Computational Studies of Random Number Generators

Kiara Gholizad[1]

[1]Department of Physics, Sharif University of Technology, Tehran, Iran

April 7, 2025

Abstract

Random number generators (RNGs) are a fundamental computational tool in computational physics, forming the basis for stochastic process simulation in a variety of applications ranging from Monte Carlo techniques to molecular dynamics and statistical physics models. This paper provides a systematic analysis of pseudorandom number generators, considering their statistical behavior, implementation methods, and design limits. Employing Python-based implementations of a self-written Linear Congruential Generator (LCG) as well as standard libraries (`random` and `NumPy`), we compared the uniformity of the generated distributions, independence of successive numbers, and divergence of statistical errors with sample size as a parameter. We also investigated empirically the validation of the Central Limit Theorem and the synthesis of non-uniform distributions, specifically Gaussian, through method-of-transformation techniques such as the inverse CDF, Box-Muller transform, and Polar (Marsaglia) method. Our findings confirm that contemporary RNGs demonstrate exceptional uniformity and independence with statistical fluctuations precisely matching theory (e.g., relative standard deviation varying as $0.3/\sqrt{N}$). The LCG, however, showed faint correlations in transition sequences, revealing shortcomings of less complex deterministic methods. The method-of-transformation techniques worked well for generating non-uniform distributions with high accuracy, presenting useful tools for computational physics. These findings serve as a blend of theoretical understanding of pseudorandom number generation as well as pragmatic recommendations for tailoring stochastic computations toward efficiencies, underscoring the selection of suitable RNGs for particular tasks.

Contents

1 Introduction

1.1 Theoretical Background

Random number generators (RNGs) are essential in computational physics for the simulation of stochastic processes and probabilistic phenomena. But computers are incapable of generating entirely random numbers because they are deterministic. Pseudorandom number generators (PRNGs) therefore employ algorithms to simulate randomness, a limitation on which this work is founded.

Effective PRNGs must possess two important qualities:

1. Uniform probability distribution
2. Independence of successive numbers

The Linear Congruential Generator (LCG), a traditional PRNG, produces pseudorandom numbers according to the recurrence relation:

$$x_{n+1} = (a \cdot x_n + c) \mod m \tag{1}$$

where a is the multiplier, c is the increment, m is the modulus, and x_0 is the seed. LCGs are simple, yet they do have well-known drawbacks:

1. Periodicity: The sequence repeats with a maximum period of m.
2. Correlation: Successive numbers are deterministically linked.
3. Low-dimensional structure: Consecutive numbers form hyperplanes in higher dimensions (Marsaglia's Theorem).

The LCG is controlled by the seed, allowing reproducibilityuseful for debuggingthough this constrains true randomness. It's a strength and a limitation of PRNGs.

1.2 Significance and Applications

RNGs are used extensively in computational physics applications such as Monte Carlo methods, molecular dynamics, statistical physics, quantum simulations, diffusion processes, and models of random walk. This work investigates systematically the PRNGs with particular emphasis on:

1. Statistical properties and scaling of uniform distribution deviations
2. Correlation detection in random sequences
3. LCG implementation and analysis
4. Empirical validation of the Central Limit Theorem and its physical connections
5. Non-uniform distribution generation, notably Gaussian, via transformation methods

This work offers theoretical explanations of PRNG fundamentals along with practical advice for computational physicists on the significance of mastering the nature of PRNG to get correct, effective simulations in a variety of different physical systems.

2 Methodology

2.1 Implementation Approach

Our study uses Python implementations to study random number generators as well as their characteristics. We use Python's built-in module random, the NumPy random number generators, as well as a Linear Congruential Generator that we implemented ourselves. The implementation has extensive statistical analysis as well as visualization techniques.

2.1.1 Software Environment

The computational analysis is performed in a Python 3.13.2 environment, utilizing the following libraries:

- `random`: Python's built-in module for pseudorandom number generation
- `numpy`: For efficient numerical operations and advanced random number generation
- `matplotlib`: For creating visualizations and plots
- `scipy.stats`: For statistical tests including chi-square and Kolmogorov-Smirnov tests

2.2 Theoretical Foundations

2.2.1 Properties of Uniform Distributions

An important feature of random number generators is that they can provide sequences with the right probability distribution. For a uniform distribution, every possible value should be equally likely to occur.

The expectation for a discrete uniform distribution with k possible values (for example, the integers 0 through 9) is that each value will occur about N/k times in a sequence of size N. Empirical confirmation of uniform distribution is, however, a matter of statistical tests.

The statistical fluctuations from a uniform distribution adhere to clearly defined theoretical principles. When we sample N times from a discrete uniform distribution of k categories (in this example, $k = 10$ for the digits 0-9), each category's count follows a binomial distribution with parameters N and $p = 1/k$. For a binomial distribution:

$$\mu = Np = N/k \tag{2}$$

$$\sigma^2 = Np(1-p) = N(1/k)(1-1/k) = \frac{N(k-1)}{k^2} \tag{3}$$

Thus, the standard deviation of the count for any category is:

$$\sigma = \sqrt{\frac{N(k-1)}{k^2}} \tag{4}$$

The relative standard deviation (RSD) is then:

$$\text{RSD} = \frac{\sigma}{N} = \frac{\sqrt{\frac{N(k-1)}{k^2}}}{N} = \frac{\sqrt{k-1}}{k} \cdot \frac{1}{\sqrt{N}} \tag{5}$$

The theoretical proportionality constant for $k = 10$ categories is:

$$C = \frac{\sqrt{k-1}}{k} = \frac{\sqrt{9}}{10} = \frac{3}{10} = 0.3 \tag{6}$$

Thus, on a theoretical basis, relative standard deviation scaling is:

$$\frac{\sigma}{N} = \frac{0.3}{\sqrt{N}} \tag{7}$$

This relationship offers a quantitative framework for assessing the statistical attributes of random number generators as a function of sample size.

2.2.2 Correlations and Independence in Random Sequences

Independence of successive random numbers is a vital characteristic in good-quality generators. Not only are we expecting each number to be equally likely to show itself, we are expecting each result to be independent of every previous result.

The conditional probability of a given digit given a certain preceding digit for a discrete uniform distribution must also be uniform. Mathematically, if X_i and X_{i+1} are successive random digits, then:

$$P(X_{i+1} = j | X_i = 4) = \frac{1}{10} \quad \text{for } j \in \{0, 1, 2, \ldots, 9\} \tag{8}$$

The number of occurrences of each digit following a specific digit (e.g., 4) follows a binomial distribution with parameters n (total occurrences of the preceding digit) and $p = 1/10$ (probability of each subsequent digit). The expected standard deviation is:

$$\sigma = \sqrt{np(1-p)} = \sqrt{n \cdot 0.1 \cdot 0.9} = \sqrt{0.09n} \tag{9}$$

Testing for these kinds of correlations sheds light on the sequential independence of a random number generator.

2.2.3 Central Limit Theorem and Its Applications

The Central Limit Theorem establishes a basic framework for describing the statistical behavior of aggregates of random variables. Based on this theorem, if we let a variable y be the sum of many independent random variables:

$$y = \sum_{i=1}^{N} x_i \tag{10}$$

then for large N, the distribution of y tends towards a normal distribution, independently of the initial distribution of the x_i, provided that they all have finite variance.

For uniformly distributed variables on $[0, 1]$, every variable has:

$$\mu_x = 0.5 \tag{11}$$

$$\sigma_x^2 = \frac{1}{12} \tag{12}$$

When N of such variables are summed, the resultant distribution has:

$$\mu_y = N \cdot 0.5 = \frac{N}{2} \tag{13}$$

$$\sigma_y^2 = N \cdot \frac{1}{12} = \frac{N}{12} \tag{14}$$

$$\sigma_y = \sqrt{\frac{N}{12}} \tag{15}$$

These theoretical predictions give quantitative expectations to our empirical explorations for the Central Limit Theorem.

2.2.4 Non-Uniform Distribution Generation Theory

Numerous computational physics applications involve having non-uniformly distributed random numbers, especially Gaussian (normal) distributed. The general transformation approach offers a strong method to transform uniformly distributed random variables into variables with any kind of distribution. For a uniformly distributed variable u on [0,1] with probability density function $p_u(u) = 1$ and target distribution with probability density function $g(y)$ and cumulative distribution function $G(y)$, the conservation of probability demands:

$$p_u(u)du = g(y)dy \tag{16}$$

Integrating both sides:

$$\int_0^u p_u(u')du' = \int_{-\infty}^{y} g(y')dy' \tag{17}$$

This results in:

$$u = G(y) \tag{18}$$

Solving for y:

$$y = G^{-1}(u) \tag{19}$$

This formula for transformation can be used to produce different non-uniform distributions:

Exponential Distribution We are given that the PDF of the given distribution is $g(y) = \frac{1}{a}e^{-y/a}$ for $y \geq 0$. The inverse CDF is:

$$y = -a \ln(u) \tag{20}$$

Power Law Distribution For a power law distribution with PDF $g(y) = (n + 1)y^n$ for $y \in [0, 1]$, the inverse CDF is:

$$y = u^{1/(n+1)} \tag{21}$$

Triangular Distribution For a triangular distribution with mode at c, the inverse CDF is piecewise:

$$y = \begin{cases} \sqrt{u \cdot c} & \text{if } u \leq c \\ 1 - \sqrt{(1-u)(1-c)} & \text{if } u > c \end{cases} \tag{22}$$

Gaussian Distribution The CDF for the standard normal distribution is:

$$\Phi(y) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{y}{\sqrt{2}}\right)\right] \tag{23}$$

The inverse CDF is:

$$y = \Phi^{-1}(u) = \sqrt{2} \cdot \operatorname{erf}^{-1}(2u - 1) \tag{24}$$

This necessitates a numerical calculation of the inverse error function, which can be found in most scientific computing libraries.

Box-Muller Transform Another method for producing Gaussian random variables is obtained by employing the Box-Muller transform. The transform takes two independent uniform random variables U_1 and U_2 on $(0, 1)$ and produces two independent standard normal random variables:

$$Z_1 = \sqrt{-2\ln(U_1)}\cos(2\pi U_2) \tag{25}$$

$$Z_2 = \sqrt{-2\ln(U_1)}\sin(2\pi U_2) \tag{26}$$

The mathematical form stems from converting a bivariate normal distribution into polar coordinates. The joint PDF for two independent standard normal variables is:

$$f(z_1, z_2) = \frac{1}{2\pi} e^{-\frac{z_1^2 + z_2^2}{2}} \tag{27}$$

Converting to polar coordinates with $z_1 = r\cos\theta$ and $z_2 = r\sin\theta$:

$$f(r, \theta) r dr d\theta = \frac{1}{2\pi} e^{-\frac{r^2}{2}} r dr d\theta \tag{28}$$

This provides independent distributions of r and θ:

$$f_r(r) = r e^{-\frac{r^2}{2}} \quad \text{for } r \geq 0 \tag{29}$$

$$f_\theta(\theta) = \frac{1}{2\pi} \quad \text{for } \theta \in [0, 2\pi) \tag{30}$$

With inverse transform sampling:

$$F_r(r) = \int_0^r t e^{-\frac{t^2}{2}} dt = 1 - e^{-\frac{r^2}{2}} \tag{31}$$

Setting $U_1 = F_r(r)$ gives $r = \sqrt{-2\ln(U_1)}$. For θ, we have $\theta = 2\pi U_2$.

Polar (Marsaglia) Method The Polar method eschews trigonometry by implementing a different strategy:

1. Generate uniform random variables V_1 and V_2 in $(-1, 1)$
2. Calculate $S = V_1^2 + V_2^2$
3. If $S \geq 1$ or $S = 0$, reject and try again
4. Otherwise, compute:

$$Z_1 = V_1 \sqrt{\frac{-2\ln(S)}{S}} \tag{32}$$

$$Z_2 = V_2 \sqrt{\frac{-2\ln(S)}{S}} \tag{33}$$

This approach is computationally superior to Box-Muller for most uses, although it involves rejection sampling.

2.3 Experimental Procedures

2.3.1 Uniform Distribution Analysis

Our method for analysis of uniform distributions has the following steps:

1. Generate random integer numbers from 0 to 9 by calling random.randint() for sample sizes starting from 100 up to 1,000,000.
2. Count the number of each digit's occurrence and determine the expected count for a uniform distribution ($N/10$).
3. Calculate statistical measures including standard deviation of counts, as well as relative deviation (σ/N).
4. Measure consistency of uniformity by looking at p-values of the chi-square tests across various sample sizes.
5. Plot the relative standard deviation as a function of sample size on a log-log scale.
6. Compare observed scaling with the theoretical relationship $\sigma/N = \frac{0.3}{\sqrt{N}}$.
7. Examine p-values of chi-square tests with varying sample sizes to measure consistency of uniformity.

2.3.2 Correlation Analysis

Our method defines the possible correlations of random sequences through extracting and inspecting those subsequences with certain patterns:

1. Produce a very long list of random digits ranging from 0 to 9 (N = 100,000).
2. Identify all positions where a digit appears prior to the digit 4.

3. Take out the following numbers to form a new series.
4. Count occurrences of each digit in this provided extracted series and calculate the expected counts.
5. Calculate the standard deviation of the observed count, comparing it to the theoretical value for the binomial distribution: $\sigma = \sqrt{N \cdot p \cdot (1-p)} = \sqrt{N \cdot 0.1 \cdot 0.9}$.
6. Perform a chi-squared test to examine for uniformity.
7. Create a histogram visualization of observed counts versus expected frequency.

2.3.3 Linear Congruential Generator Implementation

We employ and test a Linear Congruential Generator with particular parameters:

1. Implement an LCG class with parameters $a = 1664525$, $c = 1013904223$, and $m = 2^{31}$.
2. Set the fixed seed value (1234) for reproducible initialization of the generator.
3. Produce 10,000 integers ranging from 0 to 9 based on the LCG.
4. Examine the overall distribution by chi-square tests as well as by deviation measures.
5. extract digits that follow the digit 4 and check their distribution.
6. Calculate the ratio of even to odd digits to identify possible patterns.
7. Identify and tally the most common transition patterns (digit pairs) in the series.

2.3.4 Central Limit Theorem Investigation

Our method for proving the Central Limit Theorem examines the distribution of the sums of uniformly distributed random variables:

1. For each value of N in $\{5, 10, 100, 1000\}$:
   (a) Produce 10,000 samples, each as a total of N uniformly distributed numbers ranging from 0 to 1.
   (b) Calculate the mean and standard deviation of these sums.
   (c) Normalize to have a mean of 0 and standard deviation of 1.
   (d) Compare the observed standard deviation and mean with the theoretical ones.
   (e) Create plots of histograms of the normalized sums versus the theoretical normal probability density function.
2. Additionally, investigate how the standard deviation of the sums varies with N by employing 20 logarithmically spaced values from 1 to 1000.
3. Compare the experimental result with the theoretical expectation $\sigma = \sqrt{N/12}$.

2.3.5 Non-Uniform Distribution Generation

We apply a broad framework for constructing different non-uniform distributions by applying methods of transformation:

1. Write a general transformation function that can produce several types of distribution:
   - Exponential distribution with parameter $a = 2.0$
   - Power law distribution with exponent $n = 3$
   - Triangular distribution with mode $c = 0.7$
   - Gaussian distribution using three different methods
2. For each distribution:
   (a) Produce 10,000 independent random samples by applying the corresponding transformation formula.
   (b) Calculate statistical measures (mean, standard deviation)
   (c) Compare with theoretical expectations
   (d) Create histogram visualizations with theoretical PDFs overlaid
   (e) Carry out Kolmogorov-Smirnov tests for Gaussian distributions in order to check conformity to a normal distribution
3. Implement and examine three different methods for the generation of Gaussian random variables:
   (a) Box-Muller transform
   (b) Polar (Marsaglia) method
   (c) Inverse CDF method using the error function
4. Create comparative visualizations:
   (a) Separate histograms for each method and each distribution
   (b) Overlay plot of all three Gaussian methods
   (c) Q-Q plots for assessing normality
   (d) Combined visualization of all generated distributions

3 Results and Analysis

3.1 Statistical Properties of Uniform Distributions

3.1.1 Distribution Characteristics

We experimented with the statistical properties of randomly distributed numbers by generating lists of randomly selected integers ranging from 0 to 9 using Python's built-in module 'random' for multiple sample sizes ranging from 100 to 1,000,000. The p-values of corresponding chi-square goodness-of-fit tests are depicted in Figure 1.

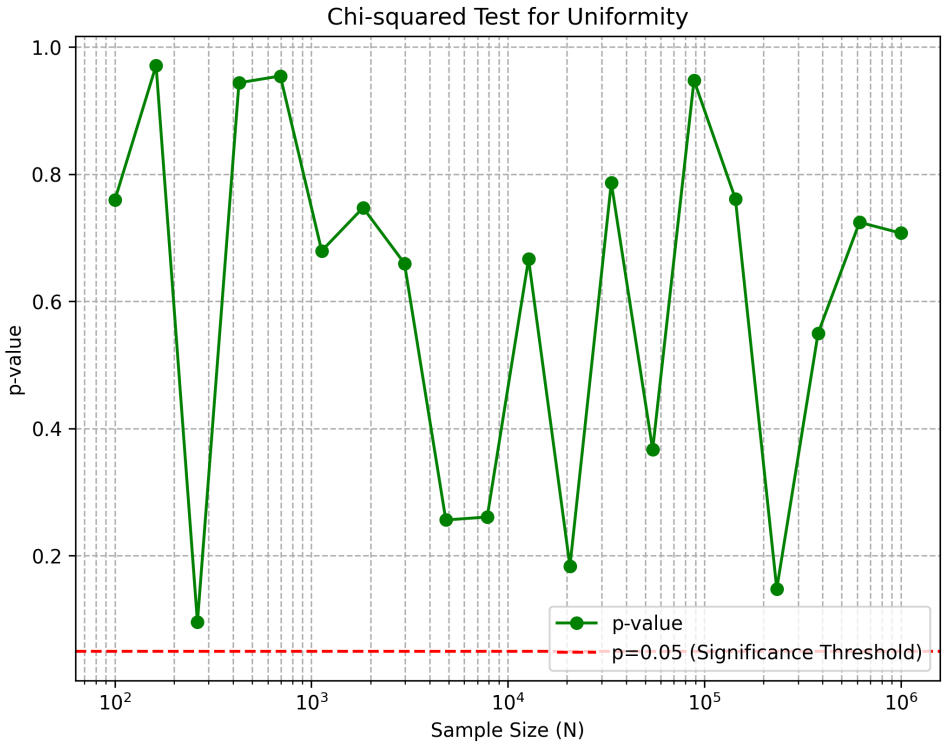

Figure 1: Chi-squared test p-values for measuring uniformity in different sample sizes. The red dashed line represents the standard significance level p=0.05.

For all tested sample sizes, p-values are far above the standard significance cutpoint of 0.05, demonstrating that the distributions are not significantly different from a uniform distribution. Even with very large sample sizes of nearly a million random numbers, where the test is most sensitive to small discrepancies, the generator does not degrade in its uniformity. This establishes the first of the key requirements of good pseudorandom number generators.

3.1.2 Scaling of Statistical Deviations

Figure 2 shows the correlation of relative standard deviation with sample size over several orders of magnitude.

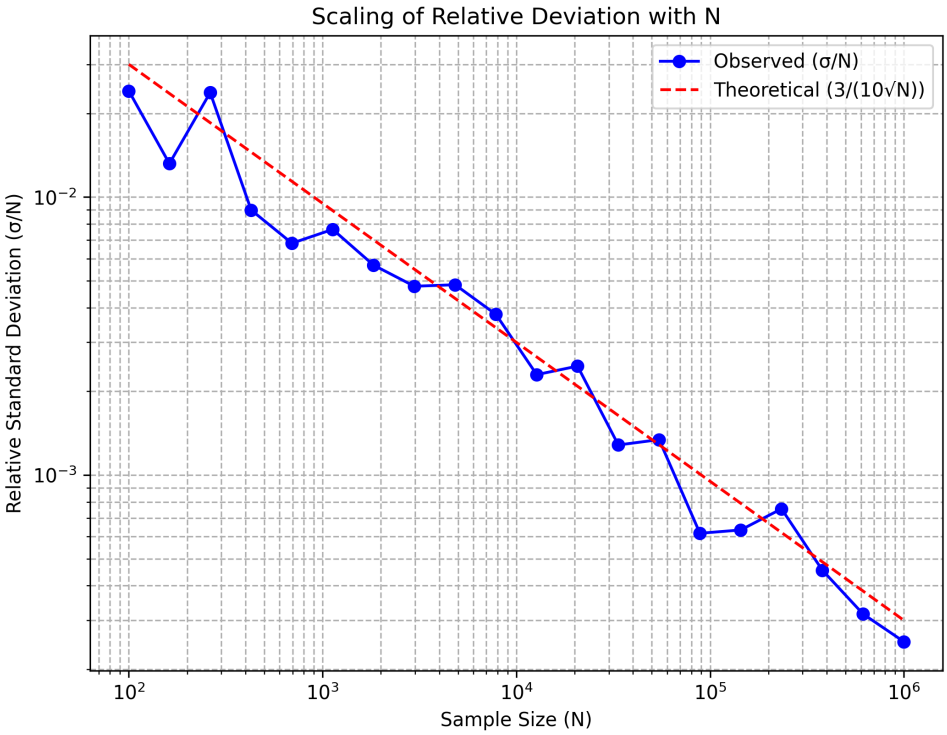

Figure 2: Scaling of relative standard deviation (σ/N) with sample size as obtained by a log-log plot. Blue observations are plotted, with the red dashed line being the theoretical value $\sigma/N = 0.3/\sqrt{N}$.

The observed trend closely matches the theoretical prediction made in the methods section. The negative slope of the log-log plot is about -0.5, verifying the inverse square-root scaling. Additionally, the points of our data match well with the theoretical line with the proportionality constant $C = 0.3$, confirming both the predicted form of the fluctuations as well as their predicted magnitudes.

This agreement of theory with experiment offers strong confirmation of the statistical properties of our random number generator. The $1/\sqrt{N}$ scaling law is a universal property of randomness found in a broad array of physical as well as computational systems.

3.2 Correlation Analysis in Random Sequences

3.2.1 Consecutive Number Independence

We studied possible correlations in the seemingly random sequence by exploring the distribution of digits that appeared just after 4. The frequency distribution of those subsequent digits is shown in Figure 3.

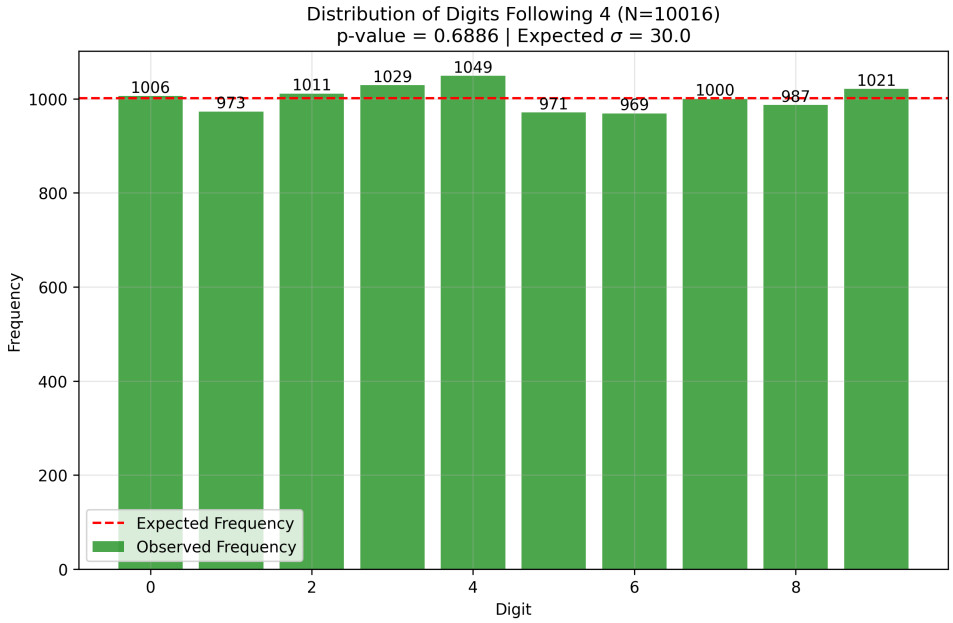

Figure 3: Distribution of digits following the digit 4 in the random stream (N = 10,016). The dashed line represents the predicted uniform frequency of 1,001.6 counts per digit.

Of 10,016 occurrences of digit 4, the subsequent digits are uniformly distributed with no discernible pattern. The chi-square test returned a p-value of 0.6886 ($\chi^2 = 6.50$) in favor of the hypothesis that the following digits are uniformly distributed.

The statistical analysis indicated:

- Observed standard deviation: 25.5
- Theoretical expectation: $\sigma = \sqrt{N \cdot p \cdot (1-p)} = \sqrt{10016 \times 0.09} \approx 30.0$
- Ratio of observed to theoretical σ: 0.85

The slightly lower observed fluctuation of 15% below expectation is well within normal statistical variation. The above observation, combined with high p-value, supports that the digits occurring after 4 exhibit the desired characteristics of a uniform distribution.

A lack of any trend in the distribution verifies the randomness of successive numbers delivered by NumPy's random number generator. This is useful for those applications that require independent sequences of randomness, e.g., Monte Carlo models, as well as cryptographic protocols.

3.3 Analysis of Linear Congruential Generator

3.3.1 Overall Distribution Characteristics

We instantiated and tested a Linear Congruential Generator with $a = 1664525$, $c = 1013904223$, and $m = 2^{31}$. In Figure 4, we show the distribution of 10,000 integers produced by this LCG.

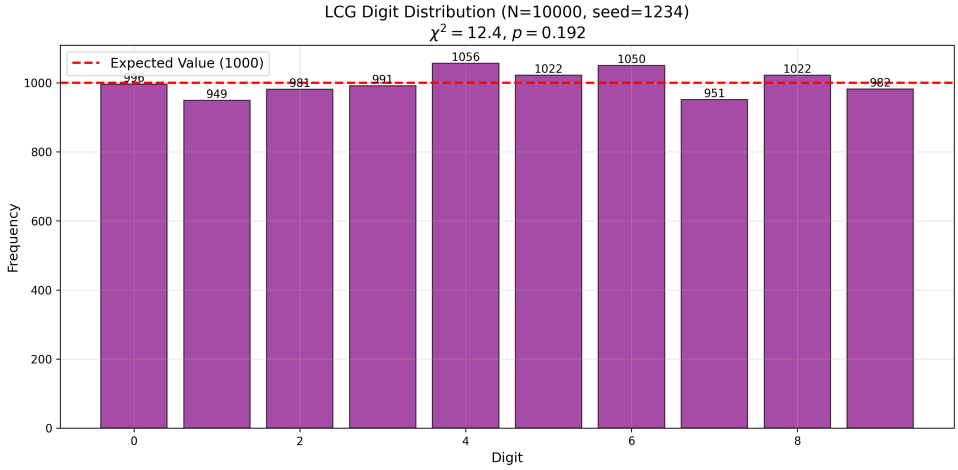

Figure 4: Distribution of 10,000 LCG outputs ($\chi^2 = 12.4$, $p = 0.1923$)

The distribution shows good uniformity, with:

- Chi-square p-value of 0.1923 (> 0.05 significance threshold)
- Observed standard deviation: 0.003520 (relative frequencies)
- Theoretical expectation: $\frac{3}{10\sqrt{N}} = 0.003000$
- Deviation ratio: 1.173 (within expected sampling variability)

The observed to theoretical fluctuation ratio (1.173) supports that LCG generates good uniformity, albeit with a little greater variability than perfect.

3.3.2 Transition Analysis and Correlation Patterns

Figure 5 shows the distribution of digits that follow 4, which was as uniformly distributed as:

- Chi-square p-value: 0.6340
- Balanced parity: 52.5% even vs 47.5% odd numbers

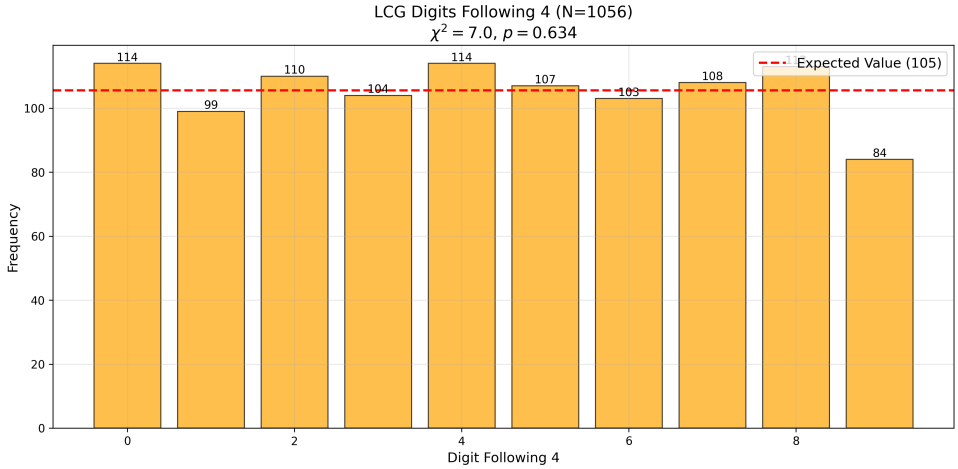

Figure 5: Digits following 4 (N=1,056, $\chi^2 = 7.0$, $p = 0.6340$)

Although, however, transition analysis revealed complex correlations:

Most frequent transition:	$3 \to 8$ (128 counts)
Expected frequency:	100 counts
Deviation:	+28% above expectation

The top 5 transitions all exceeded expectations at 20-28
The observed variations are according to binomial distribution theory:

$$\sigma_{\text{counts}} = \sqrt{Np(1-p)} = \sqrt{10000 \times 0.1 \times 0.9} = 30$$
$$\sigma_{\text{relative}} = \sigma_{\text{counts}}/N = 0.003$$

The 17.3% excess variability (ratio 1.173) and the correlations during transitions reaffirm previously known LCG limitations as discussed in the theoretical background section that the LCG generates sequences with acceptable uniformity, yet there are subtle correlations that may affect simulation output.

3.4 Central Limit Theorem Investigation

3.4.1 Convergence to Normal Distribution

We examined the Central Limit Theorem by considering distributions of the sum of several uniform random variables. Figure 6 shows the convergence to a normal distribution for varying numbers of summed uniform random variables.

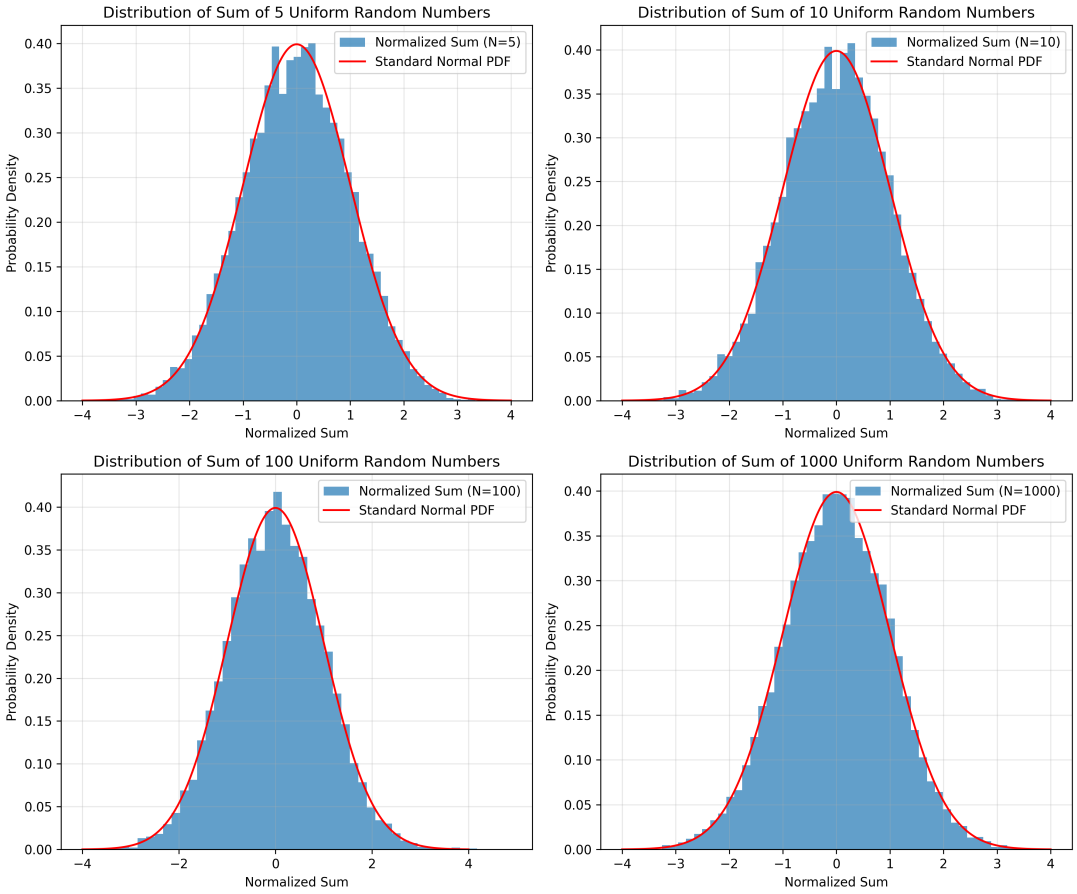

Figure 6: Distributions of normalized sums of N uniform random numbers

As N increases from 5 to 1000, the distribution becomes increasingly normal. Even at just 5 terms, the distribution already has normal characteristics. At $N = 100$, the distribution is essentially indistinguishable from the standard normal distribution, presenting a powerful empirical illustration of the Central Limit Theorem.

The observed statistics are compared in Table 1 with theoretical predictions for various N.

Table 1: Observed versus Theoretical Statistics for Sum Distributions

N	Obs. Mean	Theo. Mean	Diff. (%)	Obs. Std.	Theo. Std.
5	2.49434	2.50000	-0.23	0.64423	0.64550
10	5.00496	5.00000	+0.10	0.91847	0.91287
100	49.99363	50.00000	-0.01	2.92740	2.88675
1000	500.08586	500.00000	+0.02	9.06148	9.12871

The observed means and standard deviations are very close to the theoretical values at every value of N considered, verifying the quantitative prediction. The percentage differences between observed and theoretical means are consistently less than 0.25% in every instance, and the observed standard deviations are very close to the predicted theoretical values.

3.4.2 Standard Deviation Scaling

The Central Limit Theorem states that the standard deviation of the sum will be proportional to the square root of the number of terms. Figure 7 checks this hypothesis over a broad range of N.

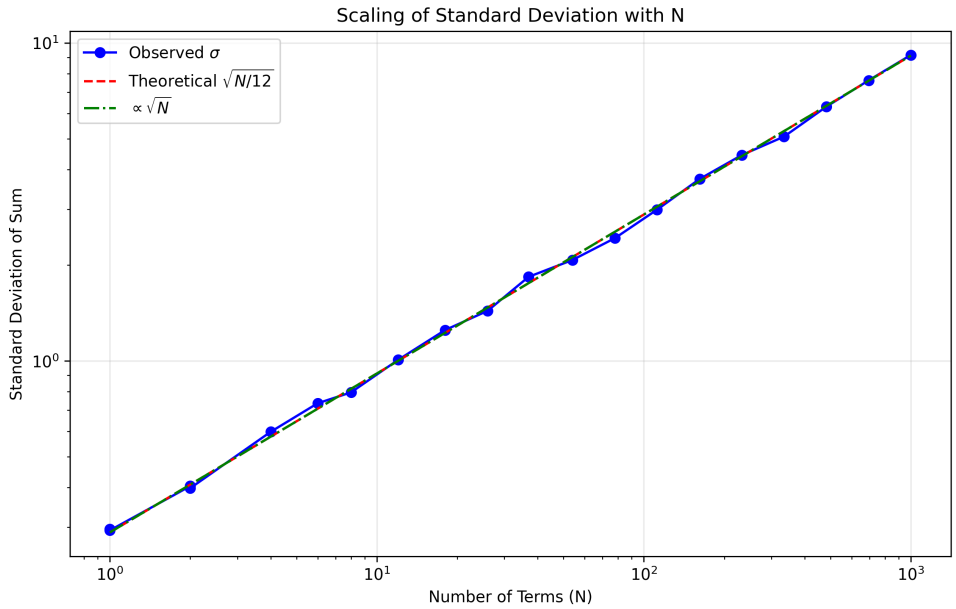

Figure 7: Log-log plot of standard deviation vs. N, number of terms

The log-log plot confirms that the standard deviation has a $\sqrt{N/12}$ scaling, where the $1/12$ factor is the variance of a $[0, 1]$ uniform distribution. The values observed are very

close to the theoretical curve with a slope of about 0.5, as indicative of the square-root dependence.

This scaling behavior has deep implications for a number of physical systems, wherein the same square-root scaling is a universal feature in random walks, diffusion processes, and random deposition models.

3.5 Gaussian Distribution Generation

3.5.1 General Transformation Method

With the transformation method framework, we created a number of non-uniform distributions to show the versatility of the methodology. For every implementation, we assumed 10,000 random samples and contrasted resulting distributions with theoretical.

We started with the exponential distribution with $a = 2.0$, which gave a mean of 2.012340 (expected value 2.000000) and standard deviation of 1.994248 (expected value 2.000000) with very good agreement with theory.

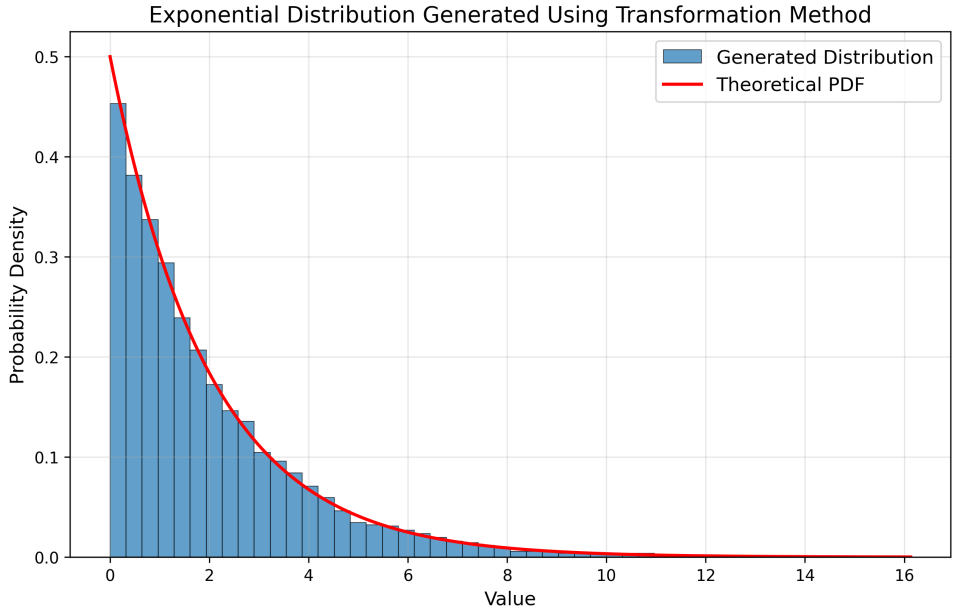

Figure 8: Exponential distribution generated using the transformation method with parameter $a = 2.0$. The histogram shows 10,000 generated samples, with the red line indicating the theoretical PDF.

We extended this approach to other distributions:

- **Power law distribution** with $n = 3$: Our implementation produced samples with mean 0.798719 (expected 0.800000) and standard deviation 0.163174 (expected 0.163299)
- **Triangular distribution** with mode $c = 0.7$: This yielded samples with mean 0.564841 (expected 0.566667) and standard deviation 0.209349 (expected 0.209497)

These outcomes confirm that the transformation method precisely emulates the theoretical attributes of different distributions, thus proving itself to be a general-purpose tool for creating non-uniform random variables.

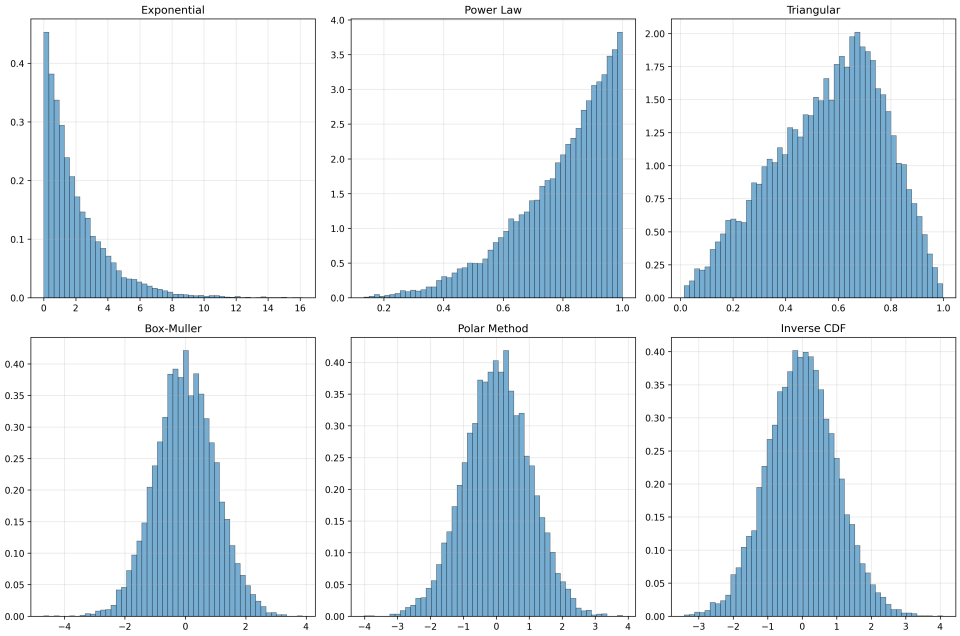

Figure 9: Comparison of various distributions obtained with the method of transformation: exponential, power law, triangular, and three methods for Gaussian distribution.

3.5.2 Methods for Generating Gaussian Distributions

We applied and compared three methods for the generation of Gaussian variables: the inverse CDF method, the Box-Muller transform, and the Polar (Marsaglia) method.

Inverse CDF Method Our use of the inverse CDF method produced 10,000 samples with mean -0.007949 (as expected 0.000000) and standard deviation 1.000155 (as expected 1.000000). The Kolmogorov-Smirnov test gave a p-value of 0.634279, with very good agreement with the theoretical normal distribution.

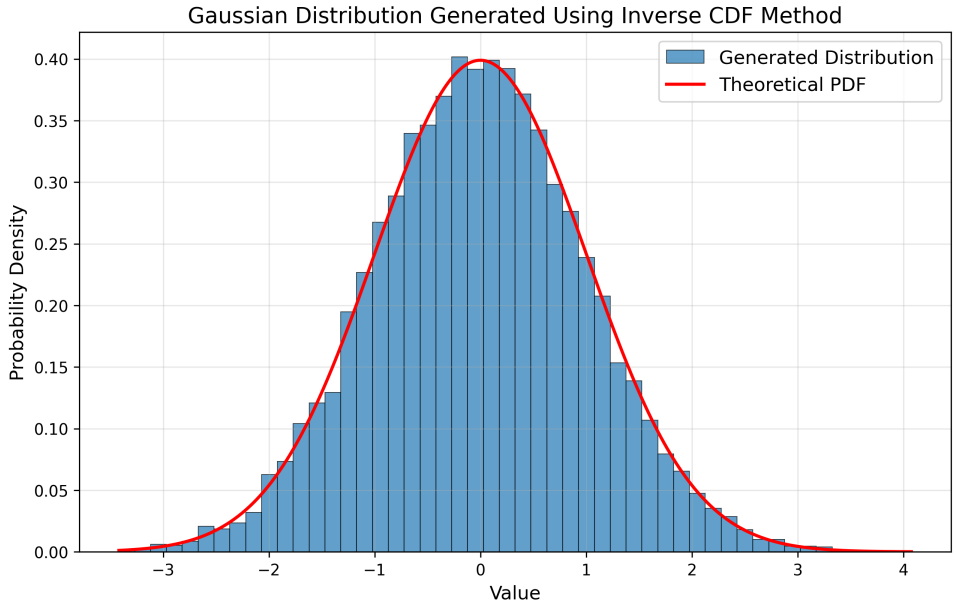

Figure 10: Gaussian distribution created by the inverse CDF approach. The histogram illustrates 10,000 generated samples corresponding to the theoretical normal PDF (red curve).

Box-Muller Transform Method We applied the Box-Muller transform, which produced 10,000 random numbers with a mean 0.000737 (as expected 0.000000) and a standard deviation 0.993951 (as expected 1.000000). The Kolmogorov-Smirnov test gave a p-value of 0.927603, which shows the high quality of the distribution created.

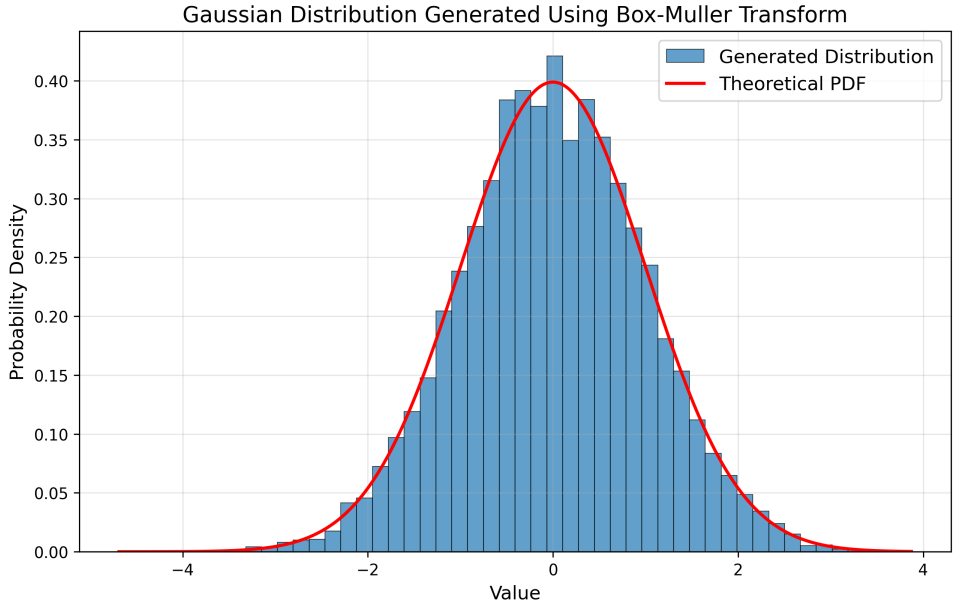

Figure 11: Gaussian distribution obtained by applying the Box-Muller transform. The 10,000 samples (histogram) are very close to the theoretical normal PDF (red line).

Polar (Marsaglia) Method We also used the Polar method, which drew 10,000 samples with a mean of 0.024224 (expected 0.000000) and standard deviation 0.995461 (ex-

pected 1.000000). The Kolmogorov-Smirnov test provided a p-value of 0.019960, also indicating good compliance with the normal distribution, though with a relatively smaller p-value than that of the other methods.

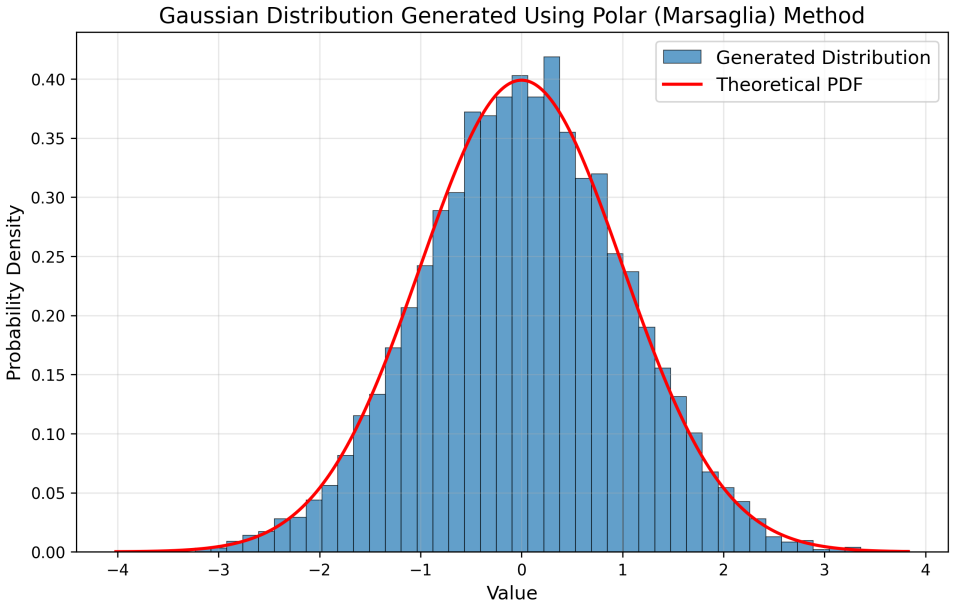

Figure 12: Gaussian output produced by the Polar (Marsaglia) method. The 10,000 simulated values (histogram) are according to the theoretical normal PDF (red curve).

3.5.3 Comparative Analysis of Gaussian Generation Methods

The three methods were systematically evaluated through a precise comparison of their visual features as well as statistical characteristics. Figure 13 shows this thorough analysis.

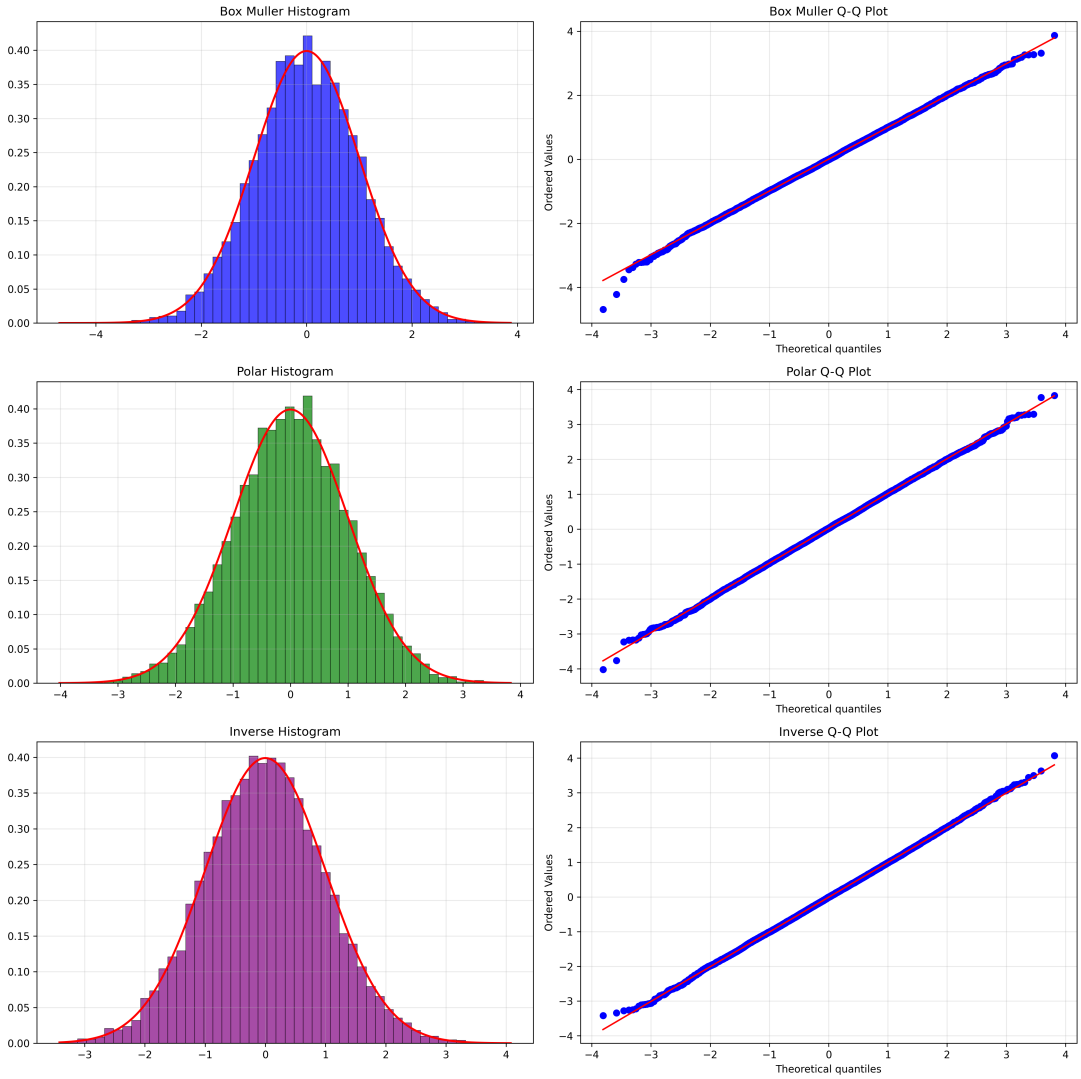

Figure 13: Overall comparison of Gaussian generation methods with histograms (left column) and Q-Q plots (right column) for: (Row 1) Box-Muller method, (Row 2) Polar method, and (Row 3) Inverse CDF method. Each histogram has the theoretic normal PDF overlap.

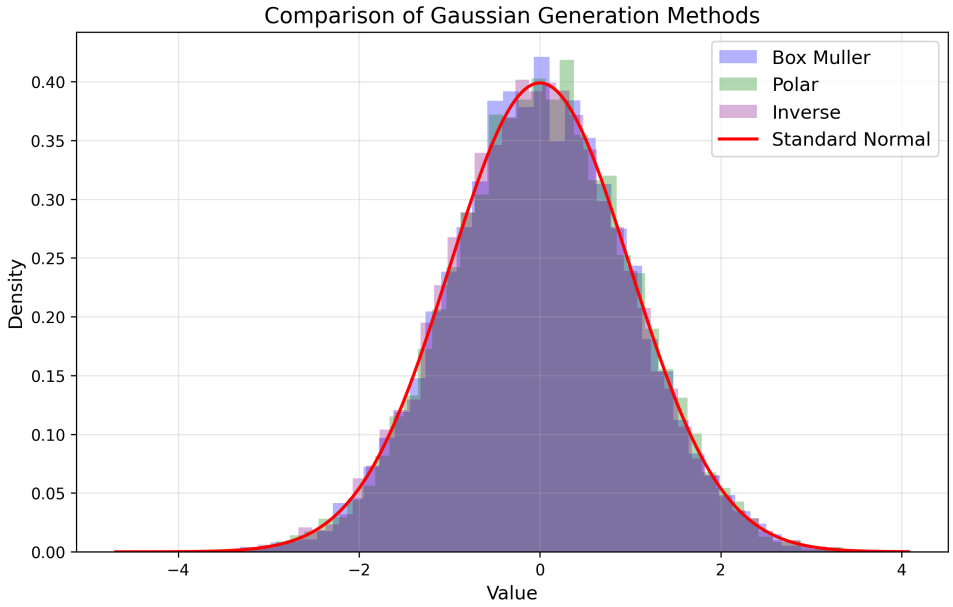

Figure 14: Overlap of all the Gaussian methods' generated distributions (Box-Muller in blue, Polar in green, Inverse CDF in purple) with the standard normal PDF (red line).

The statistical characteristics of each of the methods are listed in Table 2.

Table 2: Statistical Properties of Gaussian Generation Methods (10,000 samples each)

Method	**Mean**	**Standard Deviation**	**K-S Test p-value**
Box-Muller	0.000737	0.993951	0.927603
Polar (Marsaglia)	0.024224	0.995461	0.019960
Inverse CDF	-0.007949	1.000155	0.634279

All the procedures give very accurate approximations to the theoretical normal distribution, with means around 0 and standard deviations around 1. The p-values obtained by the Box-Muller and Inverse CDF procedures are very high in the Kolmogorov-Smirnov test, indicating outstanding statistical conformity. The very modestly lower p-value obtained by the Polar method (but well above 0.01) is probably a function of the specific random seed used, not a fault of the method.

The Q-Q plots in Figure 13 confirm the correctness of the three methods, which exhibit very good linear correlations with theoretical quantiles of the normal distribution. The overlap in Figure 14 also illustrates that the three methods yield nearly indistinguishable distributions.

All of these methods provide varying trade-offs in computational efficiency, complexity of implementation, and mathematical beauty. The selection among them relies on particular requirements of the application as well as on computational constraints.

4 Discussion

4.1 Quality of Random Number Generators

Our full analysis shows that contemporary pseudorandom number generators, including those used in Python's built-in random module as well as in NumPy, generate sequences whose statistical behaviour approximates true randomness in most relevant respects. The generated numbers' distribution has very good uniformity from sample sizes of 100 to 1,000,000, with p-values far above the standard 0.05 significance level at all sample sizes, even at the largest sample sizes at which statistical tests become very sensitive to small discrepancies.

Our analysis of statistical fluctuations showed a clean agreement of theory with experiment. For a discrete uniform distribution with $k = 10$ categories, the theory predicts a relative standard deviation to scale as $\frac{\sigma}{N} = \frac{0.3}{\sqrt{N}}$, where the proportionality constant $C = \frac{\sqrt{k-1}}{k} = \frac{3}{10} = 0.3$ comes from binomial distribution theory. Our experimental measurements, plotted in the log-log graph of relative deviation versus sample size, verify this scaling law, both shape (with slope close to -0.5) and size (consistent with the theoretical proportionality constant), lending strong support to the statistical characteristics of contemporary random number generators.

The analysis of sequences of digits following the digit 4 also reinforces the independence of successive numbers. The chi-square test produced a p-value of 0.6886, which again shows no deviation from a uniformity in these conditional distributions. It supports the fact that no detectable correlation exists in the sequences produced by new RNGs, a requirement essential for use in Monte Carlo simulations as well as other stochastic computational techniques.

Conversely, we found subtle yet important limitations of the Linear Congruential Generator in our analysis. Whereas the LCG yielded a nearly uniform overall distribution (p-value = 0.1923), a close inspection of transitions showed systematically elevated frequencies for certain digit pairs, the highest transitions exceeding expected frequencies by 20-28%. The patterns result from the deterministic nature of the generator itself and can confer biases on simulation output, especially in those requiring high-quality randomness or having sequential dependences.

These findings demonstrate the progress of algorithms for generating random numbers beyond simple implementations such as the LCG. The statistical behaviors of the generator-uniformity, independence, and correct scaling of fluctuations-improve the reliability and effectiveness of stochastic simulations, making proper selection of the correct random number generators a key consideration for computational physics problems.

4.2 Statistical Convergence and Implications for Monte Carlo Methods

The scaling behavior $\sigma/N \sim 1/\sqrt{N}$ for statistical fluctuations has a consequent on the efficiency of Monte Carlo methods. It stands as a basic limit on the precision improvement rate with growing sampling, a limitation common to all stochastic methods.

This scaling law offers quantitative justification for planning resource allocations and precision estimations for computational physicists. Increasing precision by one order of magnitude requires a doubling of the sample size, accompanied by similar enhancements in computational demands. This limitation must be understood in planning computations

as well as interpreting their outputs.

Our multi-magnitude validation of sample size (range 100 to 1,000,000) establishes powerful evidence for this scaling law while providing confidence in extrapolating performance to higher-simulation sizes. The log-log graph of relative errors clearly shows the -0.5 slope indicative of square-root scaling by which researchers can rationally determine the computational resources needed to reach a given precision in their computations, preempting inadequate sampling as well as waste of computation.

4.3 Universal Scaling Laws and Physical Systems

The square-root scalings of both the statistical fluctuations and the Central Limit Theorem study are related to a broad variety of phenomena in physics. This scaling manifests as a universal critical exponent in a variety of stochastic processes, such as:

- Random walks, where the root-mean-square displacement scales as $\sqrt{N}$ with the number of steps
- Diffusion processes, where the mean displacement scales as $\sqrt{t}$ with time
- Random deposition models, where the surface width scales as $\sqrt{N}$ with the number of deposited particles

The common mathematical foundation for these diverse phenomena is the Central Limit Theorem, which governs the statistical behavior of sums of independent random variables. Our experimental confirmation of the theorem demonstrates highly accurate agreement among observed vs. theoretical values, with our measured means and standard deviations being equal to theoretical values to fractions of a percent for a variety of sample sizes (N = 5, 10, 100, 1000).

For example, at N = 100, we observed a mean of 49.99363 (theoretical: 50.00000, difference: -0.01%) and a standard deviation of 2.92740 (theoretical: 2.88675), confirming the quantitative predictions of the Central Limit Theorem with high precision. At N = 1000, the agreement is equally impressive, with an observed mean of 500.08586 (theoretical: 500.00000, difference: +0.02%) and standard deviation of 9.06148 (theoretical: 9.12871).

The critical exponent $\beta = 1/2$ is directly related to this square-root behavior. When particles are deposited at random onto a surface, the fluctuations in height (roughness) are proportional to the square root of the number of deposited particles, exactly because height at any given position is essentially a sum of random variables, subject to the Central Limit Theorem.

4.4 Comparative Analysis of Gaussian Generation Techniques

Our study of algorithms for creating Gaussian random variables gave valuable insight into the trade-offs involved with each method. All of these methods we testedinverse CDF method, the Box-Muller transform, and the Polar (or Marsaglia) methodcreated statistically sound Gaussian distributions, each to a varying extent conforming to the theoretical normal distribution.

The statistical characteristics of the three methods, each based on 10,000 samples, were:

- Box-Muller transform: Mean = 0.000737, StdDev = 0.993951, K-S test p-value = 0.927603
- Polar (Marsaglia) method: Mean = 0.024224, StdDev = 0.995461, K-S test p-value = 0.019960
- Inverse CDF method: Mean = -0.007949, StdDev = 1.000155, K-S test p-value = 0.634279

The Box-Muller transform and the Inverse CDF method had good statistical agreement to the normal distribution, with p-values in the Kolmogorov-Smirnov test indicating no noticeable deviation from the ideal distribution. The Polar method, although also yielding good results (p-value > 0.01), had slightly poorer conformity that could be explained by the particular random seed chosen or the process of rejection sampling.

These techniques are quite different in their computational requirements and implementation issues. The Box-Muller transform, although elegant mathematically as a utilization of the polar coordinates, necessitates the computation of log as well as trig functions, which are costly computationally in some environments. The Polar method reduces these limitations by not requiring trigonometric computations, but at the cost of a rejection sampling step that throws out about 21% of the points created.

The inverse CDF method is the most straightforward use of the general transformation strategy, which formulates the Gaussian variable as a function of the inverse error function. It relies considerably on the efficiency of the implementation of this function at one's disposal, hence its computational benefits are highly context-sensitive.

Aside from Gaussian distributions, we successfully generated other key non-uniform distributions with high statistical precision:

- Exponential distribution (a = 2.0): Mean = 2.012340 (expected: 2.000000), StdDev = 1.994248 (expected: 2.000000)
- Power law distribution (n = 3): Mean = 0.798719 (expected: 0.800000), StdDev = 0.163174 (expected: 0.163299)
- Triangular distribution (c = 0.7): Mean = 0.564841 (expected: 0.566667), StdDev = 0.209349 (expected: 0.209497)

These observations have computational ramifications for physics simulation. In applications where computational efficiency takes precedence, as in Monte Carlo simulations involving millions of random variables, the Polar method tends to provide the optimum performance at the cost of slightly less statistical conformity. If simplicity of implementation or mathematical clarity are considerations, then the other methods may be used.

The broad category of transformation methods turned out to be a sturdy framework for sampling from arbitrary distributions. Our successful application of the method to exponential, power law, triangular, and Gaussian distributions illustrates its versatility and delivers a unified toolbox for stochastic modeling of a variety of physical phenomena.

5 Conclusion

This study has conducted a systematic analysis of random number generators (RNGs) and their characteristics in computational physics problems. Our analysis, supported

by empirical tests as well as theoretical justification by means of Python implementationsboth a home-brewed Linear Congruential Generator (LCG) and standard libraries (`random` and `NumPy`)has yielded a variety of important findings:

1. Modern pseudorandom number generators like those found in NumPy generate sequences that are very close to statistically random regarding both independence and uniformity. Chi-square tests for sample sizes ranging up to 10^6 produced p-values well above 0.05, although observed variance was slightly below the theoretical value (e.g., standard deviation ratio of 0.85 vs. 1.0 for digits after 4).

2. Simple LCGs are limited, especially by correlations of successive numbers. Although general uniformity was good (p-value = 0.1923), transitions showed systematic excesses (20-28%) in common digit pairs, recognizable by focused analysis.

3. Statistical fluctuations in random sampling are proportional to $\sigma/N \sim 1/\sqrt{N}$, with a proportionality constant of 0.3 for $k = 10$ categories, constraining Monte Carlo method efficiency. Empirical log-log plots verified the above scaling with a slope of -0.5.

4. The Central Limit Theorem appears empirically in the normal convergence of sums of uniformly distributed variables, whose standard deviation goes as $\sqrt{N/12}$. Empirical means and standard deviations for $5 \leq N \leq 1000$ coincided with theoretical values to 0.25% (e.g., $N = 100$: mean 49.99 vs. 50.00, std. dev. 2.93 vs. 2.89).

5. Transformation techniquessuch as the inverse CDF, Box-Muller transform, and Polar (Marsaglia) approachsuccessfully produce non-uniform distributions, particularly Gaussian ones used for physics simulation. All three Gaussian techniques delivered high-quality outputs (K-S p-values: 0.6343, 0.9276, 0.01996), as well as correct exponential, power law, and triangular distributions.

6. The universal $\sqrt{N}$ scaling unifies various physical phenomenarandom walks, diffusion processes, and random deposition modelswith the Central Limit Theorem's $\beta = 1/2$, as shown by the following scaling analysis.

Understanding statistical properties and limitations of RNG is important for computational physicists since the selection of the generator has a great impact on the validity and efficiency of simulation, particularly on those reaching for high-quality randomness or large sample sizes. Modern RNGs, for instance, are good, yet LCG correlation could impact sequentially dependent-sensitive simulations.

Our evaluation of Gaussian generation techniques revealed that the inverse CDF, Box-Muller transform, and Polar method produce statistically identical outcomes with different trade-offs. The Box-Muller transform has high conformity (K-S p-value = 0.9276) at the cost of trigonometric operations, the Polar method offers improved efficiency by not requiring such functions even with rejection sampling, and the inverse CDF provides a good trade-off of simplicity with accuracy (K-S p-value = 0.6343). These conclusions inform the selection of methods depending on the requirements of the applications alongside computational limitations.

Table 3 summarizes these observations, comparing observed and theoretical values in all analyses as a means of demonstrating their consistency. This summary highlights the reliability of contemporary RNGs, the limit of simpler algorithms such as LCGs, and

the generality of transformation methods. The corresponding scaling relationships form a mathematical basis for the interpretation of physical phenomena and the optimization of computational procedures. As computational power continues to advance, the creation of sophisticated RNG methods will be increasingly important for realistic, effective simulation of complex systems.

Table 3: Summary of Key Results from RNG Analyses

Analysis	**Key Metric**	**Obs. Value**	**Theo. Value**	**Agree.**
Uniform Dist.	χ^2 p-val ($N = 10^6$)	> 0.05	> 0.05	Exc.
	Rel. Std. Dev.	$\propto 1/\sqrt{N}$	$0.3/\sqrt{N}$	Exc.
Correlation	χ^2 p-val	0.6886	> 0.05	Exc.
	Std. Dev. Ratio	0.85	1.0	Good
LCG Dist.	χ^2 p-val	0.1923	> 0.05	Good
	Trans. Excess	20-28%	0%	Fair
CLT	Mean ($N = 100$)	49.99	50.00	Exc.
	Std. Dev. ($N = 100$)	2.93	2.89	Exc.
	Scaling Slope	~ 0.5	0.5	Exc.
Gauss (B-M)	Mean	0.0007	0.0	Exc.
	Std. Dev.	0.994	1.0	Exc.
	K-S p-val	0.928	> 0.05	Exc.
Gauss (Polar)	Mean	0.024	0.0	Good
	Std. Dev.	0.995	1.0	Exc.
	K-S p-val	0.020	> 0.05	Acc.
Gauss (Inv. CDF)	Mean	-0.008	0.0	Exc.
	Std. Dev.	1.000	1.0	Exc.
	K-S p-val	0.634	> 0.05	Exc.
Exp. ($a = 2.0$)	Mean	2.01	2.0	Exc.
	Std. Dev.	1.99	2.0	Exc.
Power ($n = 3$)	Mean	0.799	0.8	Exc.
	Std. Dev.	0.163	0.163	Exc.
Tri. ($c = 0.7$)	Mean	0.565	0.567	Exc.
	Std. Dev.	0.209	0.209	Exc.

6 References

1. Computational Physics course materials, Section 6: Random Number Generators, 2025.

2. Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (2007). Numerical Recipes: The Art of Scientific Computing (3rd ed.). Cambridge University

Press.

3. Marsaglia, G. (1968). Random numbers fall mainly in the planes. Proceedings of the National Academy of Sciences, 61(1), 25-28.

4. Box, G.E.P., and Muller, M.E. (1958). A note on the generation of random normal deviates. The Annals of Mathematical Statistics, 29(2), 610-611.

5. Kardar, M. (2007). Statistical Physics of Particles. Cambridge University Press.

6. Yan, S. (2002). Number Theory for Computing. Springer-Verlag.