{P(S_2)}{P(S_1)} = e^{4\alpha\sqrt{\frac{\varepsilon_b}{N_0}}}$$

$$\Rightarrow 4\alpha \sqrt{\frac{\mathcal{E}_b}{N_0}} = \ln \frac{P(S_2)}{P(S_1)}$$

$$\Rightarrow \boxed{\alpha = \frac{4}{4\sqrt{\mathcal{E}_b}} \ln \frac{P(S_2)}{P(S_1)}}$$

$P(S_2) > P(S_1) \rightarrow \alpha(+ve)$

$P(S_1) > P(S_2) \rightarrow \alpha(-ve)$

$P(S_1) = P(S_2) \rightarrow \alpha = 0$

Substituting $\alpha = 0$, we get -

$$P_2(0) = P(S_1)\int_{-\infty}^{0} f(y/S_1)\,dy + P(S_2)\int_{0}^{\infty} f(y/S_2)\,dy$$

$$= \frac{1}{2} \times 2 \int_{0}^{\infty} f(y/S_1)\,dy$$

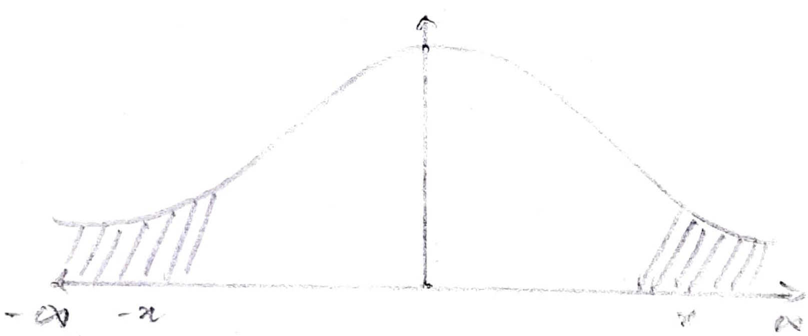
$$= \frac{1}{\sqrt{2\pi N_0/2}}\, e^{-(y-\sqrt{\mathcal{E}_b})^2/N_0}$$

$$\boxed{x = (y - \sqrt{\mathcal{E}_b}) / \sqrt{N_0/2}}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\sqrt{2\mathcal{E}_b/N_0}} e^{-x^2/2}\,dx = Q\left(\sqrt{\frac{2\mathcal{E}_b}{N_0}}\right)$$

$$\boxed{Q(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-x^2/2}\,dx = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} e^{-x^2/2}\,dx}$$

$$\boxed{\therefore Q\sqrt{\frac{2\mathcal{E}_b}{N_0}} = P(\mathcal{E})}$$



* PYTHON CODE :

import matplotlib.pyplot as plt # Provides an implicit way of plotting

import numpy as np # Support for large, multi-dimensional arrays and matrices

import math # Provides access to the mathematical functions defined by the C standard

PTO

```
t = []
xt = np.random.normal(0, 1, 10000)
for i range(len(xt)):
    if i>0.5:
        t.append(i)
    elif i<=0.5:
        t.append(0)

y=[]
for i in t:
    if i==0:
        y.append(1)
    else:
        y.append(-1)

x = np.arange(0, len(y), 1); plt.step(x,y)
plt.xlim(0, 100); plt.show()

l = p = []
for num in range(1, 11):

    final = error = xt = []
    sigma = (10* 10 ** (-num/10)/2)
    print("\n" + str(num) + " - Sigma Value: " + str(sigma))

    noise = np.random.normal(0, sigma, 10000)
    count, bins, ignored = plt.hist(noise, 1000, density = True)
    plt.plot(bins, np.ones_like(bins), linewidth = 2, color = 'r')
    plt.show()

    xt = y + noise
    plt.plot(xt); plt.title("Signal with Noise")
    plt.xlim(0, 100); plt.show()
```

```
for i in range(len(rt)):
    if rt[i] > 0:
        final.append(1)
    else:
        final.append(-1)

error_count = 0
for i in range(len(rt)):
    if final[i] != y[i]:
        error_count += 1
print("    Error count: " + str(error_count));
error.append(error_count)

plt.stem(final)
plt.xlabel("Sequence"); plt.ylabel("Amplitude")
plt.title("Demodulated Signal")
plt.xlim(0, 100) ; plt.show()

for i in range(len(error)):
    e.append(error[i]/10000)
    p.append(math.erfc(np.sqrt(num) * np.sqrt(2))) #
Returns the complementary error function of a number

plt.semilogx(p); plt.semilogy(e)
plt.xlabel("SNR in dB"); plt.ylabel("Probability of error (PAM)")
plt.show()
```

* **METHODOLOGY:**

We know that,

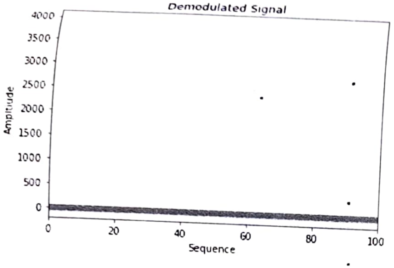
$$(SNR)_{dB} = \log_{10}\left(\frac{S}{N}\right)$$

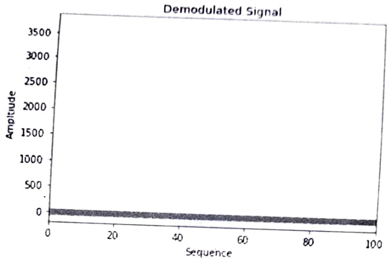
$$\Rightarrow \frac{SNR}{10} = \log_{10}\left(\frac{S}{N}\right)$$

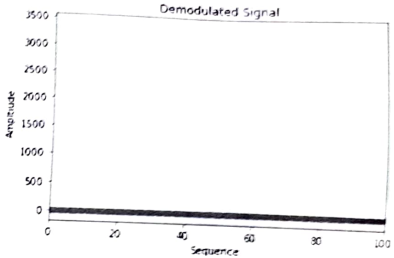
$$\Rightarrow \frac{N}{S} = 10^{\left(\frac{SNR}{10}\right)}$$

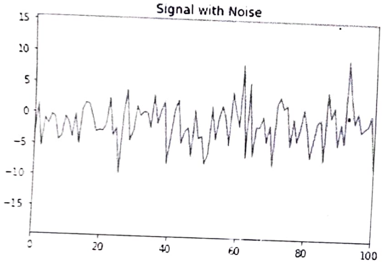
$$\Rightarrow sigma^2 = S \times 10^{\left(\frac{-SNR}{10}\right)}$$

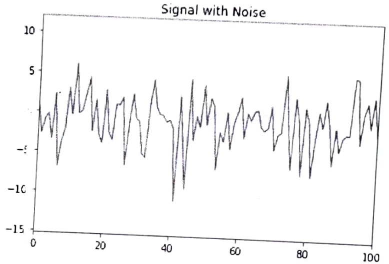
$$\Rightarrow \boxed{sigma = S^{1/2} \times 10^{\left(\frac{-SNR}{20}\right)}}$$

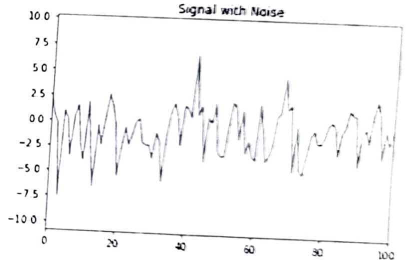

Demodulated Signal
Amplitude
Sequence

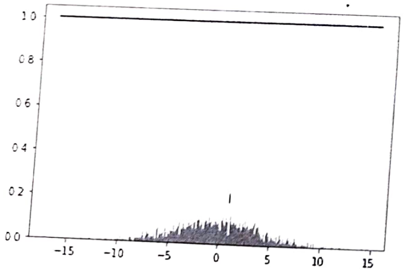

Demodulated Signal
Amplitude
Sequence

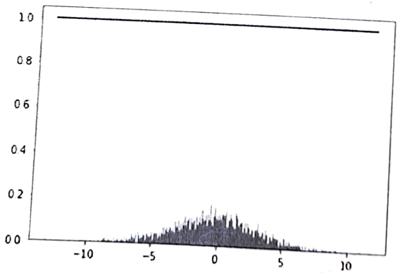

Demodulated Signal
Amplitude
Sequence

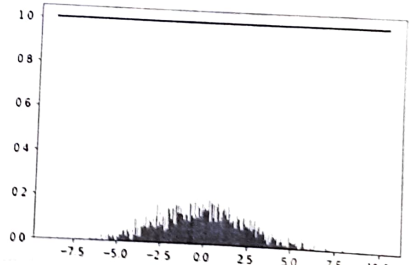

Signal with Noise

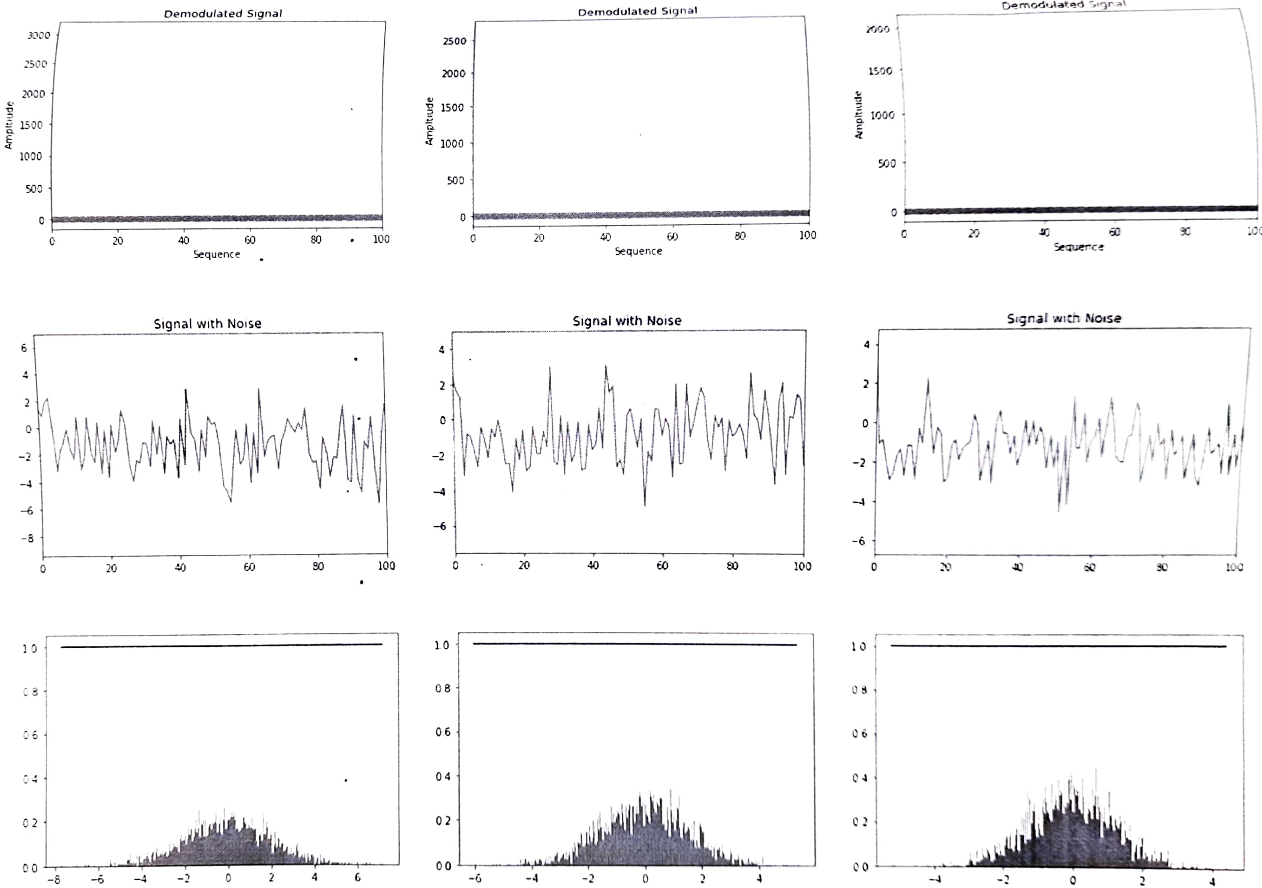

Signal with Noise

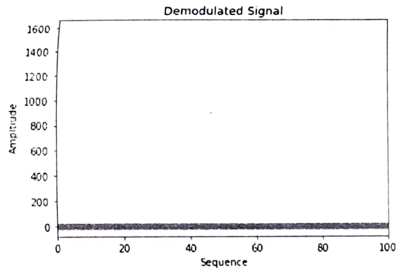

Signal with Noise

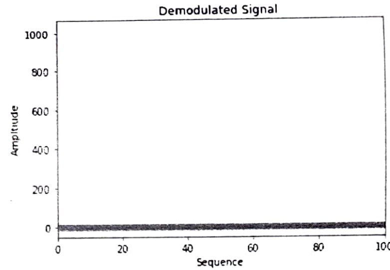


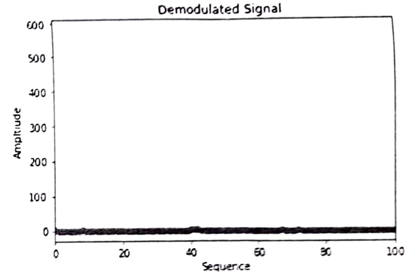


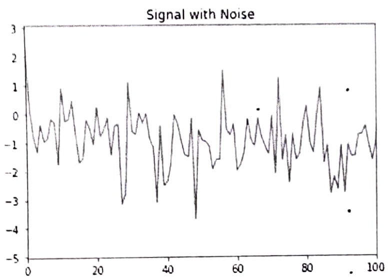


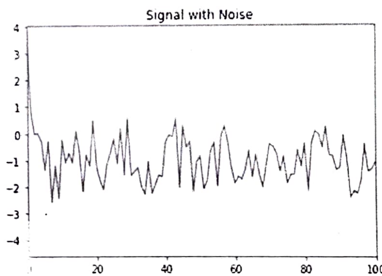

Demodulated Signal
Amplitude
Sequence
Demodulated Signal
Amplitude
Sequence
Demodulated Signal
Amplitude
Sequence
Signal with Noise
Signal with Noise
Signal with Noise

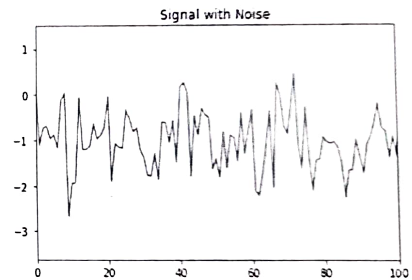

Demodulated Signal
Amplitude
Sequence

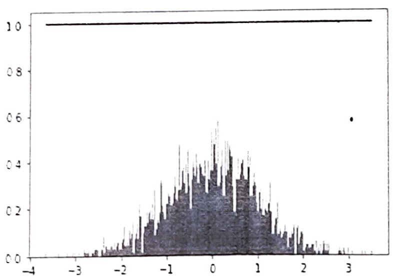

Demodulated Signal
Amplitude
Sequence

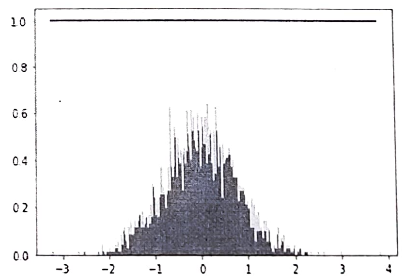

Demodulated Signal
Amplitude
Sequence

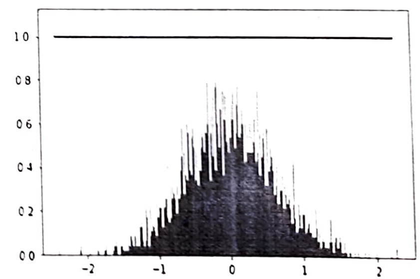

Signal with Noise

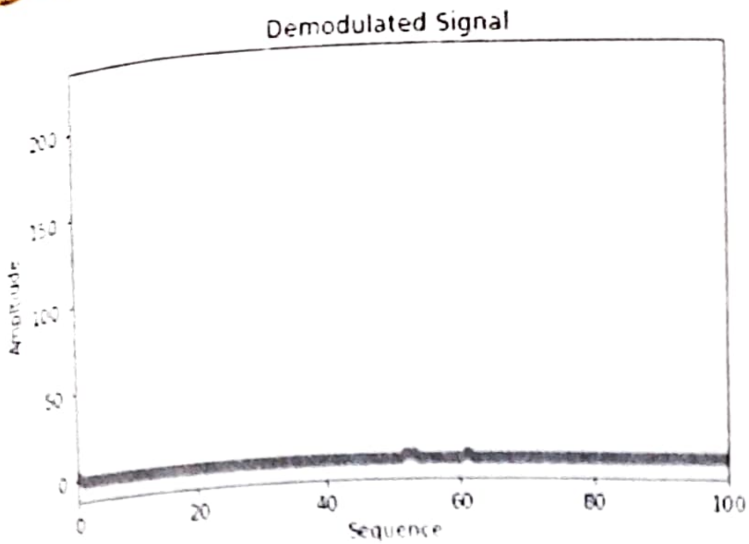

Signal with Noise

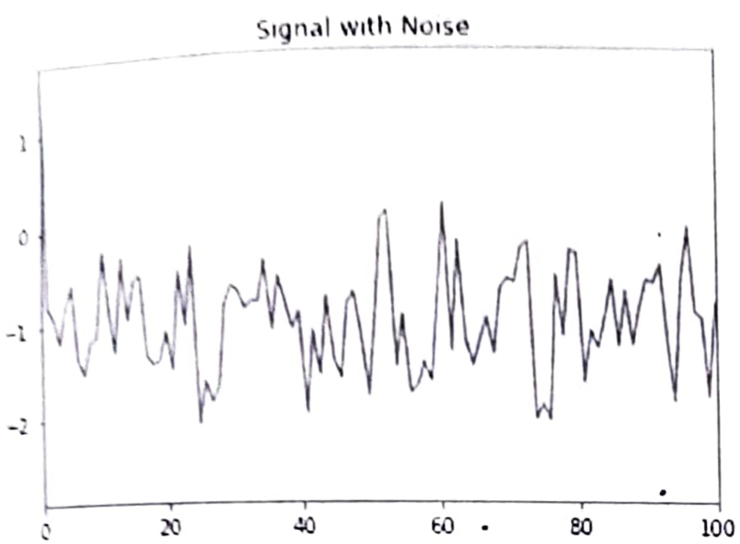

Signal with Noise

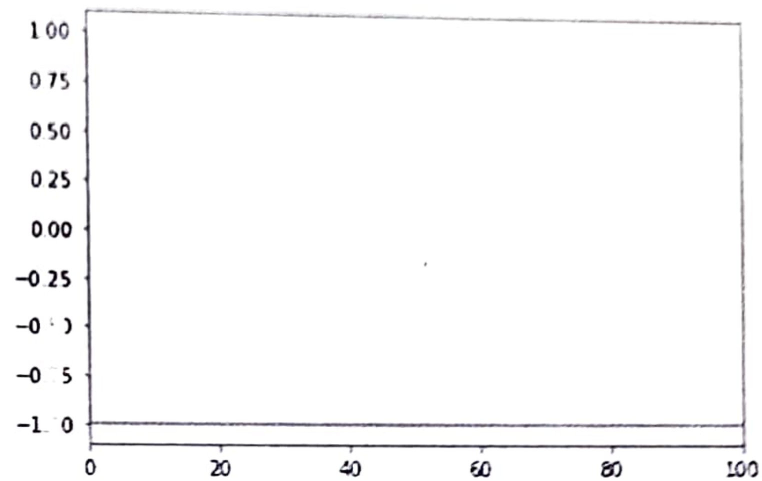


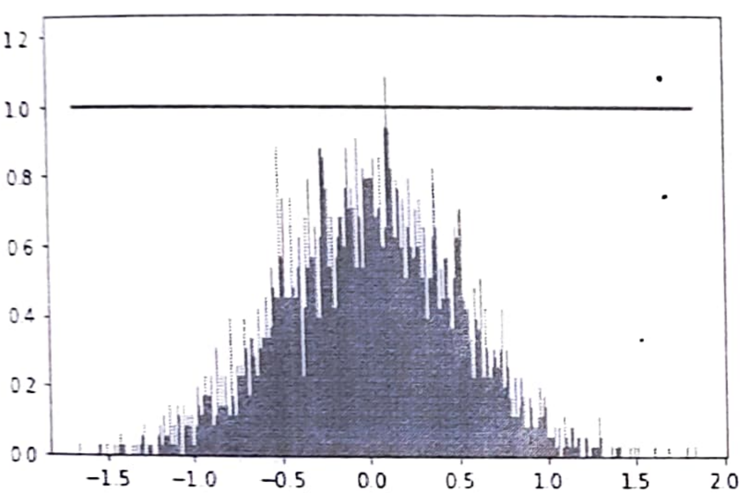


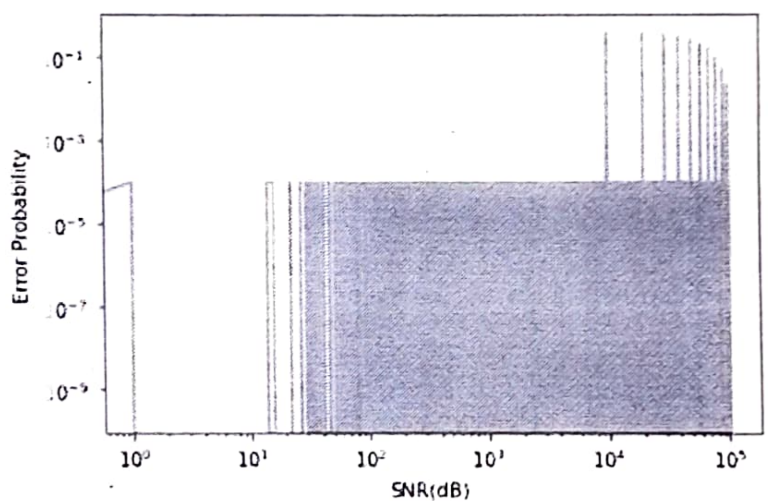



Demodulated Signal
Amplitude
Sequence


Signal with Noise






Error Probability
SNR(dB)

* OUTPUTS:

1- Sigma value : 3.9716411736214075
Error Count : 3982

2- Sigma Value : 3.1547867224009662
Error Count : 3140

3- Sigma Value : 2.5059361681363361
Error Count : 3495

4- Sigma Value : 1.9905358521467466
Error Count : 3180

5- Sigma Value : 1.5811388300841598
Error Count : 2557

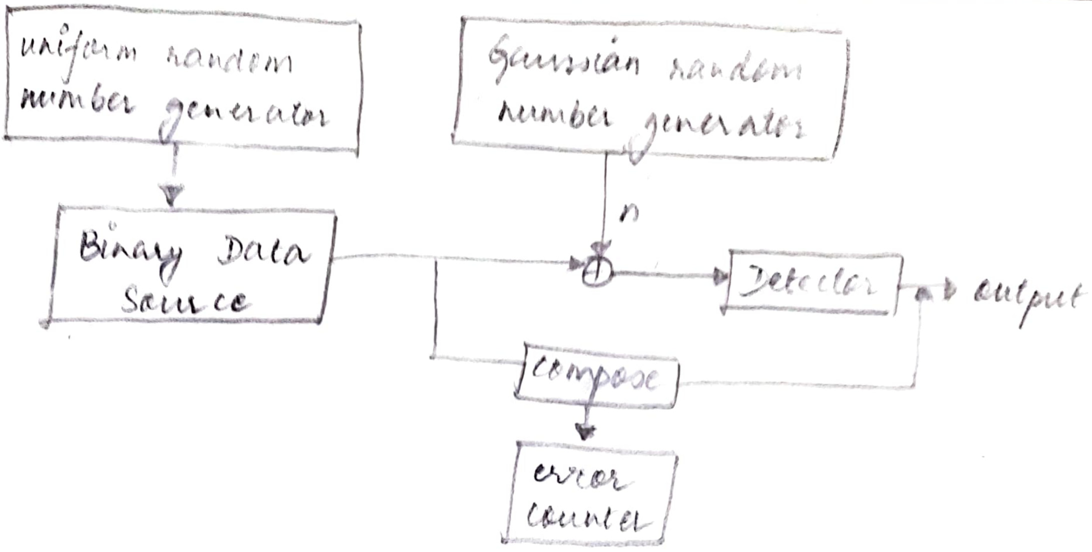
6- Sigma Value : 1.2559432157547979
Error Count : 2137

7- Sigma Value : 0.99763115748443997
Error Count : 1590

8- Sigma Value : 0.7924465962305567
Error Count : 1053

9- Sigma Value : 0.6294627058970836
Error Count : 515

10- Sigma Value : 0.5
Error Count : 226



* **RESULT:**

Thus, we have plotted the probability of error by varying SNR values and observed that with higher SNR probability of error leads to zero. All the simulation results were verified successfully.