Sinhgad Technical Educational Society's
SINHGAD COLLEGE OF ENGINEERING
VADGAON PUNE-41
Department of Electronics & Telecommunication

Sinhgad Institutes

**Experiment No. _** 9

**Subject: - Mobile Computing**

**Name of the Student:** Manas P. Patil **Roll No.** 404B041

**Date:** 8/4/24 **Marks & Signature: -**

**Subject Teacher**

E:

ot BER for AWGN Channel using python

udy AWGN channel and to measure Bit Error Rate.

TWARE & HARDWARE REQUIREMENTS:

: Unix or windows 7/8/10,

ocessor: i3/i5/i7

ftware: Python ( Jupyter Notebook) or java

**Theory:**

AWGN stands for Additive White Gaussian Noise, which is a common model used to describe the channel in digital communication systems. An AWGN channel is a type of noise channel that adds random Gaussian noise to the transmitted signal. The term "additive" refers to the fact that the noise is added to the signal, "white" means that the noise has a constant power spectral density across all frequencies, and "Gaussian" means that the noise follows a Gaussian distribution with zero mean and a given variance.

The AWGN channel is a useful model for many real-world communication scenarios, since noise in communication channels often has these characteristics. For example, thermal noise in electronic components and devices can be modeled as AWGN, as can radio frequency interference and atmospheric noise in wireless communication systems. The Gaussian distribution is also a natural model for many types of noise, since it arises from the central limit theorem, which states that the sum of many independent random variables tends to follow a Gaussian distribution.

The effects of AWGN on a digital communication system can be analyzed using statistical methods such as signal-to-noise ratio (SNR) analysis and bit error rate (BER) analysis. The SNR is defined as the ratio of the signal power to the noise power, and is a measure of the quality of the received signal. The BER is the fraction of received bits that are in error compared to the transmitted bits, and is a measure of the accuracy of the communication system. Both the SNR and BER depend on the power of the transmitted signal, the bandwidth of the channel, and the characteristics of the noise, such as its variance.

In summary, the AWGN channel is a widely used model for noise in digital communication systems, and its effects on the system can be analyzed using statistical methods. Understanding the AWGN channel is important for designing and optimizing communication systems for different noise environments.

**Procedure:**

1. Set up a communication system consisting of a transmitter, a channel, and a receiver. The transmitter should generate a sequence of bits and modulate them using BPSK modulation. The channel should add noise to the signal according to the AWGN model. The receiver should demodulate the signal and decode the received symbols back to bits.

2. Set the signal-to-noise ratio (SNR) to a desired value, e.g., 10 dB. You can use a signal generator and a noise generator to simulate the transmitted signal and the noise, respectively. Alternatively, you can use a hardware transmitter and receiver and set the SNR by adjusting the power levels of the transmitted and received signals.

3. Transmit a large number of bits, e.g., one million. Measure the bit error rate (BER) by comparing the received bits to the transmitted bits and counting the number of errors. The BER can be expressed as the ratio of the number of errors to the total number of bits transmitted.

4. Repeat step 3 for different values of SNR, e.g., 5 dB, 0 dB, -5 dB, etc. Plot the BER as a function of SNR. You should observe that the BER decreases as the SNR increases, and eventually approaches zero as the SNR becomes very large.

5. Compare your results to the theoretical BER curve for BPSK modulation in the presence of AWGN, which is given by the formula:

**BER = 0.5*erfc(sqrt(SNR))**

where erfc is the complementary error function and SNR is the signal-to-noise ratio in linear scale (not dB).

You can use this formula to calculate the expected BER for different values of SNR and compare it to your measured BER. You should observe good agreement between theory and experiment for high SNR values, but some deviation for low SNR values due to the statistical nature of the noise. you can gain a better understanding of the effects of noise on digital communication systems and how the BER depends on the SNR. This can be useful for designing and optimizing communication systems for different noise environments.

Python program that implements the experiment to measure the bit error rate in the presence of an AWGN model:

```
import numpy as np
import matplotlib.pyplot as plt

# Set the parameters
n_bits = 1000000     # Number of bits to be transmitted
SNRdBs = np.arange(-10, 11, 1)  # SNR range in dB
SNRs = 10**(SNRdBs/10) # SNR range in linear scale

# Generate random bits
bits = np.random.randint(0, 2, n_bits)

# Loop over SNR values
BERs = []
for SNR in SNRs:
   # BPSK modulation
   symbols = 2*bits - 1

   # Add noise
   noise_power = 1/SNR
   noise = np.sqrt(noise_power)*np.random.randn(n_bits)
```

```
    received = symbols + noise

    # BPSK demodulation
    decoded_bits = (received >= 0).astype(int)

    # Calculate the bit error rate
    BER = np.sum(bits != decoded_bits) / n_bits
    BERs.append(BER)

# Plot the results
plt.semilogy(SNRdBs, BERs)
plt.xlabel('SNR (dB)')
plt.ylabel('Bit Error Rate')
plt.title('Bit Error Rate vs. SNR for BPSK modulation with AWGN')
plt.grid(True)
plt.show()
```

In this program, we first set the number of bits to be transmitted (n_bits) and the SNR range in dB (SNRdBs). We then generate a random sequence of bits using NumPy's randint() function. We loop over the SNR values in the range and for each value, we perform BPSK modulation by mapping each bit to a symbol (-1 or 1). We add noise to the signal according to the AWGN model using NumPy's randn() function. We decode the signal by comparing the received symbols to a threshold of 0, and convert the decoded symbols back to bits. Finally, we calculate the bit error rate by counting the number of bits that are not correctly decoded and dividing by the total number of bits transmitted. We store the BER values in a list BERs.

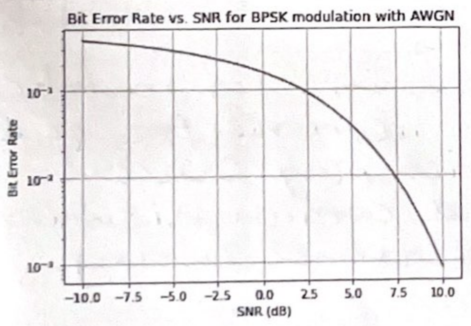
We then plot the BER values as a function of SNR using Matplotlib's semilogy() function to show the results on a logarithmic scale. We label the x-axis as SNR in dB, and the y-axis as Bit Error Rate. We also add a title and gridlines to the plot. The show() function displays the plot.

By running this program, you can observe how the bit error rate decreases as the SNR increases and eventually approaches the theoretical BER curve for BPSK modulation in the presence of AWGN. You can modify the parameters n_bits and SNRdBs to change the number of bits transmitted and the SNR range, respectively, and see how the results change.

**Conclusion**

In this experiment we studied one bit error rate vs SNR for BPSK modulation with AWGN channel

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

# Set the parameters
n_bits = 1000000      # Number of bits to be transmitted
SNRdBs = np.arange(-10, 11, 1)   # SNR range in dB
SNRs = 10**(SNRdBs/10)  # SNR range in linear scale

# Generate random bits
bits = np.random.randint(0, 2, n_bits)

# Loop over SNR values
BERs = []
for SNR in SNRs:
    # BPSK modulation
    symbols = 2*bits - 1

    # Add noise
    noise_power = 1/SNR
    noise = np.sqrt(noise_power)*np.random.randn(n_bits)
    received = symbols + noise

    # BPSK demodulation
    decoded_bits = (received >= 0).astype(int)

    # Calculate the bit error rate
    BER = np.sum(bits != decoded_bits) / n_bits
    BERs.append(BER)

# Plot the results
plt.semilogy(SNRdBs, BERs)
plt.xlabel('SNR (dB)')
plt.ylabel('Bit Error Rate')
plt.title('Bit Error Rate vs. SNR for BPSK modulation with AWGN')
plt.grid(True)
plt.show()
```



[ ]: