PROJECT 2 REPORT

RANDOM SIGNAL THEORY

APRIL 25, 2020

HARSHITH CHITTAJALLU
ASU ID- 1218707243

Table of Contents

Introduction

In this Project, there were two tasks: The first task was to generate N random variables that are exponentially distributed with parameter 1 and find the average (i.e., add the random variables and divide by N). then we plot this average versus N. After that, we plot the PDF of this Random Variable (having the average of sum of exponential distributed random variables) for several values of N and give our observations.

Now we do the same as above, except we subtract the mean, and normalize with $\sqrt{N}$ instead of N from the random variable which has the sum of N exponential random variables (Not the Random variable which has the average). We do this many (say M) times, independently, which will give us M numbers at the end. After that, we draw a histogram of all those random variables generated and compare our results under different values of n to the actual PDF of the new Random variable.

Then, the next task was about bit error probability, where we consider the following model for a digital communication system:
$r = \sqrt{E}s + n$ where $s \in \{-1, 1\}$ with equal probability, and n is additive white Gaussian noise with zero mean and variance 1, and E is the signal to noise ratio. The receiver decides $s = +1$ if $r > 0$, and decides $s = -1$, if $r \leq 0$. We estimate the error probability using Monte Carlo simulations by generating many samples and making as many decisions for s under different values of E, such that the error probability ranges from 0.5 to 0.00001 and then plot the error probability versus 10 log10(E), where the y axis is spaced logarithmically (semilog). We also theoretically derive the error probability of this binary communication system as a function of the CDF of a Gaussian random variable and plot it along with our Monte Carlo estimates. Then, we do the same for Laplace distributed noise with mean zero and variance 1. We finally compare the Laplace case with the Gaussian case to check which is better.

Theoretical calculation-1

Let X_i be 'N' i.i.d exponential random variables with parameter λ=1 (or basically, the same random variable with 'N' trails)

Then, Sum of N exponential random variables is a random variable, $S_N \sim$ Erlang (N,λ) = ~Erlang(N,1)

Sample average of the sum of N exponential random variables $M_N = S_N/N$

Pdf of $S_N = \frac{\lambda^N x^{N-1} e^{-\lambda x}}{(N-1)!}$

Cdf of $S_N = \frac{\gamma(N,\lambda x)}{(N-1)!}$

Mean of $S_N, E[S_N] = \mu = N \div \lambda = N$

Standard Deviation of $S_N = \sigma = \sqrt{\frac{N}{\lambda^2}} = \sqrt{N}$

Mean of $M_N = \mu_{M_N} = E\left[\frac{S_N}{N}\right] = \frac{E[S_N]}{N} = 1$

Law of large numbers theorem states that the sample average of a Random variable tends towards its mean, with large enough number of trials: $M_N \to \mu_{M_N}$ as $N \to \infty$

Pdf of $M_N = \frac{S_N}{N} \sim$Erlang(N,Nλ) [∵ If X~ Erlang(N,λ) then Y=aX ~ Erlang(N,$\frac{\lambda}{a}$)] $= \frac{(N\lambda)^N x^{N-1} e^{-N\lambda x}}{(N-1)!}$

Let $Z = \frac{S_N - N}{\sqrt{N}}$ [given in the question], which is of the form $Z = \frac{S_N - \mu}{\sigma}$

Thus we can apply central limit theorem on Z

The central limit theorem (CLT) establishes that, in some situations, when independent random variables are added, their 'properly normalized sum' tends toward a standard normal distribution

Or, if $Z = \frac{S_N - \mu}{\sigma}$

when $N \gg 1 \to Z \sim \phi(0,1)$

Pdf of Z is determined by obtaining the CDF

P[Z <= z]

Let x = $\sqrt{N}z + N$ then, dx = $\sqrt{N}dz$

$= P\left[\frac{S_N - N}{\sqrt{N}} \leq z\right] = P\left[S_N \leq \sqrt{N}z + N\right] =$

$\int \frac{\lambda^N \left(\sqrt{N}z+N\right)^{N-1} e^{-\lambda\left(\sqrt{N}z+N\right)}}{(N-1)!} \sqrt{N}dz$

$=\frac{\gamma(N,\lambda(\sqrt{N}z+N))}{(N-1)!}\sqrt{N}$

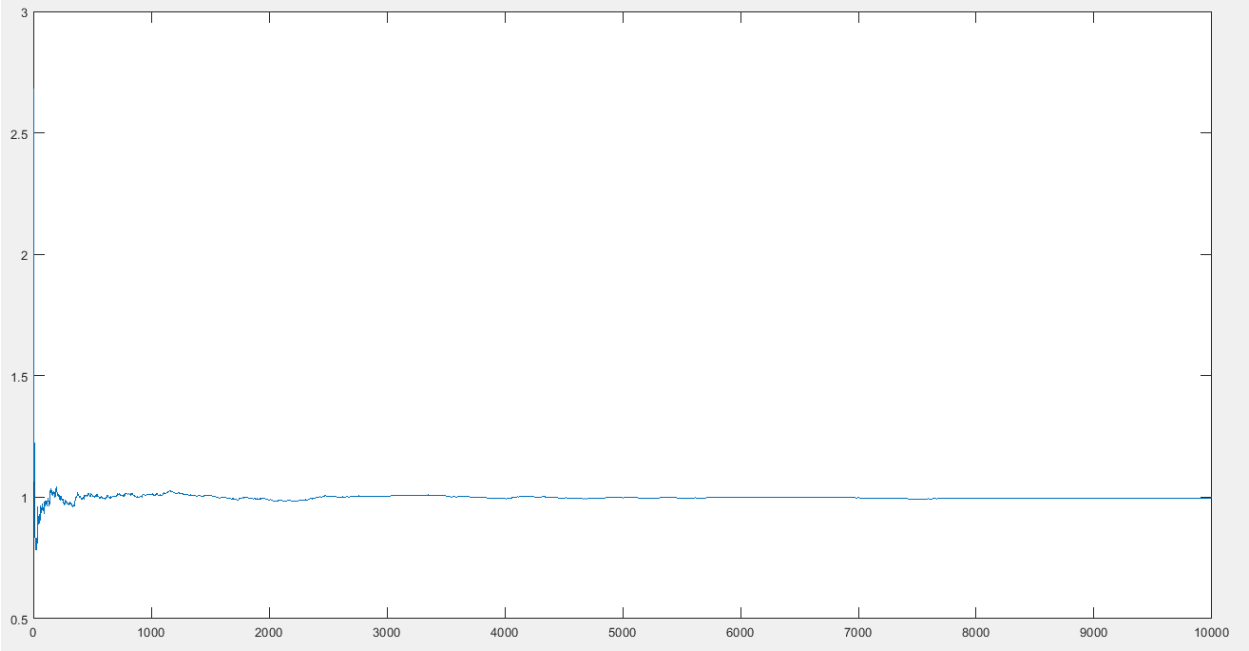
Pdf of Z = $\frac{d}{dz}\left(\frac{\gamma(N,\lambda(\sqrt{N}z+N))}{(N-1)!}\sqrt{N}\right)$

Which means that pdf of Z is = $\frac{\lambda^N \left(\sqrt{N}z+N\right)^{N-1} e^{-\lambda\left(\sqrt{N}z+N\right)}}{(N-1)!}\sqrt{N}$

Results & Conclusion -1

1) When we plot the values of M_N with increasing values of N, we observe that: $M_N \rightarrow \mu_{M_N}$ as $N \rightarrow \infty$

Which is in accordance with the weak law of large numbers theorem.

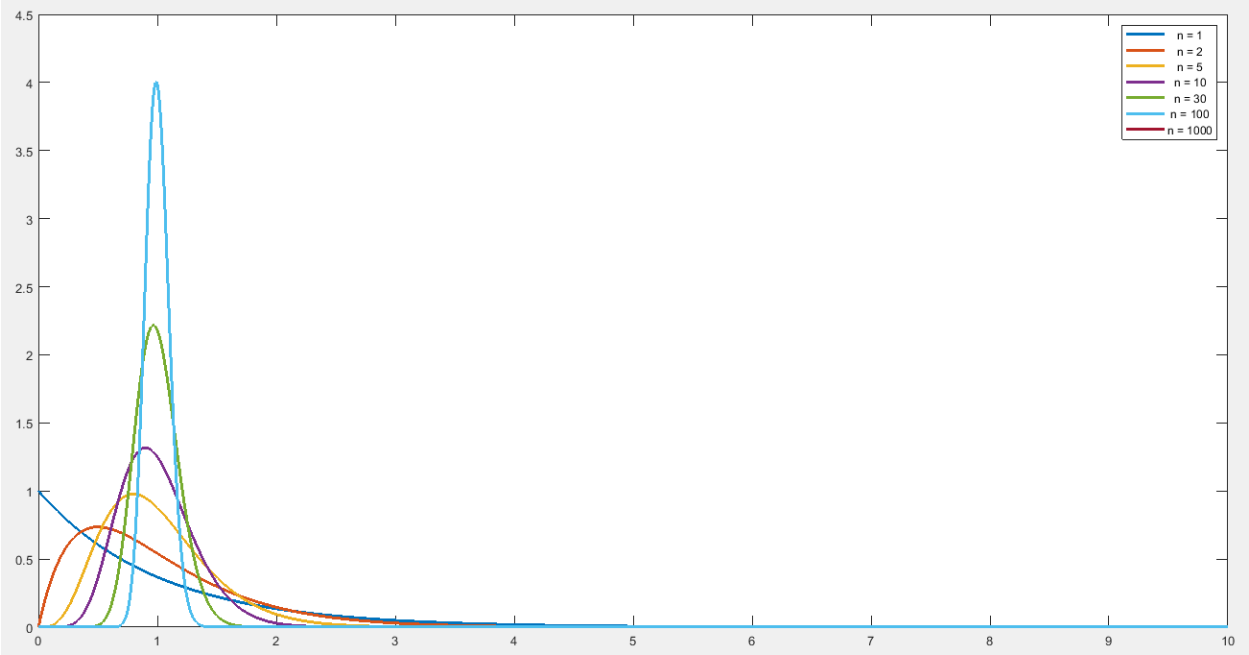


2) When we plot the pdf of M_N for various increasing values of N, we observe that the pdf concentrates at 1.

This is related to the law of large numbers, as it states that the probability of a random variable being equal to its mean becomes 1 when the number of trails approaches infinity. In other words,

$P\left[M_N \cong E[M_N]\right] \cong 1$ when $N \rightarrow \infty$

The proof of this is in the result below, where we can see that the pdf concentrates heavily at 1, which is the expected value of M_N. This means that the probability of getting ANY value from the pdf of M_N is almost equal to 1 (Expected value of M_N) at sufficiently large values of N.

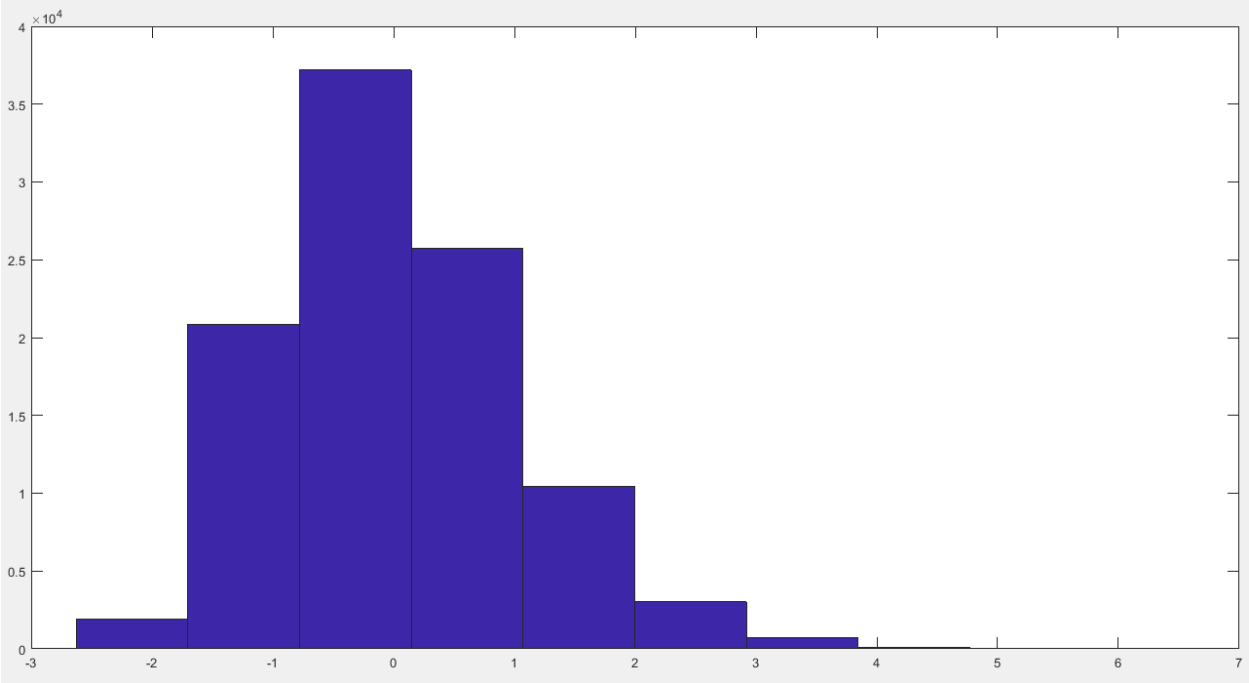


3) Now, we were asked to take the random variable S_N and subtract it with its mean and divide by $\sqrt{N}$. However, since $\sqrt{N} = \sigma_{S_N}$(Standard deviation of S_N), it is the equivalent of 'normalizing' the random variable.

Let this new random variable be 'Z'

According to central limit theorem, this random variable 'Z' will have a standard normal distribution when N is sufficiently large.

We now take the histogram of 'M' random samples from the random variable Z. To do that, we just generate 'N' exponentially distributed numbers with parameter =1, sum them up, subtract with their mean (which is equal to N) and divide them by $\sqrt{N}$.



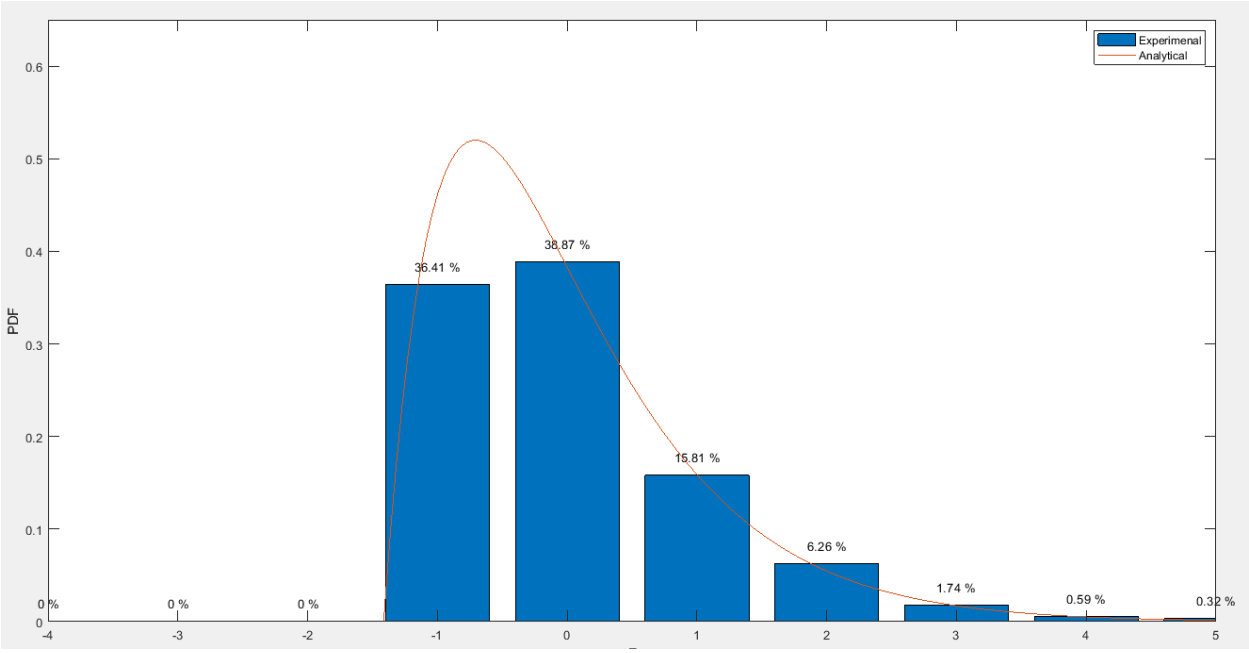
The frequency of the values of Z among 'M' values will represent the pdf of Z for large samples of M. In the above plot, N=10.

We know this happens due to Monte-Carlo Estimation, which states that in principle, Monte Carlo methods can be used to solve any problem having a probabilistic interpretation. By the law of large numbers, integrals described by the expected value of some random variable can be approximated by taking the empirical mean (a.k.a. the sample mean) of independent samples of the variable. When the probability distribution of the variable is parametrized, we use a Markov chain Monte Carlo (MCMC) sampler. The central idea is to design a judicious Markov chain model with a prescribed stationary probability distribution. That is, in the limit, the samples being generated by the MCMC method will be samples from the desired (target) probability distribution.

In other words, for large samples of any random variable, the histogram will represent the pdf of that random variable. We can approximately fit the histogram inside the pdf curve by scaling the frequency by a multiplier of 1/M.

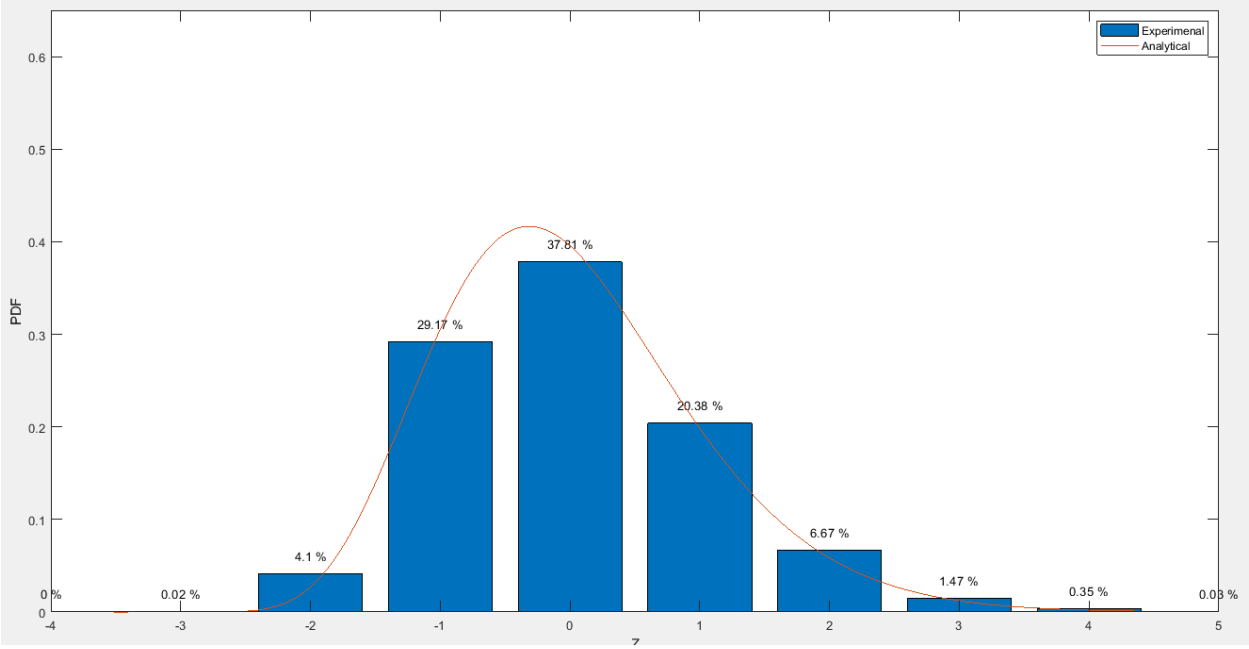
Now, when we plot the pdf of Z and compare it with the histogram obtained, we see that the histogram matches the pdf, when we scale the histogram to its frequency (i.e divide the histogram by M) we can do this as we calculated the pdf of Z, $= \frac{\gamma(N,\lambda(\sqrt{N}z+N))}{(N-1)!}\sqrt{N}$

When N=2, the plot is:



Here we observe that the pdf does not represent the normal distribution.

But When N=10, the plot becomes:



The above plot proves the central limit theorem, which states that the probability density function of Z will represent a Standard Normal Distribution for sufficiently large values of N. The new frequencies of the ‘M’ Z samples in the scaled histogram fitting inside the

pdf curve further provides additional evidence for the theorem. Here as N>=10, the pdf of Z becomes a standard Normal distribution.

Theoretical calculation-2

Given equation:

$$r_i = \sqrt{E}s_i + n_i$$

Where, s is a Bernoulli random variable with value as either -1 or 1 with an equal probability of p = 0.5 each.
And n is the additive Gaussian noise, with a standard normal distribution.
It is given that the machine calculates s as '-1' IF r<=0 and s as '1' IF r>0

But the actual value of s may not be equal to the value of s predicted by the machine. This can cause an error, for example when we know s= 1 but the machine calculates s=-1 instead, due the randomness of the value of n, changing the value of r. The total probability of an error occurring is:

$$P_{error} = P[(r > 0, s = -1) \cup (r \leq 0, s = 1)]$$
$$= (P[s = -1] \times P[r > 0 | s = -1]) + (P[s = 1] \times P[r \leq 0 | s = 1])$$

$$= \frac{1}{2}P[r > 0 | s = -1] + \frac{1}{2}P[r \leq 0 | s = 1]$$

$$= \frac{1}{2}P\left[\sqrt{E}(-1) + n > 0\right] + \frac{1}{2}P[\sqrt{E}(1) + n \leq 0]$$

$$= \frac{1}{2}P\left[n > \sqrt{E}\right] + \frac{1}{2}P[n \leq -\sqrt{E}]$$

Since n is additive white Gaussian noise, we know that the probability of n is symmetric around y axis as n has a standard gaussian pdf.

$$= \frac{1}{2}(2)P\left[n \leq -\sqrt{E}\right] = \phi(-\sqrt{E})$$

Therefore, $P_{error} = \phi(-\sqrt{E})$

Now, if we generate 'M' such samples of s and r, then according to monte-carlo estimation,

$$P_{error} \cong \frac{Number\ of\ n\ values\ \leq\ -\sqrt{E}}{M}$$

We can find the range of 'E' values where the probability of error is from 0.5 to 0.00001 from the following:

$$0.5 = \phi\left(-\sqrt{E}\right), 0.00001 = \phi(-\sqrt{E})$$

On calculating the 'Z' scores, we find that the value of E for error probability to be 0.5 is $E= 0$ and the value of E for the error probability to be 0.00001 is $E = 2$. Thus the range of E is from 0 to 2.

When n has a laplace (0,1) distribution,

The error probability is STILL $= P\left[n \leq -\sqrt{E}\right]$ as laplace distribution is also symmetric around y-axis

from the pdf of laplace, we can estimate the error probability as

$$\int_{-\infty}^{-\sqrt{E}} \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$

Since x<μ

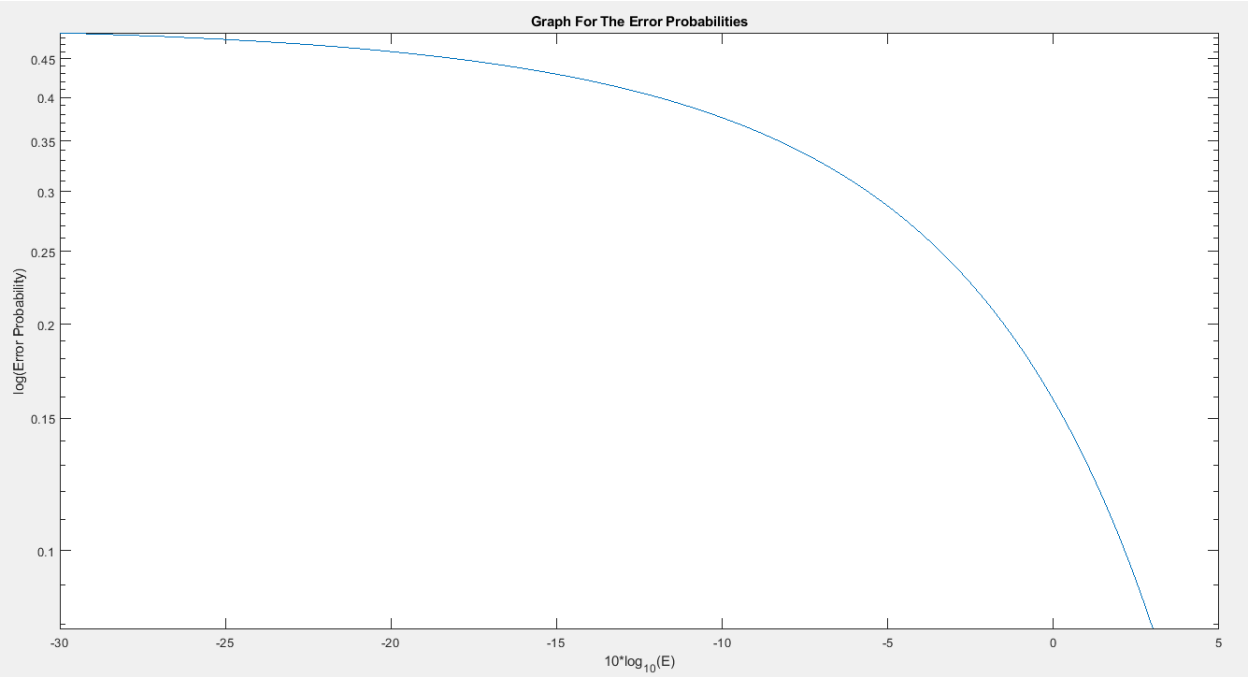
And variance = $\frac{1}{2b^2}$ => σ = $\sqrt{2}$ [given b = 1]

Thus the error probability of laplace noise after simplification becomes

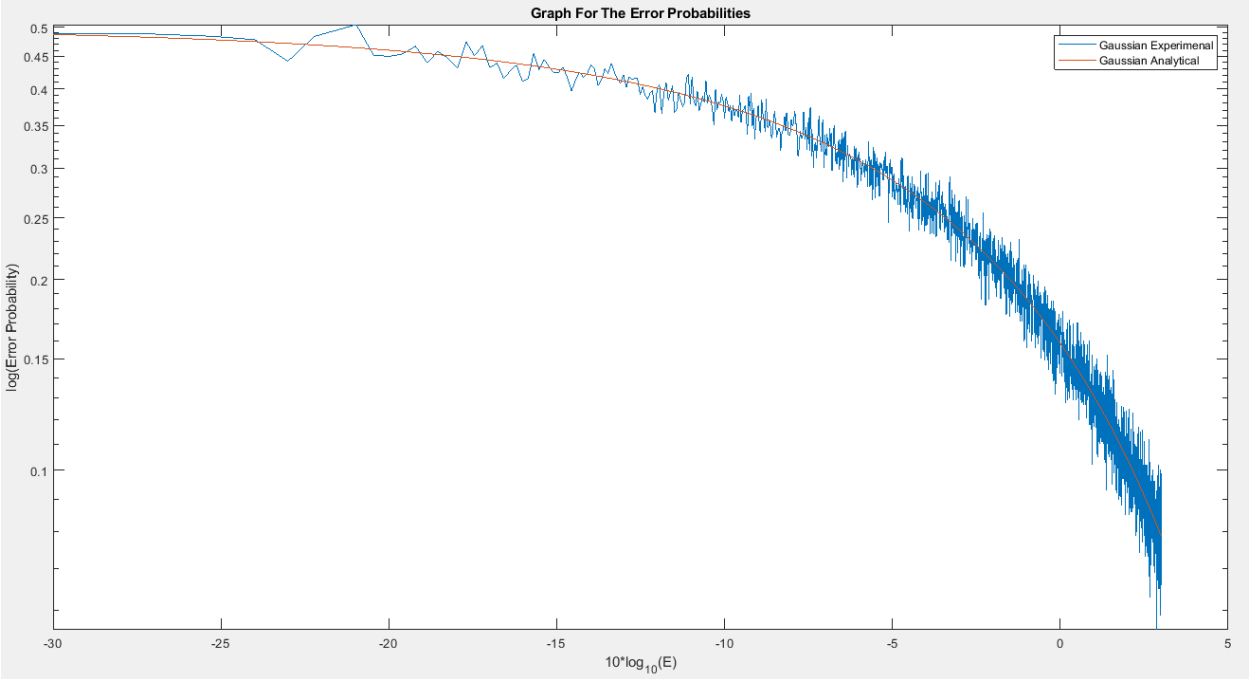
$$P_{error} = \frac{1}{2} e^{-\sqrt{E}}$$

We can now plot the theoretical error probability variation of laplace noise with E. For the monte-carlo simulation, we will create a new MATLAB function for generating 'm' laplace distributed random numbers, just like we generate a standard normal random number using the function randn(m,1)

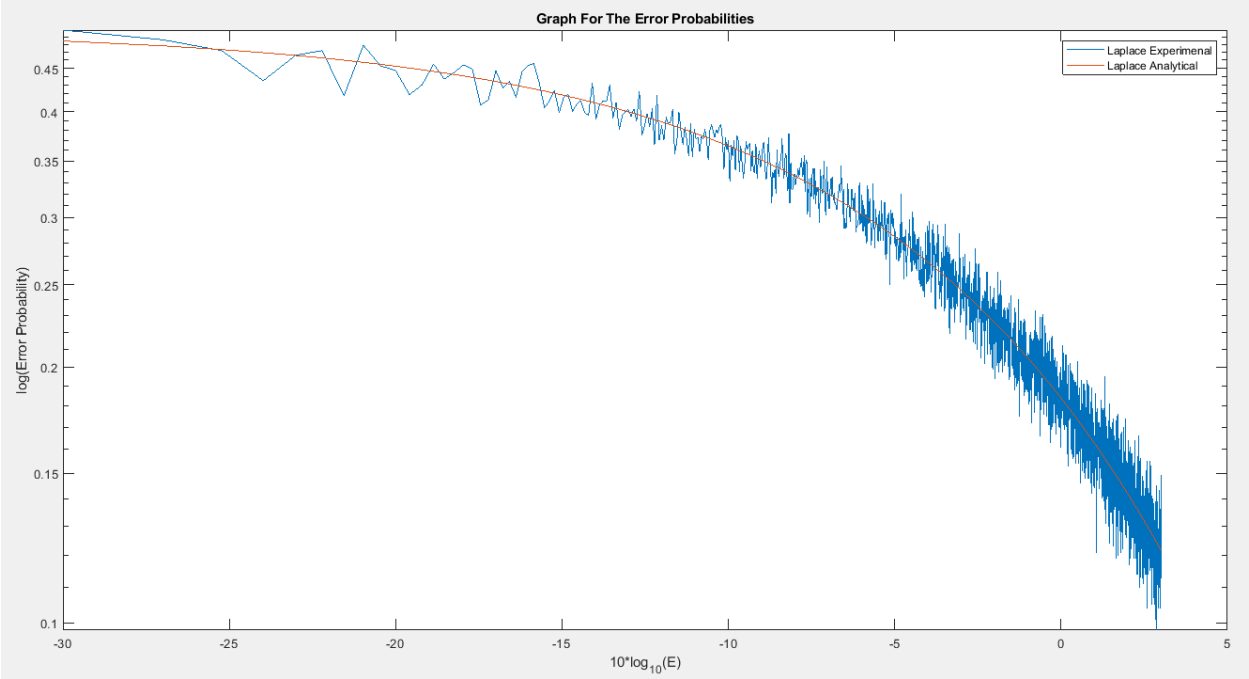
Results & Conclusion-2



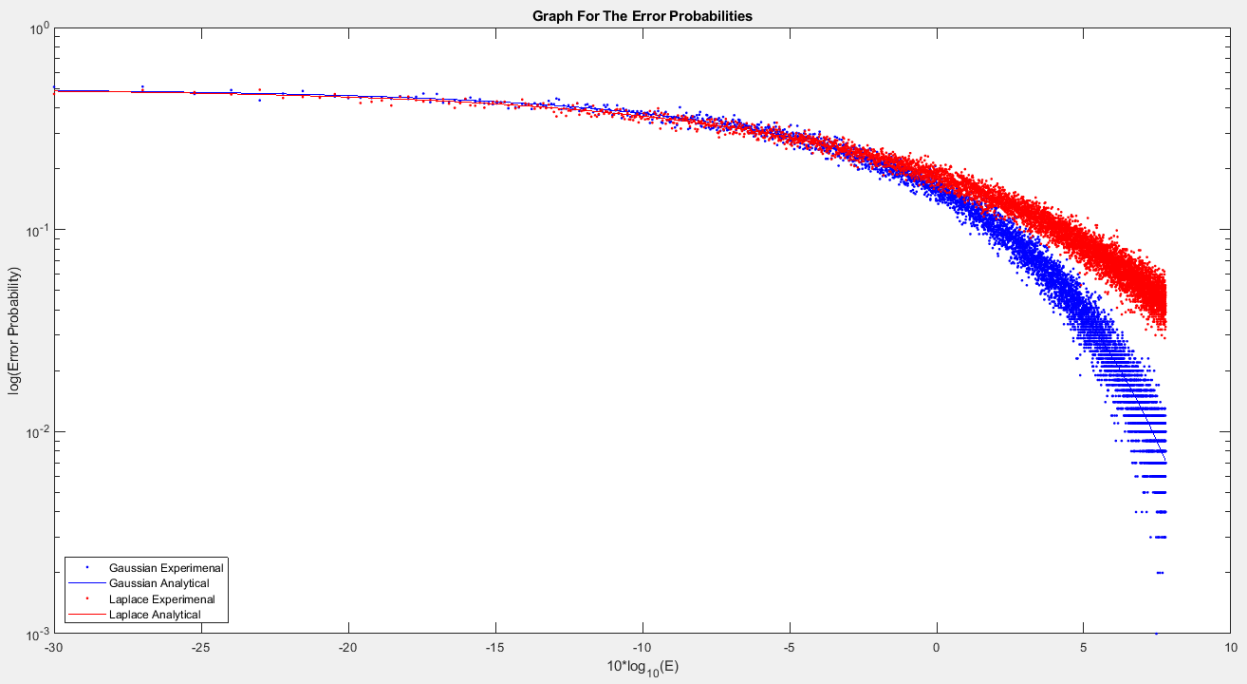
1) The above plot gives the range of the theoretical error probability for additive white Gaussian noise plotted as a semilog graph in y-axis with change in E shown in 10logE (as the scaled X-axis of E)



2) This above plot shows us how our monte carlo estimates compare to the theoretical probability curve. Based on the result we can say that they are matching.



3) We can see similar results for the laplace distribution as well in the above plot, as our monte carlo estimates match the theoretical error probability.



4) When we compare the error probability curves for Gaussian and Laplace, they are similar till E=2 but when E>2, the error probability

curve for Gaussian noise will undergo a much steeper drop compared to the error probability of the Laplace noise. This proves that the Gaussian noise has less probability of error as E increases. Thus the Gaussian noise is better than the Laplace distributed noise for higher values of E.

Code used for simulations

-Copy pasted directly from MATLAB, copy directly onto MATLAB to simulate

1. Generate N random variables that are exponentially distributed with parameter 1 and find the average (i.e., add the random variables and divide by N). Plot this average versus N:

```
N = 10000;
data = exprnd(1, N, 1); %function for generating an array of exponentially
distributed random numbers with parameter = 1
mu = zeros(size(data));
mu(1) = data(1); %Array for storing various averages of the sum of
exponentially distributed random variables
for i = 2:N
    mu(i) = (mu(i-1)*(i-1)+data(i))/(i); %Loop which makes mu store the
averages for increasing values of N from 1 to N
end
plot(mu)
```

2. Plot the PDF of the average of sum of exponential random variables for several values of N

```
narray=[1 2 5 10 30 100 1000]; %various values of n
for n = narray
    x=0:0.001:10;
    y=n^n/factorial(n-1) * x.^(n-1).*exp(-n*x); %plot the pdf of the average
of the sum of exponentially distributed random variables


    plot(x,y,'Linewidth',2)
    hold on %for plotting multiple graphs
end
hold off
legend(num2str(narray','n = %i'))
```

3. Draw a histogram of these M numbers using the hist command.

```
clear all
close all
N = 10;
```

```
M = 100000;
Z = zeros(1,M);
for i = 1:M
    data = exprnd(1, N, 1);
    Z(i) = (sum(data)-N)/sqrt(N); %the new
end

hist(Z)
```

4. when N = 2 and N = 10, Comparing the 'scaled' histogram with the actual distribution

```
clear all
close all
N = 10; %You can change the value of N=2 to get a different pdf

M = 10000 %you can increase the number of trials but is processor intensive
Z = zeros(1,M);
for i = 1:M
    data = exprnd(1, N, 1);
    Z(i) = (sum(data)-N)/sqrt(N);
end

[a, bin] =   hist(Z,-4:5); %extending the x axis from -4 to 5 so we can see
the samples and the curve
 bar(bin, a/M)
 for i = 1:length(bin)
     text(bin(i), a(i)/M+.02, num2str(a(i)/M*100, '%g %%'),
'HorizontalAlignment', 'center') %for showing the percentage samples of Z in
the histogram
 end
 x = linspace(min(bin), max(bin), 199); %we print the range of x values for
which the pdf is defined, as a function of the histogram frequencies from
minimum to maximum
%  x=0:0.001:3;
y=N*(sqrt(N)*x+N).^(N-1).*exp(-(sqrt(N)*x+N))/(sqrt(N)*factorial(N-1)); %this
is the pdf of Z
hold on
plot(x,y);
xlabel('Z')
ylabel('PDF')
legend('Experimenal','Analytical')
axis([-4 5 0 .65])
```

5. Error probability of the binary communication system as a function of the CDF of a Gaussian random variable and plotting it along with Monte Carlo estimates

```
clear all
close all
E = 0:0.001:2; %range of E for error probability to vary from 0.5 to 0.00001
```

```
p = zeros(size(E));
for i = 1:length(E)
    m=1000;
    n = randn(1,m); %generate m random numbers distributed standard gussianly
    t = binornd(1,0.5,1,m); % generate m Bernoulli random numbers with
probability = 0.5 each
    s = zeros(1,m);
    s(t==1) = 1;
    s(t~=1) = -1;

    r = sqrt(E(i)).*s + n;
    s1 = zeros(size(m));
    s1(r<=0) = -1;
    s1(r>0) = 1;
    wrong = sum(s1~=s);
    p(i)=wrong/m;
end


p2=normcdf(-sqrt(E)); %theoretical estimation of error probability due to
Gaussian noise
semilogy(10*log10(E),p,10*log10(E),p2);

title("Graph For The Error Probabilities");
xlabel('10*log_{10}(E)');
ylabel("log(Error Probability)");
legend('Gaussian Experimenal','Gaussian Analytical')
```

6) Compare the Monte Carlo simulation with the theoretical results of laplace(0,1) noise

NOTE: in order for this code to work, you must create a function file in the same folder where you access your code file. The code uses the function to generate laplace distributed random numbers. You can create a new function file provided in the code below or use the file provided along with this project

```
%%%%%%% Code starts below %%%%%%%
clear all
close all
E = 0:0.001:2;
p = zeros(size(E));
for i = 1:length(E)
   m=1000;
   n = laprnd(1,m,0,sqrt(2));  % sqrt(2) is the standard deviation of the laplace distribution
   t = binornd(1,0.5,1,m);
   s = zeros(1,m);
   s(t==1) = 1;
   s(t~=1) = -1;
```

```
    r = sqrt(E(i)).*s + n;
    s1 = zeros(size(m));
    s1(r<=0) = -1;
    s1(r>0) = 1;
    wrong = sum(s1~=s);
    p(i)=wrong/m;
end


p2=0.5*exp(-sqrt(E));
semilogy(10*log10(E),p,10*log10(E),p2);
%loglog(E,p,E,p2);
title("Graph For The Error Probabilities");
xlabel('10*log_{10}(E)');
ylabel("log(Error Probability)");
legend('Laplace Experimental','Laplace Analytical')

%%%%%% Laplace random variable Function%%%%%%

function y  = laprnd(m, n, mu, sigma)
%LAPRND generate i.i.d. laplacian random number drawn from laplacian distribution
%   with mean mu and standard deviation sigma.
%   mu      : mean
%   sigma   : standard deviation
%   [m, n]  : the dimension of y.
%   Default mu = 0, sigma = 1.
%   For more information, refer to
%   http://en.wikipedia.org./wiki/Laplace_distribution
%   Author  : Elvis Chen (bee33@sjtu.edu.cn)
%   Date    : 01/19/07
%Check inputs
if nargin < 2
    error('At least two inputs are required');
end
if nargin == 2
    mu = 0; sigma = 1;
end
if nargin == 3
    sigma = 1;
end
% Generate Laplacian noise
u = rand(m, n)-0.5;
b = sigma / sqrt(2);
y = mu - b * sign(u).* log(1- 2* abs(u));
end
```

7) Compare the error probabilities of Gaussian and Laplace distributions NOTE: you must need the same function file to run this code as well

```
clear all
close all
E = 0:0.001:6; %E max value is increased to see the comparison between
Gaussian and laplace
```

```
p = zeros(size(E));
for i = 1:length(E)
    m=1000;
    n = randn(1,m);
    t = binornd(1,0.5,1,m);
    s = zeros(1,m);
    s(t==1) = 1;
    s(t~=1) = -1;

    r = sqrt(E(i)).*s + n;
    s1 = zeros(size(m));
    s1(r<=0) = -1;
    s1(r>0) = 1;
    wrong = sum(s1~=s);
    p(i)=wrong/m;
end


p2=normcdf(-sqrt(E));
semilogy(10*log10(E),p,'b.',10*log10(E),p2,'b-');
%loglog(E,p,E,p2);
title("Graph For The Error Probabilities");
xlabel('10*log_{10}(E)');
ylabel("log(Error Probability)");


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p = zeros(size(E));
for i = 1:length(E)
    m=1000;
    n = laprnd(1,m,0,sqrt(2));
    t = binornd(1,0.5,1,m);
    s = zeros(1,m);
    s(t==1) = 1;
    s(t~=1) = -1;

    r = sqrt(E(i)).*s + n;
    s1 = zeros(size(m));
    s1(r<=0) = -1;
    s1(r>0) = 1;
    wrong = sum(s1~=s);
    p(i)=wrong/m;
end

hold on
p2=0.5*exp(-sqrt(E));
semilogy(10*log10(E),p,'r.',10*log10(E),p2,'r-');

title("Graph For The Error Probabilities");
xlabel('10*log_{10}(E)');
ylabel("log(Error Probability)");
legend('Gaussian Experimenal','Gaussian Analytical','Laplace
Experimenal','Laplace Analytical','Location','Southwest')
hold off
```

function y = laprnd(m, n, mu, sigma)
%LAPRND generate i.i.d. laplacian random number drawn from laplacian distribution

```
%   with mean mu and standard deviation sigma.
%   mu      : mean
%   sigma   : standard deviation
%   [m, n]  : the dimension of y.
%   Default mu = 0, sigma = 1.
%   For more information, refer to
%   http://en.wikipedia.org./wiki/Laplace_distribution
%   Author  : Elvis Chen (bee33@sjtu.edu.cn)
%   Date    : 01/19/07
%Check inputs
if nargin < 2
   error('At least two inputs are required');
end
if nargin == 2
   mu = 0; sigma = 1;
end
if nargin == 3
   sigma = 1;
end
% Generate Laplacian noise
u = rand(m, n)-0.5;
b = sigma / sqrt(2);
y = mu - b * sign(u).* log(1- 2* abs(u));
end
```

.

Bibliography

[1] Mathworks, "mathworks.com," 4 2 2020. [Online]. Available: https://www.mathworks.com/videos/working-with-arrays-in-matlab-69022.html.

[2] S. M, Intuitive Probability and Random processes using MATLAB, Rhode Island: Springer, 2006.

[3] J. Chen, "Investopedia," Investopedia, 1999. [Online]. Available: https://www.investopedia.com/terms/m/montecarlosimulation.asp. [Accessed 5 2 2020].