BER vs SNR for BPSK and ASK Modulation

Name: Sachin Kumar
Roll No: 220104026
Section: D
Lab No. : 7

October 18, 2024

1 Introduction

In this report, we simulate the Bit Error Rate (BER) of both Binary Phase Shift Keying (BPSK) and Binary Amplitude Shift Keying (ASK) over an Additive White Gaussian Noise (AWGN) channel. We also compute the theoretical Bit Error Probability (Pb) for BPSK and ASK using the Q-function and compare the results.

2 Mathematical Background

The theoretical Bit Error Probability (Pb) for BPSK is given by:

$$P_b^{\text{BPSK}} = Q\left(\sqrt{2 \cdot \frac{E_b}{N_0}}\right)$$

For ASK, the Bit Error Probability is given by:

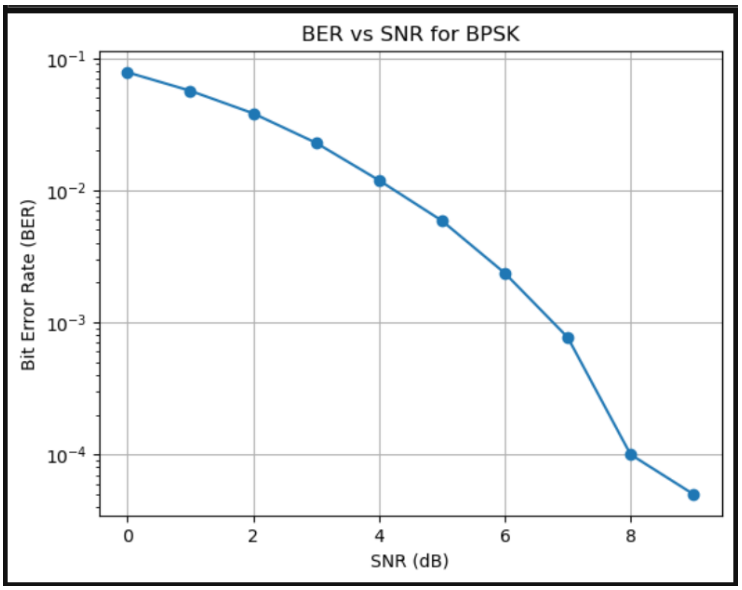
$$P_b^{\text{ASK}} = Q\left(\sqrt{\frac{E_b}{N_0}}\right)$$

where $Q(x)$ is the Q-function:

$$Q(x) = \frac{1}{2}\,\text{erfc}\left(\frac{x}{\sqrt{2}}\right)$$

In both cases, the SNR (in dB) is converted to a linear scale using:

$$\frac{E_b}{N_0} = 10^{\frac{\text{SNR}_{\text{dB}}}{10}}$$



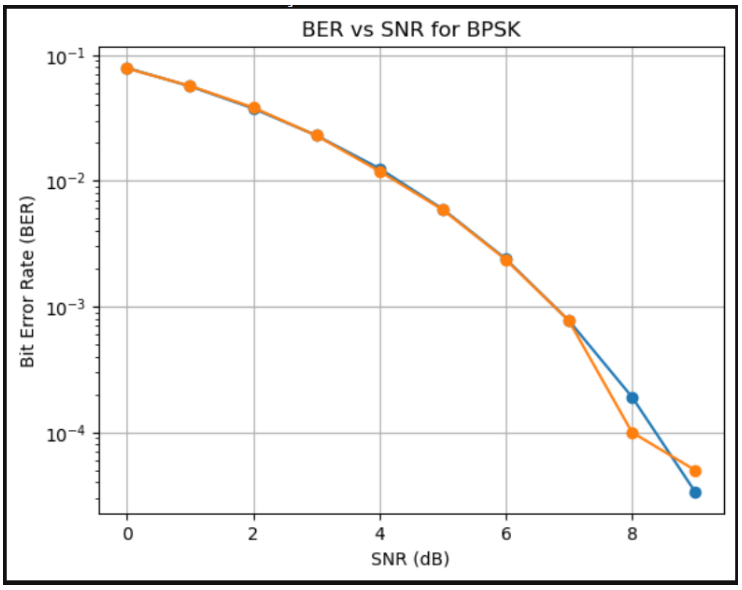
3 Simulated BER for BPSK

The Python code for simulating the BER of a BPSK system is as follows:

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

# Assuming necessary parameters and variables (snr_db, symbols, bits, N, etc.) are defined

for i, snr in enumerate(snr_db):
    snr_linear = 10**(snr / 10)  # Convert SNR from dB to linear scale
    noise_power = 1 / snr_linear  # Calculate noise power
    noise = np.sqrt(noise_power / 2) * np.random.randn(N)  # Generate AWGN noise

    received_signal = symbols + noise  # Received signal (symbols + noise)
    received_bits = received_signal > 0  # Decision based on thresholding

    errors = np.sum(bits != received_bits)  # Count the errors
    ber[i] = errors / N  # Calculate BER

    print(f'SNR = {snr} dB, BER = {ber[i]:e}')  # Print results

# Plotting the simulated BER after the loop
plt.figure()
plt.semilogy(snr_db, ber, 'o-')
```

```
plt.grid(True)
plt.xlabel('SNR (dB)')
plt.ylabel('Bit Error Rate (BER)')
plt.title('BER vs SNR for BPSK')
plt.show()
```

4 BER for BPSK and ASK

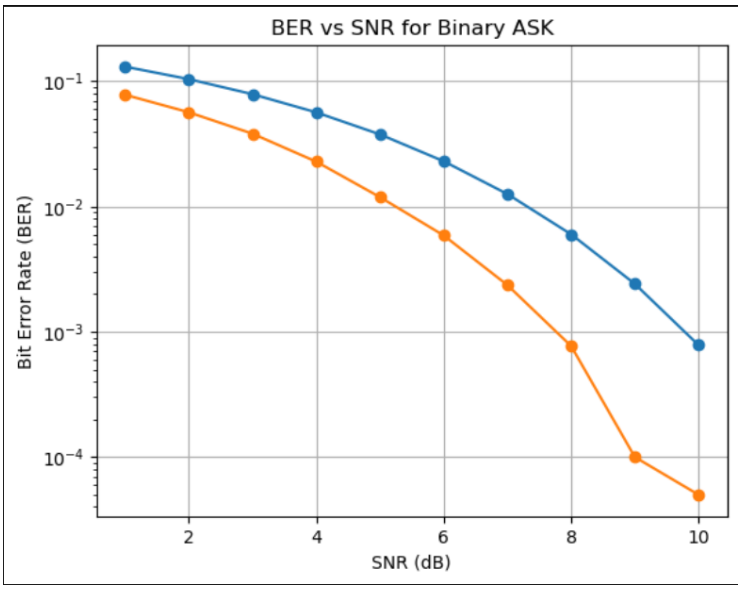



```
import numpy as np
import scipy.special as sp

# Define the Q-function
def Q_function(x):
    return 0.5 * sp.erfc(x / np.sqrt(2))
Eb_N0 = 10**(snr_db / 10)
Pb_bpsk = Q_function(np.sqrt(2 * Eb_N0))

print(f"Bit Error Probability (Pb) for BPSK: {Pb_bpsk}")

plt.figure()
plt.semilogy(snr_db, Pb_bpsk, 'o-')
plt.semilogy(snr_db, ber, 'o-')
plt.grid(True)
```

```
plt.xlabel('SNR (dB)')
plt.ylabel('Bit Error Rate (BER)')
plt.title('BER vs SNR for BPSK')
plt.show()
```

5 BER for BPSK and ASK

The Bit Error Probability (Pb) for both BPSK and ASK is computed using the Q-function as follows:




```
# Define the Q-function
def Q_function(x):
    return 0.5 * sp.erfc(x / np.sqrt(2))

snr_db = np.arange(1, 11)

Eb_N0 = 10**(snr_db / 10)
Pb_ask = Q_function(np.sqrt(Eb_N0))
for snr, pb in zip(snr_db, Pb_ask):
    print(f"SNR (dB): {snr}, Bit Error Probability (Pb) for ASK: {pb}")
# Plot the results
plt.figure()
plt.semilogy(snr_db, Pb_ask, 'o-')
plt.semilogy(snr_db, ber, 'o-')
```

```
plt.grid(True)
plt.xlabel('SNR (dB)')
plt.ylabel('Bit Error Rate (BER)')
plt.title('BER vs SNR for Binary ASK')
plt.show()
```

6 Conclusion

As expected, the theoretical Pb for BPSK is lower than for ASK at the same SNR levels. The simulated BER for BPSK closely follows the theoretical predictions, demonstrating the effectiveness of BPSK in mitigating noise. ASK, however, shows higher BER at the same SNR, indicating that BPSK is more robust in noisy environments.