Analysis of Random Walk Simulations on Fractals

Course: CS226 - Math Modeling Applications - Fall 2024

Instructor: Stepanyan Varazdat

Students: Yuri Dolukhanyan, Sergey Martirosyan, Maria Nazaryan

Date: December 10, 2024

American University of Armenia

Contents

1 Introduction

Fractals are fascinating geometric structures that appear similar at various levels of magnification. They are not just abstract mathematical concepts but also have practical implications in various scientific fields.

Random walk simulations provide a powerful method to analyze the behavior of particles or entities moving through fractal structures. This report explores the intersection of these two domains, focusing on how random walks behave when constrained by fractal geometries.

The task of this project involves the implementation of fractals using Python and graph theory. Specifically, we will create graph representations of well-known fractals, such as the Sierpinski triangle and the Dragon Curve. Once these fractals are implemented as graphs, we will perform random walk simulations on them to analyze the movement of a particle across the fractal structure. Through this process, we aim to study the random walk behavior on the given fractal. Additionally, we will analyze the resulting data from the random walk simulations to determine from which type of distribution the random walks may originate. This report will describe the process of implementing the fractals, running the random walk simulations, and analyzing the results in terms of their statistical properties.

1.1 What are Fractals?

Fractals are complex geometric shapes that exhibit self-similarity, meaning they look similar at any scale. In mathematics, a fractal is a geometric shape containing detailed structure at arbitrarily small scales, usually having a fractal dimension strictly exceeding the topological dimension. Many fractals appear similar at various scales.

Examples include the Sierpinski Triangle, Dragon Curve, etc. Also, fractals are characterized by their non-integer, or fractional, dimensions, which distinguish them from traditional Euclidean shapes.

1.2 What is a Random Walk?

A random walk is a mathematical model used to describe paths consisting of a succession of random steps. In mathematics, a random walk, sometimes known also as a drunkard's walk, is a stochastic process that describes a path that consists of a succession of random steps on some mathematical space.

It has applications in various fields, such as physics, biology, and economics. When random walks are applied to fractals, they can provide valuable insights into phenomena like diffusion processes, the influence of connectivity on movement, and the scaling behaviors that emerge in complex systems.

2 Content Layout

2.1 Sierpiński Triangle Fractal

The Sierpiński triangle, also called the Sierpiński gasket or Sierpiński sieve, is a fractal with the overall shape of an equilateral triangle, subdivided recursively into smaller equilateral triangles. Originally constructed as a curve, this is one of the basic examples of self-similar sets, i.e., it is a mathematically generated pattern that is reproducible at any magnification or reduction. It is named after the Polish mathematician Wacław Sierpiński, but appeared as a decorative pattern many centuries before the work of Sierpiński.

2.1.1 Dimensionality of the Fractal

Properties

For integer number of dimensions d, when doubling a side of an object, 2^d copies of it are created, i.e., 2 copies for 1-dimensional object, 4 copies for 2-dimensional object, and 8 copies for 3-dimensional object. For the Sierpiński triangle, doubling its side creates 3 copies of itself. Thus the Sierpiński triangle has *Hausdorff dimension*

$$\frac{\log 3}{\log 2} \approx 1.585, \tag{1}$$

which follows from solving $2^d = 3$ for d.

The area of a Sierpiński triangle is zero (in *Lebesgue measure*). The area remaining after each iteration is $\frac{3}{4}$ of the area from the previous iteration, and an infinite number of iterations results in an area approaching zero.

2.1.2 Representation of the Fractal in Python

Constructing the Sierpiński Triangle using Graph Theory in Python

To construct the Sierpiński triangle fractal in Python, we can use graph theory concepts to represent the fractal as a tree structure. The idea involves building the fractal dynamically by extending the tree structure as needed, simulating

a "random walk" that drives the construction process. Below is a detailed explanation:

i. Tree Representation

The Sierpiński triangle is represented as a tree, where each node corresponds to a triangle in the fractal. The tree starts with an initial root node, which is the first level of the tree. Each node in the tree can have exactly three children, corresponding to the three smaller triangles created at each step of the fractal's construction.

ii. Initial Setup

At the start, we have a tree with two levels:

- **Level 0**: Contains only the root node (let's call it node 0).
- **Level 1**: Contains three child nodes of the root (nodes 1, 2, and 3).

Additionally, for illustration purposes, we may initialize further levels. For example:

- **Level 2**: Contains the children of nodes 1, 2, and 3 (nodes 4 through 12).

iii. Dynamic Extension

The tree is not constructed all at once. Instead, it grows dynamically as a random walk traverses through it:

- A *random walk* is a process where a "walker" starts at the root and randomly moves to one of the child nodes at each step (it can also go back starting from the second step it makes).
- When the walker reaches a **leaf node** (a node with no children), the algorithm generates the next level by adding three new child nodes to this leaf. This step extends the tree and creates the next level of the fractal.

iv. Random Walk and Fractal Growth

The random walk ensures that only the parts of the fractal near the walker's path are constructed. This approach avoids building the entire fractal at once, which would be computationally expensive for large iterations. By dynamically adding levels as needed, we efficiently grow the fractal while maintaining its self-similar structure.

v. Resulting Structure

Over time, the tree structure corresponds to the hierarchical arrangement of the Sierpiński triangle, with smaller triangles appearing at each level. This process visually resembles the fractal's recursive nature, where each triangle splits into three smaller ones at each step.
This method provides an intuitive and computationally efficient way to construct the Sierpiński triangle fractal, leveraging graph theory and random walks to dynamically build its self-similar structure.

2.1.3 Analysis of the Results for the Sierpiński Triangle Fractal Simulations

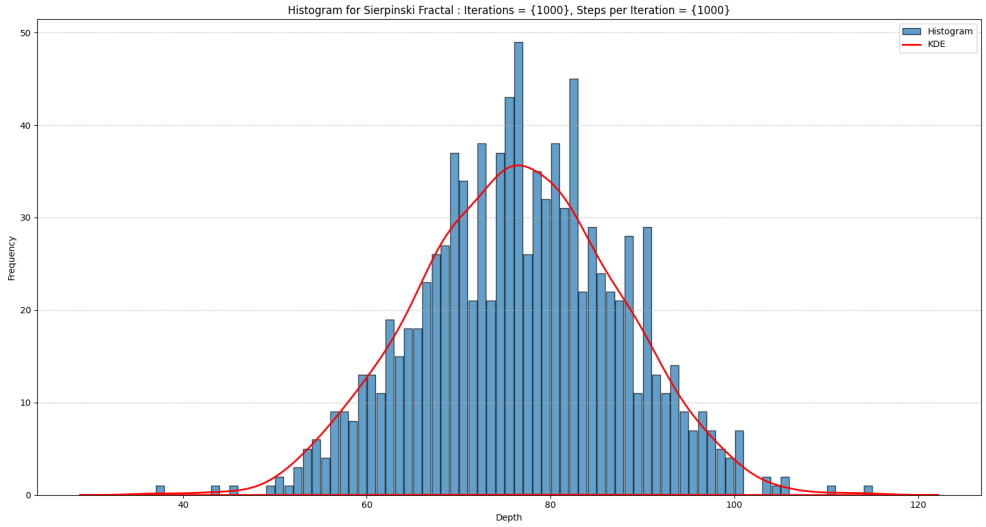
Statistical Observations from Random Walk Simulations on Sierpiński Triangle Fractal

In this project, we performed multiple simulations and analyzed the resulting data. The observations from these simulations provide insights into the distribution and statistical properties of the data as the number of simulations and steps increases. Below is a detailed explanation of the findings:

i. Emergence of a Normal Distribution

After running a large number of simulations and steps, we observed that the resulting data tends to follow a normal distribution. This is supported by the following points:

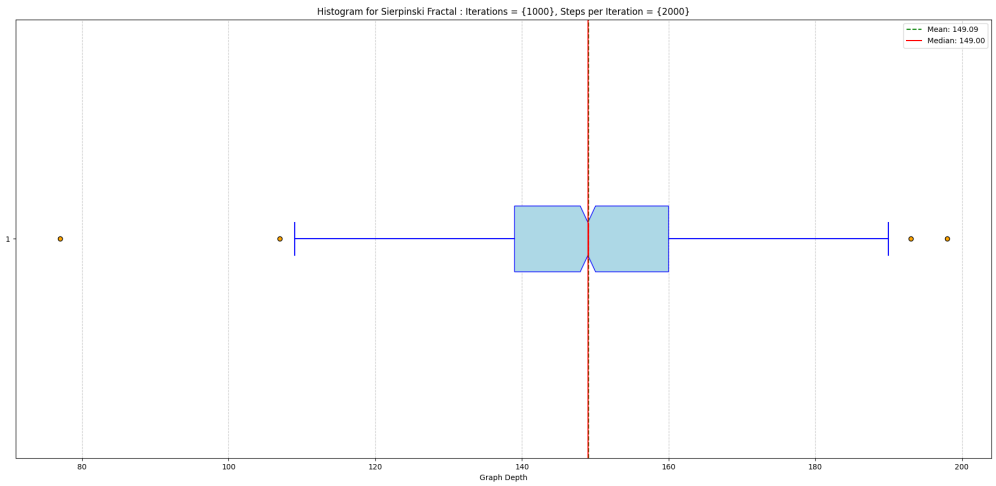
- The data exhibits the characteristic bell-shaped curve of a normal distribution.
- The mean of the distribution is non-zero ($\mu \neq 0$), which indicates a systematic bias or shift in the data's central tendency.
- To confirm the normality of the data, we conducted Shapiro-Wilk tests for different values of N (number of simulations). These statistical tests, implemented in Python, provided evidence supporting the normality of the data across varying sample sizes.



ii. Behavior of the Mean and Median

As the number of simulations (N) increases, another interesting pattern emerges:

- The **mean** and **median** of the data converge, becoming nearly equal for large N. This behavior aligns with theoretical expectations for data from a normal distribution, where symmetry ensures the mean and median are identical in the limit.
- To visualize this phenomenon, we plotted **boxplots** for the data at different simulation levels. These boxplots clearly demonstrate the convergence of the mean and median values as N increases.



iii. Supporting Evidence and Tools for Conclusion

The following steps were undertaken to support these observations:

- Python code was used to generate simulations, perform the Shapiro-Wilk normality tests, and create boxplots.
- These scripts are attached to the project for reference and reproducibility. They can be used to verify the statistical properties of the data and to reproduce the visualizations.
- The boxplots provide a visual summary of the data distribution, highlighting the relationship between the mean and median and offering further confirmation of the normality for large N.

Through a combination of statistical tests and visualizations, we observed that the data from large simulations aligns well with a normal distribution, with the mean and median converging for large N. This behavior is consistent with the Central Limit Theorem, which suggests that data from independent simulations tends to normality as the sample size increases. The accompanying Python codes serve as a detailed implementation of these analyses.

2.2 Dragon Curve Fractal

The Dragon Curve fractal is another intriguing structure that arises from iterative folding patterns. It has a distinct appearance, with self-similar spirals. The Heighway Dragon, which we used in the implementation and simulations, can be constructed from a base line segment by repeatedly replacing each segment by two segments with a right angle and with a rotation of 45° alternatively to the right and to the left.

2.2.1 Dimensionality of the Fractal

Properties

- As a *space-filling curve*, the dragon curve with initial segment length 1, has area of 1/2, as can be seen from its tilings of the plane.
- The boundary of the set covered by the dragon curve has infinite length, with fractal dimension

$$2 \log_2 \lambda \approx 1.523627086202492, \tag{2}$$

where

$$\lambda = \frac{1 + \sqrt[3]{28 - 3\sqrt{87}} + \sqrt[3]{28 + 3\sqrt{87}}}{3} \approx 1.69562076956 \tag{3}$$

is the real solution of the equation

$$\lambda^3 - \lambda^2 - 2 = 0. \tag{4}$$

2.2.2 Representation of the Fractal in Python

Constructing the Dragon Curve using Graph Theory in Python

i. Generating the Dragon Curve

To construct the Dragon Curve fractal, we represent it as a sequence of points on a 2D plane, where each edge of the fractal connects consecutive points. The construction process starts by generating these points and continues iteratively. The key steps are as follows:

- We initialize the curve with two starting points represented as complex numbers: $0 + 0i$ (the origin) and $1 + 0i$ (a point 1 unit to the right of the origin).
- At each iteration, the curve is extended by generating additional points. This is done by rotating and reversing the existing points relative to the last point of the curve.
- The iterative process ensures that the resulting graph reflects the self-similar, recursive structure of the Dragon Curve.

ii. Python Implementation

The following Python function generates the Dragon Curve up to a specified number of iterations:

```
# Dragon Curve up to a specified number of iterations
def generate_dragon_curve(n):
    # Initialize the curve with the first two points
    curve = [0 + 0j, 1 + 0j]
    for _ in range(n):
        # Generate the next segment by rotating and reversing
        curve_extension = [1j * (p - curve[-1]) + curve[-1]
                           for p in reversed(curve[:-1])]
        curve.extend(curve_extension)
    return curve
```

- The parameter `n` determines the number of iterations to perform.
- The resulting fractal graph will have 2^n edges after completing the specified number of iterations.

iii. Dynamic Extension and Dragon Curve Fractal Growth

Once the Dragon Curve is generated, we can simulate a random walk along the fractal:

- The random walk starts from the initial point $0 + 0i$.
- If the number of steps in the random walk is large enough, the walker may reach the endpoint of the fractal, i.e., the last edge.
- When this happens, the curve is extended dynamically:
  - The current curve is doubled and connected to the existing fractal structure.
  - This process increases the fractal's size, generating a new graph with $2^{(n+1)}$ edges. On the other hand, all this part is not that efficient as we would like it to be, as for large number of n it would not be really smart to allocate all this data for the fractal...

iv. Resulting Structure

Through this iterative and dynamic process, the Dragon Curve fractal evolves, displaying its characteristic self-similarity and infinite complexity. The Python code for the full algorithm, including random walk and dynamic extension, is attached to the project for reference.
This method provides a computationally clear way to construct and extend the Dragon Curve fractal. By combining graph theory, iterative algorithms, and random walks, we can dynamically generate and explore the fractal's infinite structure.

2.2.3 Analysis of the Results for the Dragon Curve Fractal Simulations

Random walks on the Dragon Curve highlight interesting trapping phenomena, where the walker often gets confined to certain regions before escaping. This feature is characteristic of fractals with loops and twists.

Statistical Observations from Random Walk Simulations on Dragon Curve Fractal

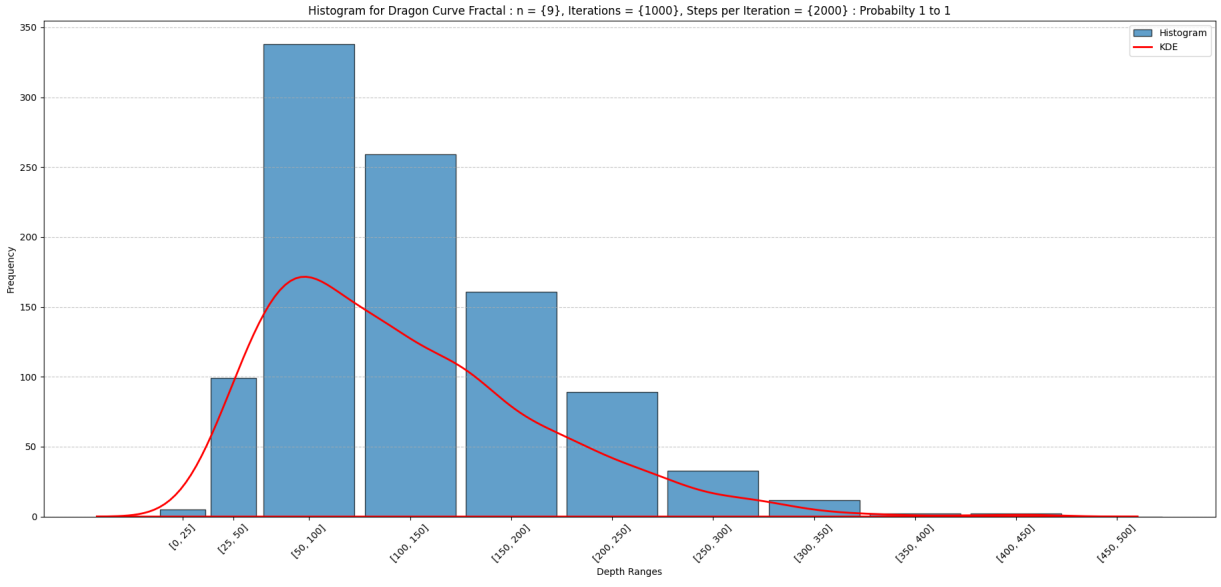
Through a series of simulations and analyses, we examined the behavior of random walks on the Dragon Curve fractal and explored the statistical properties

of the resulting data. This section describes the findings, including connections to the Tracy-Widom distribution, the phenomenon of depth trapping, and the effects of altering movement probabilities.

i. Emergence of the Tracy-Widom Distribution

After running a large number of simulations and steps, we observed that the resulting data tends to look like the Tracy-Widom distribution CDF. This conclusion is based on the following observations:

- The histogram data closely matches the theoretical distribution of Tracy-Widom, as confirmed by conducting large number of simulations for different values of steps.
- Python scripts were used to perform these Kolmogorov-Smirnov (KS) tests and to try to verify the fit between the empirical data and the theoretical distribution. But unfortunately the results consistently indicated that the $(p - value)$ of the given data was tending to 0, so we are not able to tell whether it is really from Tracy-Widom distribution or not.



ii. Depth Trapping Phenomenon

As the number of simulations increases, we observed a distinct pattern in the random walk's behavior:

- The mean depth of the random walk gradually approaches a certain limit, which we interpret as being relatively shallow compared to the fractal's total potential depth.

- This limitation arises due to the **trapping phenomenon**, where the walker frequently becomes confined to specific regions of the fractal before eventually escaping.
- These trapping events reduce the likelihood of the walker reaching significantly greater depths, particularly for long walks and large simulations.

iii. Effect of Modifying Movement Probabilities

To further explore the behavior of the random walk, we experimented with altering the probability of moving forward:

- By default, the probability of moving forward was set to $1:1$, corresponding to an unbiased random walk.
- We adjusted this probability to other ratios, introducing a bias in the walk's direction. This modification had a notable impact:
  - The walker was more likely to reach greater depths in the fractal, as observed in the resulting data.
  - The walk avoided being trapped in repeating cycle loops of the fractal, allowing it to explore new regions more effectively.
- Histograms of the resulting depths clearly show that biased walks lead to deeper explorations compared to the default setting, highlighting the fractal's sensitivity to changes in walk dynamics.

iv. Supporting Evidence and Tools

The following steps and tools were used to validate these findings:

- Python scripts were developed to perform the simulations, conduct Kolmogorov-Smirnov tests, and generate histograms of the walk's depth.
- These scripts are attached to the project for reproducibility and further analysis.
- Visualization tools such as histograms and distribution plots were used to illustrate the statistical properties of the data and the effects of altering movement probabilities.

3 Overall Conclusion

3.1 Summary of the Analysis and Research

In this report, we explored random walk simulations on fractal geometries, specifically the Sierpinski triangle and the Dragon Curve. Fractals present unique challenges and insights for modeling random walks due to their non-Euclidean structures and self-similar properties. The simulations reveal complex and fascinating dynamics of random walks on both fractals.

- For the Sierpiński Triangle Fractal, through a combination of statistical tests and visualizations, we observed that the data from large simulations aligns well with a normal distribution, with the mean and median converging for large $N - (CLT)$.
- For the Dragon Curve Fractal: considering the effects of trapping and movement probability modifications, highlights the interplay between geometry and randomness in determining the walk's behavior. These findings underscore the fractal's intricate structure and its influence on statistical phenomena.

4 Reference Page

References

[1] Wikipedia contributors, "Sierpiński triangle," *Wikipedia, The Free Encyclopedia*, available at: https://en.wikipedia.org/wiki/Sierpi%C5%84ski_triangle.

[2] Wikipedia contributors, "Dragon curve," *Wikipedia, The Free Encyclopedia*, available at: https://en.wikipedia.org/wiki/Dragon_curve.

[3] Wikipedia contributors, "Normal distribution," *Wikipedia, The Free Encyclopedia*, available at: https://en.wikipedia.org/wiki/Normal_distribution.

[4] Wikipedia contributors, "Tracy–Widom distribution," *Wikipedia, The Free Encyclopedia*, available at: https://en.wikipedia.org/wiki/Tracy%E2%80%93Widom_distribution.

[5] Wikipedia contributors, "Shapiro–Wilk test," *Wikipedia, The Free Encyclopedia*, available at: https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test.

[6] Wikipedia contributors, "Kolmogorov–Smirnov test," *Wikipedia, The Free Encyclopedia*, available at: https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test.

[7] Daniel Shiffman, "Random Walks," *Nature of Code*, available at: https://natureofcode.com/random/#random-walks.

[8] Mohammed Elkomy, "Stochastic Fractal Search Python," GitHub repository, available at: https://github.com/mohammed-elkomy/stochastic-fractal-search-python.

[9] OpenAI, "ChatGPT," *ChatGPT*, available at: https://chatgpt.com/.