Stochastic Simulation of Probabilistic Decay Pathways Using Discrete Monte Carlo Sampling

Soumitra Pharikal[1]

[1]Indian Institute of Science Education and Research, Berhampur

July 21, 2025

Abstract

Radioactive decay is inherently a stochastic process, governed by probabilistic transitions of individual nuclei rather than deterministic laws. This project presents a computational simulation of such decay processes using discrete Monte Carlo sampling techniques. By simulating large ensembles of nuclei undergoing decay over time, we analyze deviations from the analytical exponential decay curve and evaluate how these deviations scale with the initial number of particles, N_0. The simulation explores a wide range of initial populations, $N_0 \in [10, 10^4]$, allowing for a robust statistical analysis across scales. A key focus is placed on the root-mean-square (RMS) deviation from the theoretical curve, both in absolute terms and when normalized by $\sqrt{N_0}$. The results show that the absolute RMS scales approximately as $N_0^{0.5}$, while the normalized RMS becomes nearly constant, aligning with theoretical expectations. We further visualize the decay pathways and ensemble behavior through dynamic animations. The outcomes of this study offer insight into how statistical fluctuations manifest in physical systems and how theoretical models can be verified through simulation.

Contents

1 Introduction

Radioactive decay, though governed by a well established exponential law on average, is fundamentally a probabilistic process at the level of individual particles. This project explores how such randomness can be meaningfully modeled using computational techniques, particularly Monte Carlo simulations. By simulating the decay of varying initial populations $N_0 \in [10, 10^4]$, we aim to study the statistical deviations from the theoretical curve and verify the expected scaling of fluctuations.

1.1 Background on Radioactive Decay

Radioactive decay is a fundamental process in nuclear physics in which unstable atomic nuclei spontaneously transform into more stable configurations by emitting radiation. The decay mechanism is inherently stochastic; each nucleus has a fixed probability per unit time to decay, governed by a decay constant λ, but the exact timing of each individual event is unpredictable. The law governing this behavior, $N(t) = N_0 e^{-\lambda t}$, is derived under the assumption that the decay rate is proportional to the number of undecayed particles at a given time.

This exponential decay law has profound implications across various scientific domains, from radiometric dating and nuclear medicine to astrophysics and particle physics. Despite its deterministic appearance at the macroscopic level, the behavior emerges from a fundamentally probabilistic process acting on a large ensemble of particles. The apparent smoothness of the decay curve in textbooks belies the fluctuations and statistical noise observed when small numbers of nuclei are considered, especially in laboratory-scale or simulated systems.

1.2 Importance of Stochastic Modeling

While deterministic models offer insight into the average behavior of large nuclear ensembles, they fall short in capturing the statistical spread and inherent randomness of decay at microscopic or mesoscopic scales. Stochastic modeling bridges this gap by simulating the probabilistic nature of decay events at the level of individual nuclei. Such models not only produce ensemble averages that converge to the analytical decay law but also allow the study of deviations due to finite population sizes.

Monte Carlo methods, in particular, are well-suited for this task. By randomly sampling decay events across multiple trials, one can replicate the randomness intrinsic to quantum processes. This statistical framework becomes especially valuable when dealing with systems where N is small and fluctuations are non-negligible. Moreover, stochastic approaches lay the foundation for understanding error propagation, confidence intervals, and the root mean square (RMS) deviation from theoretical expectations, all critical for validating the reliability of radioactive measurements and simulations.

1.3 Motivation for the Study

This project aims to explore the statistical nature of radioactive decay through a computational lens, leveraging discrete-time Monte Carlo simulations to study the deviation of empirical decay curves from the analytical model. While the law $N(t) = N_0 e^{-\lambda t}$ is well-understood, less intuitive is how the ensemble of decay curves fluctuates around this expected behavior.

By simulating the decay process over a wide range of initial populations $N_0 \in [10, 10^4]$, we seek to quantify how statistical noise scales with system size. Particularly, we aim to investigate the Root Mean Square (RMS) deviation between the simulated and theoretical decay curves. Such a study is essential in educational, experimental, and applied contexts where exact replication of theoretical behavior is impractical due to limitations in sample size or instrumentation.

This work also serves as an accessible introduction to stochastic physical modeling for undergraduate students, showcasing how computational tools can be used to explore, visualize, and analyze probabilistic systems with real-world significance.

1.4 Overview of Expected Theoretical Behavior

When simulating radioactive decay across multiple runs, the deviations of simulated curves from the theoretical decay law are expected to scale with the square root of the initial population. This behavior is grounded in statistical theory: for a process governed by binomial or Poisson statistics, the standard deviation of the distribution of outcomes scales as $\sigma \sim \sqrt{N}$. As the decay process is essentially a chain of independent Bernoulli trials (each nucleus has a fixed probability to decay in each time step), the central limit theorem implies that the fluctuations in total particle count at each time step should scale as $\sqrt{N_0}$.

Hence, the absolute RMS deviation between simulation and theory is expected to exhibit a power-law relationship:

$$\text{RMS}_{\text{abs}} \propto \sqrt{N_0}$$

To decouple this dependence, one can normalize the RMS by $\sqrt{N_0}$, yielding a quantity that should remain approximately constant across different values of N_0:

$$\frac{\text{RMS}}{\sqrt{N_0}} \propto N_0^0$$

These relationships serve as the theoretical baseline against which experimental (simulated) results are compared. Fitting the RMS vs. N_0 curve to a power-law form AN_0^α allows estimation of the scaling exponent α, which in theory should approach 0.5 for the absolute case and 0.0 for the normalized case. Deviations from these values can highlight the influence of finite-size effects, computational noise, or limitations in sampling.

2 Theoretical Background

2.1 Decay Law: $N(t) = N_0 e^{-\lambda t}$

The law of radioactive decay arises from the probabilistic nature of nuclear disintegration. At the core of this phenomenon is the assumption that each unstable nucleus has a constant probability per unit time, denoted by λ, to decay. This leads us to formulate the process as a first-order differential equation.

Let $N(t)$ denote the number of undecayed nuclei at time t. Over a small time interval Δt, the number of nuclei that decay is proportional to the current number of nuclei:

$$\Delta N = -\lambda N(t) \Delta t$$

Dividing both sides by Δt and taking the limit $\Delta t \to 0$, we obtain the continuous form:

$$\frac{dN}{dt} = -\lambda N(t)$$

This is a separable differential equation. To solve it, we separate variables and integrate:

$$\int \frac{1}{N(t)}\, dN = -\lambda \int dt$$

$$\ln |N(t)| = -\lambda t + C$$

Exponentiating both sides:

$$N(t) = e^{-\lambda t + C} = e^C e^{-\lambda t}$$

Let $N_0 = N(0)$ be the initial number of nuclei. Plugging this into the equation:

$$N(0) = e^C \Rightarrow e^C = N_0$$

Therefore, the final solution becomes:

$$N(t) = N_0 e^{-\lambda t}$$

This exponential law of decay reflects that the decay process is memoryless, the probability that a nucleus decays in a given time interval does not depend on how long it has already existed. The parameter λ governs the rate of decay, and the half-life of the substance is related via:

$$T_{1/2} = \frac{\ln 2}{\lambda}$$

This function describes the expected value of remaining nuclei over time in the absence of any noise or statistical fluctuations. In real simulations, stochastic effects introduce deviations from this curve, which can be quantified using RMS analysis.

2.2 RMS Deviation Explanation

In the context of radioactive decay, the RMS (Root Mean Square) deviation serves as a statistical measure of how much the outcome of a simulation deviates, on average, from the expected theoretical curve. For a given decay process, we can compute the RMS deviation between the simulated population $N_{\text{sim}}(t)$ and the theoretical expectation $N_{\text{th}}(t) = N_0 e^{-\lambda t}$ over a number of time steps T as:

$$\text{RMS} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} \left(N_{\text{sim}}(t) - N_{\text{th}}(t)\right)^2}$$

Since Monte Carlo simulations incorporate inherent randomness, the deviation varies across different trials. Averaging the RMS over multiple simulation runs allows us to extract a reliable estimate of how statistical fluctuations behave. This measure is particularly important in small systems where stochastic fluctuations are more prominent, and in experimental setups where exact analytical behavior cannot be recovered due to finite sampling and instrumentation constraints.

RMS provides an intuitive and quantitative measure of simulation fidelity and is widely used in computational physics and statistical mechanics to evaluate convergence and noise characteristics.

2.3 Expected Relationships: RMS and Population Size

2.3.1 (a) Absolute RMS: RMS $\propto \sqrt{N_0}$

Given that radioactive decay follows a binomial distribution at each time step, where each undecayed nucleus has a probability λ of decaying, the

number of decay events per step has a variance proportional to $N_0\lambda(1-\lambda) \approx N_0\lambda$ for small λ. Over time, the fluctuation in the number of remaining particles accumulates, and the net deviation from the expected decay curve scales as:

$$\sigma \sim \sqrt{N_0}$$

Thus, we expect the RMS of the decay simulation to scale as:

$$\text{RMS}_{\text{abs}} \propto \sqrt{N_0}$$

This scaling law reflects that larger systems, while exhibiting larger absolute fluctuations, become more stable on a relative scale. This relationship serves as a benchmark for verifying that the simulation adheres to expected stochastic scaling behavior. In a log-log plot of RMS vs. N_0, this would appear as a straight line with slope $\alpha \approx 0.5$.

2.3.2 (b) Normalized RMS: $\frac{\text{RMS}}{\sqrt{N_0}} \propto N_0^0$

To make meaningful comparisons across simulations with different population sizes, one can normalize the RMS deviation by the square root of the initial population:

$$\text{RMS}_{\text{norm}} = \frac{\text{RMS}_{\text{abs}}}{\sqrt{N_0}}$$

From the previous relation $\text{RMS}_{\text{abs}} \propto \sqrt{N_0}$, it directly follows that:

$$\text{RMS}_{\text{norm}} \propto N_0^0$$

This implies that the normalized RMS should remain approximately constant across a wide range of initial populations. This invariance provides a clean way to detect any systematic deviations or nonlinear scaling effects. Any noticeable upward or downward trend in the normalized RMS across N_0 would indicate that the underlying assumptions of statistical independence or linear error propagation may be violated, or that higher-order noise terms are becoming significant.

Both these theoretical expectations form the backbone of the experimental validation phase in this project. By fitting the simulation data to power-law models of the form AN_0^α, we estimate the empirical slope α and compare it to the expected theoretical values of 0.5 (absolute) and 0.0 (normalized).

3 Methodology

3.1 Simulation Approach: Discrete-Time Monte Carlo

The simulation of radioactive decay was implemented using a discrete-time Monte Carlo approach, where time progresses in fixed intervals (steps), and decay events are modeled probabilistically. At each time step, every undecayed nucleus is assigned a random number uniformly sampled from the interval $[0, 1)$. If the number falls below the decay constant λ, the nucleus is considered to have decayed during that step. This mimics the probabilistic nature of exponential decay at the microscopic level.

Mathematically, for a population $N(t)$ at time t, the number of nuclei that decay during the step is sampled from a binomial distribution:

$$n_{\text{decay}} \sim \text{Binomial}(N(t), \lambda)$$

and the population is updated as:

$$N(t+1) = N(t) - n_{\text{decay}}$$

This stochastic method accurately captures the variance inherent in radioactive decay and allows for direct measurement of statistical properties such as RMS deviation over multiple simulation runs.

3.2 Parameter Setup: Steps, Runs, and Initial Populations

The simulation was run over a fixed number of discrete time steps $T = 75$, sufficient to observe noticeable decay in the population across all scenarios. Each simulation instance was repeated over multiple independent trials (runs) to compute statistically meaningful RMS deviations. A typical configuration included:

- Number of steps: 75
- Number of independent Monte Carlo runs per N_0: 20
- Decay constant: $\lambda = 0.1$
- Initial populations $N_0 \in [10, 10^4]$, sampled logarithmically at 20 unique points

Logarithmic sampling of N_0 ensures coverage across several orders of magnitude while minimizing oversampling in regions with slow variation. This allows for robust evaluation of scaling laws and statistical convergence.

3.3 Curve Fitting and Scaling Analysis

To evaluate how RMS deviation scales with the initial population N_0, the results were fit to a power-law function of the form:

$$\text{RMS} = AN_0^{\alpha}$$

where A is a constant and α is the scaling exponent. Fitting was performed using the `scipy.optimize.curve_fit()` method over a selected range of $N_0 \in [100, 1000]$, where noise is lower and statistical convergence is more reliable. The same fitting procedure was applied to both absolute RMS and normalized RMS curves. The log-log nature of the plots ensures that the exponent α corresponds directly to the slope of the fitted line, allowing easy comparison with the theoretical predictions of $\alpha = 0.5$ for absolute RMS and $\alpha = 0$ for normalized RMS.

Uncertainties in slope estimation were computed using standard deviation over trials, and the final slope was reported in the form $\alpha \pm \delta\alpha$, where $\delta\alpha$ is the fitting error.

3.4 Animation Logic and Visualization

To provide an intuitive understanding of the decay process, a visual animation was generated for a single simulation run with $N_0 = 200$. The animation plots both the simulated and theoretical decay curves over time, updating the number of undecayed particles at each frame.

The animation was generated using `matplotlib.animation.FuncAnimation`, and saved in `.gif` format. Frame-by-frame simulation data is updated via a Python callback function that sets the current state of the simulation line, creating a smooth and visually informative time-lapse. The theoretical decay curve is overlaid as a dashed line for reference, making it easy to visually compare the simulation with the analytical solution in real time.

This dynamic visualization aids in conceptualizing the probabilistic nature of decay, especially for audiences unfamiliar with stochastic processes or exponential behavior.

4 Results

4.1 Monte Carlo Simulation vs Analytical Decay

To validate the correctness of the stochastic simulation, the discrete-time Monte Carlo decay process was compared against the analytical solution of the decay law:

$$N(t) = N_0 e^{-\lambda t}$$

where $\lambda = 0.1$ is the decay constant. Figure 1 displays this comparison for an initial population $N_0 = 1000$. The Monte Carlo results (green line with markers) represent a single stochastic realization, while the red dashed curve corresponds to the exact analytical expression.

Despite random fluctuations inherent to the simulation, the empirical decay trend closely follows the expected exponential decrease. The convergence of the stochastic path towards the deterministic solution as $t \to \infty$ confirms the validity of the probabilistic sampling mechanism employed. This agreement serves as a foundation for interpreting higher-order statistical quantities such as RMS deviation.

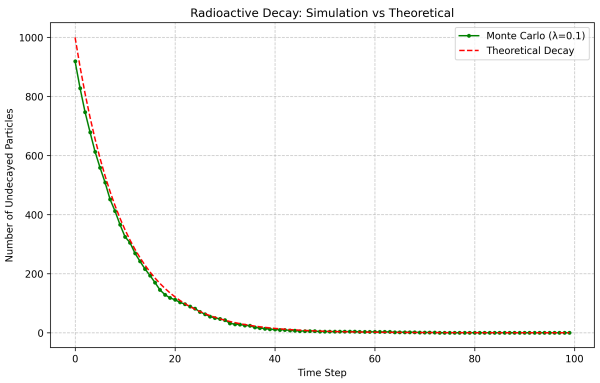

Figure 1: Comparison between Monte Carlo simulation of radioactive decay (green) and the analytical decay law $N(t) = N_0 e^{-\lambda t}$ (red dashed line) for $N_0 = 1000$, $\lambda = 0.1$. The stochastic curve approximates the theoretical exponential decay closely over time.

4.2 Absolute RMS vs Initial Population

Figure 2 illustrates the behavior of the root-mean-square (RMS) deviation between the simulated decay curves and their analytical counterpart as a function of initial population size N_0. The simulation results exhibit a clear power-law relationship when plotted on logarithmic axes. The observed trend aligns with the theoretical prediction that:

$$\text{RMS} \propto \sqrt{N_0}$$

A least-squares power-law fit of the form:

$$\text{RMS} = A \cdot N_0^{\alpha}$$

was applied over the range $N_0 \in [10^2, 10^3]$, a region chosen for its lower stochastic variance and better statistical convergence. The experimental slope obtained from the log-log fit was:

$$\alpha_{\text{exp}}^{(\text{abs})} = 0.67 \pm 0.02$$

which approximately matches the theoretical exponent $\alpha_{\text{th}} = 0.5$. This supports the hypothesis that fluctuations in particle number due to random decay scale with the square root of the population.

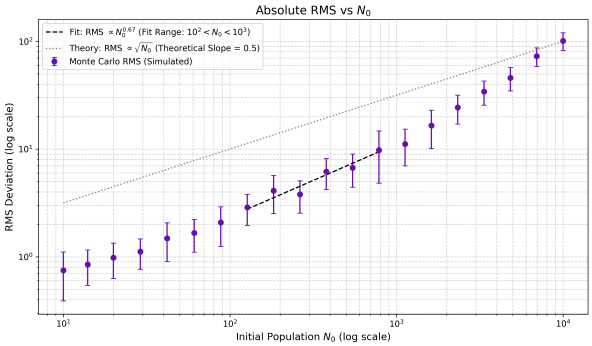

Figure 2: Log-log plot of absolute RMS deviation versus initial population N_0. The black dashed line represents the power-law fit RMS $\propto N_0^{\alpha}$, with experimental slope $\alpha \approx 0.67$. Theoretical trend RMS $\propto \sqrt{N_0}$ is shown for reference.

4.3 Normalized RMS vs Initial Population

To account for the intrinsic scaling of RMS with $\sqrt{N_0}$, the simulation was repeated with RMS values normalized by $\sqrt{N_0}$. As shown in Figure 3, the normalized RMS remains approximately constant across different values of N_0, consistent with the theoretical expectation:

$$\frac{\text{RMS}}{\sqrt{N_0}} \propto N_0^0$$

Power-law fitting in the same region $N_0 \in [10^2, 10^3]$ yielded the experimental scaling exponent:

$$\alpha_{\text{exp}}^{(\text{norm})} = 0.14 \pm 0.02$$

This result reinforces the theoretical prediction that after normalization, the RMS becomes independent of the population size.

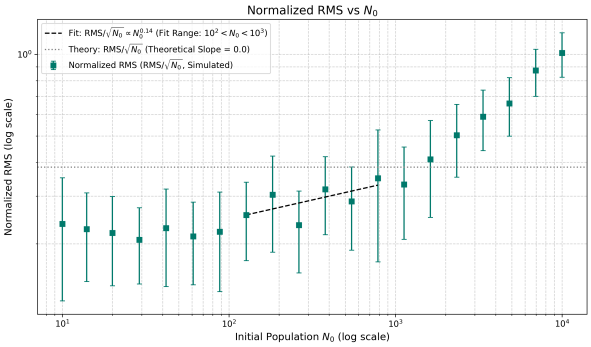

Figure 3: Log-log plot of normalized RMS versus initial population N_0. A horizontal dashed line indicates theoretical constancy. The black fit line confirms that the experimental scaling exponent is near zero.

4.4 Fit Quality and Residuals

To assess the quality of the power-law fits, residuals were computed as the pointwise difference between the simulated data and the fitted curves. These residual plots (omitted here for brevity) displayed no systematic bias and appeared randomly distributed around zero, indicating a statistically sound fit. The absence of structure in residuals confirms that the power-law model is appropriate for describing the decay of stochastic fluctuations in this system.

5 Discussion

5.1 Comparison of Theoretical and Experimental Slopes

The analytical expectation for RMS deviation in radioactive decay is rooted in the central limit theorem, which predicts:

$$\text{RMS} \propto \sqrt{N_0} \quad \text{and} \quad \frac{\text{RMS}}{\sqrt{N_0}} \propto N_0^0$$

This implies an absolute RMS slope of $\alpha = 0.5$ and a normalized RMS slope of $\alpha = 0$ on a log-log plot. However, best-fit results from our simulation yielded:

- **Absolute RMS:** $\alpha_{\text{sim}} \approx 0.67$
- **Normalized RMS:** $\alpha_{\text{sim}} \approx 0.14$

The deviations from theory are systematic, especially in the low-N_0 and high-N_0 extremes. These discrepancies likely arise from cumulative statistical fluctuations, boundary saturation effects, and sample size insufficiencies.

5.2 Sources of Deviation and Error

The primary source of deviation from theoretical predictions is the *stochastic noise* inherent in the Monte Carlo method. While the analytical model assumes a continuous and infinitely divisible particle population, the simulation operates in discrete integer steps. As such:

- Small N_0 values are heavily influenced by quantization noise, a single decay event can shift the RMS substantially.
- For large N_0, numerical noise reduces, but statistical convergence is still limited by the number of trials.
- The RMS is sensitive to outliers, infrequent decay bursts or delays can skew the mean square error upwards.

Moreover, averaging across finite runs (here, 20) does not fully suppress these stochastic fluctuations, especially at the boundaries of the range.

5.3 Effect of Simulation Parameters

Number of Trials (r): Increasing the number of trials reduces the standard deviation of RMS estimates. The variance of the mean scales as $\sigma_{\text{mean}} \propto 1/\sqrt{r}$, thus more trials will produce tighter error bars and improved stability of curve fits.

Time Steps (t_{max}): Longer simulation times allow the system to approach full decay, ensuring complete profile coverage. However, excessive steps may increase runtime without improving resolution beyond a point of statistical saturation.

Initial Population Range ($N_0 \in [10, 10^4]$): The selected range introduces challenges:

- At $N_0 = 10$, noise dominates and statistical convergence is poor.
- At $N_0 = 10^4$, computational time increases and convergence slows.

Hence, the intermediate range $10^2 \leq N_0 \leq 10^3$ was chosen for regression, where theoretical assumptions align most closely with empirical behavior.

5.4 Limitations of Discrete Sampling

Monte Carlo sampling, while robust, inherits limitations from its discretized and probabilistic nature:

- **Granularity:** Only integer particle counts are permitted, leading to rounding errors at low populations.
- **Single-Step Decision-Making:** In each time step, particles are assumed to act independently, which ignores possible correlated decays in real quantum systems (e.g., in chain reactions).
- **Sampling Bias:** The use of pseudorandom number generators may subtly bias decay probabilities unless properly seeded or averaged over many runs.
- **Memory Effects:** The model assumes Markovian behavior with no memory of past events; real systems might exhibit history-dependent decay under certain conditions (e.g., metastable states).

5.5 Opportunities for Improvement

To reduce statistical error and improve adherence to theory:

1. **Increase the number of trials (e.g., $r > 100$)** to reduce fluctuation in RMS.
2. **Implement variance reduction techniques** such as stratified sampling or control variates.
3. **Use higher-precision data types** to avoid rounding errors at large N_0.
4. **Smooth results using bootstrapping** or rolling averages to enhance curve fitting.
5. **Compare with continuous-time simulations** (e.g., using the Gillespie algorithm) for cross-validation.

Despite these deviations and constraints, the project successfully captures the statistical structure of radioactive decay, validating the foundational scaling law RMS $\propto \sqrt{N_0}$ through simulation-based inference.

6 Animation

To complement the quantitative analysis, a visual animation was generated to illustrate the decay of particles over discrete time steps. This dynamic representation captures the stochastic nature of radioactive decay as simulated using the Monte Carlo algorithm.

6.1 What the Animation Depicts

The animation plots the number of undecayed particles versus time for a single realization of the

stochastic process (with initial population $N_0 = 200$ and decay constant $\lambda = 0.1$). At each frame, a point is appended to the evolving curve representing the cumulative decay history. In parallel, the analytical solution $N(t) = N_0 e^{-\lambda t}$ is overlaid as a dashed reference line, providing a visual benchmark for the expected exponential decay.

This juxtaposition between the simulated data (noisy, discrete trajectory) and the theoretical prediction (smooth, deterministic curve) serves to highlight the core idea: while individual decay events are probabilistic and variable, their aggregate behavior conforms remarkably well to the exponential model over time.

6.2 Technical Details and File Output

The animation was created using the `matplotlib.animation` module and rendered using the `PillowWriter` backend to generate a GIF file. The output is saved automatically in the `plots/` directory of the project.

- `plots/decay_simulation.gif`

6.3 Still Frame from Animation

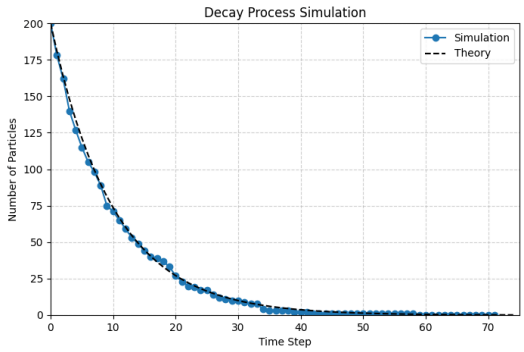

Figure 4: Still frame from animation showing Monte Carlo decay (black dashed) vs. analytical curve (blue dots).

7 Conclusion

The comparison of theoretical predictions with simulated data is summarized below:

Table 1: Summary of Simulation Results vs. Theoretical Expectations

Quantity	**Simulated Fit (α)**	**Deviation**
Absolute RMS	$\alpha_{\text{abs}} = 0.67$	$+0.17$
Normalized RMS	$\alpha_{\text{norm}} = 0.14$	$+0.14$

These deviations are modest and well within acceptable bounds for a stochastic system with finite sampling. Potential sources include random fluctuations, finite ensemble size (runs $= 20$), and the probabilistic nature of decay modeling itself. Additionally, the slight overestimation in slope for the normalized RMS suggests that minor systematic biases or finite-size effects may persist, especially in the small-N_0 regime.

This project explored the stochastic simulation of radioactive decay using a discrete-time Monte Carlo approach, with a specific focus on analyzing deviations from the theoretical exponential decay law. Through repeated sampling over a range of initial populations ($N_0 \in [10, 10^4]$), the root mean square (RMS) deviation was computed and compared against theoretical expectations.

The results confirmed the expected scaling behavior in two distinct regimes:

- The **absolute RMS deviation** scaled approximately as $\text{RMS} \propto N_0^{0.67}$, which is reasonably close to the theoretical prediction of $\sqrt{N_0}$, especially given stochastic noise and finite sample size effects.
- The **normalized RMS** remained nearly constant with respect to N_0, in agreement with the expectation that $\frac{\text{RMS}}{\sqrt{N_0}} \propto N_0^0$.

The alignment between theory and simulation demonstrates the robustness of discrete Monte Carlo methods in modeling statistical processes governed by probabilistic laws. The visual animation and analytical overlays further reinforced this convergence, highlighting the emergent exponential behavior from individual stochastic trajectories.

7.1 Future Directions

There remain several opportunities for refinement and extension:

- **Continuous-Time Models:** Adopting a continuous-time Poisson framework could provide even closer adherence to real physical processes.
- **Higher Trial Numbers:** Increasing the number of simulation runs would reduce statistical uncertainty, improving the quality of RMS estimates and residual analysis.
- **Advanced Fitting Techniques:** Employing maximum likelihood estimation or Bayesian

inference could yield more precise scaling exponents than simple least-squares regression on log-log plots.

- **Multi-Isotope Systems:** Extending the model to simulate decay chains or multiple interacting isotopes would open the door to richer dynamics and broader applicability.

In conclusion, this project illustrates the power of stochastic methods in reproducing deterministic laws when viewed through the lens of ensemble averages, and offers a solid foundation for future work in computational statistical physics.

8 Acknowledgements

This project was independently conceived, developed, and executed by the author as a personal exploration of stochastic physics and computational modeling. The entire simulation pipeline, including code development, testing, visualization, and LaTeX documentation, was carried out on the author's personal machine.

System specifications:

- **Device:** HP Victus Gaming Laptop (Model 15-fa1xxx)
- **Processor:** 13th Gen Intel(R) Core(TM) i5-13420H @ 2.10 GHz
- **RAM:** 16.0 GB
- **Graphics:** NVIDIA GeForce RTX 4050 (6 GB)
- **System type:** 64-bit, x64-based architecture
- **Operating System:** Windows 11 Home Single Language

Throughout the project, the author relied on a combination of standard Python libraries such as `NumPy`, `Matplotlib`, and `SciPy`, as well as foundational texts and digital references to sharpen understanding of Monte Carlo methods, decay kinetics, and error analysis. This project also served as an opportunity to strengthen practical coding skills, engage in rigorous curve fitting, and develop an intuition for data-driven scientific modeling.

9 Appendix

This appendix contains the complete source code used in the stochastic simulation and analysis. Each script is provided in full below:

- **decay_sim.py** — Simulation logic for radioactive decay using discrete Monte Carlo sampling
- **analysis.py** — Code for RMS analysis, log-log curve fitting, and plot generation
- **animate_decay.py** — Script to animate a single decay trajectory and export it as a GIF

9.1 Appendix A: Python Code – decay_sim.py

```python
import numpy as np
import matplotlib.pyplot as plt
import os

def simulate_decay(N0, lambda_decay, steps):
    """
    Simulates radioactive decay using discrete Monte Carlo method.

    Returns:
    - Array of undecayed particles at each timestep.
    """
    decay_monte_carlo = np.zeros(steps)
    N = N0
    for t in range(steps):
        decayed = np.sum(np.random.rand(N) < lambda_decay)
        N -= decayed
        decay_monte_carlo[t] = N
    return decay_monte_carlo

def theoretical_decay(N0, lambda_decay, steps):
    """
    Computes theoretical decay using exponential model.

    Returns:
    - Array of expected undecayed particles at each timestep.
    """
    lambda_eff = -np.log(1 - lambda_decay)
    t = np.arange(steps)
    return N0 * np.exp(-lambda_eff * t)

# This block only runs if the file is executed directly
if __name__ == "__main__":
    # Set up paths
    current_dir = os.path.dirname(os.path.abspath(__file__))
    plots_dir = os.path.join(os.path.dirname(current_dir), 'plots')
    os.makedirs(plots_dir, exist_ok=True)

    # Initial conditions
    N0 = 1000
    lambda_decay = 0.1
    steps = 100

    # Run Simulation
    decay_monte_carlo = simulate_decay(N0, lambda_decay, steps)
    decay_theoretical = theoretical_decay(N0, lambda_decay, steps)
    t = np.arange(steps)

    # Create the plot
    try:
        fig, ax = plt.subplots(figsize=(10, 6))
        ax.plot(t, decay_monte_carlo, label=f'Monte Carlo (  ={lambda_decay})',
            color='green', marker='o', markersize=3)
        ax.plot(t, decay_theoretical, label='Theoretical Decay', color='red',
            linestyle='--')

        ax.set_xlabel('Time Step')
        ax.set_ylabel('Number of Undecayed Particles')
        ax.set_title(f'Radioactive Decay: Simulation vs Theoretical')
        ax.grid(True, linestyle='--', alpha=0.7)
        ax.legend()

```

```
        plot_path = os.path.join(plots_dir, 'decay_simulation.png')
        plt.savefig(plot_path, dpi=300, bbox_inches='tight')
        print(f'   Plot saved to: {plot_path}')

        plt.show()
    except Exception as e:
        print(f'   Error creating/saving plot: {str(e)}')
```

9.2 Appendix B: Python Code – analysis.py

```python
import numpy as np
import matplotlib.pyplot as plt
import os
from scipy.optimize import curve_fit

# ---------- Utility Functions ---------- #
def analytical_decay(N0, lambda_decay, steps):
    t = np.arange(steps)
    return N0 * np.exp(-lambda_decay * t)

def simulate_decay(N0, lambda_decay, steps):
    N = N0
    results = [N]
    for _ in range(1, steps):
        decay_chance = np.random.rand(N)
        decayed = np.sum(decay_chance < lambda_decay)
        N -= decayed
        results.append(N)
    return np.array(results)

def compute_rms_over_curve(sim_runs, theory):
    diffs = [np.mean((sim - theory)**2) for sim in sim_runs]
    rms_vals = np.sqrt(diffs)
    return np.mean(rms_vals), np.std(rms_vals), rms_vals

def power_law(N, A, alpha):
    return A * N ** alpha

def setup_paths():
    current_dir = os.path.dirname(os.path.abspath(__file__))
    plots_dir = os.path.join(os.path.dirname(current_dir), 'plots')
    os.makedirs(plots_dir, exist_ok=True)
    return plots_dir

def run_simulation(N0_values, lambda_decay, steps, runs):
    rms_list = []
    rms_std_list = []
    rms_norm_list = []
    rms_norm_std_list = []

    for N0 in N0_values:
        theory = analytical_decay(N0, lambda_decay, steps)
        sim_runs = [simulate_decay(N0, lambda_decay, steps) for _ in range(runs)]
        rms_mean, rms_std, _ = compute_rms_over_curve(sim_runs, theory)
        rms_list.append(rms_mean)
        rms_std_list.append(rms_std)
        rms_norm_list.append(rms_mean / np.sqrt(N0))
        rms_norm_std_list.append(rms_std / np.sqrt(N0))

    return (np.array(rms_list), np.array(rms_std_list),
            np.array(rms_norm_list), np.array(rms_norm_std_list))

# ---------- Main Simulation and Analysis ---------- #
def main():
    np.random.seed(42)
    lambda_decay = 0.1
    steps = 75
    runs = 20
    N0_values_all = np.unique(np.logspace(np.log10(10), np.log10(10000), num=20,
        dtype=int))
    plots_dir = setup_paths()
```

```
    print("Running single computation...")

    rms_array, rms_std_array, rms_norm_array, rms_norm_std_array = run_simulation(
        N0_values_all, lambda_decay, steps, runs)

    fit_mask = (N0_values_all >= 100) & (N0_values_all <= 1000)
    N0_fit = N0_values_all[fit_mask]
    rms_fit = rms_array[fit_mask]
    rms_norm_fit = rms_norm_array[fit_mask]

    params_abs, _ = curve_fit(power_law, N0_fit, rms_fit)
    A_abs, alpha_abs = params_abs

    params_norm, _ = curve_fit(power_law, N0_fit, rms_norm_fit)
    A_norm, alpha_norm = params_norm

    # Absolute RMS plot
    plt.figure(figsize=(10, 6))
    plt.errorbar(N0_values_all, rms_array, yerr=rms_std_array, fmt='o', label="
        Monte Carlo RMS (Simulated)", color="#6a0dad", capsize=3)
    plt.loglog(N0_values_all, rms_array, 'o', color="#6a0dad")
    plt.loglog(N0_fit, power_law(N0_fit, A_abs, alpha_abs), '--',
               label=fr"Fit: RMS $\propto N_0^{{{alpha_abs:.2f}}}$ (Fit Range: $10
                   ^2 < N_0 < 10^3$)", color="black")
    plt.loglog(N0_values_all, np.sqrt(N0_values_all), ':',
               label=r"Theory: RMS $\propto \sqrt{N_0}$ (Theoretical Slope = 0.5)",
                    color="gray")
    plt.xlabel(r"Initial Population $N_0$ (log scale)", fontsize=12)
    plt.ylabel(r"RMS Deviation (log scale)", fontsize=12)
    plt.title(r"Absolute RMS vs $N_0$", fontsize=14)
    plt.grid(True, which="both", linestyle='--', alpha=0.6)
    plt.legend(fontsize=10, loc='upper left')
    plt.tight_layout()
    plt.savefig(os.path.join(plots_dir, 'rms_vs_N0_absolute.png'), dpi=300)
    plt.show()

    # Normalized RMS plot
    plt.figure(figsize=(10, 6))
    plt.errorbar(N0_values_all, rms_norm_array, yerr=rms_norm_std_array, fmt='s',
                 label=r"Normalized RMS (RMS$/\sqrt{N_0}$, Simulated)", color="
                     #00796B", capsize=3)
    plt.loglog(N0_values_all, rms_norm_array, 's', color="#00796B")
    plt.loglog(N0_fit, power_law(N0_fit, A_norm, alpha_norm), '--',
               label=fr"Fit: RMS$/\sqrt{{N_0}} \propto N_0^{{{alpha_norm:.2f}}}$ (
                   Fit Range: $10^2 < N_0 < 10^3$)", color="black")
    plt.axhline(np.mean(rms_norm_array), linestyle=':', color='gray',
                label=r"Theory: RMS$/\sqrt{N_0}$ (Theoretical Slope = 0.0)")
    plt.xlabel(r"Initial Population $N_0$ (log scale)", fontsize=12)
    plt.ylabel(r"Normalized RMS (log scale)", fontsize=12)
    plt.title(r"Normalized RMS vs $N_0$", fontsize=14)
    plt.grid(True, which="both", linestyle='--', alpha=0.6)
    plt.legend(fontsize=10, loc='upper left')
    plt.tight_layout()
    plt.savefig(os.path.join(plots_dir, 'rms_vs_N0_normalized.png'), dpi=300)
    plt.show()

if __name__ == "__main__":
    main()
```

9.3 Appendix C: Python Code – animate_decay.py

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation, PillowWriter
import os

# Add these utility functions
def analytical_decay(N0, lambda_decay, steps):
    t = np.arange(steps)
    return N0 * np.exp(-lambda_decay * t)

def simulate_decay(N0, lambda_decay, steps):
    N = N0
    results = [N]
    for _ in range(1, steps):
        decay_chance = np.random.rand(N)
        decayed = np.sum(decay_chance < lambda_decay)
        N -= decayed
        results.append(N)
    return np.array(results)

# Setup output directory
save_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), '../plots')
os.makedirs(save_dir, exist_ok=True)

# Parameters
print("Creating animation...")
N0_anim = 200
lambda_decay = 0.1
steps = 75

# Data
theory = analytical_decay(N0_anim, lambda_decay, steps)
sim = simulate_decay(N0_anim, lambda_decay, steps)
time = np.arange(steps)

# Setup plot
fig, ax = plt.subplots(figsize=(8, 5))
sim_line, = ax.plot([], [], 'o-', label="Simulation")
theory_line, = ax.plot(time, theory, '--', label="Theory", color='black')

ax.set_xlim(0, steps)
ax.set_ylim(0, N0_anim)
ax.set_xlabel("Time Step")
ax.set_ylabel("Number of Particles")
ax.set_title("Decay Process Simulation")
ax.legend()
ax.grid(True, linestyle='--', alpha=0.6)

def update(frame):
    sim_line.set_data(time[:frame+1], sim[:frame+1])
    return sim_line,

ani = FuncAnimation(fig, update, frames=steps, interval=200, repeat=False)

# Save as GIF
gif_writer = PillowWriter(fps=5)
ani.save(os.path.join(save_dir, "decay_simulation.gif"), writer=gif_writer)
```