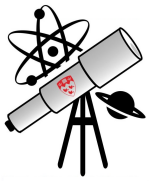
PHYS 339 LAB



# Modeling Random Distributions with Monte-Carlo

Yael Demers (260904387), Aidan Lewandowski (260910544), Jessie Zhao (260890021)

McGill University Department of Physics

January 23, 2023

**Abstract**

In this lab, we generate random numbers using Monte-Carlo Simulations with a linear probability distribution and Gaussian distribution, whereas Gaussian distribution is composed of uniform and exponential distribution. 70% of our data fit the linear model, and for the other set of data, 95.7% of the data lies within two standard deviations from the mean and therefore fit the Gaussian distribution. Our results also show the central limit theorem, with both mean and standard deviation converging to their true value for increasing sample size.

# 1 Introduction

The Monte Carlo method uses random numbers to simulate outcomes; these objects could arise 'naturally' as part of modeling a real-life system [1]. The use of Monte Carlo methods allows us to repeat the experiments for a significantly large amount of iterations and to gather the analytical data of interest. In this lab, we generate random numbers with linear, uniform, exponential, and Gaussian distributions and obtain the associated probability density and cumulative distribution functions respectively.

A probability density function p(x) (PDF) describes the probability distribution for random variables. The linear probability density function has the shape of a linear function with certain limits. The limits must be carefully defined since a linear function is continuous from negative infinity to positive infinity. Still, we cannot have a probability less than 0 or greater than 1.

Uniform distribution has a constant probability distribution function between two bounding parameters, i.e. a horizontal straight line within the limits. Uniform distribution plays a vital role in generating random variables. Most random variables are generated on the (0,1) interval, and we can apply a transformation to uniform random variables to get other distributions.

The exponential distribution is a function containing an exponential term widely used in real life. It has the property of being memoryless if it is of non-negative real numbers. Every point on the graph can be a new beginning point of the same distribution. Every instant is like the beginning of a new random period, which has the same distribution regardless of how much time has already elapsed.

Gaussian distribution also has an exponential term but a different exponent order. One of the reasons that Gaussian distribution is important is the central limit theorem. The central limit theorem states the convergence of probability distributions of functions of an increasing number of one- or multi-dimensional random variables to a normal distribution or

related distributions. [2].

We also need to compute the cumulative distribution functions from p(x). A cumulative distribution function c(x) (CDF) is the probability that getting an outcome with x' < x. The function can be generated from the PDF as follows [3]:

$$c(x) = \int_{-\infty}^{x} p(x')dx' \tag{1}$$

# 2 Experimental Methods

## 2.1 7.1 Data Collection

The objective of the first section was to generate a random number distribution using the following linear probability density defined from 0 to 1 using Eq.2.

$$p(x) = Ax \tag{2}$$

The end goal of this section was to have three plots: a scatter plot with randomly distributed numbers, one with the PDF modeling the points, and one with the CDF modeling the points. To create the scatter plot, we took 1000 points with the x-axis as 0-1000 trials (designated as variable *trials*), and the y-axis as an array of randomly generated values between 0 and 1 (designated as variable *values*). To compute the PDF, we generated an array of x values and then returned the linear probability density function given in Eq.2. To compute the CDF, we used the equation defined above Eq.1 to take the integral, which yields a new equation of $\frac{Ax^2}{2}$. To determine the normalization condition we used our bounds (0,1) and integrate the PDF to yield a value of A = 2, which we then defined as a global variable. We then applied the reverse rule $x = c^{-}1(r)$ to our CDF to find the inverse of the cumulative probability density. Plot Fig.2a shows the random distribution of the 1000 trials generated. Plot Fig.2b shows the PDF function without normalization. To compare the PDF, we created a histogram with the number of bins equal to the the variable *binNumber* (which was 20 in this case). To create error bars on the second graph, we generated ten different sets of random numbers.

Each point on the graph is the mean of the counts per bin. The error bars are the standard deviation of the ten mean values generated per bin. Plot Fig.2c shows the CDF function for this dataset. To compare the CDF, we created a histogram with the number of bins equal to the the variable *cumBinNumber* (which was 100).

## 2.2 7.2 Data Collection

The second section is to generate a random number distribution with Gaussian distribution by utilizing the Box-Muller transformation. The 2D Gaussian probability distribution function is described as in Eq.3, and can be transformed to polar coordinates as in Eq.4.

$$p(x,y) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2}{2\sigma^2}) exp(-\frac{y^2}{2\sigma^2}) dxdy \tag{3}$$

$$p(r,\theta) = \frac{1}{2\pi\sigma^2} exp(-\frac{r^2}{2\sigma^2}) rdrd\theta \tag{4}$$

We use the variable substitution $\mu = \frac{r^2}{2\sigma^2}$. Then the distribution is composed of the following:

$$M(\mu) = exp(-\mu), \Theta(\theta) = \frac{1}{2\pi} \tag{5}$$

where $M(\mu)$ is an exponential PDF and $\Theta(\theta)$ is a uniform PDF over the interval $[0, 2\pi]$. We generated random numbers with these two PDFs and then compute their associated distributions respectively. Similarly to 7.1, we conducted 1000 trials. The generation for theta values was relatively simple, and was done by multiplying $2\pi$ to a set of randomly generated numbers between 0 and 1. To generate our $\mu$ values, we followed a similar procedure given in the tutorial to compute the CDF for an exponential function.

We computed the CDF of $M(\mu) = exp(-\mu)$ and found the reverse of it by using the formula given in the tutorial $-(1/a)log(1-x)$. Then in place of $x$ we randomly generated numbers between 0 and 1. To get our r values for our polar Gaussian Eq.4, we reformulated our original variable substitution to be $r = \sqrt{2\mu\sigma^2}$. By putting these generated numbers in terms of r, we can now convert back to our Cartesian Gaussian Eq.3 from polar coordinates. Our y values were found by the relation $y = rsin(\theta)$ and our x values were found by the relation

$x = rcos(\theta)$. We then used this conversion to obtain Fig.3, which shows the one-dimensional Gaussian distribution where x is held fixed. We also made a histogram with the random x and y numbers generated to display the Gaussian distribution obtained from this dataset, as shown in Fig.11.

## 2.3 7.3 Data Collection

In part 3 of this experiment, we used three different random number generators - uniform, linear, and Gaussian- to find the probability distribution associated with the mean-N value as N gets large. We also sought to find the standard deviation of the mean N-value distribution as a function of N. We created three separate functions to generate our random values for each distribution type. For our uniform function, we used the $r.random()$ function to generate a random number between 0 and 1 for each trial in our dataset.

For our linear function, the function was comprised of two components. We created a *reverse* function to take the integral of our inputted data on the interval $[-\infty, x]$, similarly to Eq.1. Our inputted data, in this case, was a *np.zeros* array. After being integrated, our linear function returned the mean and standard deviation of our inputted data. Our Gaussian model equation followed the same process that used in 7.2 to return our x and y values, and then returned the mean and standard deviation values of the y-values only (because we want to keep our x-axis of our plot held constant).

For each probability distribution, we took 1000 trials each of N-values N = 25, 100, 500, and 1000. Then for we plotted the mean and standard deviation results as N increases while holding x constant. The plots for this are shown in Figures 8, 10, and 6 in the results section below. To see a diagram of the experimental setup (in this case, the flow of logic of the program), refer to Fig.1.

## 2.4 Methodology of the Experiment

Below can be found a flow chart detailing the steps taken to perform the three aforementioned parts of the experiment. Consequently, all results discussed in the subsequent sections of this work were obtained via this work flow.

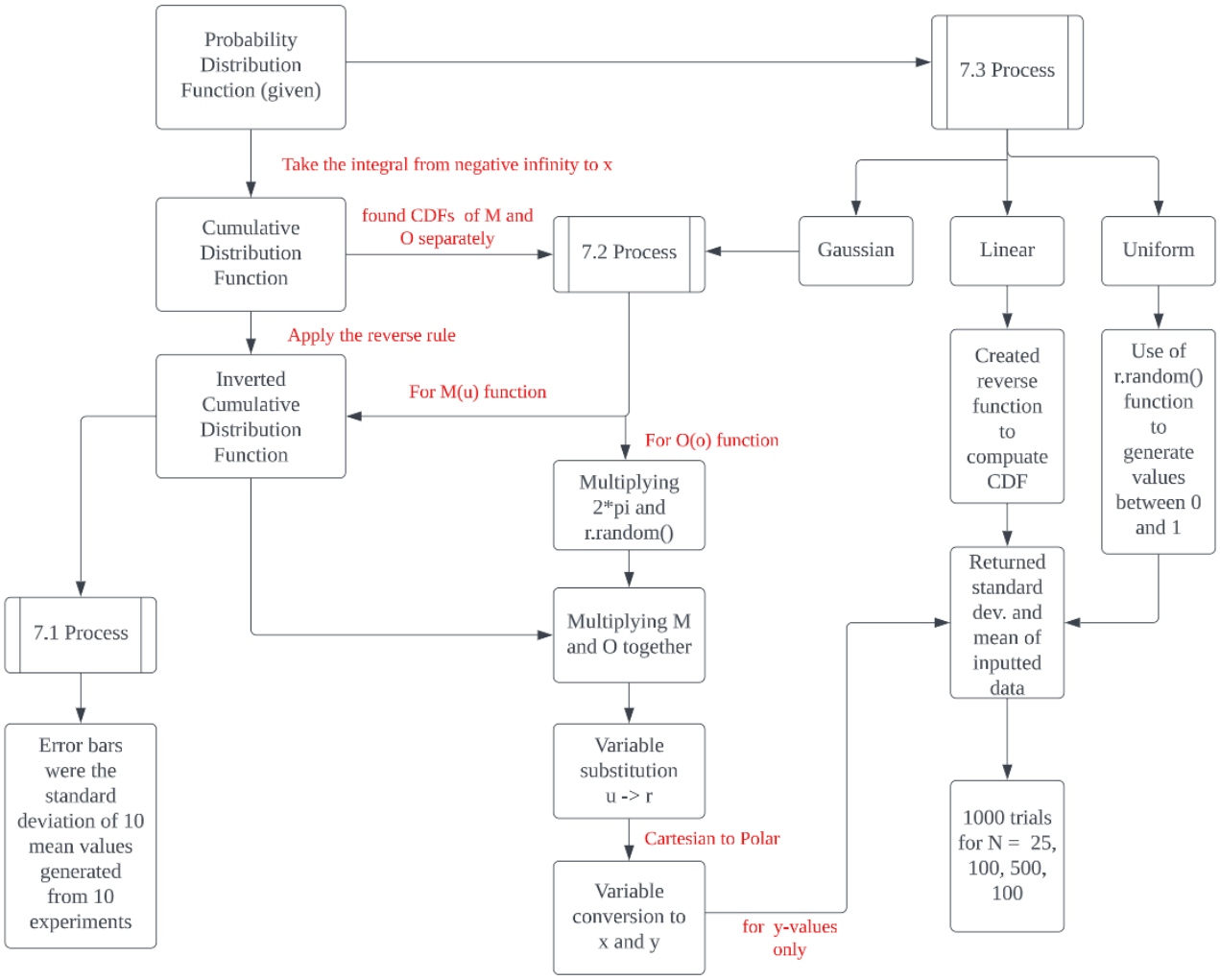



Figure 1: *The flow of logic of the program(s) created for this experiment.* This flow chart is summarizing the steps taken to get the results that are shown in subsequent sections of this academic endeavour.

# 3 Results

| **Technique Used** | **Reasons for Selection** |
|---|---|
| Uniform Distribution | · simple to create a flat distribution<br>· easy to generate random numbers<br>· produces a uniform spread of data |
| Linear Distribution | · produces data that follows a linear trend<br>· easy to model natural phenomena |
| Exponential Distribution | · memoryless property<br>· easy to model natural phenomena, especially when time is concerned |
| Gaussian Distribution | · data is concentrated around central mean<br>· models natural variation around central value |

Table 1: *Chosen techniques used through the experiment and corresponding reasoning.* In this experiment, several different methods were utilized to model our randomly distributed data. This table details each method used and the reasons for choosing that model. Sources for these reasons can be found in the References section: for uniform distribution [4], for linear distribution [5], and for Gaussian distribution [6].

## 3.1 7.1 results

| Bin Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Mean ($\mu$) | 6.9 | 9.8 | 16.2 | 21.7 | 24.0 | 28.8 | 34.6 | 39.7 | 42.0 | 44.2 |
| Standard deviation ($\sigma$) | 1 | 3 | 3 | 4 | 5 | 5 | 6 | 5 | 8 | 6 |
| Bin Number | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Mean ($\mu$) | 52.5 | 56.4 | 63.8 | 66.8 | 66.6 | 77.7 | 79.0 | 81.4 | 87.7 | 100.2 |
| Standard deviation ($\sigma$) | 7 | 5 | 7 | 6 | 8 | 9 | 7 | 7 | 10 | 4 |

Table 2: The row data for the means and standard deviations of counts with linear distribution. We choose the bin number to be 20.

The random variables were generated with a linear distribution. The associated graphs are shown in Fig.2. Fig.2a shows the 1000 random values that are generated from 0 to 1 with linear model; we chose the bin number to be 20. The data points and their standard deviation in Fig.2b are shown in Table 2. Using floating points as our random values introduces

uncertainty due to the limitations of floating point arithmetic. These sources of uncertainty are often negligible but need to be considered when operations are performed on them. Some float point values are added when obtaining the means and standard deviation of our data sets. These would typically propagate the uncertainties. However, given that our data points are within the same order of magnitude and later normalized due to the nature of both the mean and standard deviation, those uncertainties are negligible [7]. The negligible uncertainty for float point values also applies in the following 7.2 section.

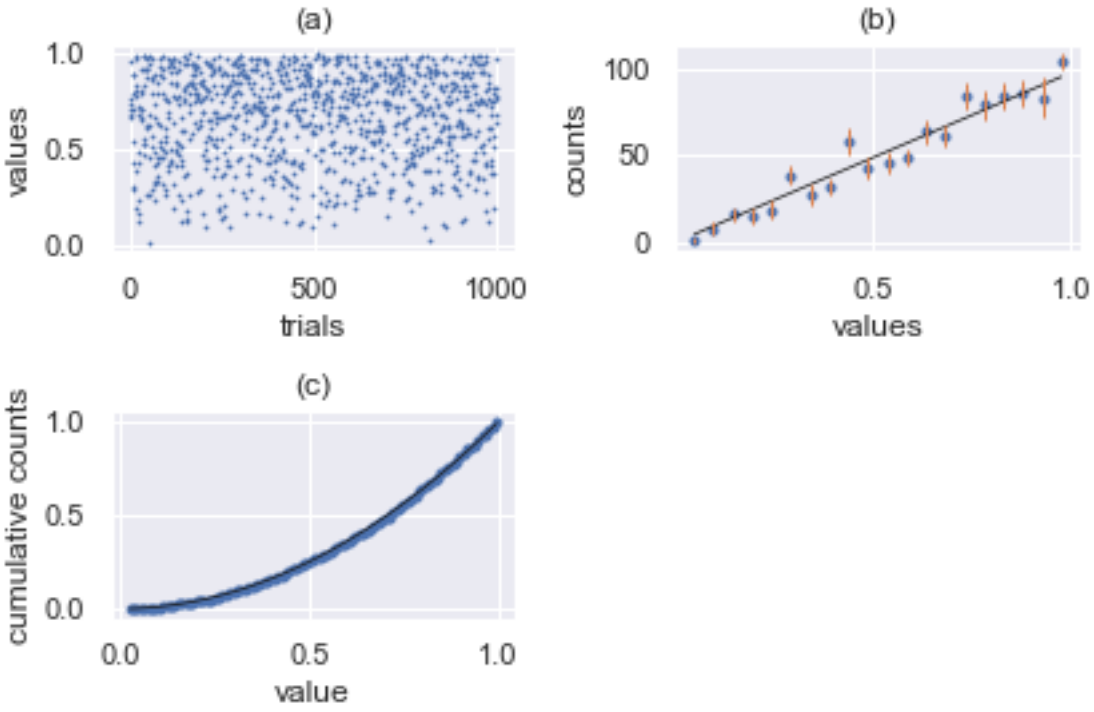



Figure 2: *Summary plot of random generated data obtained from a linear distribution* (a) 1000 random data points generate from the implemented linear model. (2) Data point counted per bin of width 20 compared to the distribution PDF as well as respective standard deviations per bin count. (3) The CDF of the linear model and corresponding generated random data.

## 3.2 7.2 results

Fig.3 shows the spread of 1000 data points using our Gaussian Distribution. In this plot the x parameter is held fixed, and there is a concentration of points between the -1 to 1 y-values, but random scattering along the x-parameter. This indicates a concentration about

the mean, which would be 0. Fig.4 is after normalization for the data and the fit line. We performed a simulation to get the percentage of our dataset that was within one standard deviation (which was 67.8%) and within two standard deviations (which was 95.7%), as indicated on the graph. To see this code, refer to Fig.17 in Appendix 2. The uncertainty of this percentage calculation would be ±0.05%. The line of best fit for Fig.4 is based on the Gaussian line of best fit curve. Visually, it is clear to see that the simulated data closely matches the intended Gaussian fit.

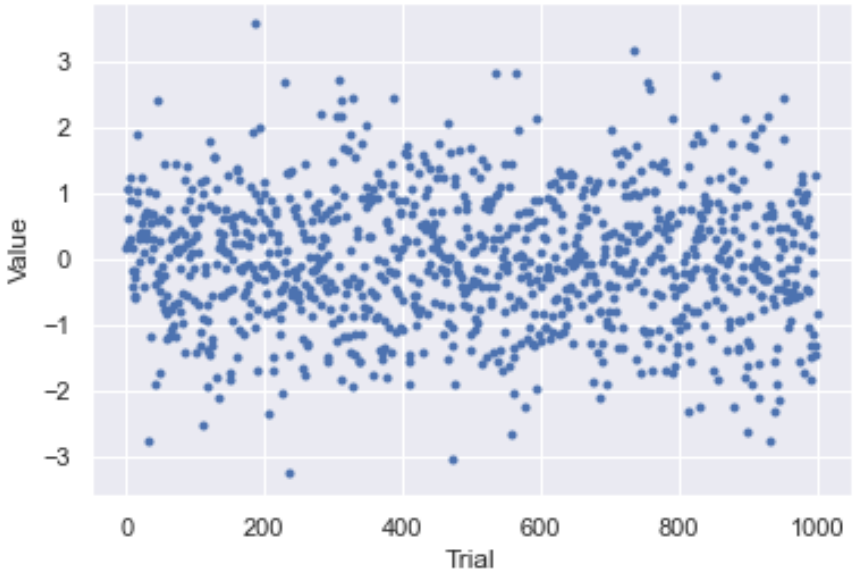



Figure 3: *1000 random data points generated from the implemented 2D Gaussian distribution.* As seen in the plot, the distribution of points is concentrated about the mean (0), but the points themselves appear to be randomly distributed throughout the x parameter.

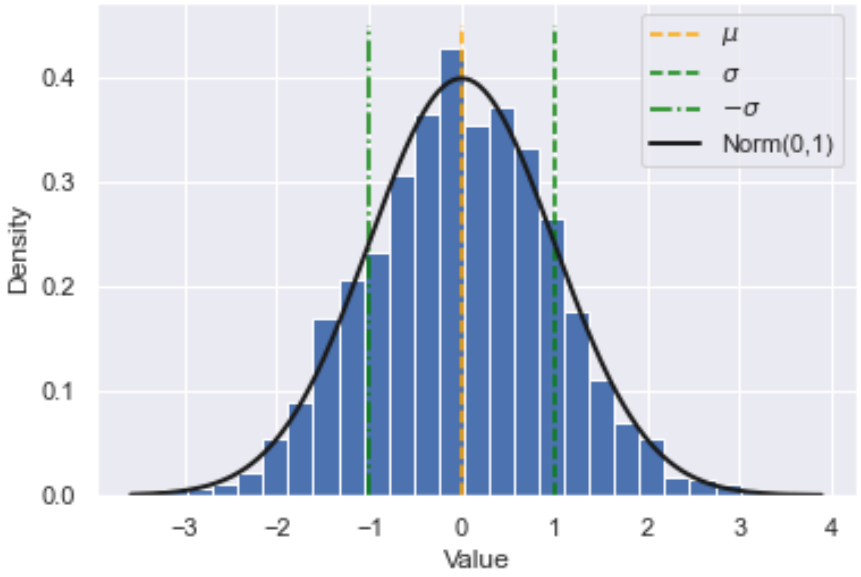



Figure 4: *Graphical density of Gaussian-distributed data points.* The information represented in this histogram shows the amount of data point found per bin (bin count of 25) after Normalization. In black can be found the the true Normalisation curve and in yellow and green can be found the location of the true mean ($\mu$) and standard deviation ($\sigma$), respectively. The Gaussian black curve is the theoretical line of best fit based on the data.

## 3.3 7.3 results

Fig.5 shows four distributions of means of gaussian-distributed experiments. These sub-figures show the means values counted per bin over 1000 experiments with varying sample sizes. Fig.5*a*, Fig.5*b*, Fig.5*c* and Fig.5*d* thus show the distributions of 1000 means of Gaussian data sets with sample sizes of 25, 100, 500 and 1000, respectively. Furthermore, Fig.5 demonstrates how both the value of the mean and the standard deviation of Gaussian distributed experiments vary with increasing sample size $N$. Both statistics can be observed as 1000 experiments were made ranging from a sample size of 1 to a sample size of 5000.

Analogous plots for both uniformly-distributed and linearly-distributed data can be found in sections A.1 and A.2 of the appendices. Those plots were omitted from this section for conciseness.

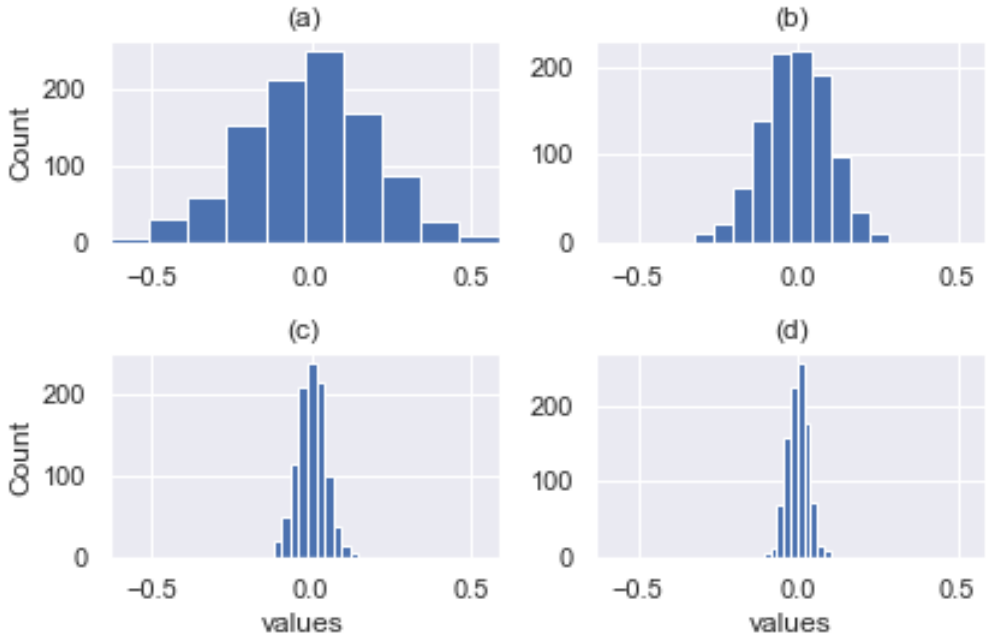



Figure 5: *Histograms showing the data spread of the mean 1000 randomly Gaussian-distributed experiment as sample size N increases.* (a) Distribution of means of experiment with sample size equal to 25. (b) Distribution of means of experiment with sample size equal to 100. (c) Distribution of means of experiment with sample size equal to 500. (d) Distribution of means of experiment with sample size equal to 1000.

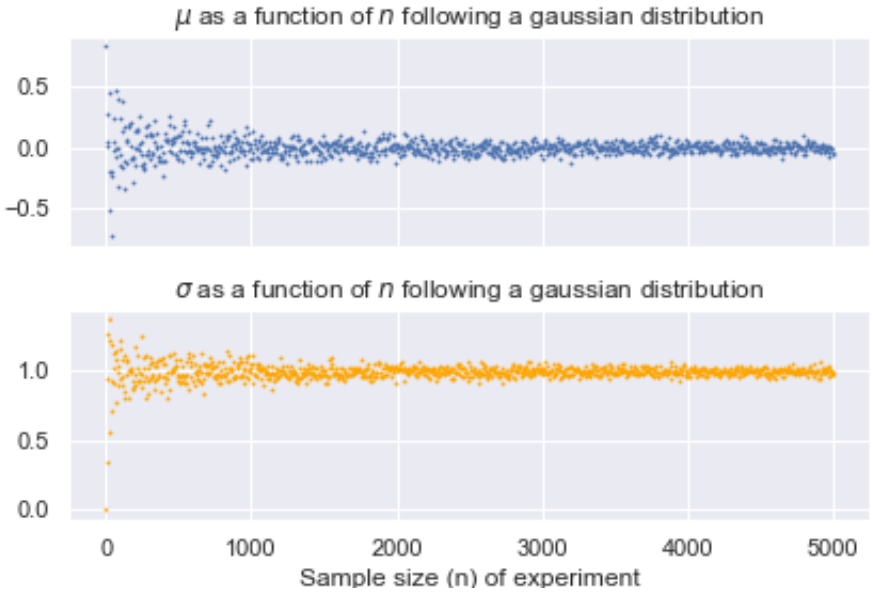



Figure 6: *Observed behaviors of mean and standard deviation values with Gaussian distribution as the number of sample (N) increases per experiment.*

# 4 Discussion

## 4.1 7.1 Discussion

Since the data was generated with a linear probability density between 0 and 1, the probability density is higher when the value is approaching 1. And therefore, we are expecting to detect more data when the value is larger. The plot in Fig.2a agrees with our expectations. To be more quantitative, the data we generated fits the linear model well because $\frac{14}{20}$ of the means and their error bars intersect with the fit line in Fig.8b. For a fit to represent the data well, one should expect at least 70% of the error bars to go through the best fit line. This shows that the generated data fits the linear model [8]. If we repeat the experiment for much more times and take the mean values and the standard deviations again, we should expect to see that all the error bars go through the linear model. A more detailed discussion regarding how sample size affects the mean and the standard deviation will be presented in Sec.4.3. The standard deviation tends to increase when the values of data points increase, but it does not follow a strict pattern. Also, in Fig.2c, the data follows the cumulative distribution function.

## 4.2 7.2 Discussion

Referring to the 2D Gaussian model in the appendix, Fig.11, all of the data points are condensed to an approximately [-1,1] area in both the x and y directions. If we fix one axis, such as in Fig.3, the y-values will follow a Gaussian distribution. We can visualize this distribution in Fig.4, in which we have overlaid a Gaussian fit line. As stated before, our results for the standard deviation calculations was 67.8% for one standard deviation and 95.7% for two standard deviations. The 68-95-99.7 rule for Gaussian distribution states that approximately 68% of the data will fall within one standard deviation of the mean, 95% within two standard deviations, and 99.7% within three standard deviations [9]. While our plot closely follows our desired Gaussian distribution, there are a few bins that have a higher density than expected. This is likely due to the fact that our random number generation yielded greater values than expected. In a future experiment, a greater number of simulations could be run for the random number generation, and then the mean values of this generation

would be taken. This would likely show a more precise model of the Gaussian distribution of the data. We could also take a new *random.seed*() in our code to see how the results in our model might change.

## 4.3 7.3 Discussion

As seen in Fig.5, when taking the means of 1000 gaussian-distributed experiments with sample size $N = 25$, the range at which said means can be found is bounded by [-0.525, 0.525]. However, as this process is repeated with increased sample size, one can note the spread of the mean values diminish. Consequently, the bell curve shape describing the behavior of the means becomes thinner. This implies that the standard deviation of the distribution becomes smaller as the sample size of each experiment increases. Mathematically, this is caused by the fact that the sample size is located in the denominator of the standard deviations of the sampling distributions. The same behaviors can be observed when looking at Fig.7 and Fig.9 for the uniform and the Gaussian-distributed experiments. Their initial means span [-0.500, 0.700] and [-0.650, 0.850] considerably decrease in the upper limit of $N$.

One can also notice that Fig.5, Fig.7 and Fig.9 agrees with the central limit theorem. That is, in the upper limit of sample size $N$ (i.e. with a sufficiently large sample from a population) the means are normally distributed even though the initial distributions were not normally distributed.

We know from Appendix C that the theoretical means and standard deviations for the given distributions are $\mu_U = 0.5$, $\sigma_U = 0.29$, $\mu_L = 0.67$, $\sigma_L = 0.24$ and $\mu_G = 0$, $\sigma_G = 1$. Then, with the usage of Fig.6, Fig.8 and Fig.10 one notices that as we approach the upper limit of sample size $N$, both the mean and standard deviation of the uniform, the linear and the Gaussian-distributed experiments converge towards their corresponding theoretical values. Additionally, one might also notice that for all three figures mentioned above, the standard deviation of the first experiment with sample size $N = 1$ is $\sigma = 0$. This is due to the fact that there is not spread in the data. The same would happen if all data points of an experiment would have the same value (which is technically the case here, but with $N = 1$).

# 5 Conclusions

In the first part of this experiment, 70% of the data fits with the linear model, and according to the 70% rule with the error bars, the linear line of best fit is in accordance with the data. 30% of the error bars did not fit within the model, particularly when the probability density value was approaching 1. However, later in section 7.3, we showed that a larger sample size results in the mean and standard deviation approaching their true value, thus increasing the number of experiments results in a better fit. We demonstrated in Section 7.2 that a random number generation for a Gaussian model would result in a Gaussian distribution of the data. We proved this by showing that 67.8% of the data spread fit within one standard deviation and 95.7% within two standard deviations. We also plotted a Gaussian fit line, which visually showed that our data closely followed a Gaussian distribution. For Section 7.3, we showed that the data distribution became thinner, and thus the standard deviation decreased. The decrease in standard deviation was due to the increased sample size. The bell curve behaviour with our various distributions was in accordance with the Central Limit Theorem, as discussed above. As the number of N samples increased, the distribution of means was compressed about the theoretical mean value of each distribution. As it stands, increasing the number of experiments in Sections 7.1 and 7.2 could improve the accuracy of the results and yield similarly successful results to Section 7.3.

# References

[1] Dirk Kroese et al. "Why the Monte Carlo method is so important today". In: *Wires Computational Statistics* (Nov. 2014), pp. 386–392. DOI: https://doi.org/10.1002/wics.1314.

[2] Hans Fischer. "Introduction". In: *A History of the Central Limit Theorem: From Classical to Modern Probability Theory*. New York, NY: Springer New York, 2011, pp. 1–16. ISBN: 978-0-387-87857-7. DOI: 10.1007/978-0-387-87857-7_1.

[3] H. Pishro-Nik. *Introduction to probability, statistics, and random processes*. Kappa Research LLC, 2014. Chap. 4.1.1. ISBN: 9780990637202.

[4] M.F. Schilling. *Fundamentals of Computational Neuroscience*. Oxford University Press, 2013. ISBN: 0199568413.

[5] S. Aggarwal S. Sood. "Linear Regression in Regression Analysis". In: (). URL: https://medium.com/@devraj.agarwal/simple-linear-regression-parameter-estimates-explained-c8da2466bed6.

[6] S. Janusonis. *Encyclopedia of Applied and Computational Mathematics*. Springer Berlin, 2013. ISBN: 978-3-540-70528-4.

[7] Oracle Inc. "What Every Computer Scientist Should Know About Floating-Point Arithmetic". In: (). URL: https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html.

[8] Department of Physics Astronomy University of Pennsylvania. *Graphical Representation of Data*. URL: https://www.physics.upenn.edu/sites/default/files/Graphing_Data.pdf.

[9] Wayne W.Lamorte. "The Standard Normal Distribution". In: (). URL: https://sphweb.bumc.bu.edu/otlt/mp-modules/bs/bs704_probability/bs704_probability9.html.

# A First Appendix (Additional Figures)

## A.1 Uniform Distribution

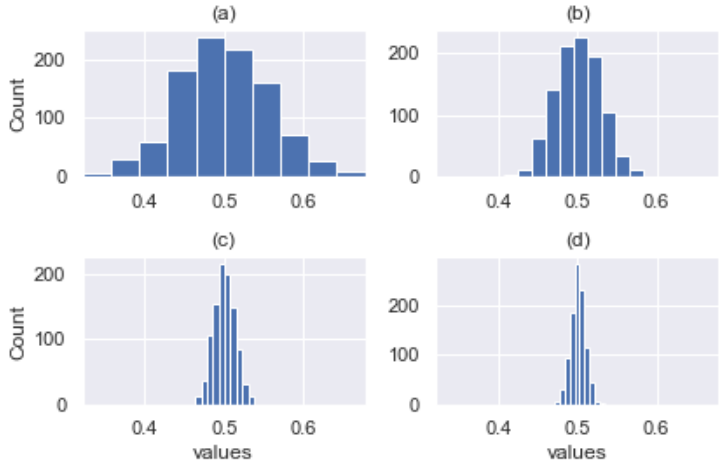



Figure 7: *Histograms showing the data spread of the mean 1000 randomly uniform-distributed experiment as sample size n increases.* (a) Distribution of means of experiment with sample size equal to 25. (b) Distribution of means of experiment with sample size equal to 100. (c) Distribution of means of experiment with sample size equal to 500. (d) Distribution of means of experiment with sample size equal to 1000.

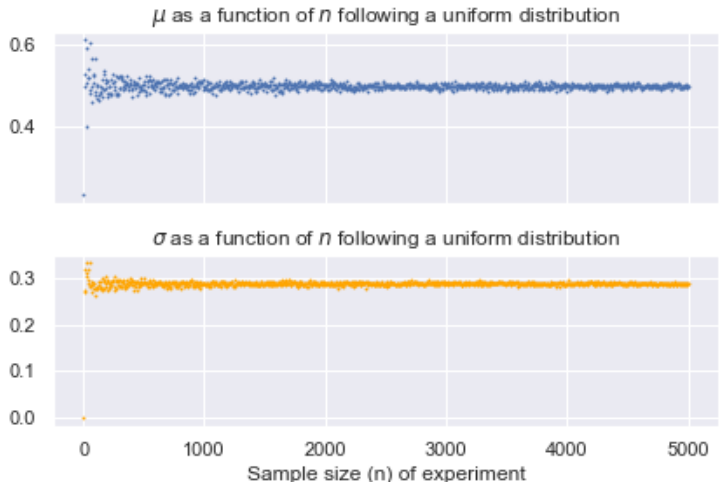



Figure 8: *Observed behaviors of mean and standard deviation values with Uniform distribution as the number of sample (n) increases per experiment.*

## A.2 Linear Distribution

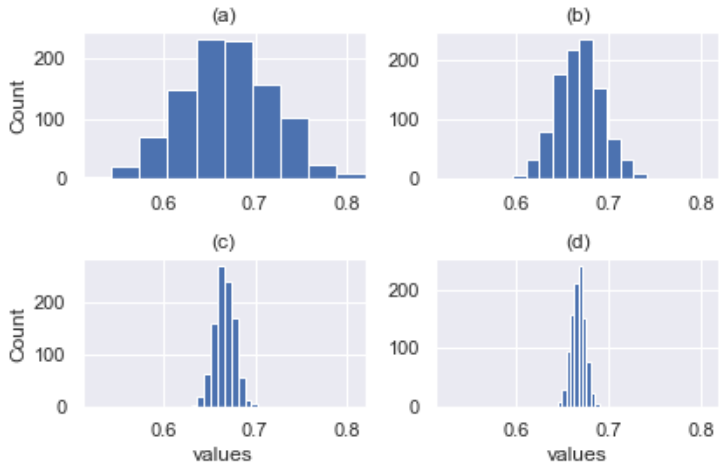



Figure 9: *Histograms showing the data spread of the mean 1000 randomly linear-distributed experiment as sample size N increases.* (a) Distribution of means of experiment with sample size equal to 25. (b) Distribution of means of experiment with sample size equal to 100. (c) Distribution of means of experiment with sample size equal to 500. (d) Distribution of means of experiment with sample size equal to 1000.

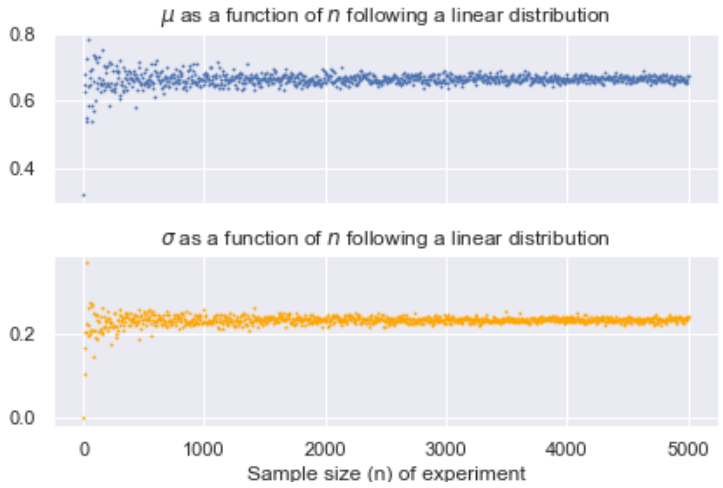



Figure 10: *Observed behaviors of mean and standard deviation values with Linear distribution as the number of sample (N) increases per experiment.*

## A.3 Gaussian Distribution

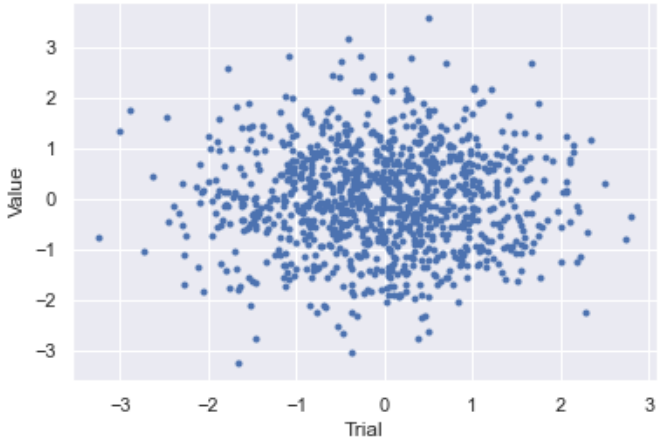



Figure 11: *1000 random data points generated from the implemented 2D Gaussian distribution on both the $x$ and $y$ axes.*

# B Second Appendix (Python Scripts)

Here follows important snippet of codes used to generate the data and plots used in this work.

For complete scripts, plots and project architecture please refer to the project repository.

*uniform_model.py*

```
6   def UNI(nsamp):
7       values=np.zeros(nsamp)
8       trials=list(range(nsamp))
9       for i in trials:
10          values[i]=r.random()
11
12      return([np.mean(values), np.std(values)])
```

Figure 12: *Slice of code outputting the mean and standard deviation of a normal experiment with sample variable sample size (nsamp).*

*linear_model.py*

```
20  #----------- Linear PDF & CDF ----------#
21  def linearModelPDF(A, x):
22      x = np.array(x)
23      return A*x
24
25  # CDF is obtained by integrating PDF from -inf to inf
26  def linearModelCDF(A, x):
27      x = np.array(x)
28      return A*(x**2)/2
29
30  #--------- Apply reverse rule ---------#
31  def reverse(data):
32      for i in range(len(data)):
33          data[i] = np.sqrt(2*r.random()/A)
34
```

Figure 13: *Slice of code defining the PDF(), CDF() and reverse() functions needed to generate Linear-distributed random data.*

```
36    def LIN(n):
37        data =np.zeros(n)
38        reverse(data)
39        return [np.mean(data), np.std(data)]
```

Figure 14: *Slice of code outputting the mean and standard deviation of a linear experiment with sample variable sample size (N).*

## *gaussian_model.py*

```
14  def gaussian_model(n, A=1, sigma=1):
15      #------------ Mus and Thetas -----------#
16      values=np.zeros(n)
17
18      moose = np.zeros(n)
19      rs = np.zeros(n)
20      thetas = np.zeros(n)
21
22      #------------ Mus and Thetas -----------#
23      for i in range(len(values)):
24          moose[i] = -(1/A) * np.log(1 - r.random())
25          thetas[i] = r.random()*2*np.pi
26
27      rs = np.sqrt(2*moose*sigma**2)
28
29      ys = rs*np.sin(thetas)
30
31      xs = rs*np.cos(thetas)
32
33      return [xs, ys]
```

Figure 15: *Slice of code outputting an array containing two Gaussian random distributions (on $x$ and $y$) with a default normalisation constant $A = 1$ and $\sigma = 1$.*

```
35  def GAU(n): # Used for data generation of section 7.3
36      values = gaussian_model(n)[1]
37      return([np.mean(values), np.std(values)])
```

Figure 16: *Slice of code outputting the mean and standard deviation of a gaussian experiment with sample variable sample size (N).*

```python
def get_percentage_within_std(data, mu, std):
    one_std = 0
    two_std = 0
    for x in data:
        if abs(x-mu) <= std:
            one_std += 1
        if abs(x-mu) <= 2*std:
            two_std += 1
    return [one_std/len(data), two_std/len(data)]

in_one_std, in_two_std = get_percentage_within_std(ys, np.mean(ys), np.std(ys))
```

Figure 17: *Slice of code calculating the percentage of Gaussian-distributed data found within one and two standard deviations of the mean.*

## *mean_distribution.py*

```python
#---------- Experiment Definition--------#
def mean_uniform(n, m): # n = samples, m = nb of exp
    out = np.zeros(m) # index 0 is mean, 1 is std
    for i in range(m):
        result_uni = UNI(n)
        out[i] = result_uni[0]

    return out
```

Figure 18: *Slice of code outputting an array of size m containing the means of N data points generated uniformly m times.* Similar slice of codes can be found in the aforementioned repository for both the Linear and Gaussian distributions. These slice of code are used to generate the data for Figures 5, 7, 9.

```python
for i in range(1000):
    curr_uni = UNI(5*i+1)
    stats_uni[0][i], stats_uni[1][i] = curr_uni[0], curr_uni[1]

    curr_lin = LIN(i+1)
    stats_lin[0][i], stats_lin[1][i] = curr_lin[0], curr_lin[1]

    curr_gau = GAU(i+1)
    stats_gau[0][i], stats_gau[1][i] = curr_gau[0], curr_gau[1]

lin_xs = np.linspace(1, 5000, 1000)
```

Figure 19: *Slice of code populating arrays of size* $= 1000$ *by means and standard deviations of experiment with increasing sample size (Uniform, Linear and Gaussian).* Similar slice of codes can be found in the aforementioned repository for both the Linear and Gaussian distributions. These slice of code are used to generate the data for Figures 6, 8, 10.

# C Third Appendix (Derivations)

## Uniform Distribution

Let a=0, b=1 be the boundaries of possible generated data value. Then,

$$\begin{aligned}\mu_U &= \frac{a+b}{2} \\ &= 0.50 \\ \sigma_U &= \sqrt{\frac{(b-a)^2}{12}} \\ &= 0.2886... \\ &\approx 0.29\end{aligned}$$

## Linear Distribution

$$\begin{aligned}\mu_L &= \int_0^1 2x \cdot x^2 \cdot dx \\ &= \frac{2}{3} \\ &\approx 0.67 \\ \sigma_L^2 &= E(x^2) - E(x)^2 \\ &= \int_0^1 2x \cdot x^2 \cdot dx - (\int_0^1 2x \cdot x^2 \cdot dx)^2 \\ &= \frac{1}{2} - \frac{4}{9} \\ \sigma_L &= 0.2357... \\ &\approx 0.24\end{aligned}$$

## Gaussian Distribution

Theoretically, the Gaussian Distribution has a mean $\mu_G = 0$ and standard deviation $\sigma_G = 1$.