AMS 595 - Assignment1
Monte Carlo Estimation of π

Amol Arora, SBUID: 116491705

September 15, 2024

Contents

1 Github Link

https://github.com/amol1202/AMS595-Assignment1

2 Introduction

Monte Carlo methods are widely used for numerical integration and estimation tasks. In this report, I work on demonstrating the Monte Carlo method to estimate the value of π. This report includes conducting experiments using a fixed number of points, achieve a specified precision, and visualize the results with an iterative process that generates a simulation. The relationship between computational cost and precision is also analyzed.

3 Monte Carlo Estimation with Fixed Number of Points

3.1 Methodology

Estimating the value of π can be done by randomly generating points inside a unit square and counting how many fall within a quarter circle. The ratio of points inside the circle to the total number of points gives an estimate for π:

$$\pi \approx 4 \times \frac{\text{Number of points inside quarter circle}}{\text{Total number of points}}.$$

I have used increasing values for the number of random points: $10^2, 10^3, 10^4, 10^5, 10^6, 10^7$.

3.2 Results

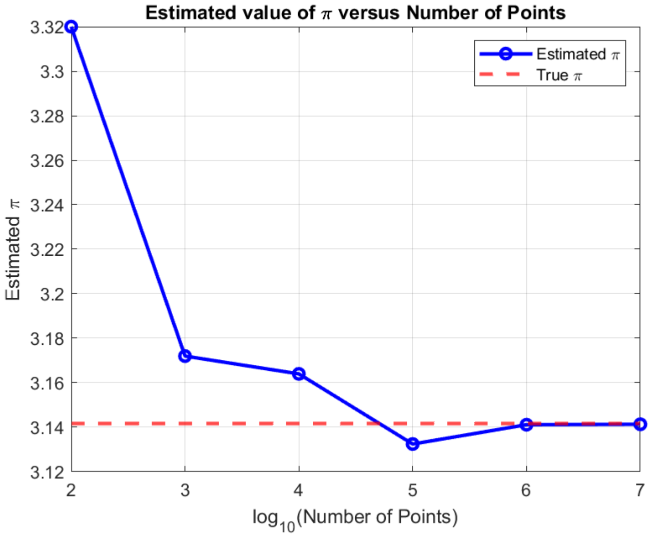

Figure 1: Estimated value of π versus number of points (log scale).

As shown in Figure 1, the estimated value of π approaches the true value as the number of points increases. The horizontal red dashed line represents the true value of π.

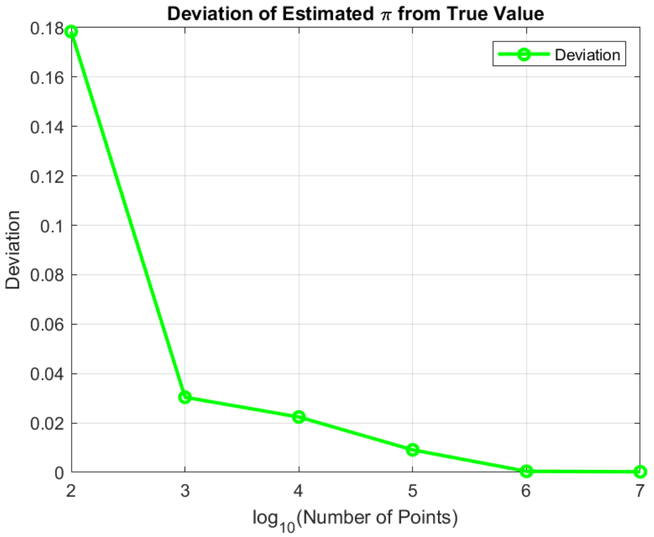

Figure 2: Deviation of estimated π from the true value.

Figure 2 shows how the deviation from the true value of π decreases as more points are sampled, indicating that the method converges.

3.3 Execution Time

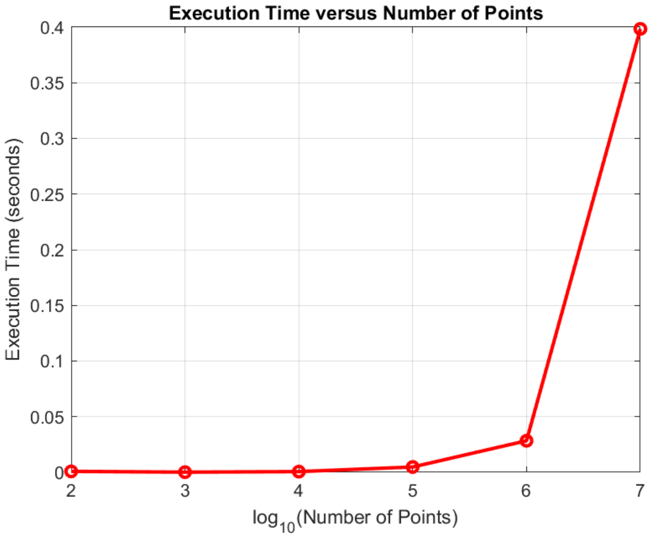

Figure 3: Execution time versus number of points (log scale).

Also in Figure 3, we can see that the execution time increases with the number of points. Larger datasets require more computational effort, which is reflected in the growing time complexity.

3.4 Precision vs Computational Cost

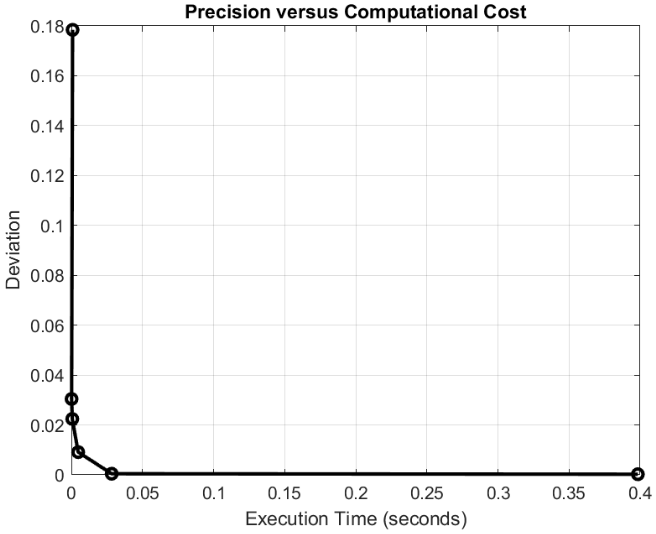

Figure 4: Precision versus computational cost (execution time).

Figure 4 shows the trade-off between computational cost (execution time) and precision. As the number of points increases, precision improves but at a higher computational cost.

4 Monte Carlo Estimation to a Specified Precision

4.1 Methodology

In the second part of the experiment, the aim is to achieve a target precision (e.g., 0.001). The algorithm iteratively increases the number of random points until the deviation between successive estimates of π falls below the specified precision threshold. The loop terminates when the target precision is met or when a maximum number of iterations is reached.

4.2 Results

The final estimate of π for a precision target of 0.001 was obtained after several iterations. A total of N points were used, and the number of iterations was X.

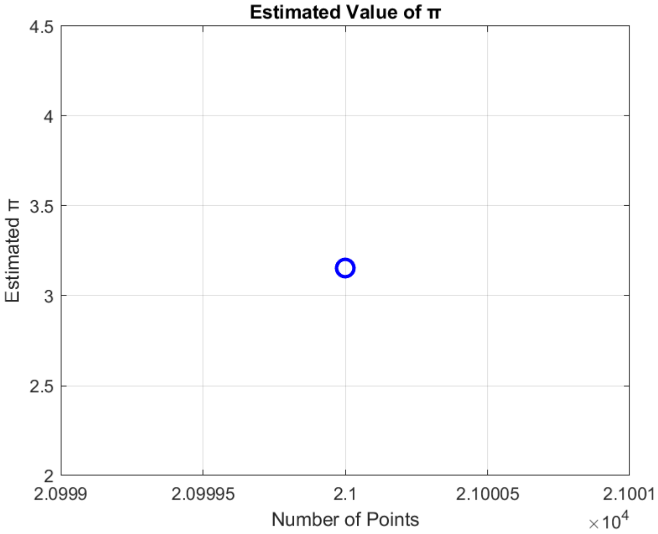

Figure 5: Estimated value of π to a specified precision.

5 Monte Carlo Estimation with Simulation

5.1 Methodology

In the third part, I created a function that estimates π to a specified precision while generating a GIF that visualizes the points inside and outside the quarter circle. This process allows for real-time visualization of the convergence of the estimate.

5.2 Results

The output is a GIF file showing the iterative process of random point generation and the final estimated value of π.

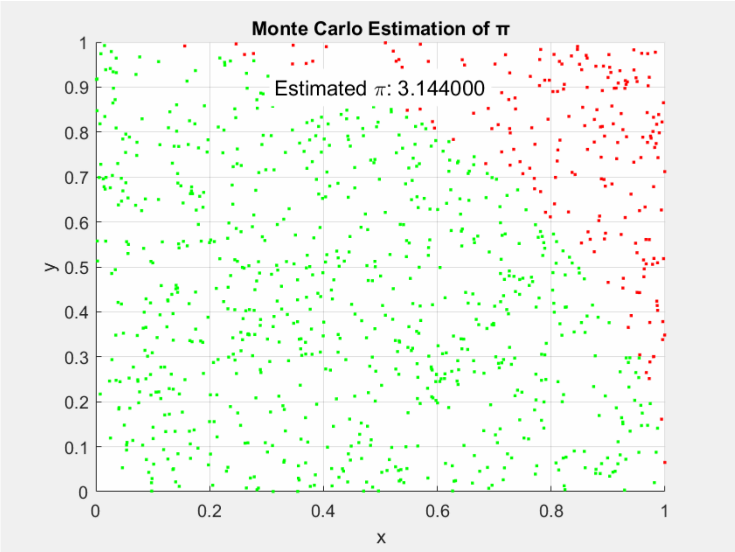

Figure 6: Visualization of Monte Carlo estimation of π (Simulation on Github).

6 Conclusion

The Monte Carlo method provides an intuitive and relatively simple approach to estimate π. However, achieving high precision requires a significant number of points, and thus greater computational cost. The trade-off between precision and execution time should be considered when selecting the number of points. The visualization through a simulation allows for real-time observation of the convergence of the estimate.

References

[1] OpenAI, ChatGPT Version 4 - `https://www.openai.com/chatgpt`. **This was used only for these purposes**:

1. Assisted in converting the image to GIF format for the Section-3 of the assignment.
2. Provided the idea to use a log scale for data points instead of a linear scale.
3. Guidance on formatting and best coding practices.

[2] `https://en.wikipedia.org/wiki/Monte_Carlo_method`

[3] `https://www.mathworks.com/help/matlab/graphics-images.html`