Matlab Project
EE 113
Probability

Jeffrey Jiang
ID: 904255069
Professor Roychowdhury

1. *Generation of binomial RVs.*

In Appendix A, I show the entire program written in order to generate the binomial random distribution pmf. I split the problem into two parts: generating the binomial random variable and answering the questions by generating a pmf. Generating the random variable was done by summing the total number of successes when generating Bernoulli random variables using the discrete random variable generation of the uniform random variable. Figure 1 shows a picture of the probability mass function generated empirically using this method of summing Bernoulli trials, where $n = 100$ and $p = 0.2$. As we see, this figure does not show exactly the same as an ideal binomial pmf, there are distinct similarities, as there is a very clear mode at around 20, which is the mean of the graph, and there is a symmetrical form.

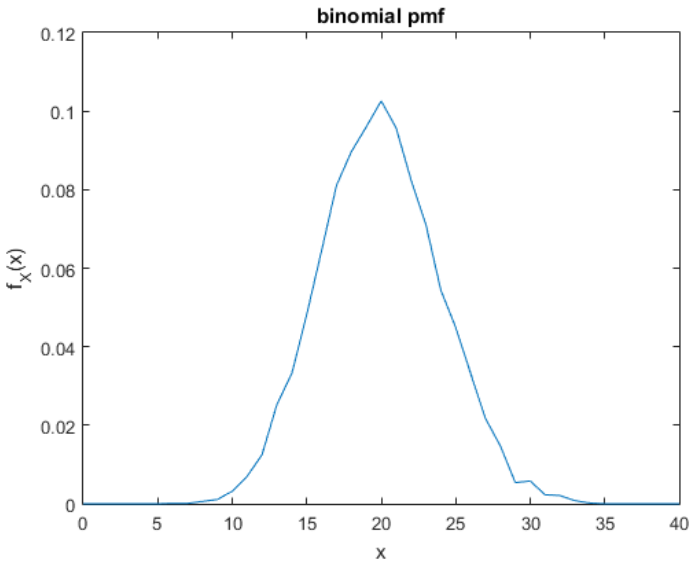

Figure 1: Empirically Generated Binomial Random Variable

In a general binomial random variable with n trials and probability p per trial, we have that the probability of any value X is given by Equation 1. Given this equation, we can determine the value of the maximum probability in the pmf of the model.

$$P[X = k] = \binom{n}{k} p^k (1-p)^{n-k} \tag{1}$$

Consider the change in change in probability from one value of k to another, $\Delta P = P[X = k+1] - P[X = k]$.

$$\Delta P = \binom{n}{k+1} p^{k+1}(1-p)^{n-k-1} - \binom{n}{k} p^k (1-p)^{n-k}$$

$$= \frac{n!}{(k+1)!\,(n-k)!} p^k (1-p)^{n-k-1}\big((n-k)p - (k+1)(1-p)\big)$$

$$= \frac{n!}{(k+1)!\,(n-k)!} p^k (1-p)^{n-k-1}\big(np - k - (1-p)\big) \tag{2}$$

Based on Equation 2, the maximum value of probability occurs at the first value of k such that $\Delta P < 0$. $\Delta P < 0 \Rightarrow np - k - (1-p) < 0 \Rightarrow k > np - (1-p)$. Since $0 < (1-p) < 1$, and k is an integer, the smallest value of k that has $\Delta P < 0$ is $\boxed{k = np}$.

In this case of the binomial random variable with $n = 100$ and $p = 0.2$, we expect the k with the maximum probability to be the average value of the binomial random variable: $k = 20$. This is also expected because we expect the binomial probability mass distribution to be a unimodal, symmetric probability distribution. Emperically in the case with the probability mass function plotted in Figure 1, we have that the point that has the highest probability is at $k = 21$, which provides strong evidence to the theoretical value that we calculated.

2. *Generation of Gaussian RV.*

Appendix B shows the MATLAB code that I used to complete this part of the project. For this part of the project, we want to generate the pdf and cdf of a Gaussian Random Variable through trial samples given by MATLAB. We generate the respective pdfs and cdfs with $t = 10, 10^4, 10^7$ samples respectively, and plot them in Figures 2, 3, and 4, where there is already super-imposed the ideal pdf on top of the sampled density function. Each of the functions also generate an output containing important information about the distribution. For this experiment, we will be finding the mean, variance, and fraction of values above certain thresholds of each randomly generated distribution.

For the case where there are 10 samples, we have Figure 2.a and the output shown just below, which shows a very strange density distribution, due to the low number of samples. However, the variance, unsurprisingly, still matches relatively closely to the expected mean and variance of the Gaussian random variables, with the mean slightly too high due to random variation alone.

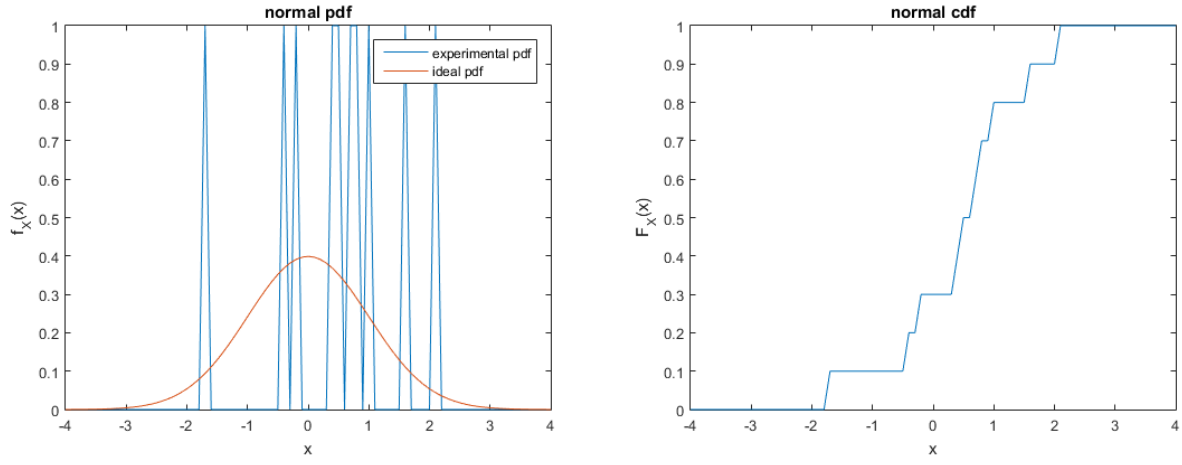

Figure 2: Probability Distributions of 10 Gaussian Distributed Samples

```
The mean is 0.477437
The variance is 1.128681
The fraction of values exceeding 1 is 0.200000
The fraction of values exceeding 2 is 0.100000
The fraction of values exceeding 3 is 0.000000
```

Another analysis that we have is to compare the fraction of values exceeding 1, 2, and 3. Based on the error function $Q(x)$, $Q(1) = 0.159$, $Q(2) = 0.0228$, and $Q(3) = 0.00135$. While the values we have are very rough (with 10 samples, the fractions occur at increments of 10), they do follow a general pattern of what fraction of values should exceed these values.

When we start adding more trials, we expect the sampled numbers to give us samples that more closely follows the density function that we have drawn. In addition, we expect to see all the values get closer to the ideal with the more samples we have. The pdf and cdf for 10000 Gaussian random variables is shown in Figure 3.

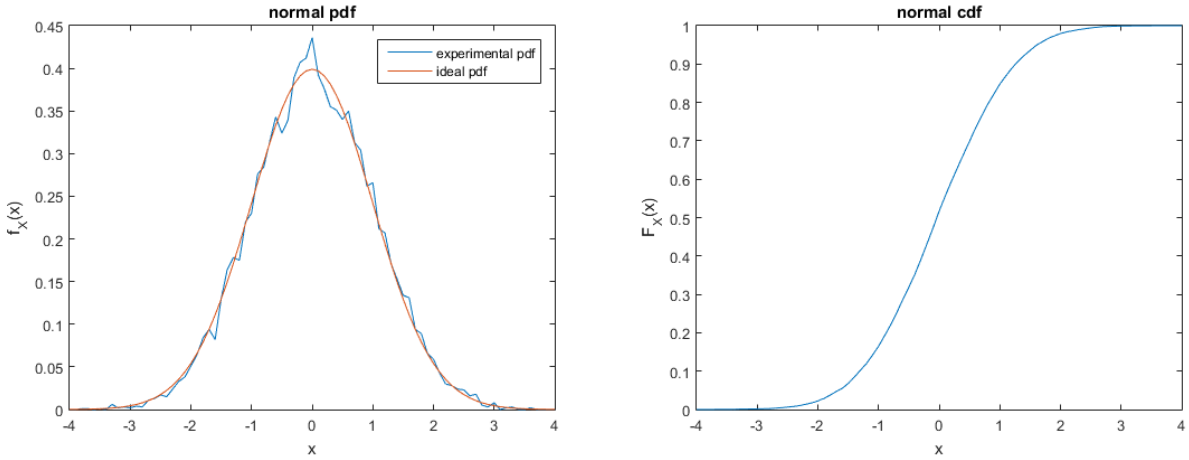

Figure 3: Probability Distributions of 10^4 Gaussian Distributed Samples

```
The mean is 0.022723
The variance is 0.993525
The fraction of values exceeding 1 is 0.152100
The fraction of values exceeding 2 is 0.020500
The fraction of values exceeding 3 is 0.000700
```

Just as expected, many of the values tended toward the actual distribution. The sampled pdf now fits much more closely to the ideal pdf, even though there are still variations at various locations probably due to some random chance that occurred. However, the cdf looks smooth and looks, in form, equivalent to the ideal cdf. In addition, most of the set values have converged toward the ideal as well. The mean is now much closer to 0 and the variance is practically 1 in this graph, which shows the convergence of the values. In addition, we can see that the fraction of values exceeding 1, 2, and 3 have normalized toward the values of the error function given by $Q(1) = 0.159$, $Q(2) = 0.0228$, and $Q(3) = 0.00135$.

Finally, we do the same experiment with $t = 10^7$ samples. We expect this to almost exactly follow the curve and have essentially the same outcome as the Gaussian distribution we expect. These plots are shown in Figure 4. As we see, the distributions are nearly perfect in that the ideal and the experimental pdfs differ only when observed very closely.

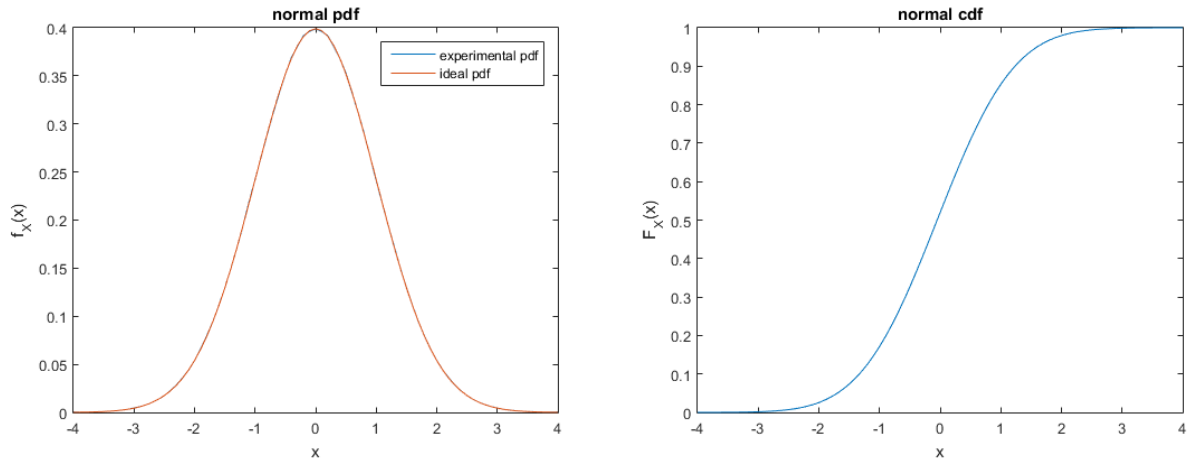

Figure 4: Probability Distributions of 10^7 Gaussian Distributed Samples

```
The mean is -0.000118
The variance is 1.000322
The fraction of values exceeding 1 is 0.146879
The fraction of values exceeding 2 is 0.020197
The fraction of values exceeding 3 is 0.001133
```

In addition, we can observe the additional values. At this point the difference between the ideal and experimental mean and variance is much less than 1%. The closeness clearly indicates that the random number generated do follow the normal distribution. However, something to note is that the fraction of values exceeding 1, 2, 3 is smaller than expected in all three cases. This is due to the fact that, despite the continuous look of the graph, the values we have are in fact discrete values. This means that $P[X \geq 1] \neq P[X > 1]$. In fact, the difference between the two values is relatively significant, being $f_X(1)dx \approx 0.025$, $f_X(2)dx \approx 0.0054$, $f_X(3)dx \approx 0.00045$. If we take this into consideration, the actual value of the error function lies within the range of the value dictated and the increment created by the discrete value. Therefore, we can assume that these samples do follow the distribution function.

3. *Simulation of Game of Craps.*

The game of craps uses multiple dice rolls in order to determine whether the player is a winner or a loser. The player wins the first round if he rolls a 7 or an 11, but loses if he rolls a 2, 3, or 12. Otherwise, whatever value he rolls becomes the target value, and the player continues to roll until he either gets the target, and wins, or rolls a 7, and loses.

Winning on the first roll is simply whether or not a 7 or an 11 is rolled. In Table 1, we note the probabilities of each of the possible rolls given two dice rolls. We note that rolling a 7 and rolling an 11 are completely disjoint events, so the expected probability of winning the first round is simply $\frac{8}{36} = \frac{2}{9} = 0.2222\ldots$. Using this information, we run a MATLAB program. The first section of Appendix C shows the program that we run to generate a certain number of dice rolls and places them into a matrix. Using this program, we get that the probability generated from 10000 rounds of dice rolls is 0.2269, which is relatively close to the actual theoretical value.

After determining the probability of winning the first round, we must consider the probability of winning given the first round has ended in not a win or a loss. In this case, we must set a target value and continue to deal with all other rolls until either the target or a seven is rolled. Assume the target is 5. In this case, define $p_5 = \frac{1}{9} = 0.111\ldots, p_7 = \frac{1}{6} = 0.1666\ldots, q_5 = 1 - p_5 - p_7 = \frac{13}{18} = 0.7222\ldots$. Using these values, we can determine the probability of winning. Clearly, if we win on the k-th roll after the first roll, that means that there were $k-1$ rolls that were neither 5 nor 7. Therefore, the probability of winning on the $(k+)1$-th roll (including the first roll now) is show in Equation 3. Then, the probability of winning given the first roll is 5 is just a sum of these values from 1 to infinity, shown in Equation 4.

$Roll$	$Probability$
2	$\frac{1}{36}$
3	$\frac{2}{36} = \frac{1}{18}$
4	$\frac{3}{36} = \frac{1}{12}$
5	$\frac{4}{36} = \frac{1}{9}$
6	$\frac{5}{36}$
7	$\frac{6}{36} = \frac{1}{6}$
8	$\frac{5}{36}$
9	$\frac{4}{36} = \frac{1}{9}$
10	$\frac{3}{36} = \frac{1}{12}$
11	$\frac{2}{36} = \frac{1}{18}$
12	$\frac{1}{36}$

Table 1: PMF of Two Dice Rolls

$$P[win\ on\ (k+1)th\ roll | first\ roll = 5] = q_5^{k-1} p_5 \quad (3)$$

$$P[win | first\ roll = 5] = \sum_{k=1}^{\infty} q_5^{k-1} p_5 = \frac{p_5}{1-q_5} = \frac{p_5}{p_5 + p_7} = \boxed{0.4} \quad (4)$$

Therefore, we see that the probability of winning after the first roll is a 5 is 0.4. Notice that Equation 4 can also be written in the following manner: $\frac{p_5}{p_5+p_7}$, which removes the need to have another variable q_5. This property also extends to every other non-winning, non-losing first run, except with different value for p_5.

We will use MATLAB to determine whether or not this theoretical probability holds true when the dice is rolled many times. We created another function in Appendix C that contains whether or not a player wins, given a specific target value. In 10000 games, we get that the probability of winning, given that the target value has been set at 5 is 0.3972, which is less than 1% less than the theoretical.

Using this information, we can determine the final probability of a full game of craps. We use total probability law to get each conditional probability.

$$\begin{aligned} P[win] = {} & P[roll = 7\ or\ 11]P[win | roll = 7\ or\ 11] + P[roll = 2,3,12]P[win | roll = 2,3,12] \\ & + P[roll = 4]P[win | roll = 4] + P[roll = 5]P[win | roll = 5] \\ & + P[roll = 6]P[win | roll = 6] + P[roll = 8]P[win | roll = 8] \\ & + P[roll = 9]P[win | roll = 9] + P[roll = 10]P[win | roll = 10] \end{aligned}$$

$$= \frac{2}{9} \times 1 + 0 + \frac{1}{12} \times \frac{\frac{1}{12}}{\frac{1}{12} + \frac{1}{6}} + \frac{1}{9} \times \frac{\frac{1}{9}}{\frac{1}{9} + \frac{1}{6}} + \frac{5}{36} \times \frac{\frac{5}{36}}{\frac{5}{36} + \frac{1}{6}} + \frac{5}{36} \times \frac{\frac{5}{36}}{\frac{5}{36} + \frac{1}{6}} + \frac{1}{9} \times \frac{\frac{1}{9}}{\frac{1}{9} + \frac{1}{6}} + \frac{1}{12} \times \frac{\frac{1}{12}}{\frac{1}{12} + \frac{1}{6}}$$

$$= \frac{2}{9} + \frac{1}{36} + \frac{2}{45} + \frac{25}{396} + \frac{25}{396} + \frac{2}{45} + \frac{1}{36} = \frac{244}{495} = 0.492929\ldots \approx \boxed{49.3\%}$$

Finally, one last part of Appendix C contains the entire craps game, which essentially just combines the two components. This allows us to determine that for 10000 runs through the game of craps, the probability of winning was 0.4959, which is less than 1% different from the expected value of winning. Therefore, this provides strong evidence for the theoretical result we calculated. Unsurprisingly, craps is a well-designed game where the game is only slightly tipped in the favor of the house.

4. *Simulation of binary transmission systems.*

Consider a binary transmission system. In this transmission system, we have that the transmitted value of the transmission system only belongs to the set $\{-2, 2\}$, which are sent with approximately equal probability. However, along the way, the values are introduced to noise that take the form of a Laplacian distribution. The following discussion deals with this problem.

The first thing about this problem is to generate the Laplacian distribution. We want to use the transformation method described, using the fact that the cdf of any random variable is non-decreasing to mean that the inverse function is defined. We know that $f_X(x) = \frac{\alpha}{2}e^{-\alpha|x|}$. We can take the integral (splitting it for when x is positive and negative) to find the cdf function.

$$F_X(x) = \begin{cases} \int_{-\infty}^{x} \frac{\alpha}{2}e^{\alpha x}dt & for\ x \le 0 \\ \int_{-\infty}^{0} \frac{\alpha}{2}e^{\alpha x}dt + \int_{0}^{x} \frac{\alpha}{2}e^{-\alpha x}dt & for\ x > 0 \end{cases} = \begin{cases} \frac{1}{2}e^{\alpha x} & for\ x \le 0 \\ 1 - \frac{1}{2}e^{-\alpha x} & for\ x > 0 \end{cases} \quad (5)$$

$$F_X^{-1}(x) = \begin{cases} \frac{1}{\alpha}\ln(2x) & for\ x \le 0 \\ -\frac{1}{\alpha}\ln\big(2(1-x)\big) & for\ x > 0 \end{cases} \quad (6)$$

As a result, we can get the inverse cdf, which is given by Equation 6. We can use this value to generate Laplacian random variables. This is shown in Appendix D. As a result of this code, we can generate graphs for the pdf and cdf of the Laplacian distribution.

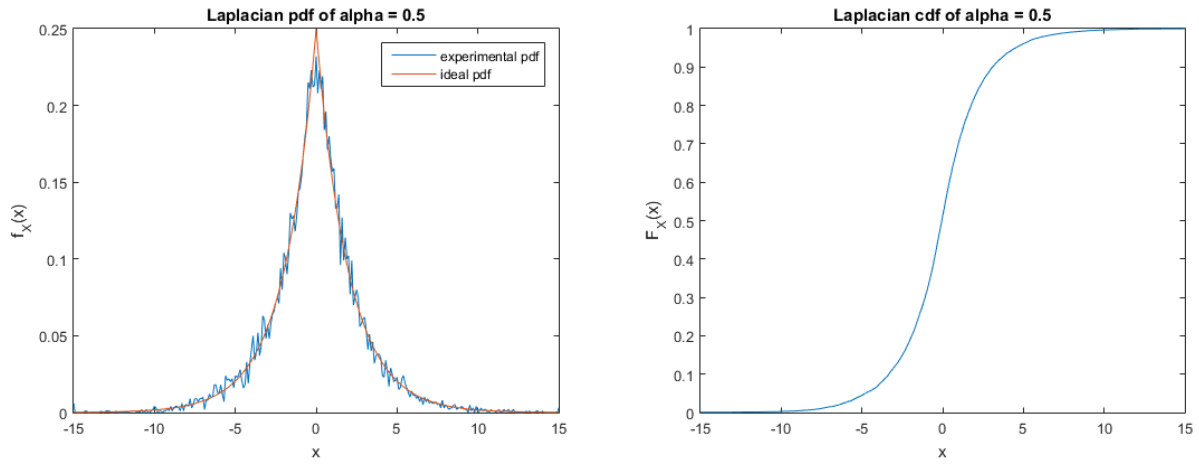

Figure 5: Laplacian pdf and cdf of $\alpha = 0.5$

First, we want to make sure that the generated Laplacian distribution looks about correct. Figure 5 shows the Laplacian pdf and cdf when $\alpha = 0.5$, while Figure 6 shows the Laplacian pdf and cdf when $\alpha = 2$, with the ideal pdf also super-imposed on the sampled pdf. In this project simulation, we use 10000 samples.

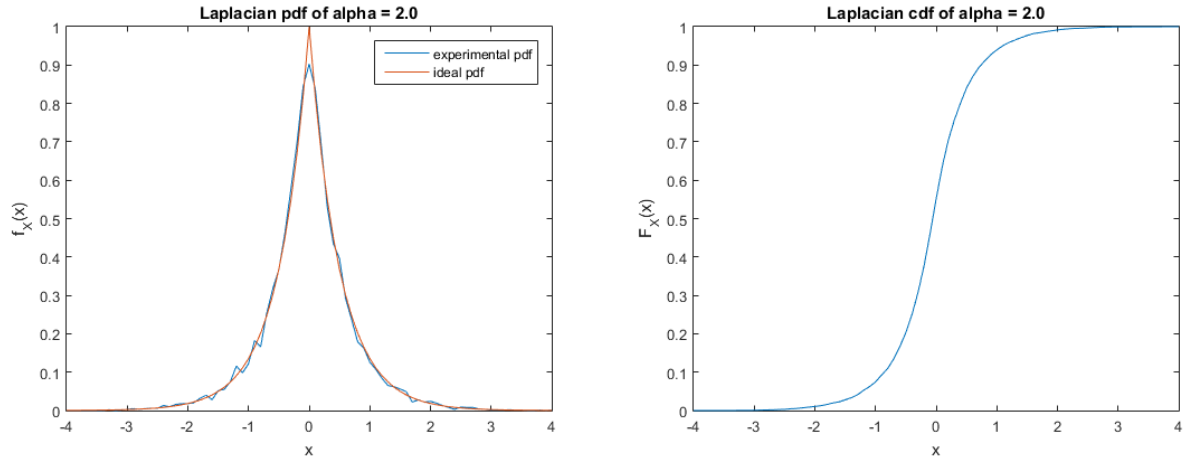

Figure 6: Laplacian pdf and cdf of $\alpha = 2$

The ideal pdf is placed in these graphs just to ensure that the transformation that we did worked as intended. Based on the views on these graphs, it looks like the transformation did work as intended as the pdf fits pretty closely to the ideal pdf. Therefore, we can conclude that we were able to properly generate a Laplacian distributed variable.

Now, based on this Laplacian distribution, we are able to simulate the signal property at hand. We create $Y = X + N$, where N is Laplacian distributed. For this part, we want to compute the accurate value of α required in order to make sure that we have $SNR(dB) = 0dB$. By the definition of the Laplacian random variable, it has a variance of $\sigma^2 = \frac{2}{\alpha^2}$. Therefore, in order to make the signal-noise ratio $0dB = 10\log_{10}\left(\frac{1}{\sigma^2}\right) = 10\log_{10}\left(\frac{\alpha^2}{2}\right)$. This result implies that $\frac{\alpha^2}{2} = 1$ resulting in $\alpha = \sqrt{2}$. We use this value to compute the received signal Y.

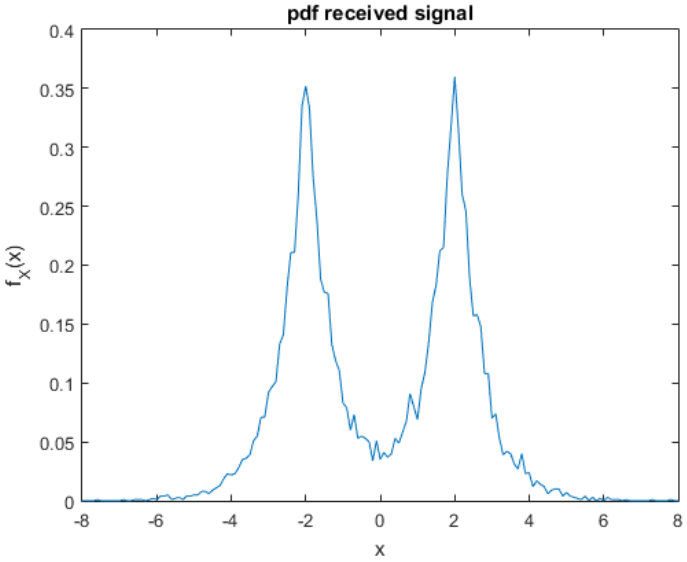

Figure 7: PDF of a Generated Signal After Noise

Next, we want to calculate the probability of error. By the total probability theorem, we have Equation 7.

$$P[error] = P[X = 2]P[N < -2|X = 2] + P[X = -2]P[N > 2|X = -2] \quad (7)$$

This is the case because the probabilities indicated for N are the equivalent to saying the probability of error given their condition. Since $P[X = 2] = P[X = -2] = 0.5$ and N is symmetrical, and N is independent of X, $P[error] = 2P[N < -2]$. By definition of the cdf, this means that we can get Equation 8 (theoretically).

$$P[error] = 2F_X(-2) \quad (8)$$

Just plugging this into Equation 5, we get that $P[error] = 0.02955$ is the theoretical value for the amount of error. For this part of the project, however, in order to empirically collect data on the amount of error, we cannot use this simplified formula. We still need to count the total number of samples that reach out of their bounds, as described in Equation 7. In the simulation, the probability of error turns out to be 0.0299, which is very close to the value of the theoretical error.

5. *Central Limit Theorem.*

Lastly, we want to simulate the Central Limit Theorem. This simulation deals with summing a certain number of uniformly distributed random variables in the range $(1,6)$. This can be done quite simply in MATLAB. Appendix E shows the generation of the pdf and cdf of the sum of uniformly distributed random variables, using 10000 sums generated from some number n uniformly distributed random variables.

We went over in lecture what the mean and variance of a sum of random variables are, but here I will go over them again analytically to ensure their accuracy. Consider a sequence of iid random variables: $X_1, X_2, \ldots$ By the definition of iid random variables, $E[X_1] = E[X_2] = \cdots = \mu$. $\sigma_1^2 = \sigma_2^2 = \cdots = \sigma^2$. Because the random variables are independent, Then, we have Equation 9 to compute the average value of the sum. This property follows due only due to the fact that the expected value is a linear property.

$$E[S_n] = E[X_1 + X_2 + \cdots + X_n]$$

$$= E[X_1] + E[X_2] + \cdots + E[X_n] = \boxed{n\mu} \quad (9)$$

Using the average value for the sum, we can also derive the variance of the sum. Equation 10 shows the entire derivation. This property uses both the independent and identical properties of the random variables, the independent to split the product of the expected values and the identical to evaluate the sum at the very end.

$$\sigma_{S_n}^2 = E[S_n^2] - (E[S_n])^2$$

$$= E[(X_1^2 + X_2^2 + \cdots + X_n^2 + 2X_1X_2 + 2X_1X_3 + \cdots + 2X_{n-1}X_n] - (n\mu)^2$$

$$= \sum_{i=1}^{n} E[X_i^2] + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E[X_i X_j] - n^2\mu^2 \xrightarrow{indep.} \sum_{i=1}^{n} E[X_i^2] + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E[X_i]\,E[X_j] - n^2\mu^2$$

$$= \sum_{i=1}^{n} E[X_i^2] + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \mu^2 - n^2\mu^2 = \sum_{i=1}^{n} E[X_i^2] + n(n-1)\mu^2 - n^2\mu^2$$

$$= \sum_{i=1}^{n} E[X_i^2] - n\mu^2 = \sum_{i=1}^{n}[E[X_i^2] - \mu^2] = \sum_{i=1}^{n} \sigma_{X_i}^2 \xrightarrow{ident.} \boxed{n\sigma^2} \qquad (10)$$

As a result, we can obtain the theoretical Gaussian pdf and cdf by using the new mean and variance to compute them and compare them to the graphs of the sum of random variables.

Figure 8 shows the base case: a single uniformly distributed random variable, without a sum. This should look nothing like a Gaussian distribution because it is not a Gaussian distribution. Yet, we can still compare it to see the huge differences.

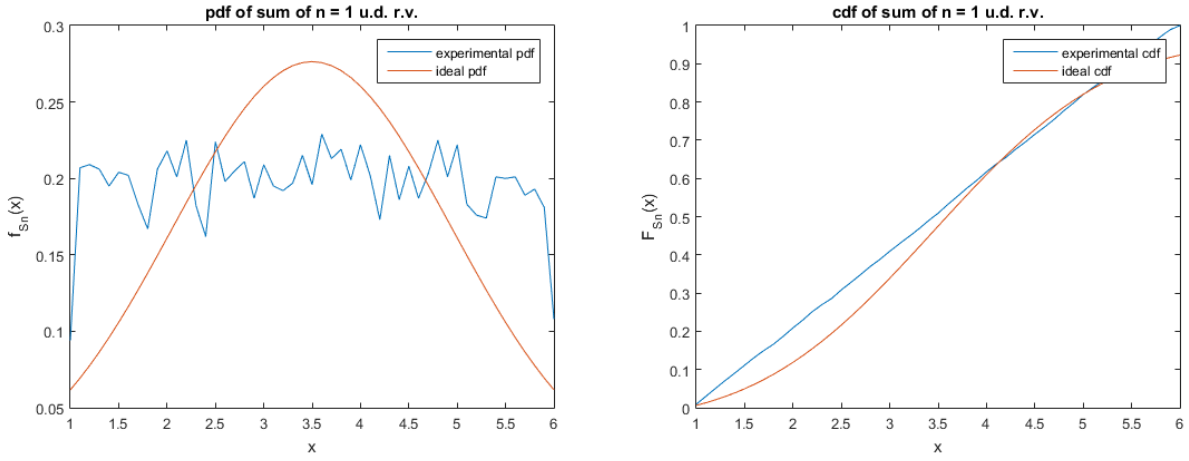

Figure 8: pdf and cdf of a Single Uniformly Distributed Random Variable using 10000 samples.

As expected, the pdf shows glaring differences between the two pdfs. However, notice that the cdf is already starting to take on a similar shape, as expected, from the cdf. More interesting things start to happen as n increases, however. The next number we choose is $n = 5$, shown in Figure 9. Notice that the form that we were expecting, with the Gaussian curve, is already starting to appear, as the general form of the samples has already begun to track the form of the Gaussian distribution. We note that the variations surrounding the Gaussian distribution is fairly large in this case. However, at this moment, the cdf of the sums has already aligned with the cdf of the ideal Gaussian distribution to a point where it is indistinguishable. This is a surprising result, as even a sum of 5 uniformly distributed random variables can create a random variable that is surprisingly close to the Gaussian distribution, especially when only considering the cdf.

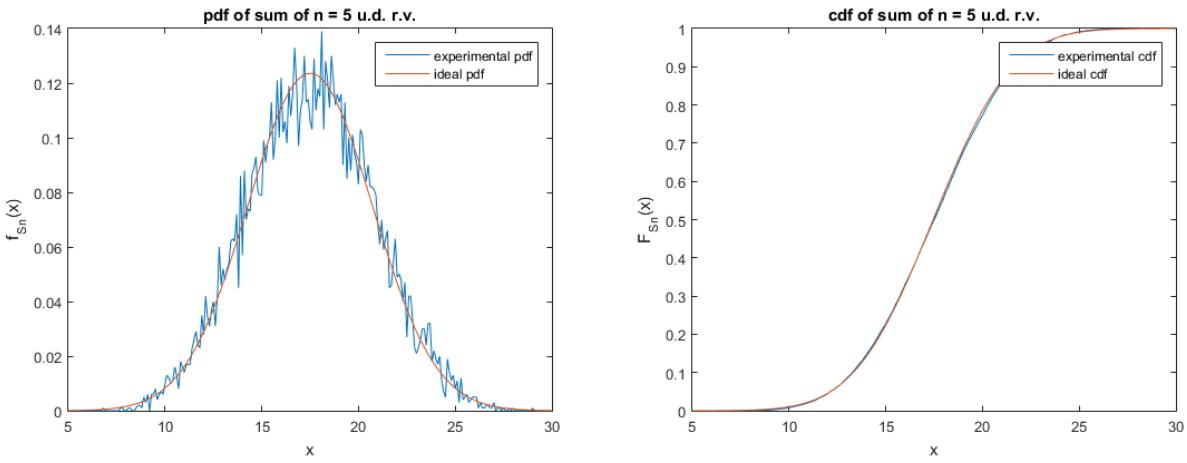

Figure 9: pdf and cdf of the Sum of 5 Uniformly Distributed Random Variables using 10000 samples.

Likewise, as the sum increases to 10 and 50 random variables, the variations shrink, somewhat, and the cdf gets closer to the ideal cdf. Figure 10 display this information and show that although Central Limit Theorem is only true as an iid random variable is summed to infinity, the limit actually approaches the Gaussian distribution very quickly, as 5 or 10 random variables is already enough to possibly use the Gaussian distribution as a model for considering the possible sum.

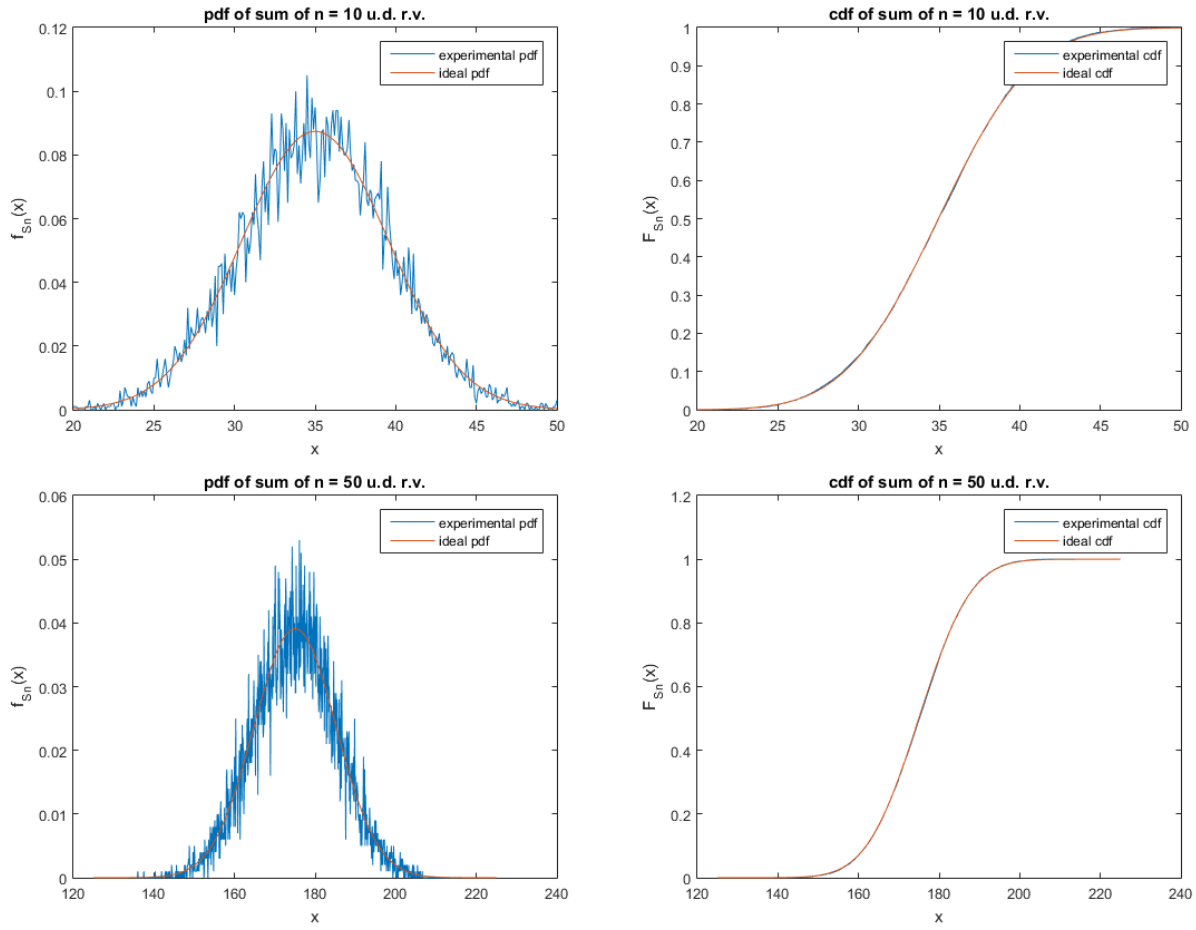

Figure 10: pdf and cdf of the Sum of 10 and 50 Uniformly Distributed Random Variables using 10000 samples.

Appendix A

Generating a Binomial Random Variable

```
function s = binom_rv(n, p, k)
s = zeros(1, k);
for i = 1:k
    s(i) = sum(floor(p + rand(1,n))); %sums n Bernoulli RVs with prob p.
end
```

Completing Tasks of Project 1

```
%generation of a discrete Bernoulli trial

%generation of a binomial random distribution
step = 1;
prob = 0.2;
n_trials = 100;
n_samples = 10000;
binom = binom_rv(n_trials, prob, n_samples);
s = 0:step:40;

y = hist(binom, s);

%plotting and labeling the pmf.
density = y / n_samples / step;
plot(s,density);
title('binomial pmf');
xlabel('x');
ylabel('f_X(x)');

%finding the k of maximum probability.
% = find the k of the maximum value.
[k,ki] = max(y);
disp_msg = sprintf('The k with maximum probability is %d', ki);
disp(disp_msg);
```

Appendix B

Completing Tasks of Project 2

```
%generation of Gaussian
step = 0.1;
n_samples = 1e7; %input value for t here.
x = randn(1,n_samples);
s = -4:step:4;

%determine mean and variance.
exp_mean = sprintf('The mean is %f', mean(x));
exp_vari = sprintf('The variance is %f', var(x));
disp(exp_mean);
disp(exp_vari);

%create the plot of the pdf
y = hist(x,s);
normal_pdf = y / n_samples / step;
figure;
plot(s, normal_pdf);
exp_label = 'experimental pdf';
hold on
ideal_pdf = exp(-s.^2/2)/sqrt(2*pi);
ideal_label = 'ideal pdf';
plot(s, ideal_pdf);
title('normal pdf');
xlabel('x');
ylabel('f_X(x)');
legend(exp_label, ideal_label);

%create the plot of the cdf
size_s = size(s, 2);
normal_cdf = zeros(1, size_s);
for i = 1:size_s
    normal_cdf(i) = sum(normal_pdf(1:i)) * step;
end
figure;
plot(s, normal_cdf);
title('normal cdf');
xlabel('x');
ylabel('F_X(x)');

%determine the fraction of values exceeding 1, 2, 3.
q1 = 1 - normal_cdf(5/step + 1);
q2 = 1 - normal_cdf(6/step + 1);
q3 = 1 - normal_cdf(7/step + 1);

msg1 = sprintf('The fraction of values exceeding 1 is %f', q1);
msg2 = sprintf('The fraction of values exceeding 2 is %f', q2);
msg3 = sprintf('The fraction of values exceeding 3 is %f', q3);
disp(msg1);
disp(msg2);
disp(msg3);
```

Appendix C

Generating a Dice Roll

```
function rolls = dice_roll(n_rolls, n_times)
rolls = zeros(1, n_times);
for i = 1:n_times
    rolls(i) = sum(floor(1 + 6 * rand(1, n_rolls)));
end
```

Determining Win After First Roll (if not completed)

```
function win = craps_ctd(target, n_repeats)
win = 0;
for i = 1:n_repeats
    roll = dice_roll(2, 1);
    while (roll ~= target) && (roll ~= 7)
        roll = dice_roll(2,1);
    end
    if roll == target
        win = win + 1;
    end
end
```

Playing a Simulated Game of Craps Many Times

```
function win = craps(n_repeats)
%result = zero(1,n_repeats);
win = 0;
for i = 1:n_repeats
    roll = dice_roll(2, 1);
    if (roll == 7) || (roll == 11)
        win = win + 1;
    elseif (roll == 2) || (roll == 3) || (roll == 12)
        %disp('You lose!');
    else
        target = roll;
        result = craps_ctd(target, 1);
        win = win + result;
    end
end
end
```

Completing Tasks of Project 3

```
% generate the probabiltiy of winning the first time
n_times = 10000;
rolls = dice_roll(2, n_times);
win_round_1 = (sum(rolls == 7) + sum(rolls == 11)) / n_times;
prob_win_1 = sprintf('The prob. of winning round 1 is %f', win_round_1);
disp(prob_win_1);

% generate the probability of winning given first roll is 5.
prob_win_5 = craps_ctd(5, n_times) / n_times;
win_5_msg = sprintf('The winning prob. w/ 1st roll 5 is %f', prob_win_5);
disp(win_5_msg);

%generate the probability of winning the game of craps.
prob_win = craps(n_times) / n_times;
win_msg = sprintf('The probability of winning is %f', prob_win);
disp(win_msg);
```

Appendix D

Generate Laplacian Random Variable

```
function X = Laplacian_distribution(alpha, n_trials)
X = zeros(1, n_trials);
uniform = rand(1,n_trials);

%the cdf of a laplacian is:
% 1/2 exp(ax) for x < 0. 1 - 1/2 exp(-ax) for x > 0.
% note that when x = 0, cdf = 1/2.

for i = 1:n_trials
    if ( uniform(i) <= 0.5 )
        X(i) = log(2*uniform(i))/alpha;
    else
        X(i) = -log(2*(1-uniform(i)))/alpha;
    end
end
end
```

Completing Tasks of Project 4

```
% Project Part 4
% part a:

%alpha = 2.
step = 0.1;
alpha = 2;
n_samples = 10000;
x = Laplacian_distribution(alpha, n_samples);
s = -4:step:4;

%getting and plotting pdf.
y = hist(x, s);
laplace_pdf = y / n_samples / step;
ideal = alpha/2 * exp(-alpha*abs(s));
exp_label = 'experimental pdf';
ideal_label = 'ideal pdf';

figure;
plot(s,laplace_pdf);
hold on;
plot(s, ideal);
title(sprintf('Laplacian pdf of alpha = %0.1f', alpha));
xlabel('x');
ylabel('f_X(x)');
legend(exp_label, ideal_label);

%getting and plotting cdf.
size_s = size(s, 2);
laplace_cdf = zeros(1, size_s);
ideal_cdf = zeros(1, size_s);
for i = 1:size_s
    laplace_cdf(i) = sum(laplace_pdf(1:i)) * step;
```

```
end
figure;
plot(s, laplace_cdf);
title(sprintf('Laplacian cdf of alpha = %0.1f', alpha));
xlabel('x');
ylabel('F_X(x)');

%part b: generate the empirical transmission values.
alpha = sqrt(2);
n_samples = 10000;
s = -8:step:8;

%generate the signal.
X = 4 * floor(0.5 + rand(1, n_samples)) - 2; %signal
N = Laplacian_distribution(alpha, n_samples); %noise
Y = X + N; %recieved signal

y = hist(Y, s);
received_pdf = y / n_samples / step;
figure;
plot(s, received_pdf);
title('pdf received signal');
xlabel('x');
ylabel('f_X(x)');

%emperically compute the error probability:
incorrectly_left = 0;
incorrectly_right = 0;
for i = 1:n_samples
    if ( X(i) > 0 && N(i) < -2 )
        incorrectly_left = incorrectly_left + 1;
    end
    if ( X(i) < 0 && N(i) > 2 )
        incorrectly_right = incorrectly_right + 1;
    end
end
n_errors = incorrectly_left + incorrectly_right;
p_errors = n_errors / n_samples;
err_msg = sprintf('The probability of error is %f', p_errors);
disp(err_msg);
```

Appendix E

Completing Tasks of Project 5

```
step = 0.1;
n_repeats = 10000;

n = 1; % TODO change to check n
s = 1:step:6; % TODO change to fit values.
s_n = zeros(1, n_repeats);
for i = 1:n_repeats
    x = 1 + 5*rand(1, n); %uniform random var (1,6)
    s_n(i) = sum(x); %an instance of s_n
end

y = hist(s_n, s);
pdf = y / n_repeats / step;
mean = n*7/2;
var = n*25/12;
ideal = exp(-(s-mean).^2/(2*var))/(sqrt(2*pi*var));
figure;
plot(s,pdf);
hold on
plot(s,ideal);
exp_label = 'experimental pdf';
ideal_label = 'ideal pdf';
title(sprintf('pdf of sum of n = %d u.d. r.v.', n));
xlabel('x');
ylabel('f_S_n(x)');
legend(exp_label, ideal_label);

%create the plot of the cdf
size_s = size(s, 2);
cdf = zeros(1, size_s);
ideal_cdf = zeros(1, size_s);
for i = 1:size_s
    cdf(i) = sum(pdf(1:i)) * step;
    ideal_cdf(i) = sum(ideal(1:i)) * step;
end
figure;
plot(s, cdf);
hold on
plot(s, ideal_cdf);
exp_label = 'experimental cdf';
ideal_label = 'ideal cdf';
title(sprintf('cdf of sum of n = %d u.d. r.v.', n));
xlabel('x');
ylabel('F_S_n(x)');
legend(exp_label, ideal_label);
```