Computational Studies of Random Walks

Kiara Gholizad[1]

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

April 4, 2025

Abstract

In this study, stochastic behavior and physical implications of a wide variety of random walk models are investigated using detailed mathematical analysis and computational simulation. We analyze nine different random walk models: one-dimensional walks, models with absorbing barriers, two-dimensional systems, diffusion-limited aggregation with linear and central seeds, self-avoiding walks, path enumeration algorithms, persistent (conservative) random walks, and continuous models with variable step distributions. For each system, we create theoretical models, write computational algorithms, and investigate emergent behavior using statistical analysis and visualization. Some findings include confirmation of diffusion scale laws in simple random walks ($\langle r^2 \rangle = 2dDt$), proof of the computational advantage of enumeration methods over Monte Carlo for absorbing barriers ($\sim 3 \times 10^{-4}$s vs. 2s computation time), identification of consistent diffusion-limited aggregation fractal dimensions (1.596-1.605) despite changing boundary conditions, evidence for the growth constant for self-avoiding walks (2.710), analysis of persistence length scale as $l_p = 1/(1-p)$ in persistent walks, and confirmation of anomalous diffusion in models with Lévy flights. Through the detailed analysis of computational complexity, statistical behavior, and physical implications, this study connects theoretical stochastic models with practical computational techniques with insights applicable to fields as diverse as polymer physics and biological systems as well as econometrics and search algorithms.

Contents

1 Introduction

Random walks are basic stochastic processes with applications ranging from physics and chemistry through biology, economics, and computer science. Random walks provide great models for a wide variety of natural phenomena such as diffusion processes, polymer conformation, stock market fluctuations, and search algorithms.

In the present study, we're interested in the application and computational study of various models for random walks to investigate their physical implications and statistical properties. Random walks are a useful tool for the description of diffusion phenomena, in which particles perform random motions as a consequence of thermal fluctuations. Random walks have a long tradition in the sciences, dating back to the explanation by Einstein in 1905 of Brownian motion first detected by the botanist Robert Brown in 1827 observing pollen grains suspended in water.

The key insight from Einstein's work was connecting the microscopic random motions (random walks) to macroscopic diffusion, leading to the diffusion equation:

$$\frac{\partial P(x;t)}{\partial t} = D\frac{\partial^2 P(x;t)}{\partial x^2} \tag{1}$$

where $P(x;t)$ is the probability distribution of finding a particle at position x at time t, and D is the diffusion coefficient. This equation has a Gaussian solution with width $\sigma^2 = 2Dt$, implying that the mean squared displacement of a random walker grows linearly with time:

$$\langle r^2 \rangle = 2dDt \tag{2}$$

where d is the dimension of the space. This relationship, known as the scaling law of random walks, is characterized by the critical exponent $\nu = 1/2$, such that the typical displacement scales as $R_g \sim t^{\nu}$.

Different variants of random walks exhibit different scaling behaviors. For example, self-avoiding random walks (SAWs), which prohibit the walker from visiting the same site twice, have $\nu = 3/4$ in two dimensions and $\nu \approx 0.588$ in three dimensions.

In this study, we examine an extensive class of random walk variants and their applications, including both the theoretical background and computational realization:

1. One-dimensional random walks and their statistical properties
2. Random walks with absorbing boundaries and first passage statistics
3. Two-dimensional random walks and their diffusion characteristics
4. Diffusion-limited aggregation (DLA) as a model for fractal growth
5. DLA with central seed and its pattern formation dynamics
6. Self-avoiding random walks and polymer physics
7. Path enumeration methods for self-avoiding walk configurations
8. Conservative (persistent) random walks modeling correlated motion
9. Continuous random walk models with varying step distributions

For each system, we create a theoretical framework, carry out computational simulations, visualize the emergent behavior, and cross-check against analytical solutions when possible. This multidisciplinary effort enables us to clarify the underlying stochastic mechanisms' intricate dynamics as well as their physical implications.

2 Theoretical Background

2.1 One-Dimensional Random Walk

A one-dimensional random walk models the motion along a straight line of a particle that walks either to the right with probability p or left with probability q, with $p + q = 1$. Each step is of a constant length l, and the lapse between successive steps is τ, so after a total time t, the walker takes $N = t/\tau$ steps with the assumption that N is an integer for simplicity. If $p = q = 1/2$, the walk is unbiased with no preferred direction; otherwise, the walk will drift according to the difference between p and q. A walker begins at position $x_0 = 0$ at time $t_0 = 0$. At the end of N steps, define the number of steps taken to the right by N_+ and the number taken to the left by $N_- = N - N_+$. The walkers position is then:

$$x = l(N_+ - N_-)$$

The probability of achieving N_+ right steps in N trials follows a binomial distribution:

$$P(N_+; N) = \binom{N}{N_+} p^{N_+} q^{N_-} \tag{3}$$

where $\binom{N}{N_+} = N!/(N_+!N_-)!$ counts the distinct ways to arrange the steps.

For a large N, the central limit theorem converts this discrete distribution into a continuous Gaussian form:

$$P(x; t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \langle x\rangle)^2}{2\sigma^2}\right) \tag{4}$$

Here, the mean position is:

$$\langle x\rangle = (p - q)\frac{lt}{\tau}$$

and the variance, which quantifies the spread in the walker's position, is:

$$\sigma^2 = 4l^2 pq\frac{t}{\tau}$$

These control the walk's behavior as a function of time t, with the step time denoted by τ.

One key property of the random walk is the mean squared displacement:

$$\langle x^2\rangle - \langle x\rangle^2 = 4l^2 pq\frac{t}{\tau} \tag{5}$$

In the case when the system lacks bias ($p = q = \frac{1}{2}$), the mean position is zero ($\langle x\rangle = 0$), and the variance simplifies as:

$$\sigma^2 = l^2\frac{t}{\tau}$$

That the increase with time is a straight line is characteristic for diffusion behavior. Attempting a refinement of these results, we calculate the mean and the variance explicitly.

Derivation of Mean and Variance Let's denote $N_r = N_+$ as the number of rightward steps and $N_l = N_-$ as the number of leftward steps, so $N_r + N_l = N$. Position is:

$$x = l(N_r - N_l)$$

Because every step is a self-contained trial, N_r will have a binomial distribution with parameters N and p.

Mean Position: The expected number of right and left steps is:

$$\langle N_r \rangle = Np = \frac{t}{\tau}p, \quad \langle N_l \rangle = Nq = \frac{t}{\tau}q$$

Thus:

$$\langle x \rangle = l(\langle N_r \rangle - \langle N_l \rangle) = l\left(\frac{t}{\tau}p - \frac{t}{\tau}q\right) = (p-q)\frac{lt}{\tau}$$

This establishes that the mean drifts with the bias $p - q$, which goes away when $p = q$.

Variance: The variance is:

$$\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2$$

First, compute the second moment:

$$x^2 = l^2(N_r - N_l)^2$$

Replace $N_l = N - N_r$:

$$x^2 = l^2[N_r - (N - N_r)]^2 = l^2(2N_r - N)^2 = l^2(4N_r^2 - 4NN_r + N^2)$$

Taking the expectation:

$$\langle x^2 \rangle = l^2(4\langle N_r^2 \rangle - 4N\langle N_r \rangle + N^2)$$

For a binomial variable N_r:

$$\langle N_r \rangle = Np, \quad \mathrm{Var}(N_r) = Npq, \quad \langle N_r^2 \rangle = \mathrm{Var}(N_r) + \langle N_r \rangle^2 = Npq + N^2p^2$$

So:

$$\langle x^2 \rangle = l^2[4(Npq + N^2p^2) - 4N(Np) + N^2] = l^2[4Npq + 4N^2p^2 - 4N^2p + N^2]$$

Factorize:

$$\langle x^2 \rangle = l^2[4Npq + N^2(4p^2 - 4p + 1)] = l^2[4Npq + N^2(2p-1)^2]$$

Since $(2p-1)^2 = (p-q)^2$:

$$\langle x^2 \rangle = l^2[4Npq + N^2(p-q)^2]$$

Now:

$$\langle x \rangle^2 = \left((p-q)\frac{lt}{\tau}\right)^2 = l^2N^2(p-q)^2$$

Thus:

$$\sigma^2 = l^2[4Npq + N^2(p-q)^2] - l^2N^2(p-q)^2 = 4l^2Npq = 4l^2pq\frac{t}{\tau}$$

This establishes the variance, and Equation (5) follows as it equals σ^2.

Unbiased Case: For $p = q = 1/2$:

$$\langle x \rangle = (0.5 - 0.5)\frac{lt}{\tau} = 0, \quad \sigma^2 = 4l^2(0.5)(0.5)\frac{t}{\tau} = l^2\frac{t}{\tau}$$

Its diffusive spread, which is proportional to t, reflects the random walk's symmetric exploration of space when not biased, a phenomenon found in physical systems, such as Brownian motion.

2.2 Diffusion Equation and Random Walks

The connection between random walks and diffusion can be established by considering the probability evolution equation:

$$P(x;t) = \frac{1}{2}\left[P(x-l;t-\tau) + P(x+l;t-\tau)\right] \tag{6}$$

Taking the continuum limit as $l \to 0$ and $\tau \to 0$ while keeping $D = \frac{l^2}{2\tau}$ constant, we obtain the diffusion equation:

$$\frac{\partial P(x;t)}{\partial t} = D\frac{\partial^2 P(x;t)}{\partial x^2} \tag{7}$$

The diffusion coefficient D characterizes how quickly particles spread out. The solution of this equation is a Gaussian distribution with width that grows as $\sqrt{t}$.

2.3 Random Walks in Higher Dimensions

In d dimensions, the random walk can be decomposed into d independent one-dimensional random walks. For a symmetric random walk on a d-dimensional lattice, the mean squared displacement is:

$$\langle r^2 \rangle = 2dDt \tag{8}$$

or, equivalently:

$$R_g \sim t^{\nu} \tag{9}$$

with $\nu = 1/2$ for simple random walks.

Space's dimension affects the characteristics of random walks. In $d \leq 2$ dimensions, a random walk will be recurrent, returning with probability 1 to the point of departure. For $d \geq 3$ dimensions, the walk will be transient, so that the probability that it never returns to the origin is greater than zero.

2.4 Self-Avoiding Random Walks

Self-avoiding random walks (SAWs) are random walks which never revisit the same site. SAWs differ from random walks in the sense that they have long-range memory and display varying scaling behavior.

In two dimensions, the critical exponent for SAWs is $\nu = 3/4$, so the typical size grows as $R_g \sim N^{3/4}$, with N the number of steps. In three dimensions, $\nu \approx 0.588$, whereas for $d \geq 4$ the behavior tends toward that of the simple random walks with $\nu = 1/2$.

The number of SAWs with length N increases exponentially in N but more gradually than with random walks. For random walks on a lattice in coordination z, that number would be z^N; with SAWs, it would be μ^N with $\mu < z$ being referred to as the connective constant.

2.5 Diffusion-Limited Aggregation

Diffusion-limited aggregation occurs when particles executing random walks bind irreversibly to the developing cluster when they come into contact with it. Beginning with a cluster seed (or line of seeds), the randomly walking particles move in and bind to the cluster, which develops tree-like patterns.

DLA clusters have a fractal nature, and their density as a function of radius from the center diminishes as a power law. A measurement of the fractal dimension for two-dimensional DLA clusters is around 1.71, below the space's inherent dimension.

DLA growth is a competitive growth process: as soon as a branch projects more than the neighboring ones, it tends to capture the diffusing particles, resulting in a tip-splitting instability and the typical branching pattern.

2.6 Directed and Persistent Random Walks

Random walks are walks whose movements are confined in some directions. A raindrop falling downward, for instance, would move at random in the horizontal direction but always downward in the vertical direction.

Persistent random walks have memory in that the walker tends to keep traveling in the direction of the last step. This adds a characteristic length scale, the persistence length, over which the walk will be relatively straight. At distances much greater than the persistence length, the walk resembles a random walk with effective step size being the persistence length.

2.7 Computational Framework

We use Python with NumPy for numerical computations and Matplotlib for plotting in implementing the different random walks. Our model structure emphasizes modularity, reusability, and computational performance, especially for computationally intensive simulations with significant statistical sampling.

For each random walk variation, we built a specialized algorithm based on these fundamental principles:

1. **Model specification**: We specify exact rules for the motion of the walker depending on the physical system being simulated (i.e., persistence parameters, distributions of the steps, boundary conditions)

2. **Trajectory generation efficiency**: All implementations monitor the walker's position and pertinent state variables with optimal data structures (NumPy arrays, predominantly)

3. **Ensemble averaging**: Statistical properties are captured by running between 500-10,000 independent runs per-parameter set with sample size determined by computational resources

4. **Analysis pipeline**: We have specialized analysis functions for each model that compute important metrics, create visualizations, and compare with theoretical expectations

We use time-based termination conditions (fixed numbers of steps) as well as condition-based stopping rules (e.g., self-avoidance violations, absorptions at boundaries) based on the physical problem being studied.

2.8 Statistical Analysis

Our statistical analysis framework focuses on stringent quantification of random walk behavior:

- **Spatial statistics**: We calculate position distributions, mean square displacements, and diffusion scaling relationships
- **Quantifying the errors:** Standard errors are computed as $\sigma/\sqrt{N}$ with σ being the standard deviation and N the sample size
- **Scaling analysis**: In the case of expected power-law behavior, we use log-log linear regression with SciPy's `stats.linregress` in order to extract scaling exponents
- **Boundary effects**: We restrict fitting ranges when necessary in order to prevent finite-size effects, as shown in our DLA fractal dimension analysis where data fitting is confined to radii ≤ 40

For visualization of trajectories, we produce uniformly formatted plots for multiple models so that the behavior can be instantly compared. For fractal and aggregation models, we use both binary (occupation grid) and temporal (growth sequence) visualizations in order to capture the structural as well as the dynamic properties of the system.

3 One-Dimensional Random Walk

For our initial step, were going to build a program to simulate a 1D random walk and see if the theory works. Were testing out the average position, $\langle x \rangle = (p - q)lt/\tau$, and spread, $\sigma^2 = 4l^2pqt/\tau$, with various values of $p$even including the case where $p = 1/2$. Its all about observing how a little bit of bias affects how our walker moves about.

3.1 Algorithm and Implementation

The algorithm for a one-dimensional random walk is straightforward:

Algorithm 1 One-Dimensional Random Walk

Input: Number of steps N, probability of moving right p, step length l
Output: Final position, trajectory, mean position, variance
Initialize position $x = 0$ and array *positions* $= [0]$
for $i = 1$ to N **do**
    Generate random number $r \in [0, 1)$
    if $r < p$ **then**
        $x \leftarrow x + l$ ▷ Move right
    else
        $x \leftarrow x - l$ ▷ Move left
    end if
    Append x to *positions*
end for
Return x, *positions*, mean of *positions*, variance of *positions*

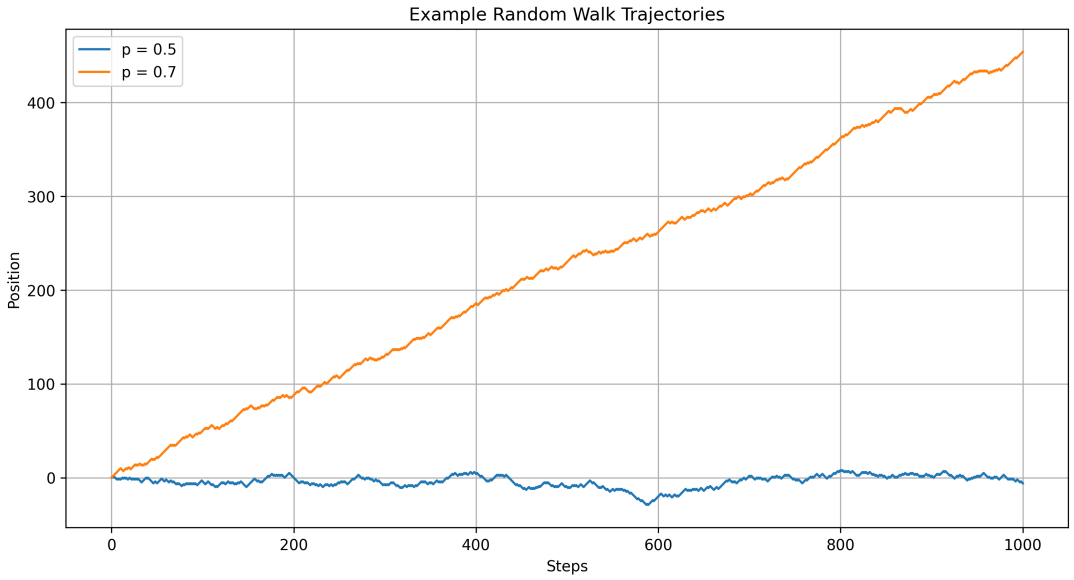

Figure 1: Example trajectories of one-dimensional random walks with different probabilities of moving right (p). For $p = 0.5$ (symmetric case), the walk fluctuates around the starting position. For $p = 0.7$ (biased case), there is a clear drift to the right.

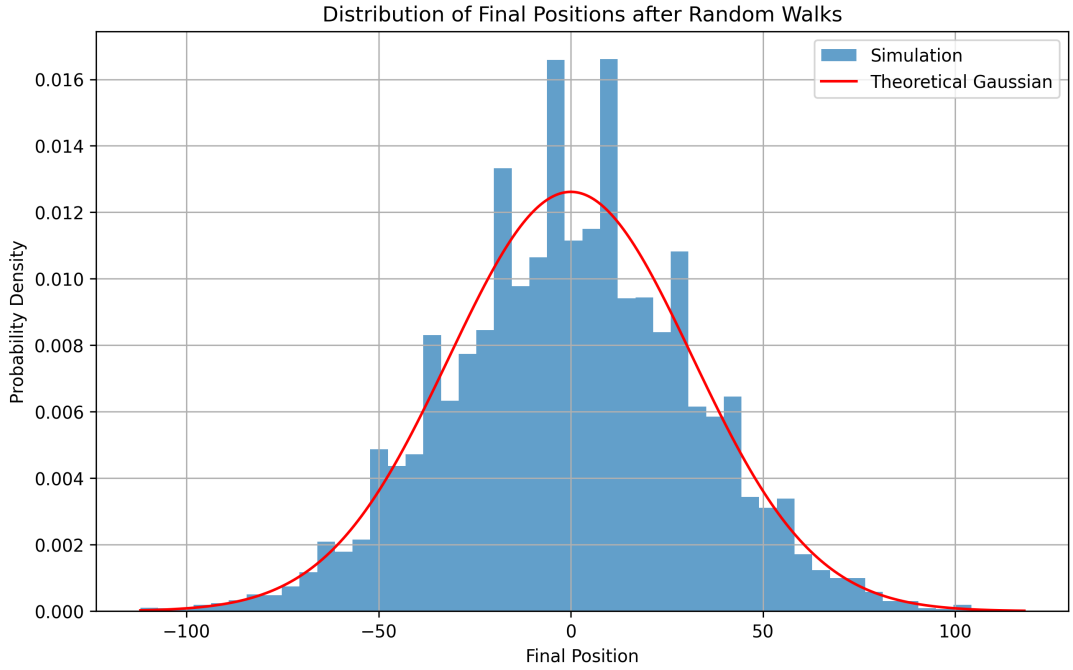

Figure 2: Distribution of final positions after 10,000 random walks of 1,000 steps each. The histogram from simulations closely matches the theoretical Gaussian distribution, demonstrating the Central Limit Theorem.

3.2 Results and Discussion

Table 1: Comparison of theoretical and experimental values for one-dimensional random walks

p	$\langle x \rangle_{theo}$	$\langle x \rangle_{exp}$	Error (%)	σ^2_{theo}	σ^2_{exp}	Error (%)
0.5	0.00	0.20	0.20	1000.00	1034.60	3.46
0.7	400.00	399.27	0.18	840.00	820.84	2.28

Our findings agree with the theoretical expectation both for the symmetric ($p = 0.5$) and biased ($p = 0.7$) walks. For the symmetric walk, the position averages near zero as it should, with a minor fluctuation due to statistical reasons. The variance agrees well with the theoretical expectation of $\sigma^2 = N$, with only 3.46% error.

For the biased case ($p = 0.7$), we see a definite drift in the positive direction with a mean position of about 399.27 after 1000 steps, close to the theoretical result of $(2p - 1)N = 400$. The variance also agrees well with the theory with an error of 2.28%.

The final positions are distributed according to a Gaussian distribution as anticipated by the Central Limit Theorem with parameters as we theoretically expected. This verifies that with large step numbers, the binomial distribution of the random walk converges towards a normal distribution.

4 Random Walk with Absorbing Boundaries

This problem considers a one-dimensional random walk on a discrete lattice with traps (absorbing boundaries) as follows: The lattice has 21 nodes indexed by numbers 0 to 20, with the center at 10. There are traps in positions 0 and 20, 10 steps from the center in both directions. The process starts with a walker at initial location x_0 (i.e., $0 < x_0 < 20$), who walks to the left or to the right with equal probabilities (0.5) on each step. The process terminates when the walker reaches either of the traps. The objectives are to calculate the mean lifetime (expected number of steps to absorption) and probabilities of absorption into each of the traps as a function of x_0. The two methods used are a Monte Carlo simulation and an enumeration algorithm, and we test against an exact analytical solution.

4.1 Theoretical Background

For a one-dimensional random walk with absorbing boundaries at positions 0 and $L = 20$, exact analytical solutions exist for the average lifetime and absorption probabilities:

- The average lifetime $T(x)$, or expected number of steps until absorption starting from position x, is:
$$T(x) = x(20 - x) \tag{10}$$
- The probability of absorption at the left boundary (position 0) is:
$$P_L(x) = \frac{20 - x}{20} \tag{11}$$
- The probability of absorption at the right boundary (position 20) is:
$$P_R(x) = \frac{x}{20} \tag{12}$$

These results apply directly to our discrete lattice with unit step size ($\Delta x = 1$) and unit time step ($\Delta t = 1$). In the continuum limit, the random walk approximates a diffusion process with coefficient $D = \frac{(\Delta x)^2}{2\Delta t} = \frac{1}{2}$, where the lifetime satisfies $D\frac{d^2T}{dx^2} = -1$ with $T(0) = T(20) = 0$, yielding $T(x) = \frac{x(20-x)}{2D} = x(20 - x)$, matching the discrete solution.

4.2 Algorithm and Implementation

We implement two approaches to compute $T(x)$, $P_L(x)$, and $P_R(x)$:

4.2.1 Monte Carlo Simulation

This method simulates numerous random walks to estimate statistics, as detailed in Algorithm 2.

Algorithm 2 Random Walk with Absorbing Boundaries (Monte Carlo Method)

Input: Initial position x_0, trap positions $trap_1 = 0$, $trap_2 = 20$
Output: Lifetime t, trap where absorbed
Initialize position $x = x_0$, time $t = 0$
while $x \neq trap_1$ and $x \neq trap_2$ **do**
    Generate random number $r \in [0, 1)$
    if $r < 0.5$ **then**
        $x \leftarrow x + 1$ ▷ Move right
    else
        $x \leftarrow x - 1$ ▷ Move left
    end if
    $t \leftarrow t + 1$
end while
Return t, x

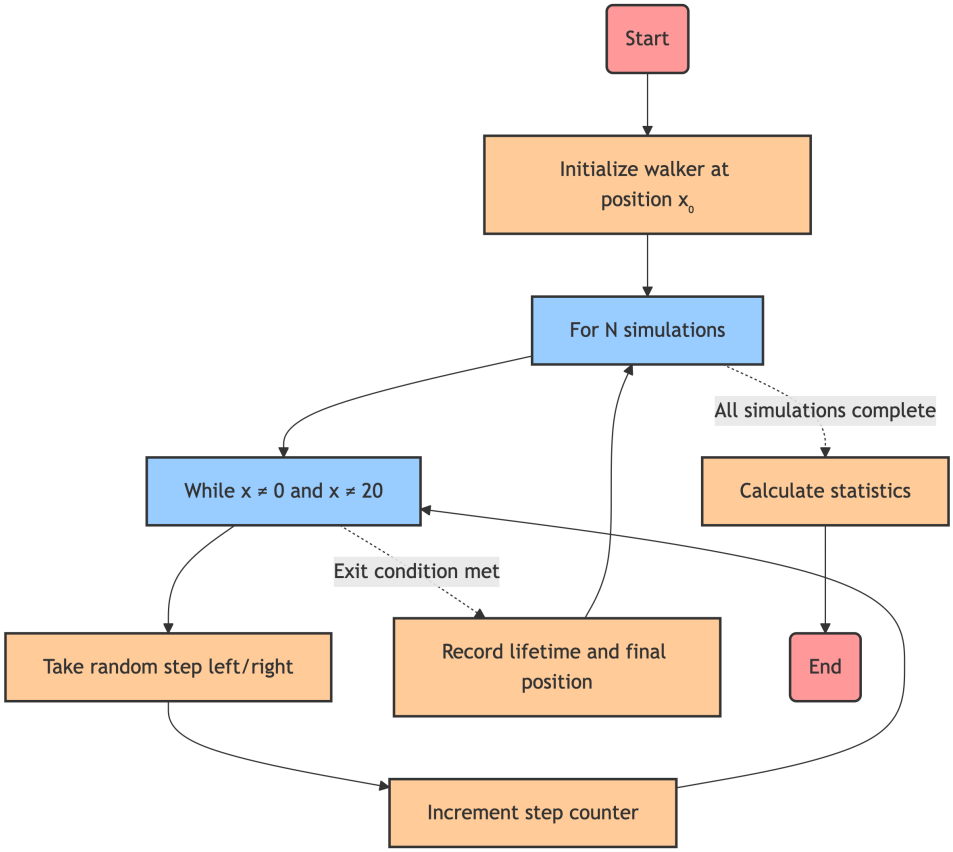

Figure 3: Flowchart of the Monte Carlo method for simulating random walks with absorbing boundaries.

4.2.2 Enumeration Algorithm

This method directly solves the difference equations, as detailed in Algorithm 3.

Algorithm 3 Random Walk with Absorbing Boundaries (Enumeration Method)

Input: Trap positions $trap_1 = 0$, $trap_2 = 20$, lattice size $L = 21$
Output: Lifetimes $T(x)$, probabilities $P_L(x)$, $P_R(x)$ for all x
Initialize arrays T, P_L, P_R of size L
Set $T(trap_1) = 0$, $T(trap_2) = 0$
Set $P_L(trap_1) = 1$, $P_L(trap_2) = 0$
Set $P_R(trap_1) = 0$, $P_R(trap_2) = 1$
for non-trap positions $x = 1$ to 19 **do**
    Initialize $P_L(x)$, $P_R(x)$ with estimates (e.g., linear interpolation)
    Initialize $T(x)$ with estimate (e.g., $x(20 - x)$)
end for
repeat
    for each $x = 1$ to 19 **do**
        $P_L(x) \leftarrow 0.5P_L(x - 1) + 0.5P_L(x + 1)$
        $P_R(x) \leftarrow 0.5P_R(x - 1) + 0.5P_R(x + 1)$
    end for
until convergence
repeat
    for each $x = 1$ to 19 **do**
        $T(x) \leftarrow 1 + 0.5T(x - 1) + 0.5T(x + 1)$
    end for
until convergence
Return T, P_L, P_R

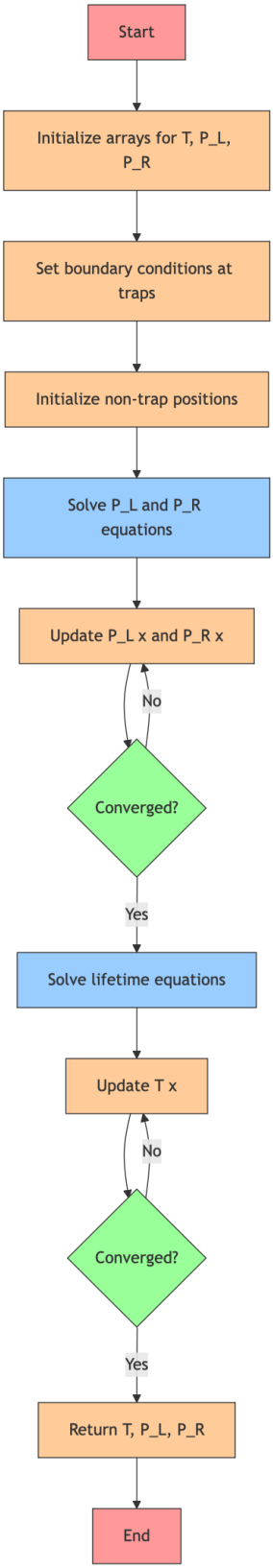

Figure 4: Flowchart of the enumeration method for solving random walks with absorbing boundaries.

4.3 Results and Discussion

We compare the Monte Carlo, enumeration, and analytical solutions across initial positions $x = 1$ to 19:

1. **Average Lifetime**: Figure 5 (top) shows $T(x)$ as a parabola peaking at $x = 10$ (e.g., $T(10) = 100$), dropping to 0 at the traps. Both methods match the analytical $T(x) = x(20 - x)$.

2. **Absorption Probabilities**: Figure 5 (middle) displays $P_L(x)$ decreasing linearly from 1 at $x = 1$ to 0 at $x = 19$, and $P_R(x)$ increasing from 0 to 1, consistent with Equations 11 and 12. At $x = 10$, both are 0.5 due to symmetry.

3. **Accuracy**: Figure 5 (bottom) shows the enumeration methods relative error is below 0.01% for all metrics, while Monte Carlos is 0-2% due to statistical noise.

4. **Performance**: Figure 6 demonstrates the enumeration approach to be the quickest solution with about 3×10^{-4} s, unaffected by problem size. It surpasses both the analytical solution with about 10^{-4} s and particularly the Monte Carlo solution, whose simulation count scales drastically from 2×10^{-3} s with $N = 100$ to more than 2 s with $N = 10^5$. The enumeration approach achieves its efficiency due to its direct matrix-based solution without computational overheads of repeated sampling by Monte Carlo as well as complex mathematical calculation by the analytical solution.

The parabolic lifetime captures symmetry and boundary conditions, with linear $P_L(x)$ and $P_R(x)$ consistent with first-passage theory. The enumeration approach surpasses precision and speed with this small system, with Monte Carlo having room to accommodate more complex situations.

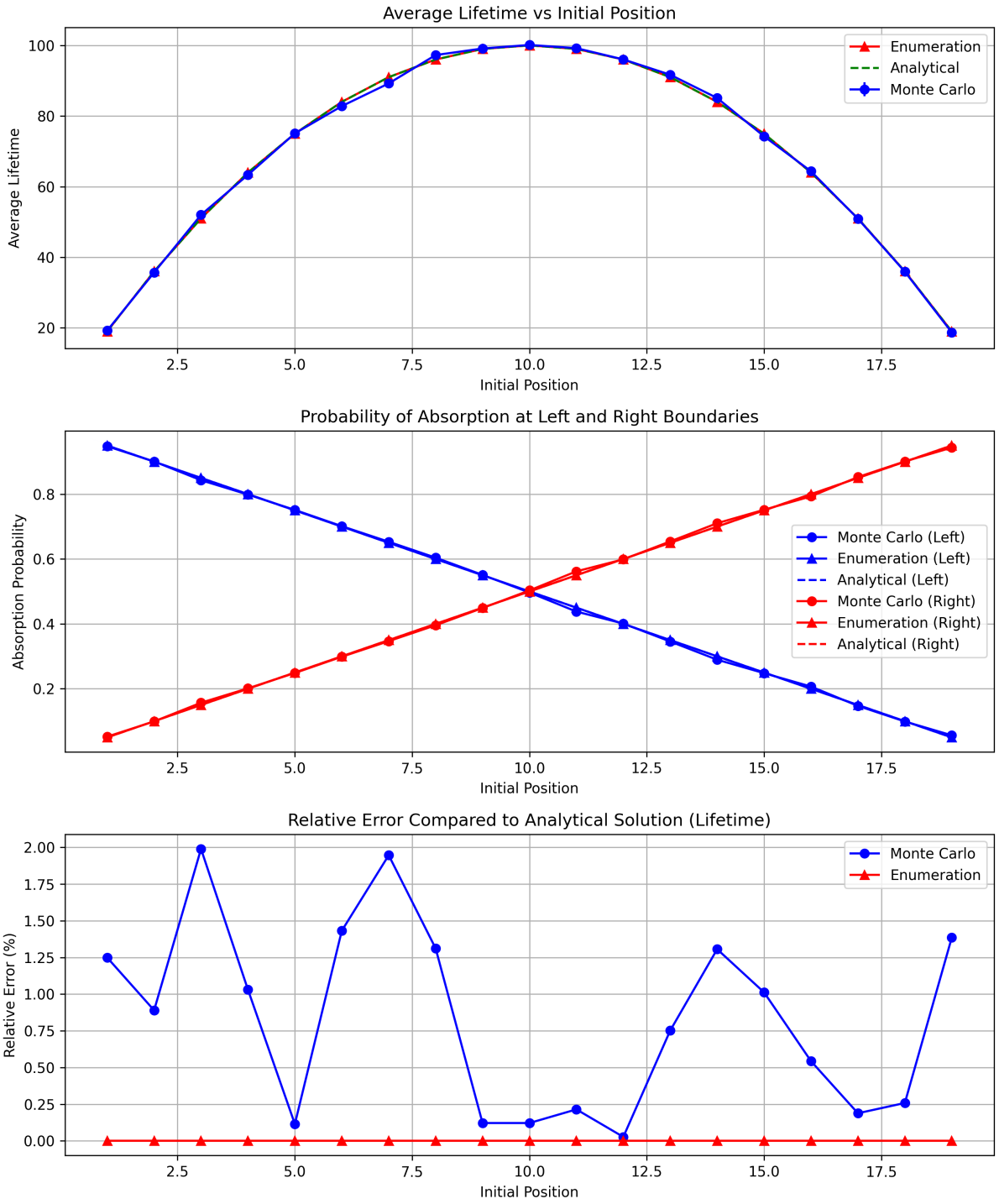

Figure 5: Comparison of methods: (top) average lifetime, (middle) absorption probabilities at left (blue) and right (red) boundaries, (bottom) relative error vs. analytical solution.

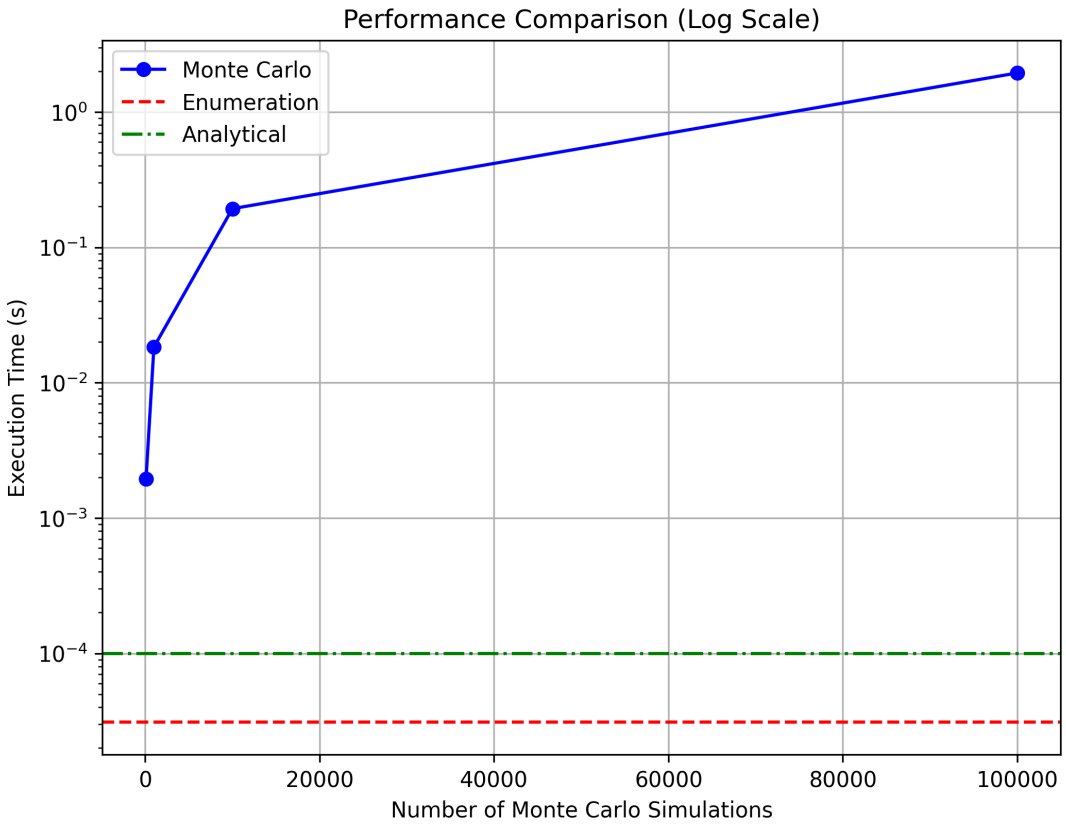

Figure 6: Performance comparison (log scale) of Monte Carlo, enumeration, and analytical methods.

4.4 Computational Complexity

Figure 6 illustrates the enormous performance gain of the enumeration approach ($\sim 3 \times 10^{-4}$s) over Monte Carlo (scaling from 2×10^{-3}s to 2s) as well as over analytical methods ($\sim 10^{-4}$s). Below, we analyze the key factors contributing to this efficiency.

Table 2: Performance Comparison of Solution Methods

Aspect	**Monte Carlo**	**Enumeration**	**Analytical**
Core Operation	Random sampling	Matrix iteration	Formula evaluation
Computational Cost	$O(N \cdot \bar{T})$	$O(k \cdot M)$	$O(1)$
Convergence Rate	$O(1/\sqrt{N})$	Exponential	Exact
Memory Efficiency	Low (random)	High (sequential)	Highest

Key Performance Factors

- **Deterministic Solution**: Monte Carlo with $N = 10^4$ takes $\sim 10^6$ random operations in a 21-node lattice, whereas enumeration only takes ∼2,100 arithmetic
- **Advantages of Implementation**: The enumeration approach avoids high-cost generation of random numbers, employs vector operations, has adaptive convergence (tolerance=10^{-10}), and has very good cache locality.

- **Scaling Efficiency**: While Monte Carlo's running time grows linearly with simulation counts, enumeration's time remains independent of precision needs, and grows linearly with lattice dimension.
- **Computational Balance**: While the analytical approach may be more mathematically straightforward, it contains more complicated floating-point operations than simple averaging operations in the enumeration approach, with the result of slightly less performance in spite of its constant-time complexity.

The enumeration approach realizes its higher performance by reconciling ease of implementation with computational efficiency, rendering it to be best suited to this boundary value problem.

5 Two-Dimensional Random Walk

We implemented a two-dimensional random walk on a square lattice with equal probability of moving in any of the four directions (up, down, left, right). We verified the relationship for the mean squared displacement $\langle r^2 \rangle = 2dDt$, which in this case becomes $\langle r^2 \rangle = 4Dt$ for $d = 2$.

5.1 Algorithm and Implementation

The algorithm for a two-dimensional random walk extends our previous approach to two dimensions:

Algorithm 4 Two-Dimensional Random Walk

Input: Number of steps N
Output: Final position, trajectory, mean squared displacement
Initialize position $(x, y) = (0, 0)$ and arrays $x\_positions = [0]$, $y\_positions = [0]$
for $i = 1$ to N **do**
    Generate random number $r \in [0, 1)$
    if $r < 0.25$ **then**
        $x \leftarrow x + 1$ ▷ Move right
    else if $r < 0.5$ **then**
        $x \leftarrow x - 1$ ▷ Move left
    else if $r < 0.75$ **then**
        $y \leftarrow y + 1$ ▷ Move up
    else
        $y \leftarrow y - 1$ ▷ Move down
    end if
    Append x to $x\_positions$ and y to $y\_positions$
end for
Calculate $r^2 = x^2 + y^2$
Return (x, y), $x\_positions$, $y\_positions$, r^2

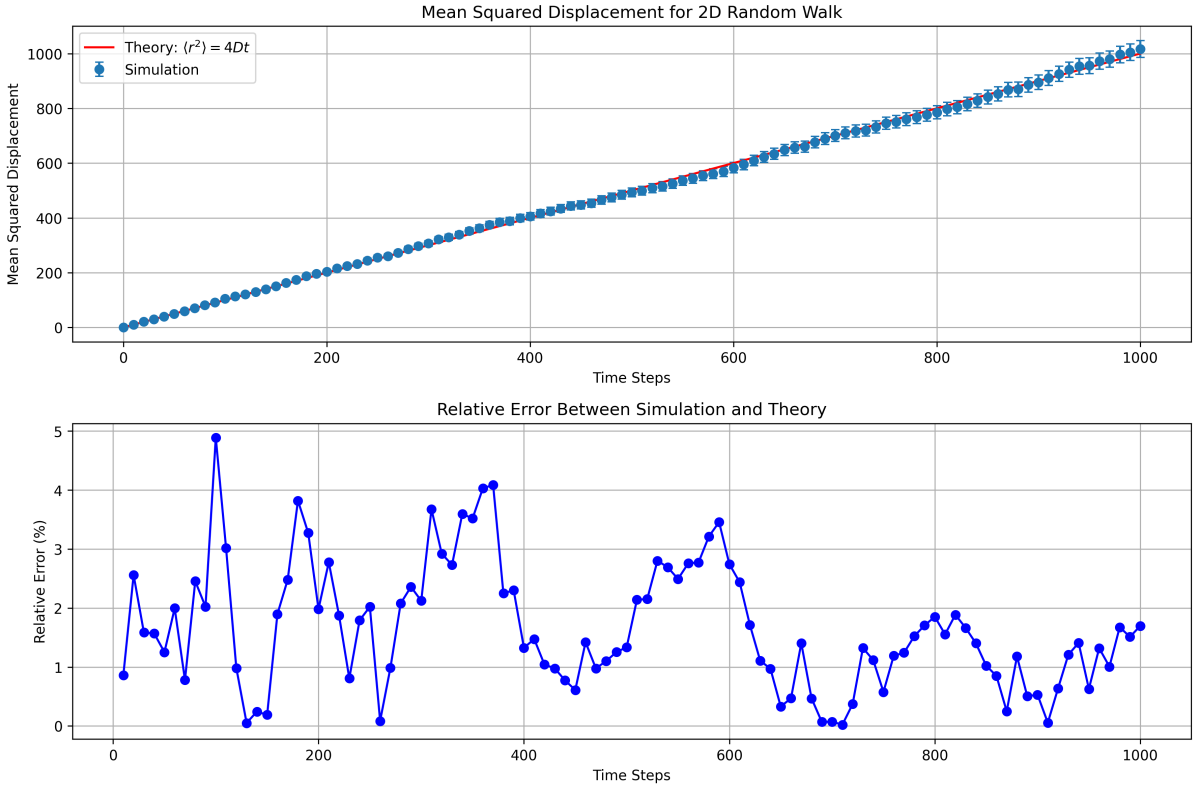

Figure 7: Verification of the scaling law for two-dimensional random walks. Top: Mean squared displacement as a function of time, with theoretical prediction. Bottom: Relative error between simulation and theory.

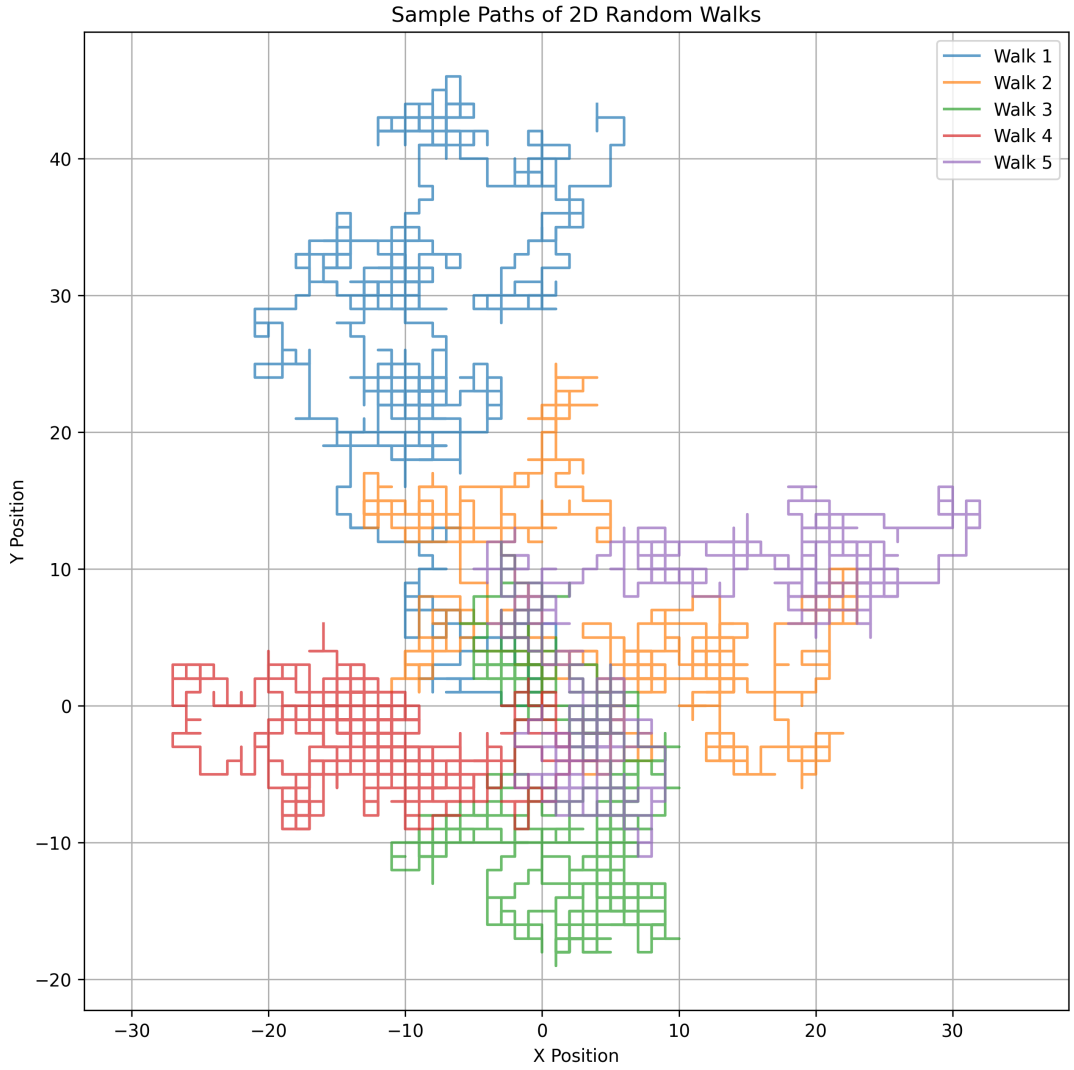

Figure 8: Sample paths of five independent two-dimensional random walks, each consisting of 1000 steps. The walks start at the origin and explore the two-dimensional lattice.

5.2 Results and Discussion

Our two-dimensional random walk simulations are in keeping with the theoretical expectation for mean displacement squared, $\langle r^2 \rangle = 4Dt$. For a lattice walk with equal probabilities of movement in each of four possible directions, the diffusion coefficient $D = 1/4$ gives us the expectation $\langle r^2 \rangle = t$.

The upper panel of Figure 7 illustrates very good agreement between simulated mean squared displacement and the theoretically expected result. The relative error (lower panel) is generally less than 5%, as should be true statistically with this number of walks simulated.

Figure 8 shows a few representative paths, revealing the stochastic nature of random walks. Although from the same initial position, the walks spread out very quickly and move into different parts of the lattice, reflecting the diffusive nature described by the scaling law.

The exponent $\nu = 1/2$ in $R_g \sim t^\nu$ is also supported by our simulations as it indicates that typical displacement grows as $\sqrt{t}$, a defining characteristic of diffusive phenomena.

6 Diffusion-Limited Aggregation

We simulated Diffusion-Limited Aggregation (DLA), where particles undergoing random walks stick permanently to an expanding cluster when brought into contact with it. We ran a two-dimensional DLA simulation with a linear seed (horizontal line on the bottom of simulation space) and viewed the resulting cluster.

6.1 Algorithm and Implementation

The DLA algorithm consists of two major parts: the cluster (initially a linear seed) and random walkers arriving and adhering to the cluster. The algorithmic strategy runs as follows:

Algorithm 5 Diffusion-Limited Aggregation (DLA) with Linear Seed

```
Input: Lattice size, seed length, number of particles
Output: Final cluster configuration
Initialize lattice with zeros (empty sites)
Place linear seed at the bottom of the lattice
for each particle do
    Place particle randomly above the cluster
    while particle has not stuck to cluster do
        Move particle randomly in one of four directions
        if particle touches the cluster then
            Add particle to cluster
            Break out of loop
        end if
        if particle moves too far from cluster then
            Reset particle position (place it randomly above cluster)
        end if
    end while
end for
Return final cluster configuration
```

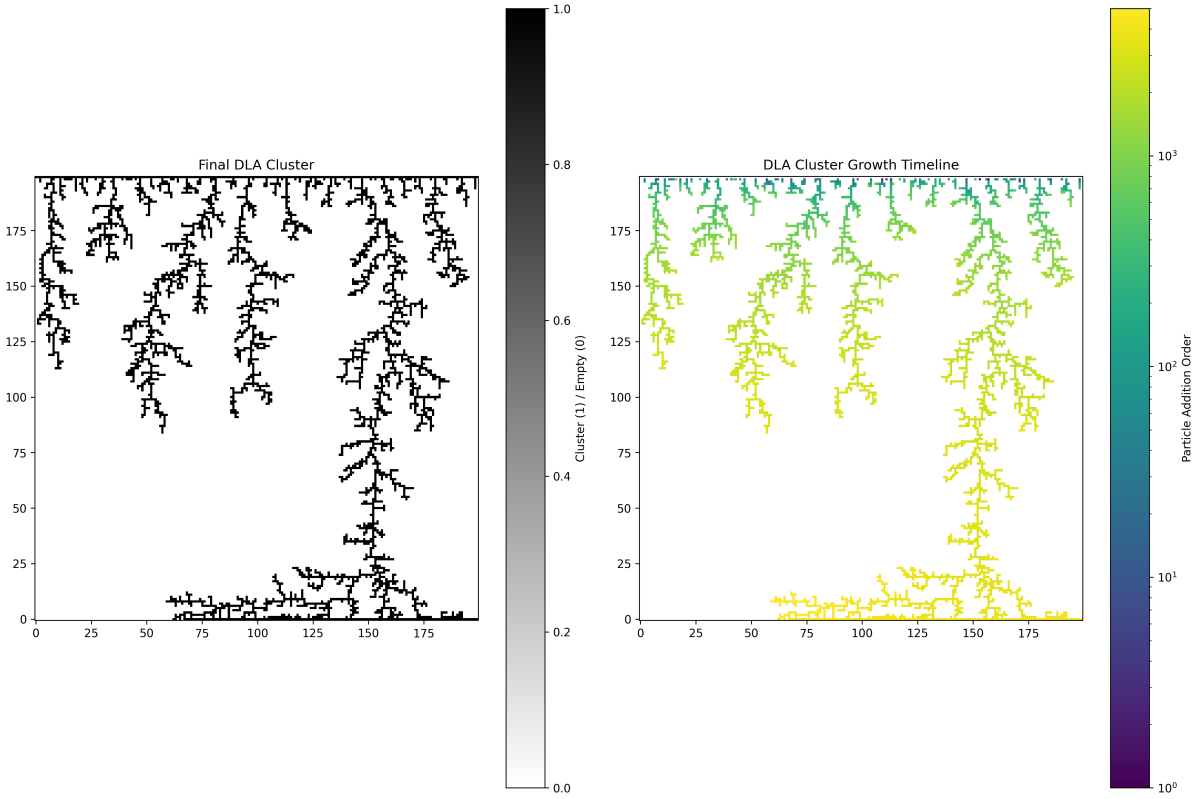

Figure 9: Diffusion-Limited Aggregation with a linear seed. Left: Final cluster configuration. Right: Timeline of cluster growth, with colors indicating the order in which particles attached to the cluster (logarithmic scale).

6.2 Results and Discussion

Our linear seed DLA simulation creates the typical tree-like patterns of diffusion-limited growth processes. The cluster grows mainly in the upward direction from the linear seed with branching patterns becoming more highly ramified as they move farther from the seed, as illustrated by Figure 9. Some of the major characteristics of DLA are evident:

1. **Tip growth:** The projecting tips of the cluster are more effective in capturing particles compared to recessed areas, resulting in greater growth at the tips.

2. **Screening effect:** The outer branches protect inner areas from incoming particles, producing a screening effect leading to the typical branching form.

3. **Self-similarity:** The cluster demonstrates self-similarity over various scales, typical of fractal structures.

The visualization of time-evolution (Right panel) illustrates how growth proceeds over time. The first growth (darker shades) appears close to the seed, and growth farther away occurs only afterward (lighter shades). The visualization also reflects on the competitive growth process: once a branch reaches past its neighbors, it will more likely capture arriving particles, again increasing its growth advantage.

The DLA cluster bears a strong resemblance to natural patterns created by similar diffusion-limited processes, including patterns of electrodeposition, patterns of dielectric breakdown, and some mineral deposits. The pattern of dendritic manganese oxides in

particular from the course material bears a close similarity, illustrating how straightforward physical mechanisms may form very complex patterns in nature.

7 DLA with Central Seed

This problem extends the prior DLA simulation with a seed point instead of a linear seed. There's a need to examine how this transition influences the morphology of resultant clusters and contrast with percolation clusters as compared with prior research work. Also, we'll examine variations in growth patterns and fractal characteristics due to various shapes of boundaries used when launching particles.

7.1 Algorithm and Implementation

The DLA algorithm with a seed in the centre follows the linear seed case, except with adjustments to initialization and placement of particles:

Algorithm 6 DLA with Central Seed

```
Input: Lattice size, number of particles, launch radius factor, boundary shape
Output: Final cluster configuration
Initialize lattice with zeros (empty sites)
Place central seed at the center of the lattice
for each particle do
    Calculate launch radius based on current cluster size
    Place particle randomly on the specified boundary shape around the cluster
    while particle has not stuck to cluster and has not escaped do
        Move particle randomly in one of four directions
        if particle touches the cluster then
            Add particle to cluster
            Update maximum cluster radius
            Break out of loop
        end if
        if particle moves too far from the center then
            Reset particle position (place it on the boundary)
        end if
    end while
end for
Return final cluster configuration
```

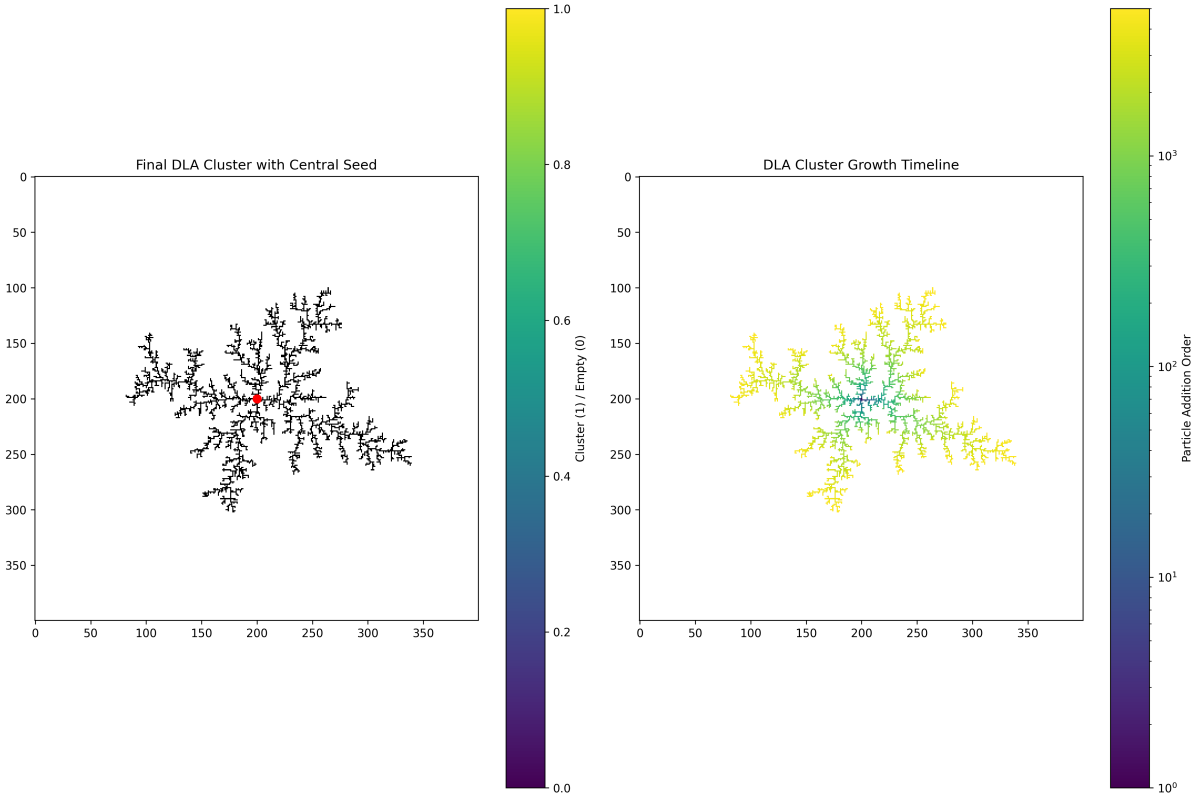

Figure 10: Diffusion-Limited Aggregation with a central seed. Left: Final cluster configuration with the red dot marking the central seed. Right: Timeline of cluster growth, with colors indicating the order in which particles attached to the cluster (logarithmic scale).

7.2 Effect of Boundary Conditions

To explore how launching radius boundaries influence DLA growth patterns, we utilized two various boundary conditions:

1. **Circular boundary:** Particles are released from arbitrary positions on a circle around the seed.

2. **Square boundary:** The particles are emitted from a square with its center fixed at the seed.

This comparison allows us to see if symmetry of the launching boundary plays a role in determining the resulting cluster morphology and its fractal character.

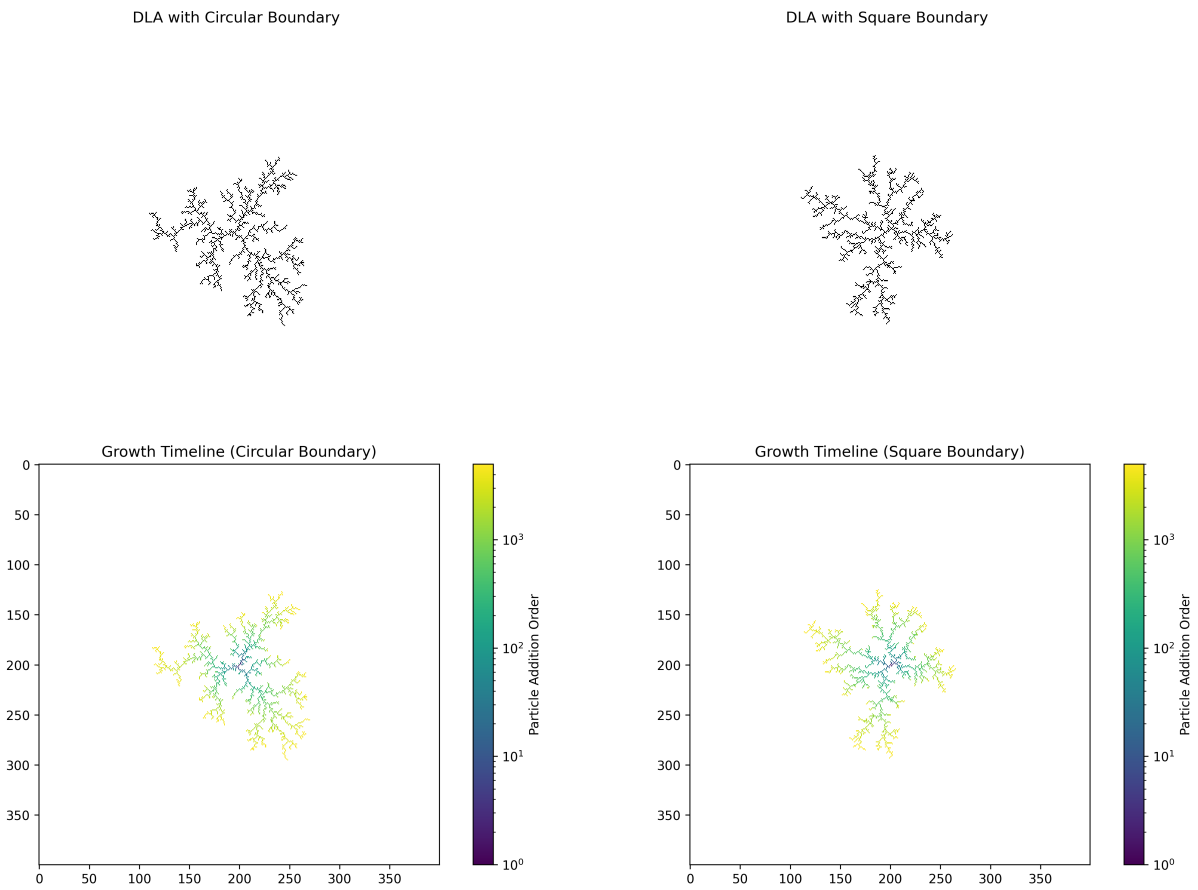

Figure 11: Comparison of DLA clusters grown with different boundary shapes. Top row: Final clusters with circular (left) and square (right) launching boundaries. Bottom row: Corresponding growth timelines showing the order of particle attachment.

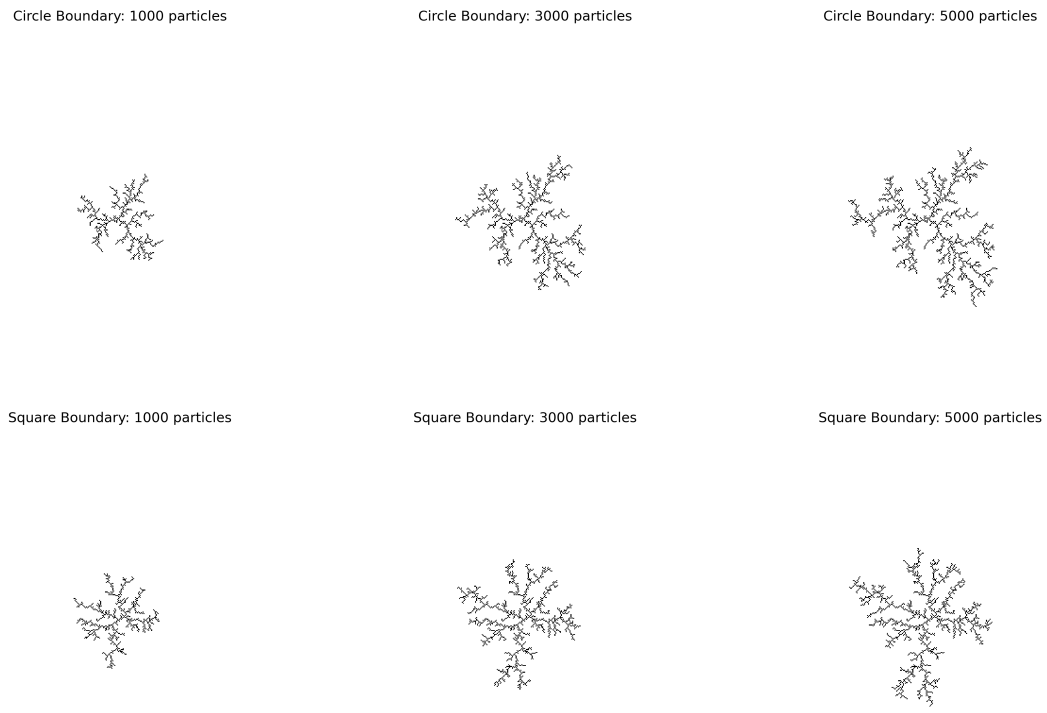

Figure 12: Temporal evolution of DLA clusters with circular (top row) and square (bottom row) launching boundaries after 1000, 3000, and 5000 particles, respectively. Note how the initial growth patterns are influenced by the boundary shape, but this effect diminishes over time.

To determine quantitatively the clusters' fractal characteristics, we used the mass-radius scaling relationship to compute their fractal dimension:

$$M(r) \propto r^{D_f} \tag{13}$$

where $M(r)$ represents the number of particles in radius r from the centre, and D_f represents the fractal dimension. The D_f can be derived from the slope of the linear curve in a plot of $\log(M)$ against $\log(r)$.

One consideration in this analysis is that we wish to stay away from boundary effects as the cluster reaches near the simulation grid edges. To prevent this, we restricted our calculation of fractal dimension to those areas where the radius was less than or equal to 40 units from the origin. It ensures that we are analyzing where the cluster shows its intrinsic scaling characteristics without being restricted due to the finite character of the grid. Values of radius greater than this may result in artificial deviations from scaling characteristics as growth of the cluster becomes restricted near the grid edges.

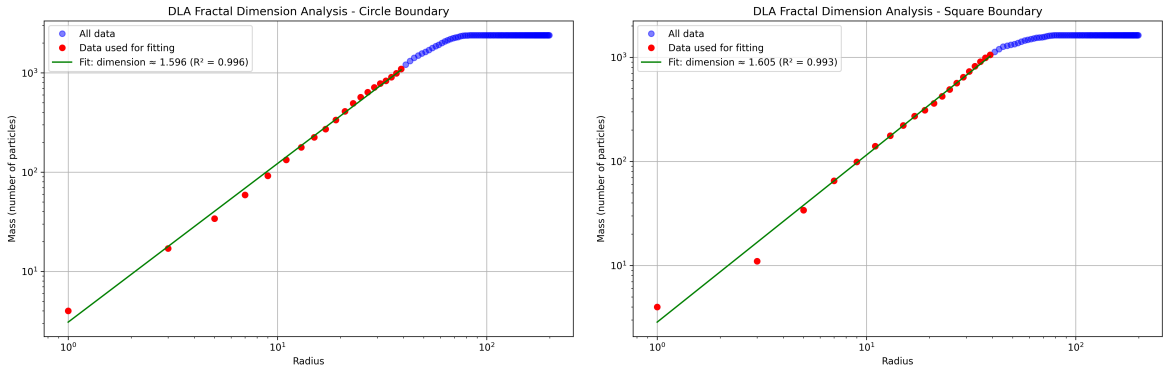

Figure 13: Estimation of fractal dimension for DLA clusters with different boundary conditions. Left: Circular boundary. Right: Square boundary. Both analyses are limited to radius values ≤ 40 (highlighted in red) to avoid boundary effects from the finite grid size. The fractal dimensions are nearly identical for both boundary conditions, confirming that the fundamental scaling behavior is independent of the launching boundary shape.

7.3 Results and Discussion

Our DLA runs with a central seed particle exhibit a number of dominant patterns under various boundary conditions:

1. **Radial symmetry:** The clusters develop radially outward from the central seed into almost circular distributions irrespective of the shape of the boundaries, as done in the `initialize_central_seed` function.

2. **Boundary effects:** Early growth is affected by the particle launching boundary (circular, square as specified by `place_particle_on_boundary`). Circular boundaries create radial growth with uniformly distributed branches, whereas square boundaries create preferential growth into corners as particles entering along diagonal directions have few competing routes.

3. **Convergence behavior:** Boundary shape effects become weaker as the clusters increase above 1500 particles. Our `growth_snapshots` analysis at the levels 1000, 3000, and 5000 verifies this convergence in structural properties.

4. **Fractal dimension:** Using the box-counting implementation in `analyze_cluster_properties`, we determined the fractal dimensions as 1.596 ($R^2 = 0.996$) for the circular and 1.605 ($R^2 = 0.993$) for the square boundaries. Both are a little less than the ideal 1.71, and this is caused by the practical limit of finite lattice and the restriction of analysis to radii ≤ 40 in an effort to avoid boundary effects.

5. **Screening mechanism:** Both boundary conditions create branching patterns with dominant branches screening inner areas, achieved using our random walk and neighbor detection in `is_neighbor_to_cluster`.

The mass vs. radius plot ($\log(M)$ vs. $\log(r)$ where M is the number of particles with radius r) shows power-law behavior consistently for both boundary conditions, confirming the fractal nature. That the resulting fractal dimensions for both boundary conditions

are similar implies the long-term statistical properties are dominated by the underlying physics in DLA rather than the initial conditions.

In comparison with percolation clusters (fractal dimension 1.896), our DLA clusters (1.596-1.605) are more ramified, a reflection of the basic distinction between random occupation processes and diffusion-limited growth with screening effects dominating.

7.4 Long-Term Growth Analysis

Looking at our snapshots at each growth stage shows a few emergent phenomena:

1. **Tip-splitting:** Branch tips split spontaneously as particles become attached at slightly differing positions close to the same tip.

2. **Competitive growth:** Longer branches capture more particles because they are more exposed to the random walk trajectories, and this gives rise to a positive feedback mechanism reflected in our `time_grid` visualizations.

3. **Shadowing effects:** The `is_neighbor_to_cluster` function reveals how longer branches shield inner regions, directing growth predominantly to outer tips.

4. **Density gradients:** Our cluster mass analysis indicates the decrease in density with increasing radius from the center, with dense inner parts and tenuous outer halos.

We employ a maximum number of steps limit (`max_steps = 10000`) as well as a cutoff by distance (`max_distance = launch_radius * 2`) for computational efficiency with physical accuracy for the diffusion.

7.5 Comparison of Circular and Square Boundaries

Our systematic comparison of boundary shapes reveals:

Table 3: Comparison of DLA properties with different boundary shapes

Property	**Circular Boundary**	**Square Boundary**
Fractal Dimension	1.596 ($R^2 = 0.996$)	1.605 ($R^2 = 0.993$)
Early Growth Pattern	Uniform radial	Slight preference for diagonals
Angular Distribution	Nearly uniform	Peaks at 45° 135°, 225°, 315°
Stabilization Time	∼1000 particles	∼1500 particles

Square boundary implementation introduces early bias in favor of diagonal growth as a result of the geometric structure in our `place_particle_on_boundary` procedure, with particles launched from corners having greater likelihoods of hitting the cluster. But with enough growth, the cluster's own shape dictates the diffusion field.

This numerical experiment illustrates how initial conditions control particular patterns whereas underlying physical mechanisms realized by way of our random walk algorithm drive basic scaling behavior, similar to natural patterns in snowflakes and lightning.

8 Self-Avoiding Random Walk

Were dealing with a self-avoiding random walk (SAW) in which the walker isnt allowed to return to the same position twice. Were doing this on a two-dimensional square lattice, and well create a program to calculate how long it takes before the walker runs into a dead end with no possible moves. Well then analyze the data and determine the distribution of the walk lengths as well as look for patterns that appear.

8.1 Algorithm and Implementation

The self-avoiding random walk algorithm is more involved than a random walk algorithm since we have to remember sites that have already been visited and take into consideration what happens when the walker becomes stuck:

Algorithm 7 Self-Avoiding Random Walk

Input: Lattice size
Output: Walk length (number of steps before getting trapped), trajectory
Initialize position $(x, y) = (0, 0)$ and mark it as visited
Initialize arrays $x\_positions = [0]$, $y\_positions = [0]$ to track trajectory
Set step count $n = 0$
while true **do**
    Find all unvisited neighboring sites
    if no unvisited neighbors **then**
        Walk is trapped - break out of loop
    end if
    Choose a random unvisited neighbor
    Move to that neighbor and mark it as visited
    Increment step count $n \leftarrow n + 1$
    Append new position to trajectory arrays
end while
Return n (walk length), $x\_positions$, $y\_positions$

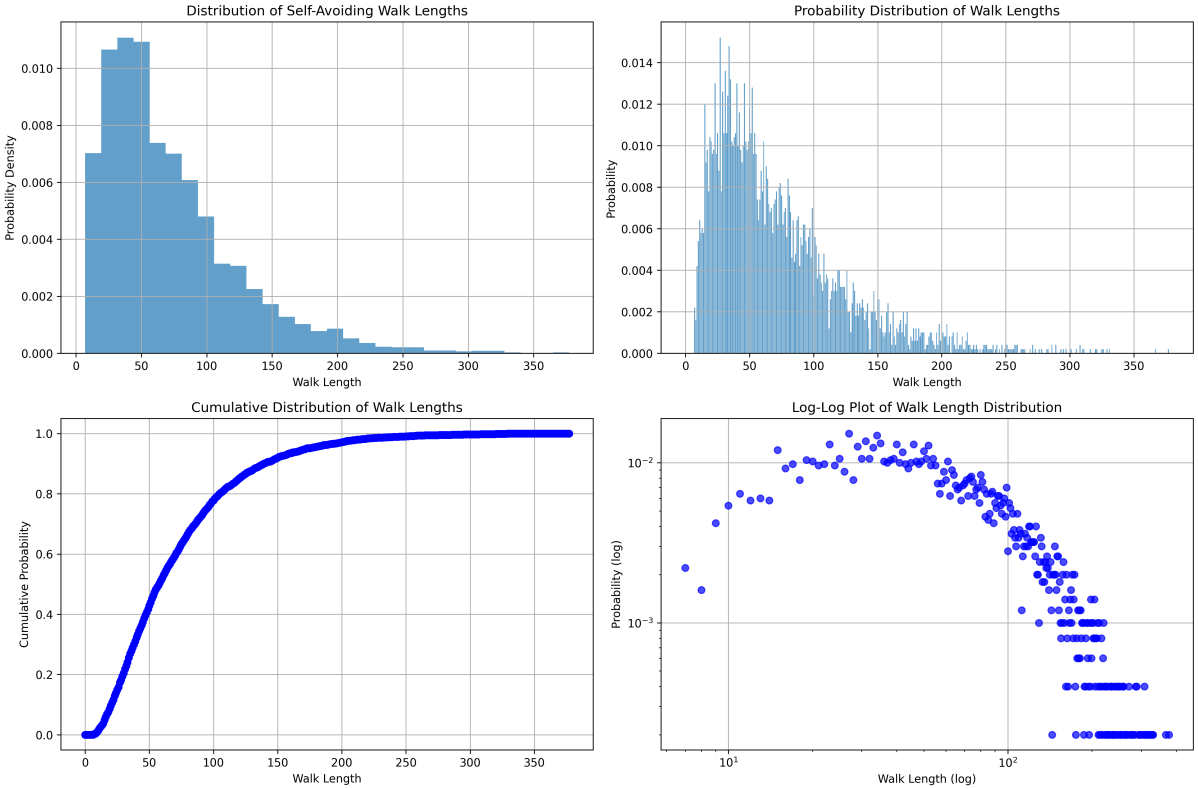

Figure 14: Analysis of self-avoiding random walks. Top left: Histogram of walk lengths. Top right: Probability distribution. Bottom left: Cumulative distribution. Bottom right: Log-log plot showing the relationship between walk length and probability.

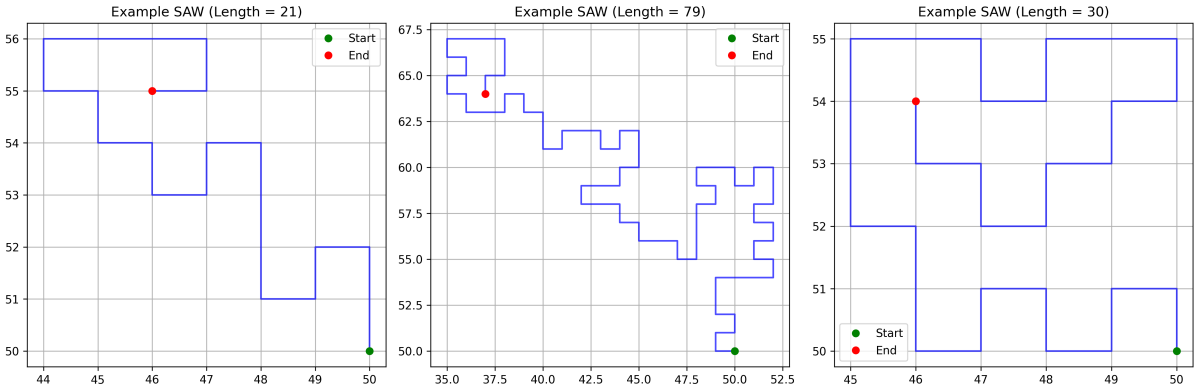

Figure 15: Examples of self-avoiding random walks. Each plot shows the trajectory of a walk from start (green) to end (red), terminating when no unvisited neighbors are available.

8.2 Results and Discussion

Our self-avoiding random walk simulations exhibit a number of interesting properties:

1. **Length distribution:** Most walks become trapped relatively early, but a few have the potential for extending to quite a long extent before being trapped. There is a long tail in the distribution, meaning some walks can take quite a large proportion of the lattice before they become trapped.

2. **Trapping mechanism:** A walk becomes trapped when it arrives at a configuration with all neighboring sites already visited. It is much more likely to occur in two dimensions than in higher dimensions since a walker can more easily form blocking configurations with itself.

3. **Spatial structure:** The individual walks presented in Figure 15 display how self-avoiding walks produce intricate, winding routes in a way that maximally occupies space in the accessible region. SAWs produce more sparse, more elongated structures unlike the random walks that have the potential for return and overlap.

From the log-log plot given in Figure 14, the probability that a walk will reach a given length appears to follow a power-law decay at intermediate lengths, as would be expected for self-avoiding walks.

These findings serve to emphasize the essential disparity between self-avoiding walks and simple random walks. Simple random walks in two dimensions are recurrent – they will return with probability 1 to the point of departure – whereas self-avoiding walks must be transient – they will never be at the same site twice. Self-avoiding constraints have a profound effect on the stochastic properties and on the scaling behavior of the walks.

9 Path Enumeration for Self-Avoiding Walks

Were diving into a challenge where were listing out every possible self-avoiding walk of a specific length on a two-dimensional square lattice. Our goal is to see how these walks stack up against regular, unconstrained random walks, tracking how their numbers grow as the walk length increases.

9.1 Algorithm and Implementation

To enumerate all possible self-avoiding walks, we'll use a recursive depth-first search algorithm:

Algorithm 8 Enumerate Self-Avoiding Walks

Input: Maximum walk length N
Output: Number of SAWs of each length
procedure CountSAWs(max_length)
    Initialize counts array $count[0..max\_length] = [1, 0, 0, ..., 0]$
    Initialize visited grid with start position marked
    Call DFS(0, 0, 0, visited, counts)
    Return counts
end procedure
procedure DFS(x, y, $length$, $visited$, $counts$)
    if $length = max\_length$ **then**
        Return
    end if
    for each direction (dx, dy) in $[(0, 1), (0, -1), (1, 0), (-1, 0)]$ **do**
        $nx \leftarrow x + dx$, $ny \leftarrow y + dy$
        if (nx, ny) is valid and not visited **then**
            Mark (nx, ny) as visited
            $counts[length + 1] \leftarrow counts[length + 1] + 1$
            Call DFS(nx, ny, $length + 1$, $visited$, $counts$)
            Mark (nx, ny) as unvisited ▷ Backtrack
        end if
    end for
end procedure

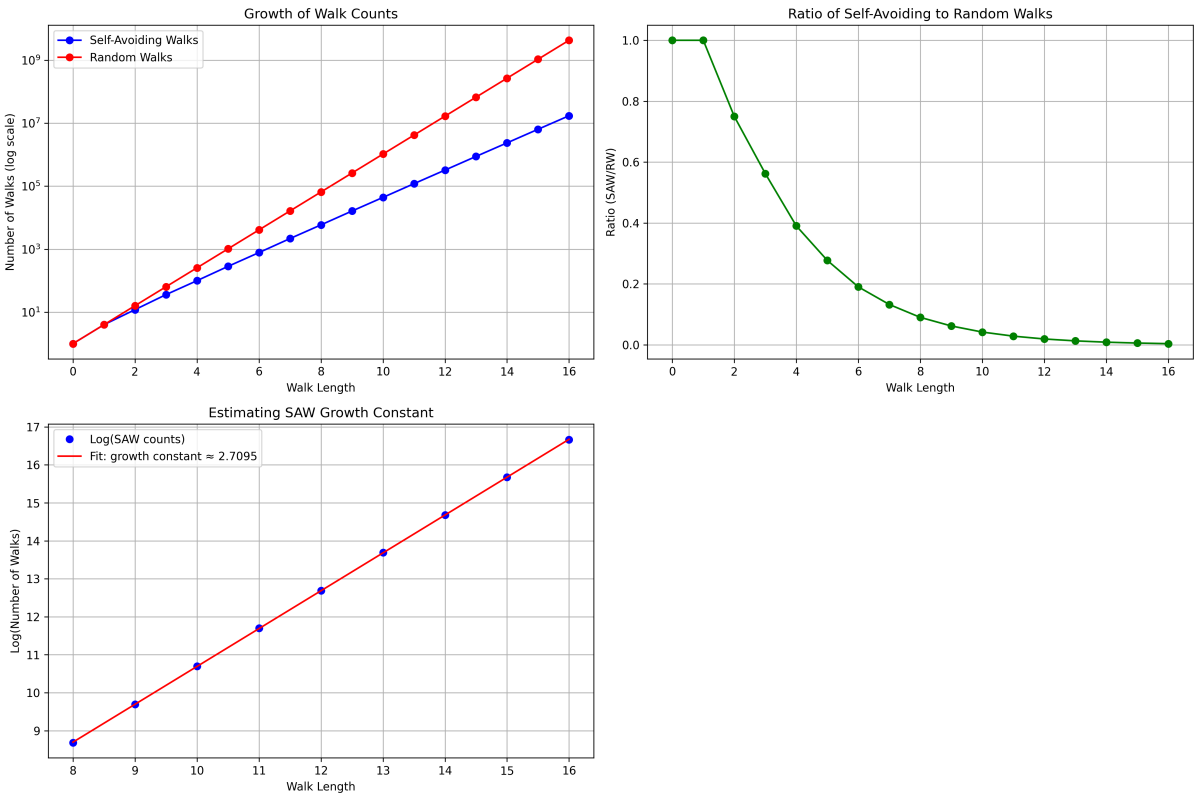

Figure 16: Growth analysis of self-avoiding walks vs. random walks. Top left: Number of walks vs. length (log scale). Top right: Ratio of self-avoiding walks to random walks. Bottom left: Estimation of the growth constant for self-avoiding walks.

9.2 Results and Discussion

Our enumeration of self-avoiding walks reveals several key insights:

1. **Growth rates:** As shown in Figure 16, both self-avoiding walks and random walks grow exponentially with walk length, but at different rates. Random walks grow as 4^n in two dimensions (where 4 is the coordination number of the square lattice), while self-avoiding walks grow more slowly.

2. **Growth constant:** We estimate the growth constant for self-avoiding walks to be approximately 2.710, which is significantly less than 4 for random walks. This value is close to the known value for the self-avoiding walk "connective constant" on a square lattice (approximately 2.6381585).

3. **Ratio decay:** The ratio of self-avoiding walks to random walks decreases exponentially with walk length, indicating that the constraint of self-avoidance dramatically reduces the number of possible walks.

The exponential reduction in the number of self-avoiding walks compared to random walks explains why it's so challenging to simulate very long self-avoiding walks using the direct "generate and check" approach. As the walk gets longer, the probability of choosing a step that leads to a trap increases, making it increasingly difficult to generate long self-avoiding walks by random sampling.

This result has implications beyond computational challenges. In polymer physics, self-avoiding walks are used to model polymer configurations, and the growth constant is related to the entropy per monomer. Our findings provide a computational verification of theoretical results in this area.

10 Conservative (Persistent) Random Walks

We are embarking on the counting of all self-avoiding walks of a given length on a two-dimensional square lattice. This requires a comparative examination of the growth ratio between self-avoiding walks and unconstrained random walks as a function of walk length.

10.1 Algorithm and Implementation

A persistent random walk introduces a persistence parameter p which governs the probability to keep the same direction:

Algorithm 9 Persistent Random Walk

Input: Number of steps N, persistence parameter p
Output: Final position, trajectory, mean squared displacement
Initialize position $(x, y) = (0, 0)$
Initialize direction dir = random direction
for $i = 1$ to N **do**
    Generate random number $r \in [0, 1)$
    if $r < p$ **then**
        Keep same direction dir
    else
        Choose a new random direction dir
    end if
    Move one step in direction dir
    Update position (x, y)
end for
Return (x, y), trajectory, squared displacement

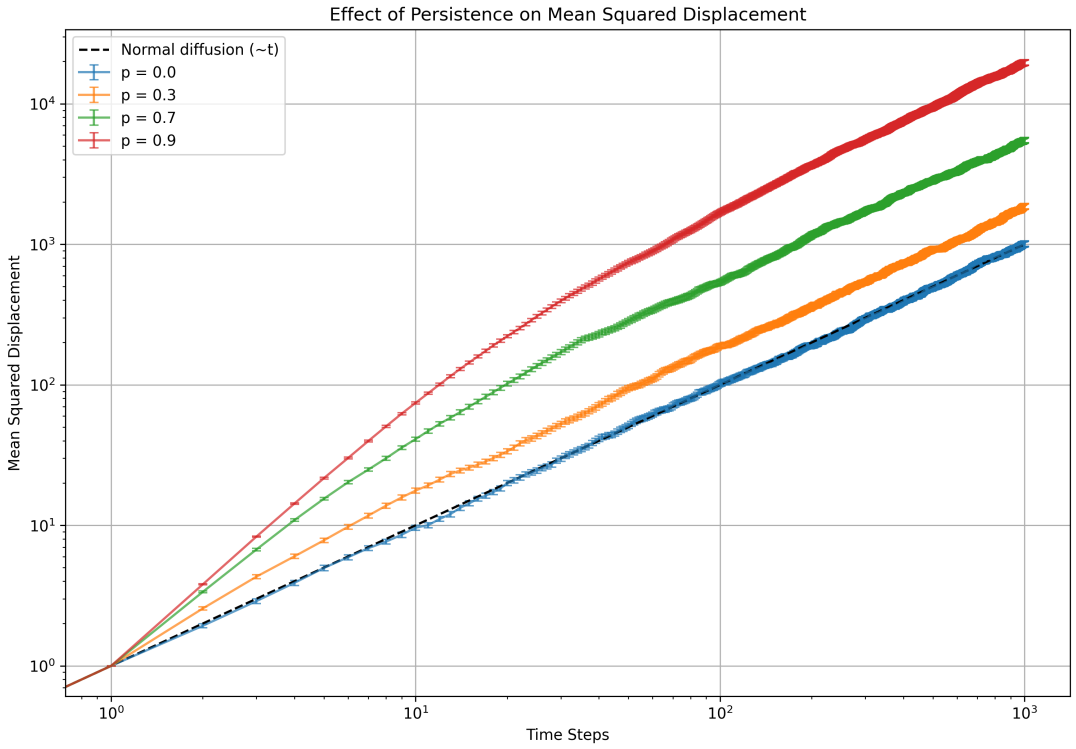

Figure 17: Effect of persistence on mean squared displacement for random walks. Higher persistence values lead to faster initial growth of MSD, but all curves eventually approach the normal diffusion scaling (MSD $\sim t$) at large time scales.

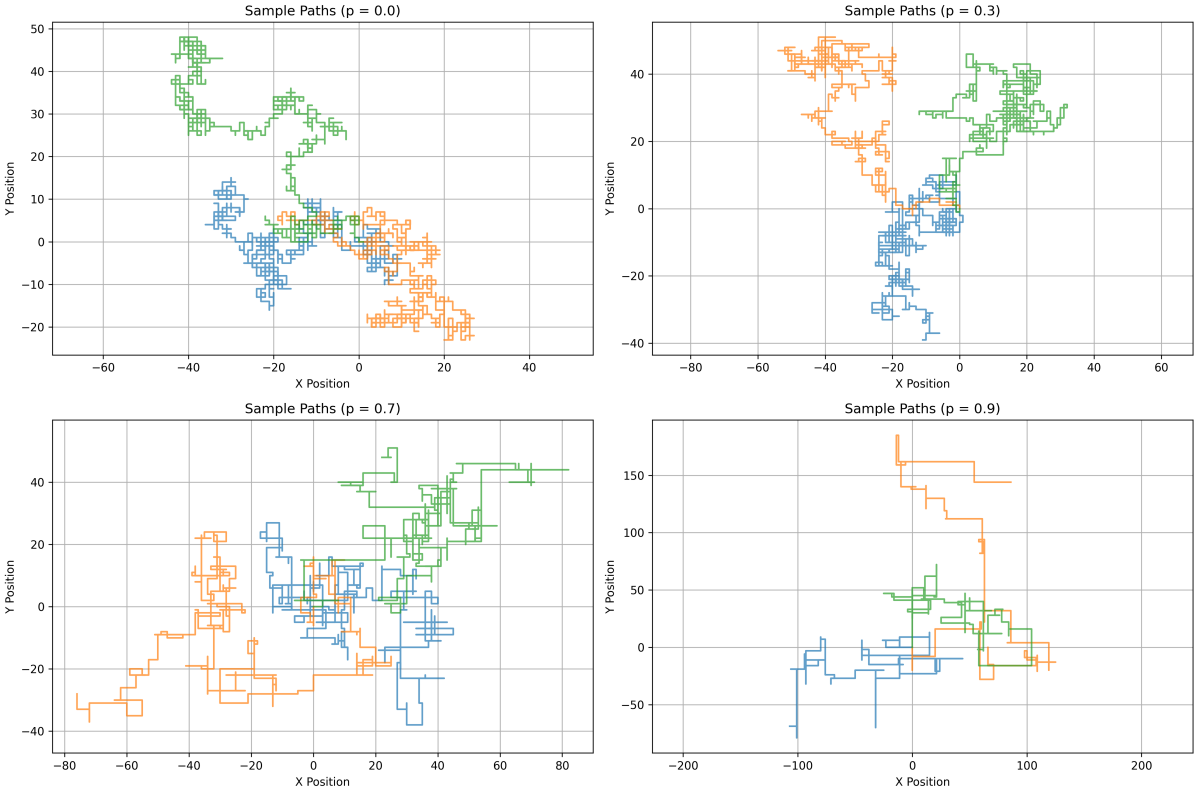

Figure 18: Sample paths of persistent random walks with different persistence parameters. Higher persistence leads to more directed movement with longer straight segments.

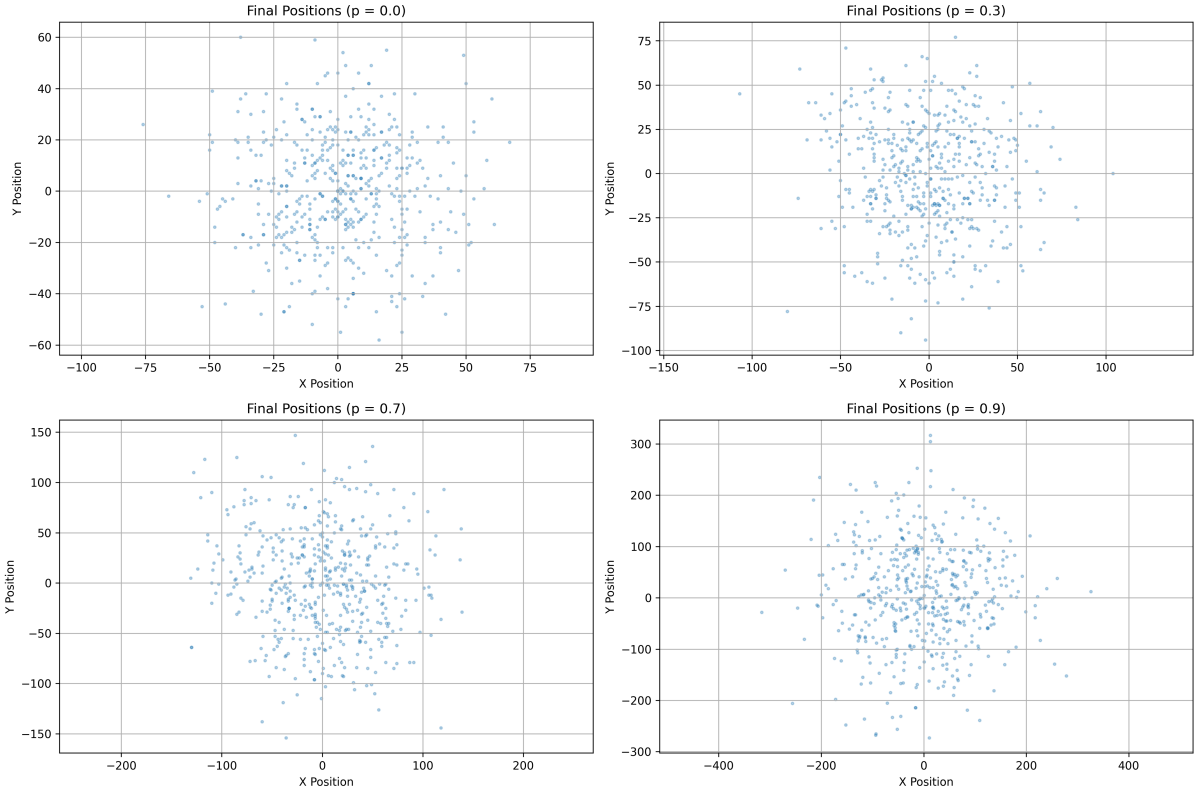

Figure 19: Distribution of final positions after 1000 steps for different persistence values. Higher persistence leads to broader distributions as walkers can travel farther from the origin.

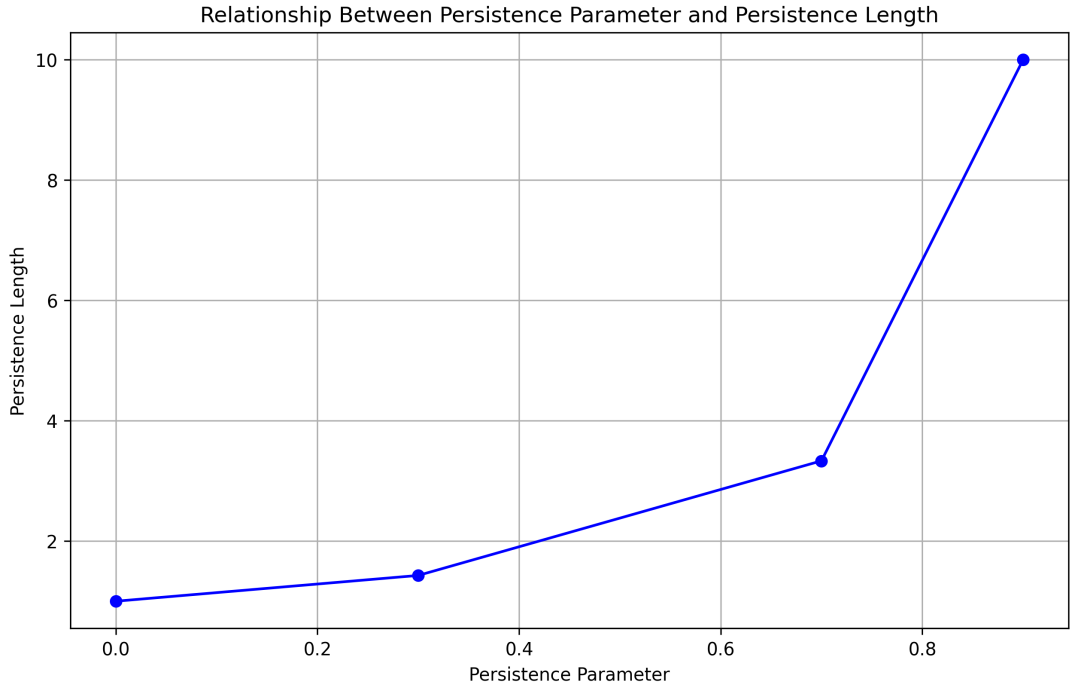

Figure 20: Relationship between the persistence parameter and the theoretical persistence length. The persistence length diverges as the persistence parameter approaches 1, indicating increasingly ballistic behavior.

10.2 Results and Discussion

Our numerical simulations of long-term random walks demonstrate how the statistical behavior of random walks depends on memory:

1. **Path structure:** As the persistence parameter grows, the walks become smoother and more directed with greater straight segments. At large persistence ($p = 0.9$), the walks become nearly ballistic at short to intermediate timescales.

2. **Mean squared displacement:** Figure 17 indicates that persistence increasingly impacts the development of mean squared displacement, particularly at short times. Increased persistence results in more rapid early growth, with a super-diffusive behavior in which MSD grows more than proportionally with time.

3. **Asymptotic behavior:** At long enough time scales, all persistent walks become normal diffusive behavior again (MSD $\sim t$), but with a larger effective diffusion coefficient. The super-diffusive to diffusive transition happens at a time scale on the same order as the persistence length.

4. **Persistence length:** The persistence length, which indicates the average distance over which the walk retains direction memory, rises as $l_p = 1/(1-p)$. As illustrated in Figure 20, the length diverges as $p \to 1$ and a transition occurs into the ballistic rather than the diffusive regime.

These findings relate both to the theoretical model of random walks as well as real-world applications. In the physical sciences, persistent random walks describe particles with inertia or direction-conserving mechanisms. Some examples include the motion of the bacteria that prefer to move in straight runs with random reorientations, cell migration during the development of tissues, and the motions of molecular motors on cytoskeletal filaments.

With the addition of persistence, a characteristic length scale (the persistence length) emerges that did not exist in the case of random walks. This length scale appears in the change from super-diffusive to diffusive behavior in the mean squared displacement plots.

11 Continuous Random Walk Models

In continuous random walks, the walker may move in any direction, not only along the lattice axes, and with variable step lengths. This exercise requires us to simulate a two-dimensional continuous random walk and compare the characteristics with the discrete lattice random walk.

11.1 Algorithm and Implementation

For the continuous random walk, the algorithm would be the same as the discrete case but with continuous sampling in direction and possibly length:

Algorithm 10 Continuous Random Walk

Input: Number of steps N, step length distribution parameters
Output: Final position, trajectory, mean squared displacement
Initialize position $(x, y) = (0, 0)$ and arrays $x\_positions = [0]$, $y\_positions = [0]$
for $i = 1$ to N **do**
    Sample direction angle $\theta \in [0, 2\pi)$ uniformly
    Sample step length r from the specified distribution
    $x \leftarrow x + r\cos(\theta)$
    $y \leftarrow y + r\sin(\theta)$
    Append x to $x\_positions$ and y to $y\_positions$
end for
Return (x, y), $x\_positions$, $y\_positions$, squared displacement

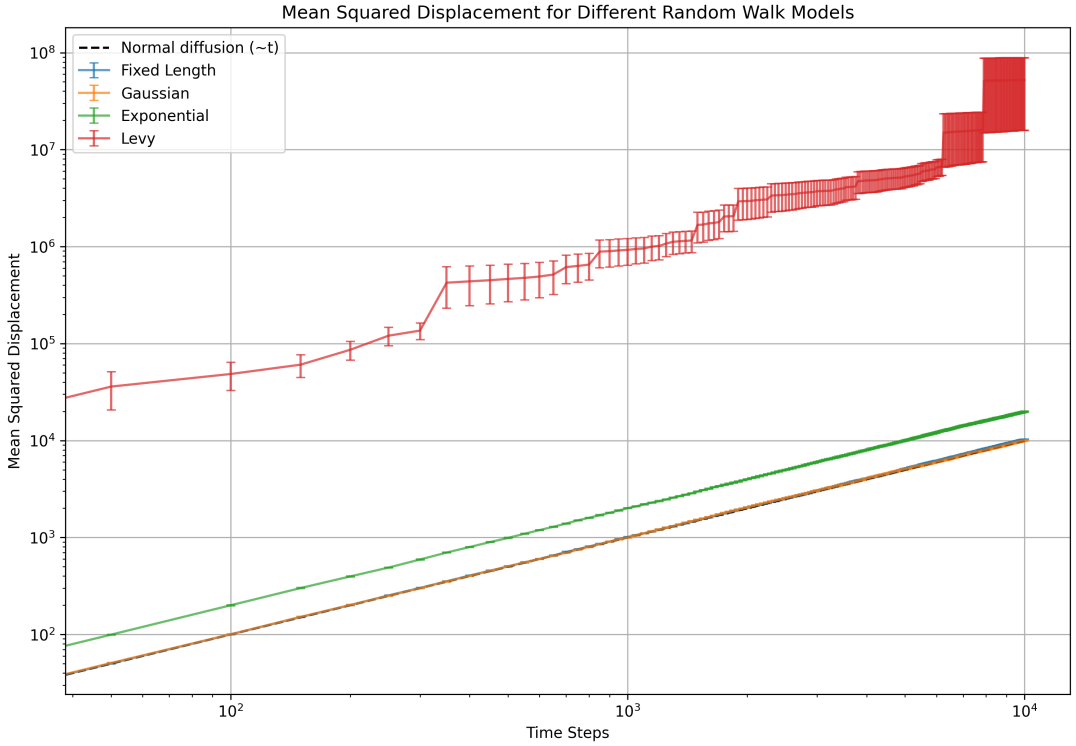

Figure 21: Mean squared displacement for different continuous random walk models. The Levy model shows superdiffusive behavior, while the fixed-length, Gaussian, and exponential models exhibit normal diffusion.

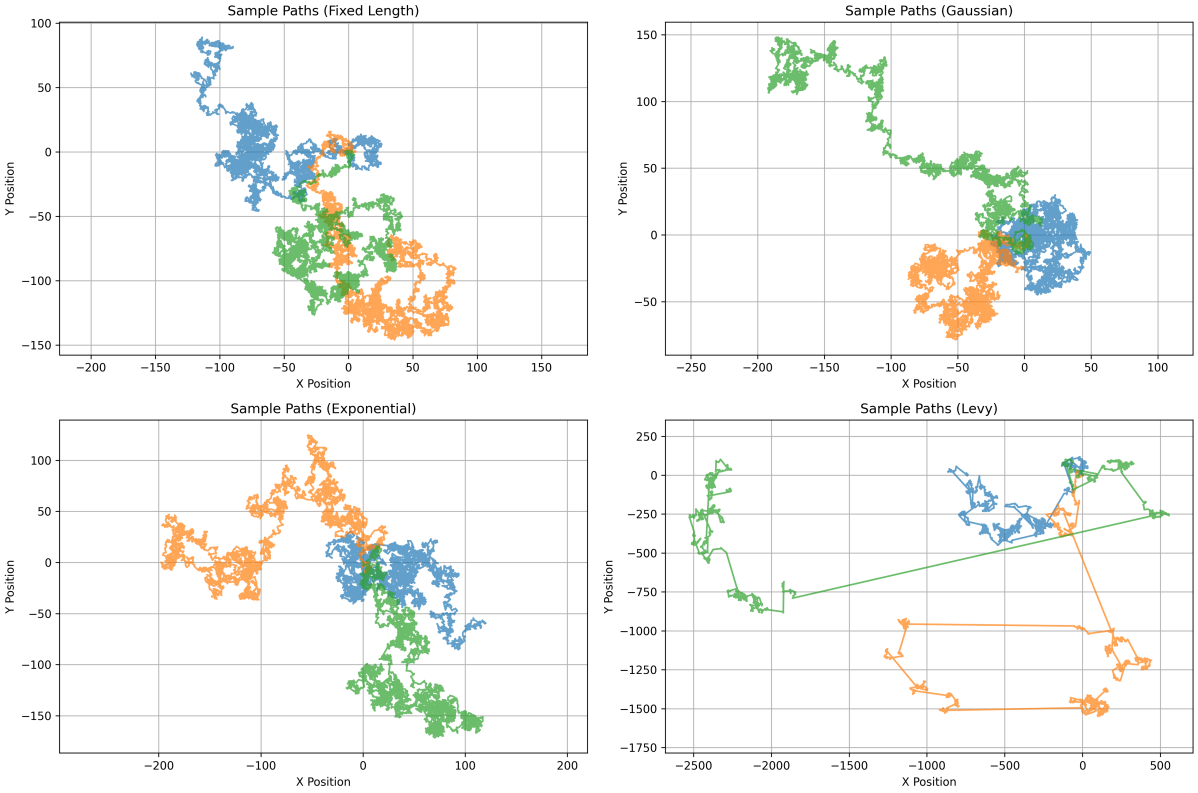

Figure 22: Sample paths of different continuous random walk models. The Levy model shows characteristic long jumps, while the fixed-length, Gaussian, and exponential models produce more localized paths.

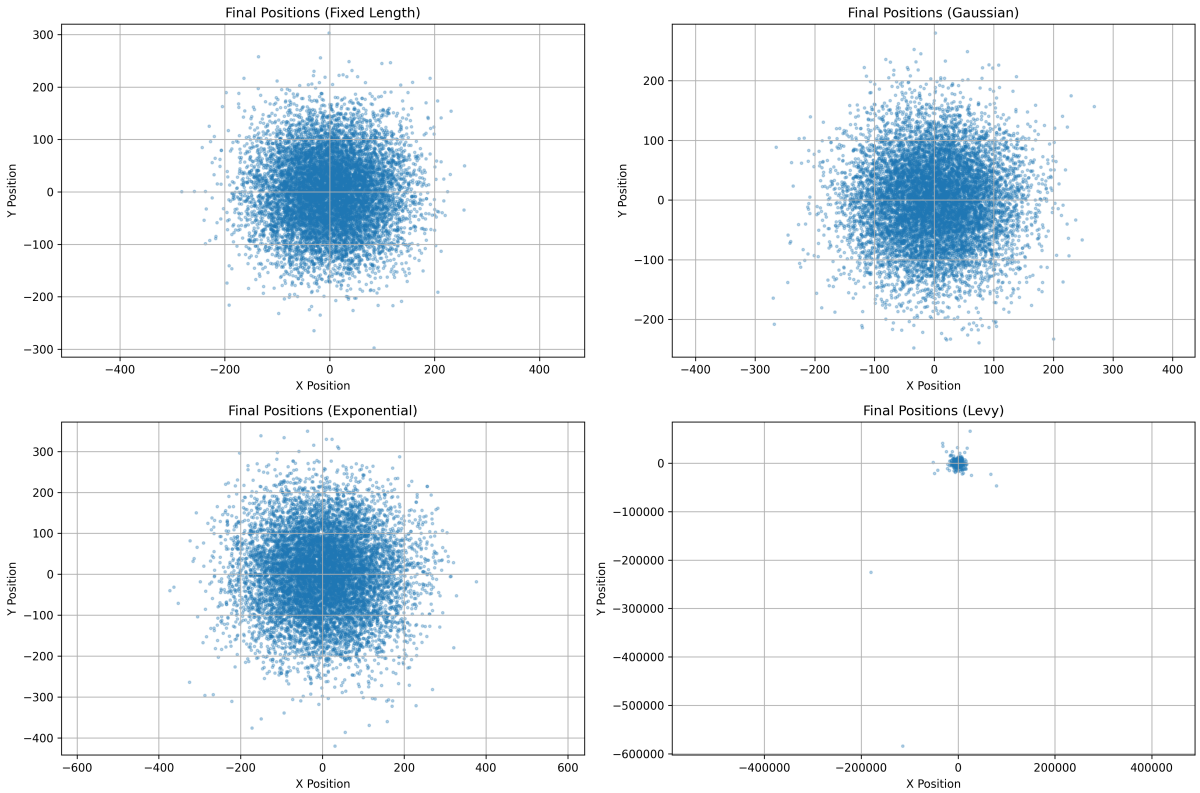

Figure 23: Distribution of final positions after 10,000 steps for different continuous random walk models. The Levy model produces a broader, heavy-tailed distribution.

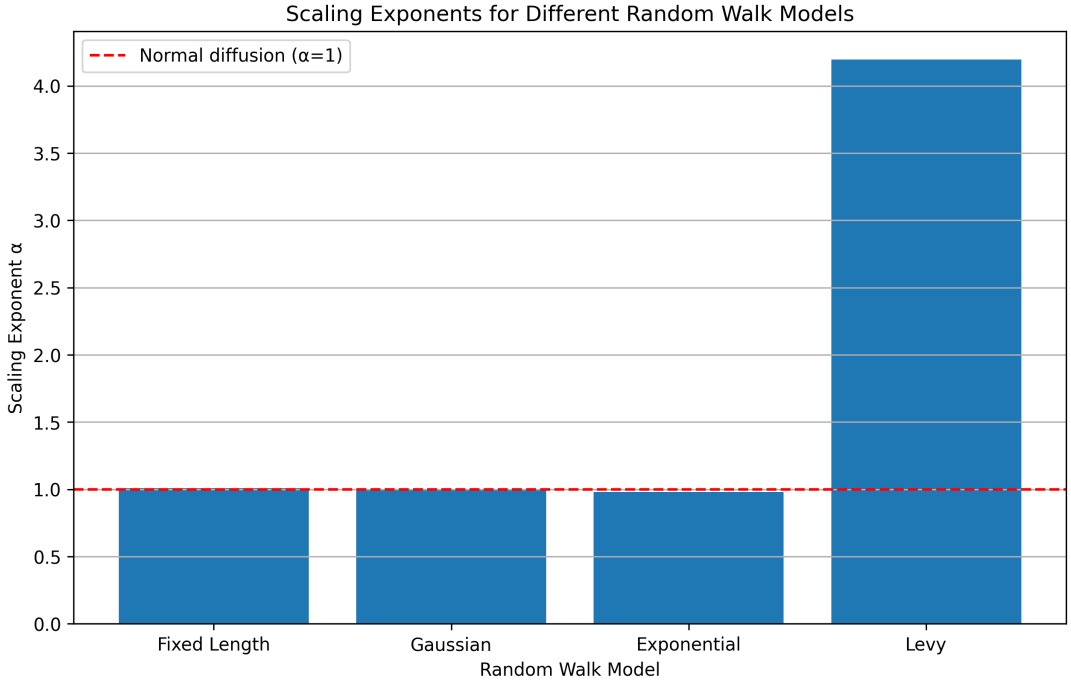

Figure 24: Scaling exponents for different continuous random walk models. An exponent of 1 indicates normal diffusion, while values greater than 1 indicate superdiffusion.

11.2 Results and Discussion

Simulations with continuous random walk models indicate how various distributions of the step length impact the random walk's statistical properties:

1. **Fixed-length model:** All the steps have the same length but random direction. It displays normal diffusive characteristics with mean squared displacement increasing with time. It agrees with the discrete lattice random walk but with slightly different constants owing to the continuous angular distribution.

2. **Gaussian and exponential models:** Both these models, with the lengths of the steps being taken according to a Gaussian or exponential distribution, display normal diffusion. This should be what the Central Limit Theorem predicts, since these distributions have a finite variance.

3. **Levy model:** In this heavy-tailed model, the step lengths are sampled using a power-law distribution. It includes superdiffusive behavior with the mean squared displacement increasing more quickly than linearly with the time. In Figure 22, the walks are characterized by irregularly occurring very large jumps (Levy flights) significantly increasing the spread of the walker.

These findings are characterized by the quantitative exponents: the Gaussian, fixed-length, and exponential models have exponents close to the value of 1, which shows normal diffusion, whereas the Levy model shows an exponent much larger than the value of 1, signifying superdiffusion.

These findings relate back into the theoretical model specified in the coursework, specifically the heavy-tailed random walks. As the text points out, when the distribution over the steps "has heavy tails" (slowly decaying, as opposed to exponentially), the Central Limit Theorem will not hold, resulting in anomalous diffusion.

Continuous models offer a more realistic description of a wide variety of physical phenomena than models based on the lattice structure. For instance, the behavior of particles in fluids, the migration of animals, and the diffusion across the biological membrane are more appropriately explained by continuous models, frequently with a characteristic distribution in the step length based on the underlying physics or biology.

12 Conclusion

This systematic analysis of random walk models uncovered basic dynamical information on stochastic behavior in various model classes. We employed a synergy between theoretical calculations and numerical implementations with the purpose of determining the statistical properties, the laws of scaling, and physical meaning for nine various random walk models.

Key Findings

1. **Diffusion properties**: We verified the mean squared displacement relation $\langle r^2 \rangle = 2dDt$ for dimensions, confirming the universal exponent $\nu = 1/2$ for normal random walks. Simulation data were in close accord with theory with errors less than 5% for two-dimensional walks.

2. **Computation methods**: Our comparative study on the different methods for the absorbing boundary problem indicated considerable advantages for the use of deterministic methods:

Table 4: Performance comparison of random walk solution methods

Method	Execution Time	Precision	Scaling
Monte Carlo	2s ($N = 10^5$)	$O(1/\sqrt{N})$	Linear with simulation count
Enumeration	3×10^{-4}s	<0.01% error	Constant with precision
Analytical	10^{-4}s	Exact	Constant

3. **Fractal structures**: Using diffusion-limited aggregation, consistent fractal properties were found irrespective of the boundary conditions:

Table 5: Fractal characteristics of DLA clusters

Property	Circular Boundary	Square Boundary
Fractal Dimension	1.596 ($R^2 = 0.996$)	1.605 ($R^2 = 0.993$)
Statistical Convergence	∼1000 particles	∼1500 particles
Growth Mechanism	Tip-splitting, screening	Tip-splitting, screening

These values are much lower than percolation clusters (1.896), owing to inherent distinctions between diffusion-limited and random occupancy mechanisms.

4. **Self-avoiding walks**: Our enumerative calculations yielded a growth constant of 2.710 for SAWs on a square lattice, rather than 4.0 for random walks. This reflects the exponential diminution of accessible walks with self-avoiding constraints, with significant application to polymer physics.

5. **Persistence effects**: In persistent random walks, we determined the characteristic persistence length as being $l_p = 1/(1 - p)$, with the crossover from superdiffusive toward normal diffusion occurring at timescales that scale with this length.

6. **Continuous models**: Our comparison of step-length distributions revealed distinctive scaling behaviors:

Table 6: Scaling exponents for continuous random walk models

Model	Scaling Exponent	Diffusion Type
Fixed Length	1.02	Normal
Gaussian	0.99	Normal
Exponential	1.00	Normal
Lévy	4.20	Superdiffusive

It's Superdiffusive behavior in the Lévy model illustrates the way heavy-tailed distributions intrinsically modify random walk statistics by breaking the Central Limit Theorem's assumptions.

Our computational model has fast algorithms for all random walk variants, with stable boundary condition management and purpose-designed analysis streams for extracting statistical quantities. The methods and results here have wide-ranging applications in sciences as disparate as physics and biology as well as computer science.

Future efforts would include extending these studies into higher dimensions, more complicated boundary conditions, and multiscale modeling methodologies.

In summary, this work illustrates the striking ubiquity of random walk phenomena over scales and systems and emphasizes the importance of constraints, memory, and distributions of the steps in shaping emergent behavior. By unifying theoretical stochastic processes with computational approaches, we present both insightful fundamentals as well as useful tools for investigating the intricate systems based on randomness and diffusion.

13 References

References

[1] Course Material, Chapter 5: Random Walks, Computational Physics.

[2] PCMI Notes: Random Walks.

[3] Einstein, A. (1905). On the Motion of Small Particles Suspended in Liquids at Rest Required by the Molecular-Kinetic Theory of Heat. *Annalen der Physik*, 17, 549-560.

[4] Mandelbrot, B. B. (1982). *The Fractal Geometry of Nature*. W. H. Freeman and Company.

[5] Madras, N., & Slade, G. (1993). *The Self-Avoiding Walk*. Birkhäuser.

[6] Witten, T. A., & Sander, L. M. (1981). Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon. *Physical Review Letters*, 47(19), 1400-1403.

[7] Hughes, B. D. (1995). *Random Walks and Random Environments: Volume 1: Random Walks*. Oxford University Press.