ECE131A Project 1: Monte Carlo Simulation and BER Analysis

Darren Saelee

August 20, 2025

Contents

1 Introduction

The purpose of this project is to study the average probability of error, or bit error rate (BER), in a binary communication system, both analytically and via Monte Carlo simulation. The system transmits either a $+S$ or $-S$ voltage to represent a binary bit ("1" or "0"). The received signal is the random variable X, and is affected by noise represented by the random variable N. The model is:

$$X = \begin{cases} S + N, & \text{if } H_1 \text{ was sent (bit "1")} \\ -S + N, & \text{if } H_0 \text{ was sent (bit "0")} \end{cases}$$

2 Gaussian Noise

2.1 Steps Taken

To evaluate the Bit Error Rate (BER), we perform a Monte Carlo simulation.

1. Initialization: We set the number of trials to $M = 10^5$ and define the signal-to-noise ratio in decibels, SNR_{dB}, ranging from 1 dB to 10 dB in 0.5 dB steps. The SNR values are converted to linear scale by
$$\text{SNR}_{\text{Linear}} = 10^{\text{SNR}_{\text{dB}}/10}.$$
We use $\sigma = 1$. The derivation for $\text{SNR}_{\text{Linear}}$ is shown in the appendix.

2. Noise Generation: For each SNR value, a Gaussian noise vector is generated using the Matlab "randn" function which returns normal distributed random values.

3. Bit Stream Generation: A stream of uniform random bits (0 through 1) is created using a pseudo-random Mersenne Twister Seed.

4. Modulation: The bit stream's values are thresholded at 0.5 and mapped to signal amplitudes: $+S$ if the bit is 1, and $-S$ if the bit is 0, where $S = \sqrt{\text{SNR}_{\text{Linear}}}$.

5. Noise Addition: Gaussian noise is added so that $X = S + N$ or $X = -S + N$, which simulates the received signal.

6. Decision Rule and Error Count: A threshold at 0 is applied. If a transmitted $+S$ yields a received value $X < 0$, it is counted as an error. Bit errors are counted across trials and divided by the total number of $+S$ transmissions to obtain the simulated BER.

7. Theoretical BER Calculation: The analytical BER for Gaussian noise is computed using the Q-function:
$$P_e = Q\big(S\big).$$

8. Plot Results: BER vs. SNR curves then are plotted on a semi-logarithmic graph for both simulated and analytical results.

2.2 Theoretical BER Derivation

Shown in the appendix, $P_e = P(E \mid H_1)$. And since probability of error given H_1, is when $P(N < -S)$. By integral Definition:

$$\text{P}(e|H_1) = \int_{-\infty}^{-S} f_N(x)\,dx = \int_{-\infty}^{-S} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\,dx$$

For Gaussian PDF with a zero mean. And with subbing $t = \frac{x}{\sigma}$ and $dt = \frac{dx}{\sigma}$, we get:

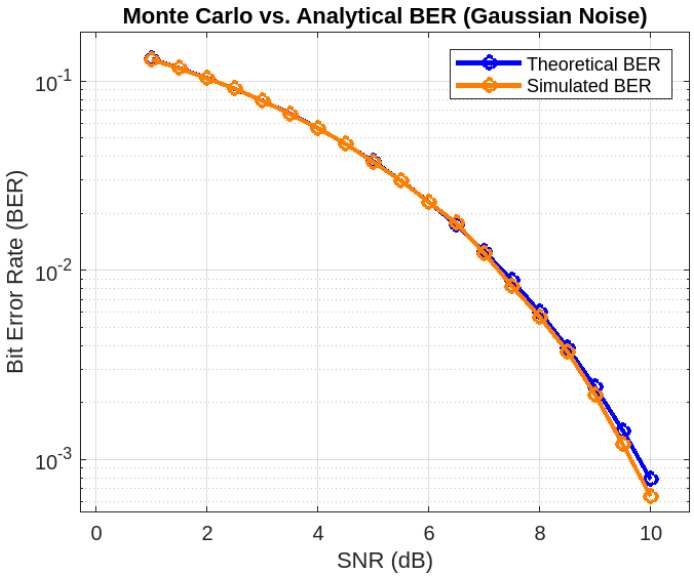
$$P_e = \int_{-\infty}^{-S/\sigma} \frac{1}{\sqrt{2\pi}} e^{-t^2/2}\, dt = \Phi\left(-\frac{S}{\sigma}\right) = Q\left(\frac{S}{\sigma}\right)$$

In our program, our theoretical BER for H_1 is calculated as

$$P_e = Q\left(\sqrt{\text{SNR}_{\text{Linear}}}\right) \text{ for } \sigma = 1 \text{ and } S = \sqrt{\text{SNR}_{\text{Linear}}} \text{ derived in the appendix}$$

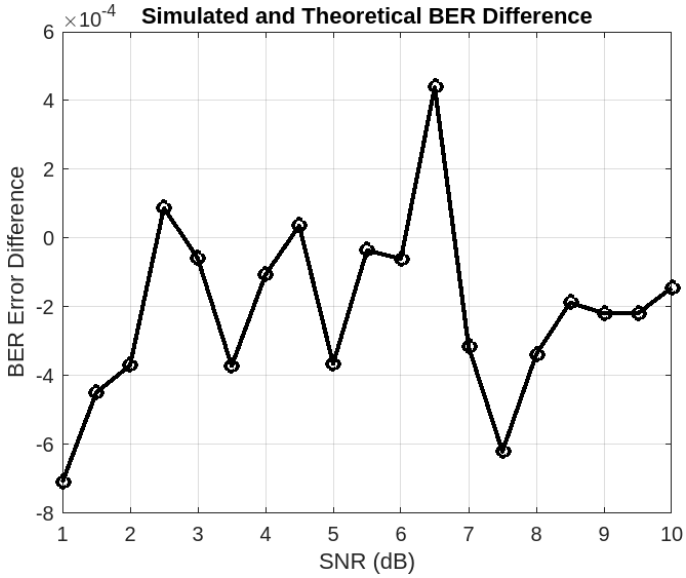
2.3 Simulation vs Theory Comparison

BER Graph: The plot below compares the simulated bit error rate (BER) in orange and the theoretical BER in blue.

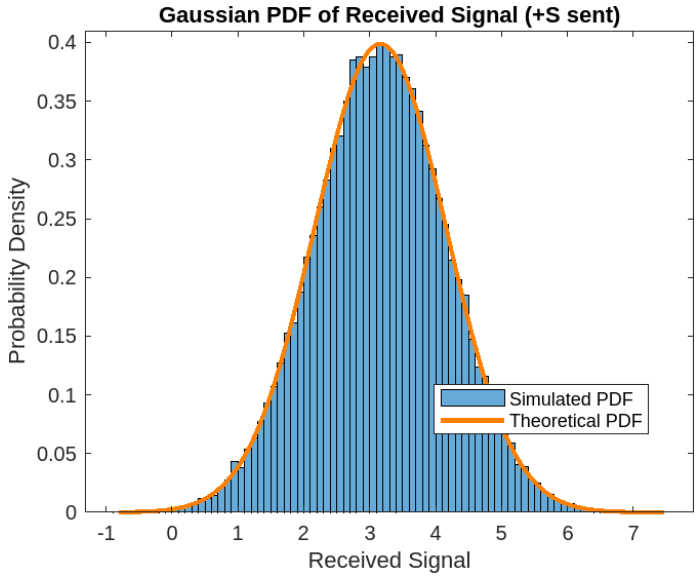


Overall, the simulated BER shows a close match to the theoretical curve. In addition, the BER decreases exponentially with increasing SNR in the semi-log plot, which makes sense because the larger the signal amplitude is, the stronger the signal is, and the signal is less prone to error. The close match of our simulation to the theoretical results and expectations supports the simulations accuracy and the theoretical calculations accuracy to predict error.

While the simulated BER slightly diverges from the theoretical line at higher SNR values, this is likely due to random variation from our finite sample sizes rather than any systematic error. Running the simulation with different random seeds or a higher seed shows that the graphs agree more strongly.



In the error graph above, we observe how differences between the simulated and theoretical BER are quite small, remaining below an order of 10^{-4}. This again shows that the simulation is accurate in predicting the theoretical behavior under Gaussian noise. The sharp fluctuations observed in the graph are likely due to randomizing noise, randomizing the bit stream, and the finite-ness of our sample size of 100,000. For example, when the sample size is increased to 10,000,000, these differences become a scale of a hundred times smaller on the order of around 10^-6.



In the Gaussian PDF plot, we see the expected bell-shaped curve of a normal distribution, with the peak centered around $x = S$ and the distribution decreasing symmetrically and exponentially from the mean peak. The peak is relatively "wide." As though most values cluster around the mean, there is a sizable density in noise values that deviate around the mean.

3 Laplacian Noise

3.1 Steps Taken

To simulate the Bit Error Rate (BER) with Laplacian noise, a Monte Carlo simulation is utilized similarly to the Gaussian experiment.

1. Initialization: First the number of trials is defined as $M = 10^5$, and the range of SNR values in decibels ($1 - 10$ with 0.5 increments in our case). The SNR in DB is then converted to linear scale for use in our calculations as previously in the Gaussian Noise simulation.

2. Noise Generation: Then we generate a uniformly distributed random variable and apply the inverse CDF of the Laplacian distribution to transform it into Laplacian noise. The inverse CDF, which will be derived in Section 3.2, is for now given by:

$$F_N^{-1}(u) = \begin{cases} \frac{1}{\alpha}\ln(2u), & 0 < u < \frac{1}{2} \\ -\frac{1}{\alpha}\ln(2(1-u)), & \frac{1}{2} \leq u < 1 \end{cases}$$

   where $\sigma = 1$, $\alpha = \sqrt{2}$.

3. Bit Stream Generation: A stream of random bits (0 through 1) is created using a uniform distribution, with a pseudo-random Mersenne Twister Seed.

4. Modulation: Each value in the bit stream vector is thresholded at 0.5 and mapped to a signal amplitude of $+S$ for 1 or $-S$ for 0. $S = \sqrt{\text{SNR}_{Linear}}$

5. Noise Addition: Laplacian noise is added (with $X = N \pm S$) to the transmitted signal to simulate the received signal.

6. Decision Rule and Error Count: We calculate the probability of error by parsing through the +signals sent and parsing if the received signal is an error or crosses the threshold ($X < 0$ for $+S$ sent). Bit errors are counted and divided by the total number of +S bits to calculate the simulated BER.

7. Theoretical BER Calculation: The theoretical BER for Laplacian noise is given by:

$$P_e = \frac{1}{2}e^{-\alpha S}$$

   where $S = \sqrt{\text{SNR}_{\text{Linear}}}$.

8. Plot Results: BER vs. SNR curves then are plotted for both simulated and analytical results for comparison.

3.2 Analytical BER Derivation

The probability density function of Laplacian noise is given by:

$$f_N(n) = \frac{\alpha}{2}e^{-\alpha|n|}$$

$$\text{Given } \sigma^2 = \frac{2}{\alpha^2}, \text{ and for } \sigma = 1, \Rightarrow \alpha = \sqrt{2}$$

Assume bit $+S$ is transmitted (H_1). An error occurs if the received signal is less than zero:

$$X = +S + N < 0 \quad \Rightarrow \quad N < -S$$

So the probability of error is:

$$P_e = P(X < 0 \mid H_1) = P(N < -S)$$

For Laplacian noise with zero mean its PDF:

$$f_N(n) = \frac{\alpha}{2} e^{-\alpha|n|}$$

and for a negative signal, or negative n the PDF corresponds to:

$$f_N(n) = \frac{\alpha}{2} e^{\alpha n}, \quad n < 0$$

And such, the analytical probability is:

$$P_e = \int_{-\infty}^{-S} \frac{\alpha}{2} e^{\alpha n}\, dn = \left[\frac{1}{2} e^{\alpha n}\right]_{-\infty}^{-S} = \frac{1}{2} e^{-\alpha S}$$

$$\Rightarrow P_e = \frac{1}{2} e^{-\alpha S}$$

3.3 CDF and Sample Generation

In order to get Laplacian noise, we use the inverse of the CDF of the Laplace distribution. Typically, the CDF function ($F_N(n)$) takes a noise value as input and gives the probability that the noise is less than or equal to that value. Therefore, we inverse the function as $F_N^{-1}(u)$, for input u from the uniform random distribution in $(0, 1)$, to obtain Laplacian-distributed noise as output. This allows us to simulate Laplacian noise using uniform random inputs.
In order to derive this equation for generating Laplacian noise, we start by finding the CDF of the Laplace function with integration.
For a zero mean Laplacian, the PDF is:

$$f_N(n) = \frac{\alpha}{2} e^{-\alpha|n|}, \quad -\infty < n < \infty$$

We can find our CDF with integration

$$F_N(n) = P[N \le n] = \int_{-\infty}^{n} f_n(t)\, dt$$

Case 1: $n < 0$

$$F_N(n) = \int_{-\infty}^{n} \frac{a}{2} e^{\alpha t}\, dt = \left[\frac{1}{2} e^{\alpha t}\right]_{-\infty}^{n} = \frac{1}{2} e^{\alpha n}$$

Case 2: $n \ge 0$

$$F_N(n) = \int_{-\infty}^{0} \frac{\alpha}{2} e^{\alpha t}\, dt + \int_{0}^{n} \frac{\alpha}{2} e^{-\alpha t}\, dt$$

$$= \frac{1}{2} + \left[-\frac{1}{2} e^{-\alpha t}\right]_0^n = \frac{1}{2} + \left(-\frac{1}{2} e^{-\alpha n} + \frac{1}{2}\right) = 1 - \frac{1}{2} e^{-\alpha n}$$

With some algebraic manipulation, we can find the inverse in terms of $u \in (0, 1)$:

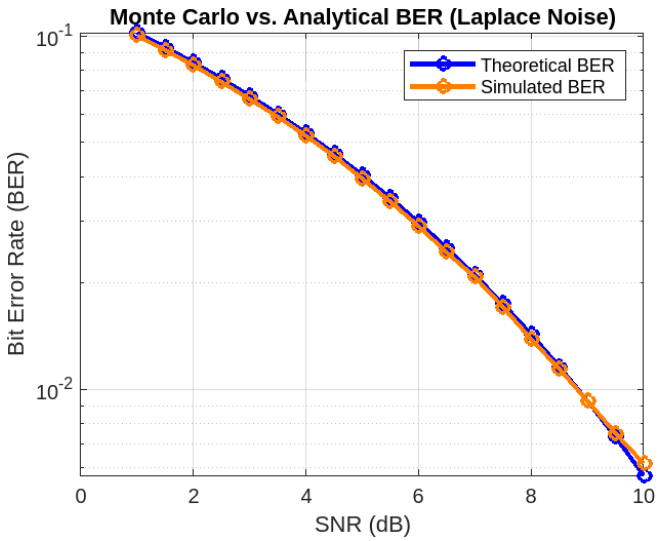
$$F_N(n) = \begin{cases} \frac{1}{2} e^{\alpha n}, & n < 0 \\ 1 - \frac{1}{2} e^{-\alpha n}, & n \ge 0 \end{cases} \Rightarrow \begin{cases} \frac{1}{\alpha} \ln 2u, & 0 < u < \frac{1}{2} \\ -\frac{1}{\alpha} \ln 2(1-u), & \frac{1}{2} \le u < 1 \end{cases}$$

from the analytical CDF derivation above: $\alpha = \sqrt{2}$

3.4 Simulation vs Theory Comparison

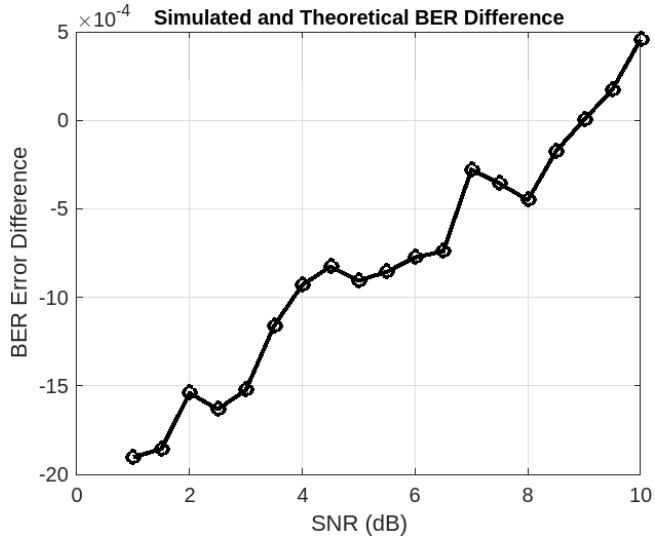
3.4.1 BER for various SNR

The plot below compares the simulated bit error rate (BER) in orange and the theoretical BER in blue.



Overall, the simulated BER closely aligns with the theoretical values derived for Laplace noise. The BER decreases exponentially with increasing SNR, which aligns closely to our analytical expectations derived above and our intuitive expectations that a stronger signal will be less error-prone when transmitted. We conclude the agreement between our simulation and the theoretical results and expectations shows the simulations' accuracy and the theoretical calculations' accuracy to predict error.

It should be noticed that there are times throughout our graph where the curves do not align exactly. However, these small variations and fluctuations are expected, since our simulation has inherent randomness from our Laplace noise, and bit streams, and also a finite number of Monte Carlo trials ($M = 10^5$).



In addition, something seemingly abnormal seen is in the error difference chart, with the uptrend of the BER difference. But trying a different seed, we see that its BER difference is indeed *random* for

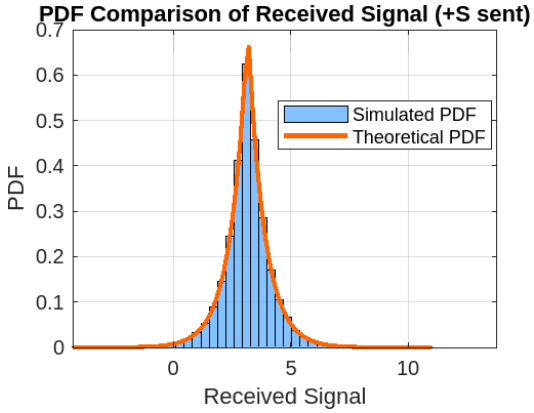
each different SNR, as we see sharp fluctuations. This shows that this pattern shown in our Simulated and Theoretical BER difference chart is a random anomaly for this data set and once again shows our simulation accurately follows analytical BER.

3.4.2 PDF and CDF Comparison

PDF Comparison: In order to compare the PDF of the simulated and theoretical signal X, we use the PDF of the Laplace noise above centered at $x = S$ and symmetric on both sides.

$$f_X(x) = \frac{\alpha}{2} \exp(-\alpha|x - S|)$$

The simulated PDF was obtained by Matlab's Histogram function so that we could see an approximate visual of the density.

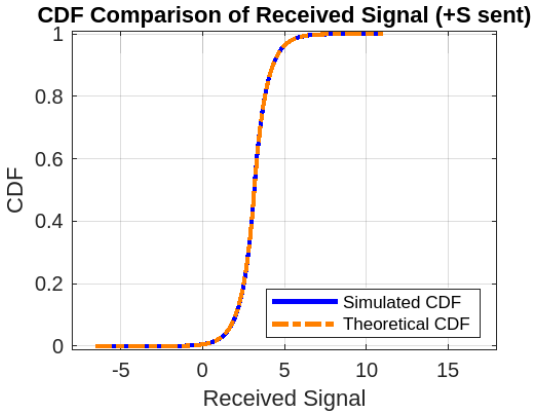


In the above plot, the PDF plot is of a typical Laplacian shape, with the peak occurring around $x = S$, and exponential decay on both sides of the mean. There are long tails which represent the amount of extreme signals received, and a pointy peak which represents a high amount of signals concentrated closely around the mean. These extreme signals represent error, and high concentration means a low spread for most samples. The CDF shows a high rate of increase at the value of $X = S$, which aligns with how the PDF peaks at that x value. There are some areas where the simulation is slightly below or above the theoretical PDF curve, but the dataset overall staying within the PDF curve shows the accuracy of the simulation.

CDF Comparison: The following CDF graph's theoretical component utilizes the theoretical CDF of the received signal found earlier, centered around S:

$$F_X(x) = \begin{cases} \frac{1}{2} \exp\left(\alpha(x - S)\right), & x < S \\ 1 - \frac{1}{2} \exp\left(-\alpha(x - S)\right), & x \geq S \end{cases}$$

The simulated CDF was obtained using the Matlab empirical cumulative distribution function over the received signal samples.



The comparison plot shows strong agreement between the simulated and theoretical CDF's. The CDF curve rises steeply around $x = S$, consistent with how the area underneath the high signal density correlates to a higher change in CDF. The simulated graph follows along the theoretical curve well, and confirms the received samples are accurately follow the theoretical Laplacian form.

4 Conclusion

This experiment successfully demonstrated that bit error rate under different signal-to-noise ratios can be reasonably estimated using Monte Carlo simulation. In both Gaussian and Laplacian noise cases, the simulated BER curves closely followed the theoretical predictions, as seen in the SNR vs. BER graphs. Additionally, the differences between the simulated and analytical results were relatively small, falling within the order of 10^{-3} and 10^{-4}, as shown in our Gaussian and Laplace BER difference graphs, respectively.

A clear detail that we repeatedly saw from the SNR vs BER graphs is that as SNR increases, the BER decreases. This agrees with our expectations, as a stronger signal amplitude relative to the noise makes the received signal more distinguishable, thereby reducing the chances of error.

When comparing the two types of noise, we found that Gaussian noise resulted in a lower BER than Laplacian noise for the SNRs we tested (1 dB–10 dB). For instance, at a SNR of 10 dB, the Gaussian BER reached approximately 10^{-3}, while the Laplacian BER was closer to 5×10^{-2}. This gap is significant, and within our context of a SNR from 1 dB to 10 dB, we can conclude Gaussian noise leads to fewer bit errors.

These results align well with the theoretical properties of the Laplace and Gaussian distributions. For our Laplace graph, though the sharp peak indicates that many signals are concentrated close to the mean value, the heavy tails represent a higher chance of large deviations from the mean. These large deviations correspond to larger, extreme noise values and therefore higher bit error rates. Meanwhile, the Gaussian distribution has a rounder peak and shorter tails, which results in more moderate noise fluctuations and fewer errors on average.

In terms of future experiments, we could improve accuracy by increasing the number of trials in the simulation. A larger sample size would reduce random variation and bring the simulated results closer to the theoretical curve. Of course, this would depend on the amount of error required for an experiment.

A Appendix

A.1 Extra Derivations

A.1.1 Finding Signal From Our Signal Noise Ratio in DB

We first get the linear signal from the signal in db using the following equation:

$$\text{SNR(dB)} = 10\log_{10}(\text{SNR}_{\text{Linear}})$$
$$\Rightarrow \text{SNR}_{\text{Linear}} = 10^{\text{SNR(dB)}/10}$$

Using the equation below, we get the signal amplitude using its variance. This is the signal value S, which we use in our theoretical and simulated calculations.

$$\text{SNR}_{\text{Linear}} = \frac{S^2}{\sigma^2}$$
$$\Rightarrow S = \sqrt{\text{SNR}_{\text{Linear}}}; \sigma = 1$$

A.1.2 Probability of Bit Error Derivation

Recall that X is the received data random variable:

$$X = \begin{cases} S + N, & H_1 = \text{"1"} \\ -S + N, & H_0 = \text{"0"} \end{cases}$$

We assume $\text{P}(H_0) = \text{P}(H_1) = \frac{1}{2}$. We decide with threshold $t_0 = 0$, if the signal is above 0 we guess it is 1 and if it is below 0 we guess it is 0:

- If $x \geq t_0 = 0$, we decide H_1 was sent;
- If $x \leq t_0 = 0$, we decide H_0 was sent.

* We used these bounds to derive the analytical probability of error above as: The probability of an error when H_1 was sent:

$$\text{P}(e|H_1) = \int_{-\infty}^{0} f_1(x)\,dx$$

The probability of an error when H_0 was sent:

$$\text{P}(e|H_0) = \int_{0}^{\infty} f_0(x)\,dx$$

The average probability of error $P(e)$ is the sum of the error probabilities of each hypothesis.

$$P(e) = P(e|H_1)P(H_1) + P(e|H_0)P(H_0)$$

Since we assumed both hypotheses are equally likely, this simplifies to:

$$P(e) = \frac{1}{2}P(e|H_1) + \frac{1}{2}P(e|H_0)$$

Since our noise distribution is symmetric, the two conditional error probabilities are equal:

$$P(e|H_1) = P(e|H_0)$$
$$\Rightarrow P(e) = P(e|H_1) = P(e|H_0)$$

We use this equation to calculate both analytical and simulated errors, as we only need to calculate the error for one hypothesis (in our work, we use H_1).

A.2 Code Listings

A.2.1 Gaussian Main Script

```
snr_db = 1:0.5:10;
snr_linear = 10.^(snr_db./10);
sigma = 1;
BER_sim = zeros(size(snr_db)); % Simulated Monte Carlo BER
BER_theory = zeros(size(snr_db)); % Analytical BER

for i = 1:length(snr_db)
    % generate Gaussian Noise Vector
    N = generate_gaussian_noise(M, 2000);

    % generate Info Bits with Uniform Distribution
    bit_stream = generate_info_bits(M, 2001);

    % assigns Signal Amplitudes its signs depending on threshold (
       modulation)
    [S_modulated, bit_mask, S] = modulate(bit_stream, snr_linear(i));

    % now add Noise to Modulated Signals
    S_recv = S_modulated + N;

    % calculates Perecentage of errors in S, Signal
    P_sim = calculate_error(S_recv, bit_mask);
    BER_sim(i) = P_sim;

    % compute Analytical BER with Qfunction
    P_thr = qfunc(S);
    BER_theory(i) = P_thr;
end

% plot THR vs SIM probability error
figure;
semilogy(snr_db, BER_theory, 'b-o', 'LineWidth', 2, 'DisplayName', '
   Theoretical BER');
hold on;
semilogy(snr_db, BER_sim, '-o','Color' ,[1, 0.5, 0], 'LineWidth', 2, '
   DisplayName', 'Simulated BER');
grid on;
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)');
title('Monte Carlo vs. Analytical BER (Gaussian Noise)');
legend('show');

% plot error graph
error_diff = BER_sim - BER_theory;
figure;
plot(snr_db, error_diff, 'k-o', 'LineWidth', 2);
```

```
xlabel('SNR (dB)');
ylabel('BER Error Difference');
        title('Simulated and Theoretical BER Difference');
grid on;

received_H1 = S_recv(bit_mask);  % samples where +S was sent

% plot theoretical PDF Graph and Simulated PDF
figure;
histogram(received_H1, 'Normalization', 'pdf');  % empirical PDF
hold on;
x_vals = linspace(min(received_H1), max(received_H1), 100);
theory_pdf = normpdf(x_vals, S, sigma);
plot(x_vals, theory_pdf, '-', 'Color' ,[1, 0.5, 0],'LineWidth', 2);
xlabel('Received Signal');
ylabel('Probability Density');
title('Gaussian PDF of Received Signal (+S sent)');
legend('Simulated PDF', 'Theoretical PDF', 'Location', 'best');

ylim([0.00 0.41])
```

A.2.2 Laplace Main Script

```
% Initializations
M = 1e5; % number of simulations
snr_db = 1:0.5:10;
snr_linear = 10.^(snr_db./10);
BER_sim = zeros(size(snr_db)); % Simulated Monte Carlo BER
BER_theory = zeros(size(snr_db)); % Analytical BER

sigma = 1;
alpha = sqrt(2); % For unit variance Laplace

for i = 1:length(snr_db)
    % Generate Laplace Noise Vector
    N = generate_info_bits(M, 2002); % generates our uniform noise
    NL = transform_to_laplace_noise(M, N, alpha); % inverse cdf

    % Generate Info Bits with Uniform Distribution
    bit_stream = generate_info_bits(M, 2003);

    % Assigns Signal Amplitudes its signs depending on threshold
    [S_modulated, bit_mask, S] = modulate(bit_stream, snr_linear(i));

    % Add Noise to get Recieved Data Vector
    S_recv = S_modulated + NL;

    % Calculates Perecentage of Errors Out of +S sent
    P_sim = calculate_error(S_recv, bit_mask);
```

```
    BER_sim(i) = P_sim;

    % Calculate Analytical Values With Equation
    P_thr = 0.5 * exp(-(alpha * S)); % Analytical error for Laplacian
       noise
    BER_theory(i) = P_thr;
end

% plot BER Vs SNR
figure;
semilogy(snr_db, BER_theory, 'b-o', 'LineWidth', 2, 'DisplayName', '
   Theoretical BER');
hold on;
semilogy(snr_db, BER_sim, '-o','Color' ,[1, 0.5, 0] ,'LineWidth', 2, '
   DisplayName', 'Simulated BER');
grid on;
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)');
title('Monte Carlo vs. Analytical BER (Laplace Noise)');
legend('show');

% plot error differences!
error_diff = BER_sim - BER_theory;
figure;
plot(snr_db, error_diff, 'k-o', 'LineWidth', 2);
xlabel('SNR (dB)');
ylabel('BER Error Difference');
title('Simulated and Theoretical BER Difference', 'FontSize', 10);
grid on;

received_H1 = S_recv(bit_mask);  % samples where +S was sent

% now we plot CDF and PDF!
% PDF Plot using histogram
figure;
histogram(received_H1, 50, 'Normalization', 'pdf','FaceColor', [0.2
   0.6 1], 'FaceAlpha', 0.6, 'DisplayName', 'Simulated PDF');
hold on;
x_vals = linspace(min(received_H1), max(received_H1), 100);
theory_pdf = (alpha / 2) * exp(-alpha * abs(x_vals - S));
plot(x_vals, theory_pdf, '-', 'Color', [1, 0.4, 0], 'LineWidth', 2,'
   DisplayName', 'Theoretical PDF');

xlabel('Received Signal');
ylabel('PDF');
title('Laplace PDF Comparison of Received Signal (+S sent)');
legend('Location', 'best');
grid on;
xlim([-4.3 13.8]);
```

```
ylim([-0.00 0.70]);

% CDF plot
[f_emp_cdf, x_cdf] = ecdf(received_H1); % get emperical cdf of
   Hypothesis 1
theory_cdf = zeros(size(x_vals));
theory_cdf(x_vals < S) = 0.5 * exp(alpha * (x_vals(x_vals < S) - S));
theory_cdf(x_vals >= S) = 1 - 0.5 * exp(-alpha * (x_vals(x_vals >= S)
   - S));

figure;
plot(x_cdf, f_emp_cdf, 'b', 'LineWidth', 2, 'DisplayName', 'Simulated
   CDF');
hold on; grid on;
plot(x_vals, theory_cdf, '-.','Color' ,[1, 0.5, 0],'LineWidth', 2, '
   DisplayName', 'Theoretical CDF');
xlabel('Received Signal');
ylabel('CDF');
title('Laplace CDF Comparison of Received Signal (+S sent)');
legend('Location', 'southeast');

xlim([-8 18])
ylim([-0.01 1.01])
```

A.2.3 Generate Gaussian Noise

```
function N = generate_gaussian_noise(M, seed)
% Generate Guassian Noise Vector with size 1xM
    rng(seed,  twister ); % twister for psuedo random
    N = randn(1, M);      % 1xM noise vector
```

A.2.4 Generate Info Bits

```
function bit_stream = generate_info_bits(M, seed)
% Create 1xM Normally Distributed Random Bit Stream
    rng(seed,  twister ); % New Psuedo random Mersenne Twister Seed
    bit_stream = rand(1,M);
```

A.2.5 Transform to Uniform Laplace Noise

```
function NL = transform_to_laplace_noise(M, U, a)
% transform uniform into Laplace Noise with Inverse Laplace
% a = alpha value, M = samples, U = Uniform Noise

    NL = zeros(1, M);
    for i = 1:M
        u = U(i);
        if u < 0.5
```

```
            NL(i) = 1/a * log(2 * u);
        else
            NL(i) = -1/a * log(2 * (1 - u));
        end
    end
end
```

A.2.6 Modulate

```
function [modulated_signal, bit_mask, S] = modulate(bit_stream,
   SNR_linear_i)
    % assign Data Vector with +-S Values
    S_squared = SNR_linear_i; % for sigma = 1
    S = sqrt(S_squared);
    bit_mask = bit_stream >= 0.5; % threshold of 0.5 asigns 0 or 1
    modulated_signal = S * (2 * bit_mask - 1); % Assign +S or -S
end
```

A.2.7 Calculate Error

```
function P_sim = calculate_error(S_recv, bit_mask)
% calculates and Returns the Perecentage of errors in Signal
    S_recv_H1 = S_recv(bit_mask); % Filter out Data recieved to only
       where +S sent
    err_flags = S_recv_H1 < 0; % Collect XX that has error

    % Calc Probability of Recieving an Error when +S sent
    S_H1_tot = sum(bit_mask); % Total count of +S sent
    total_err = sum(err_flags); % Total count of errors from +S
       recieved
    P_sim = total_err/S_H1_tot; % Probability of error with H1
```