ECE131A Project Report

Ma, Yunhui
UID: 505-179-815
Date: 12/4/2019

Introduction

In objectives of this lab is to use the Monte Carlo simulation and analytical to evaluate the average probability of error of binary data communication system. A system would receive the signal S with noise error N. In this lab, it uses two different ways including Gaussian and Laplacian random variables to generate and simulate the error. Furthermore, using the simulation results compares to analytical results.

Theoretical Calculation

Case A:

$P(e|H_1) = \int_{-\infty}^{0} f_N(x-S)dx$ $P(e) = P(e|H_1)$;

$P(e) = P(e|H_1) = \int_{-\infty}^{0} f_N(x-S)dx = \int_{-\infty}^{-S} f_N(x)dx = \Phi(-S) = 1 - \Phi(S)$

Assume, S = 1 $P(e) = 1 - \Phi(1) = 1 - 0.8413 = 0.1587$

Case B:

Laplacian random variable with $\sigma = 1$, $f_N(n) = \left(\frac{\alpha}{2}\right)e^{-\alpha|n|}$, $-\infty < n < \infty$

$P(e|H_1) = \int_{-\infty}^{0} f_N(x-S)dx = \int_{-\infty}^{0} \left(\frac{\alpha}{2}\right)e^{-\alpha|x-s|}dx.$

We need to consider the sign of (x-s), it will be very completed. However, using $P(e|H_0)$ =

$\int_{0}^{\infty} f_N(x+S)dx$ is more convenient because x+s is always positive when x is from 0 to ∞

$P(e) = P(e|H_0) = \int_{0}^{\infty} f_N(x+S)dx = \int_{0}^{\infty} \left(\frac{\alpha}{2}\right)e^{-\alpha|x+s|}dx = \left(\frac{\alpha}{2}\right)e^{-\alpha s}\int_{0}^{\infty} e^{-\alpha x}dx =$

$$\frac{\sqrt{2}}{2}e^{-\sqrt{2}\cdot s}\frac{1}{-\sqrt{2}}e^{-\sqrt{2}\cdot x}|_0^{+\infty} = 0 + \frac{e^{-\sqrt{2}\cdot s}}{2}e^{-\sqrt{2}\cdot 0} = \frac{e^{-\sqrt{2}\cdot s}}{2}$$

Assume S = 1, $P(e) = \frac{e^{-\sqrt{2}\cdot 1}}{2} = 0.12156$

Monte Carlo simulation

Case A

Step1, Generating an 1xM iid Gaussian noise vector.

- M is a positive integer number. The larger M is, the more noise samples.
- To generate different random variable, set randn('seed', 2000);
- Use MATLAB randn(1, M) function to generate the normal distribution random numbers, and store into the Gaussian noise vector N.

Step2, Generating an 1xM random binary data vector of $\pm$S values with equal probability

- Set uniformly distributed pseudo-random generator rand ('seed', 2001), and generate M random variable and store into S1 vector.
- Set the random variables which are greater or equal than 0.5 is 1, otherwise is 0. It's equivalent to $P(H_1) = P(H_2) = \frac{1}{2}$.
- Value of S depends on desired SNR (dB). Using function SS = S*(2*S2-1) to scale the input binary vector S2 to SS

Step3, Generating received vector by adding the input values and noises. X = SS + N

Step4, Collecting received data under hypothesis H_1, XX = X.*S2. since S2 is binary vector, it only collects the data when S2 is 1.

Step5, Decision for received data number under hypothesis H_1. Collect all XX data with values less or equal to 0. X1 = XX < 0, which are the error data;

Step6, Average error probability evaluation. The error probability is equal to the error results divided by the number of S2 is 1 (H_1 data sent)

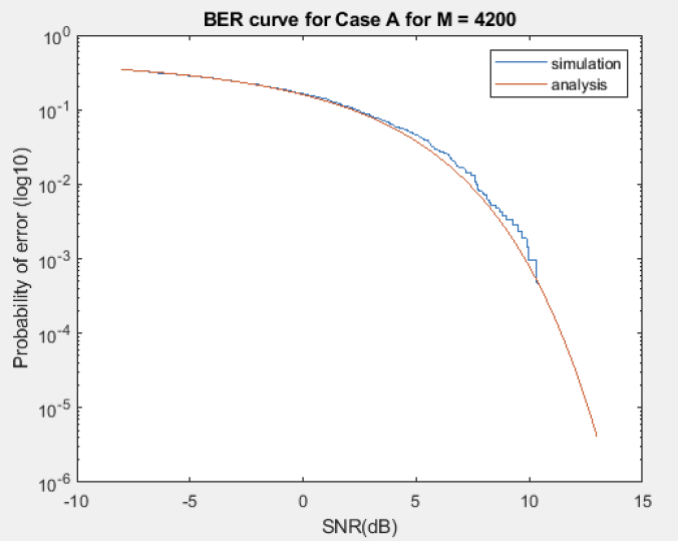
$P(e) = \frac{X11=sum(X1)}{M1=sum(S2)}$

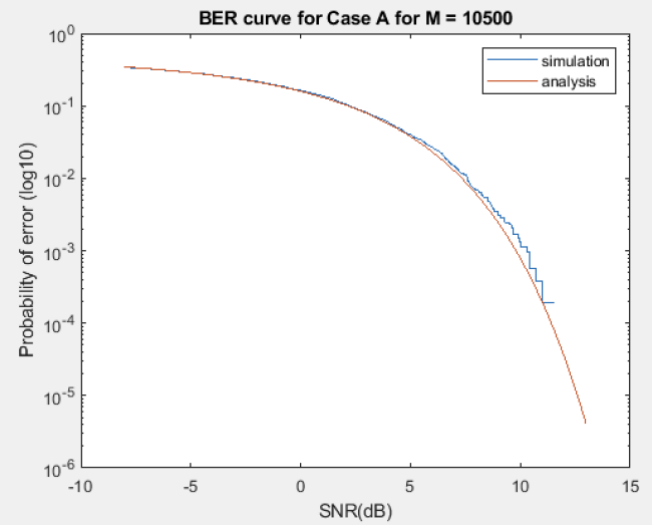
Analysis:

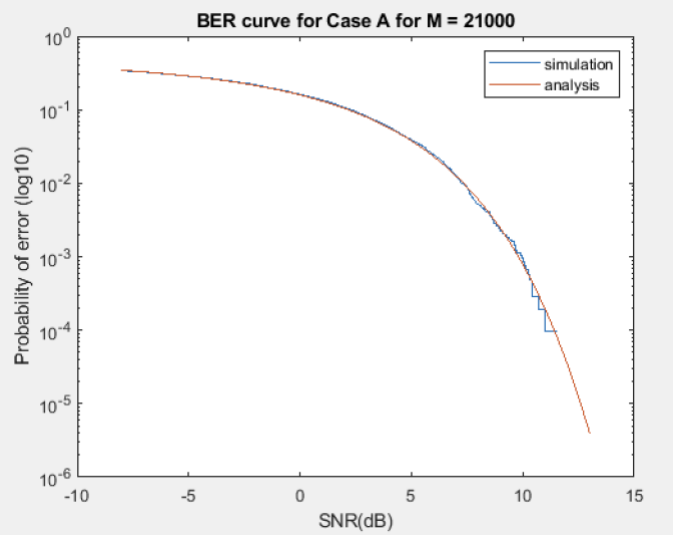
For **S = 1**, from Theoretical calculation P(e) = 0.1587

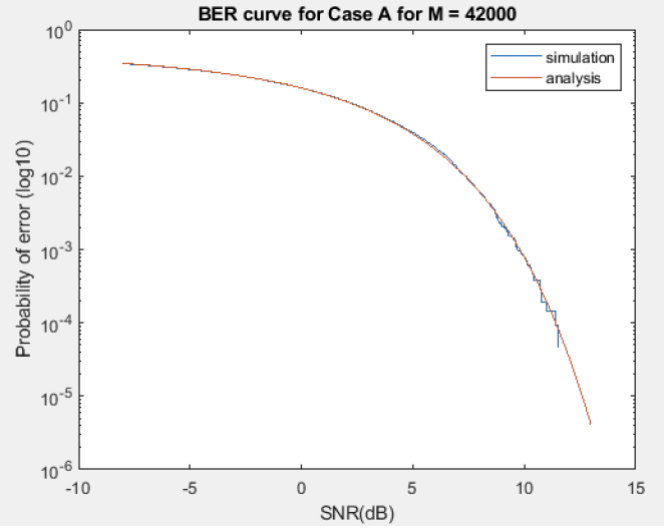
M	P(e)
100	0.1176
500	0.1521
1000	0.1628
2000	0.1663
5000	0.1629
8000	0.1649
10000	0.1648
50000	0.1597
100000	0.1591
1000000	0.1586
5000000	0.1587

If the S is a constant number, as the M increases to a very large number, the probability of error is close to the theoretical value. The more samples of simulation have, the Monte Carlo simulation value would be approximately equal to the theoretical value which is calculated from Gaussian distribution.









From the BER logarithmic curve, as the M increases, the Monte Carlo simulation curve is close to the theoretical curve. Furthermore, if M is approaching infinity, the simulation curve would be the same as the theoretical curve.

The Monte Carl simulation has no data after the SNR_dB is about equal to 12 compared to the theoretical curve. One of the reasons is when calculate the S scale. For example, if SNR_dB is 12, $S = 10^{\frac{12}{20}} \approx 3.98$. The normal distribution randn() would not generate some numbers which are less than -3.98. The calculation of X = SS + N would not get any negative value which is the error for all SS is positive. Therefore, there is no negative value error for the positive input such as sending H1, and the probability of error would be zero. And the log10(0) has no value. It won't plug any numbers into the graph.

In addition, it can see that as the SNR_dB increases, the probability of error would become smaller and finally approach to zero. Therefore, if the system processes a louder or larger signal, the noise of error could be ignored.

Case B

Step1, Use the seed = 2002 to generate the 1xM PR uniform vector N.

Step2, Find the cdf F(x) of the Laplacian rv with zero mean and $\sigma = 1$.

Step3, Find the inverse of cdf F(x).

Step4, Transform the 1xM PR uniform vector N based on the inverse of F(x) to the 1xM Laplacian vector NL. Because N is the probability from range 0 to 1, it treats N < 0.5 as negative and N > 0.5 as positive.

Step5, Generating an 1xM random binary data vector of $\pm$S values with equal probability.

Step6, Generating received vector by adding the input values and noises. X = SS + NL.

Step7,

- Collecting received data under hypothesis H_1, XX = X.*S2. since S2 is binary vector, it only collects the data when S2 is 1.
- Decision for received data number under hypothesis H_1. Collect all XX data with values less or equal to 0. X1 = XX < 0, which is the error data;
- Average error probability evaluation. The error probability is equal to the error results divided by the number of S2 is 1 (H_1 data sent)

  P(e) = $\frac{X11=sum(X1)}{M1=sum(S2)}$

$$f_N(n) = \begin{cases} \left(\frac{\alpha}{2}\right)e^{-\alpha n} & n \geq 0 \\ \left(\frac{\alpha}{2}\right)e^{\alpha n} & n < 0 \end{cases} \qquad F_N(n) = \begin{cases} \left(\frac{\alpha}{2}\right)\int_{-\infty}^{0} e^{\alpha n}dx + \left(\frac{\alpha}{2}\right)\int_{0}^{n} e^{-\alpha n}dx & n \geq 0 \\ \left(\frac{\alpha}{2}\right)\int_{-\infty}^{n} e^{-\alpha n}dx & n < 0 \end{cases}$$

$$F_N(n) = \begin{cases} \frac{1}{2}e^{\alpha n}|_{-\infty}^{0} - \frac{1}{2}e^{-\alpha n}|_{0}^{n} & n \geq 0 \\ \frac{1}{2}e^{\alpha n}|_{-\infty}^{n} & n < 0 \end{cases} \qquad F_N(n) = \begin{cases} 1 - \frac{1}{2}e^{-\alpha n} & n \geq 0 \\ \frac{1}{2}e^{\alpha n} & n < 0 \end{cases}$$

Inverse of F(x), $F_N^{-1}(n) = \begin{cases} \frac{\ln(2n)}{\alpha}, & n < 0 \\ \frac{\ln(2-2n)}{-\alpha}, & n \geq 0 \end{cases}$

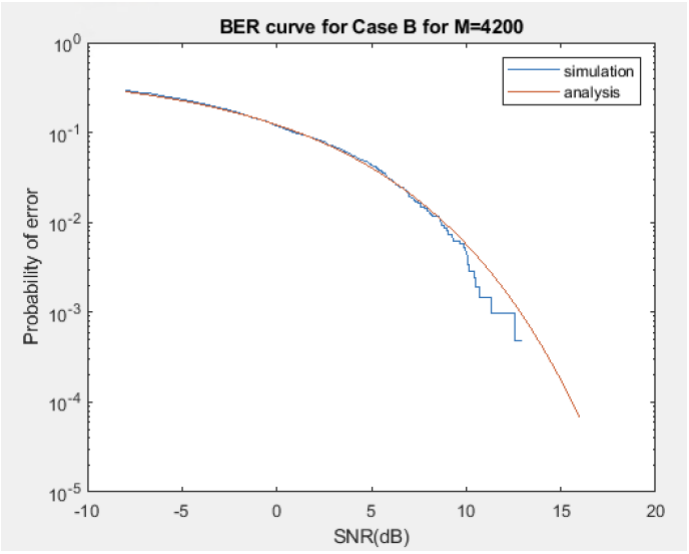
NL = $-\frac{1}{\alpha} sign(N - 0.5)\ln(1 - 2|N - 0.5|)$

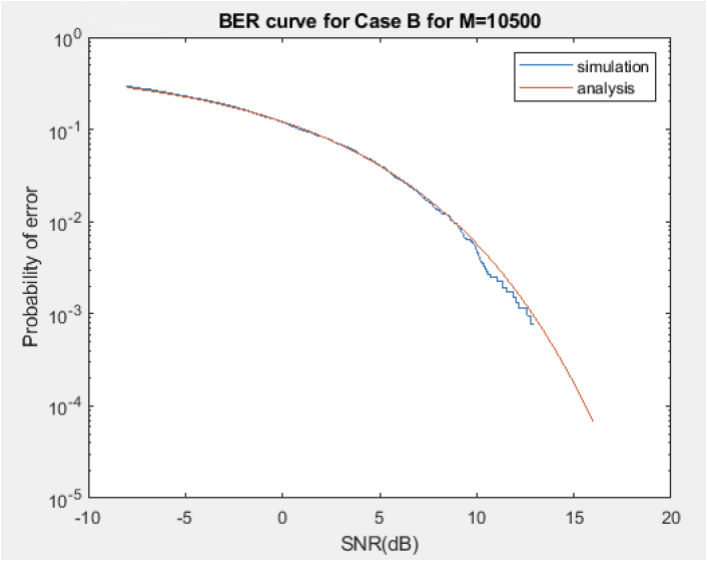
Analysis:

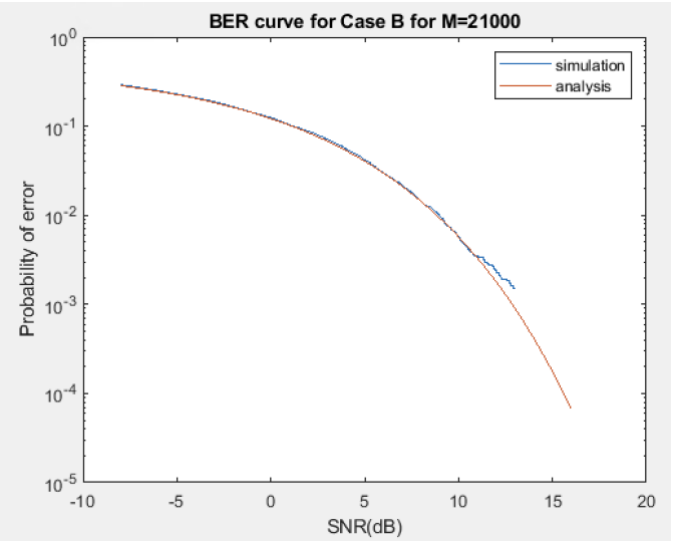
For **S = 1**, from Theoretical calculation for Laplacian distribution Pe = 0.12156

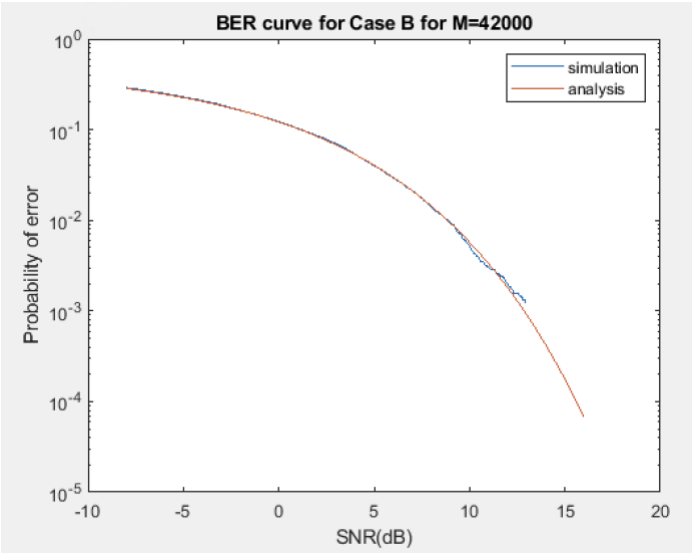
M	P(e)
100	0.1489
500	0.1079
1000	0.1061
100	0.1197
5000	0.1181
8000	0.1186
10000	0.1195
50000	0.1214
100000	0.1232
1000000	0.1216
5000000	0.1215

From this table, similar to case A, as the M increases to a very large number, the probability of error is close or equal to the theoretical value which is calculated from Laplacian distribution.









By using Laplacian random variable for noise, from the BER logarithmic curve case B, as the M increases, the Monte Carlo simulation curve is close to the theoretical curve. Furthermore, if M is approaching infinity, the simulation curve would be the same as the theoretical curve. For case B, it's more precise compared to case A.

The Monte Carl simulation also has no data after the SNR_dB is about equal to 12 compared to the theoretical curve. In this case, the noises are generated by the function.

$$\text{NL} = -\frac{1}{\alpha} sign(N - 0.5) \ln(1 - 2|N - 0.5|) \quad \text{N (0,1)}$$

As the S scale becomes a large number, the calculation of NL would not generate any negative value which are less than S scale. Therefore, it won't have any error for all SS is positive when S is too large. And the log10(0) has no value. It could not find any value in the BER curve above.

Similar to case A, as the SNR_dB increases, the probability of error would become smaller and finally approach to zero. It can have the same result that the system processes a louder or larger signal, the noise of error could be ignored.

Conclusion:

In this project, it uses two ways to simulate the noise error and compared to the theoretical result. For case A, it uses zero mean and standard deviation = 1 of Gaussian random variable for noise error N. Using Gaussian cdf for theoretical calculation. P(e) = 1 - $\Phi(S)$. The scale S is depended on the Signal-to-noise-ratio (SNR_dB). As the sample size M increase, the Monte Carlo simulation results are close to the theoretical results.

For case B, it uses Laplacian random variable with zero mean and standard deviation 1 to simulate the noise error. Base on the calculation, the simulation noise generation function is NL = $-\frac{1}{\alpha} sign(N - 0.5)\ln(1 - 2|N - 0.5|)$. The theoretical noise generation pdf if $f_N(n) = \left(\frac{\alpha}{2}\right) e^{-\alpha|n|}$, and P(e) = $\frac{e^{-\sqrt{2}\cdot s}}{2}$. Similarly, to Case A, as the sample size M increases, the Monte Carlo simulation results are close to the theoretical resules.

In order to produce less error, the signal sending to the system should have large signal-to-noise-ratio.

Appendix (MATLAB Code)

Case A

```
M = 10500;
randn('seed', 2000);
N = randn(1,M);
rand('seed', 2001);
S1 = rand(1,M);
S2 = S1 >= 0.5;
Pe = zeros(1,10500);


SNR_dB = -8;
SNR_range = -8:0.002:12.999;

for i=1:1:10500
    SNR_dB = SNR_dB + 0.002;
    S = 10.^(SNR_dB/20);
    SS = S.*(2.*S2-1);
    X = SS + N;
    XX = X.*S2;
    X1 = XX<0;
    M1 = sum(S2);
    X11 = sum(X1);
    Pe(i)=X11/M1;
end
```

```
%disp(Pe);
figure;
%semilogy(SNR_range, Pe);
semilogy(SNR_range, Pe, SNR_range_t, Pe_t);
legend('simulation', 'analysis');

title('BER curve for Case A for M = 10500');
xlabel('SNR(dB)');
ylabel('Probability of error (log10)');
```

Case B

```
M = 21000;
rand('seed',2002);
N = rand(1,M);
NL = (-1/sqrt(2))*sign(N-0.5).*log(1-2*abs(N-0.5));

rand('seed',2003);
S1 = rand(1,M);
S2 = S1 >= 0.5;

Pe_s = zeros(1,21000);
Pe_l = zeros(1,48000);

SNR_dB = -8;
SNR_range = -8:0.001:12.999;
for i=1:1:21000
    SNR_dB = SNR_dB + 0.001;
    S = 10.^(SNR_dB/20);
    SS = S*(2*S2-1);
    X = SS + NL;
    XX = X.*S2;
    X1 = XX<0;
    M1 = sum(S2);
    X11 = sum(X1);
    Pe_s(i)=X11/M1;
end
```

```
SNR_dB = -8;
SNR_range_l = -8:0.0005:15.9999
for j=1:1:48000
    SNR_dB = SNR_dB + 0.0005;
    S = 10.^(SNR_dB/20);
    Pe_l(j) = (exp(-sqrt(2).*S))/2;
end


figure;
semilogy(SNR_range, Pe_s, SNR_range_l, Pe_l);
legend('simulation', 'analysis');
title('BER curve for Case B for M=21000');
xlabel('SNR(dB)');
ylabel('Probability of error');
```