ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

ADVANCED PROBLEM SOLVING

ESTIMATION WITH MONTE CARLO METHODS

By: Tigist Wondimneh - UGR/5506/17

Monte Carlo Methods Report

1. Introduction

Monte Carlo methods are a class of numerical techniques that leverage random sampling to approximate solutions to mathematical problems. They are particularly useful for problems involving high-dimensional integrals or cases where analytical solutions are challenging to compute.

In this report, we explore the application of Monte Carlo methods to:

1. Estimating the value of π\pi.
2. Performing numerical integration in 1D and 2D.
3. Analyzing the error and convergence behavior as the number of random samples (N) increases.

Each task is supported by visualizations of convergence and error behavior, along with theoretical analysis of the results.

2. Tasks

Task 1: Estimation of π

Objective: To estimate the value of π by randomly sampling points in a square enclosing a unit circle and calculating the fraction of points that fall inside the circle.

Methodology:

1. Generate N random points (x,y) in the range [0,1]×[0,1]
2. Determine if each point lies inside the unit circle using: x2+y2 ≤1
3. Estimate π using the formula:

$$\pi \approx 4 \cdot \frac{\text{Number of Points Inside Circle}}{\text{Total Number of Points}}$$

Results:

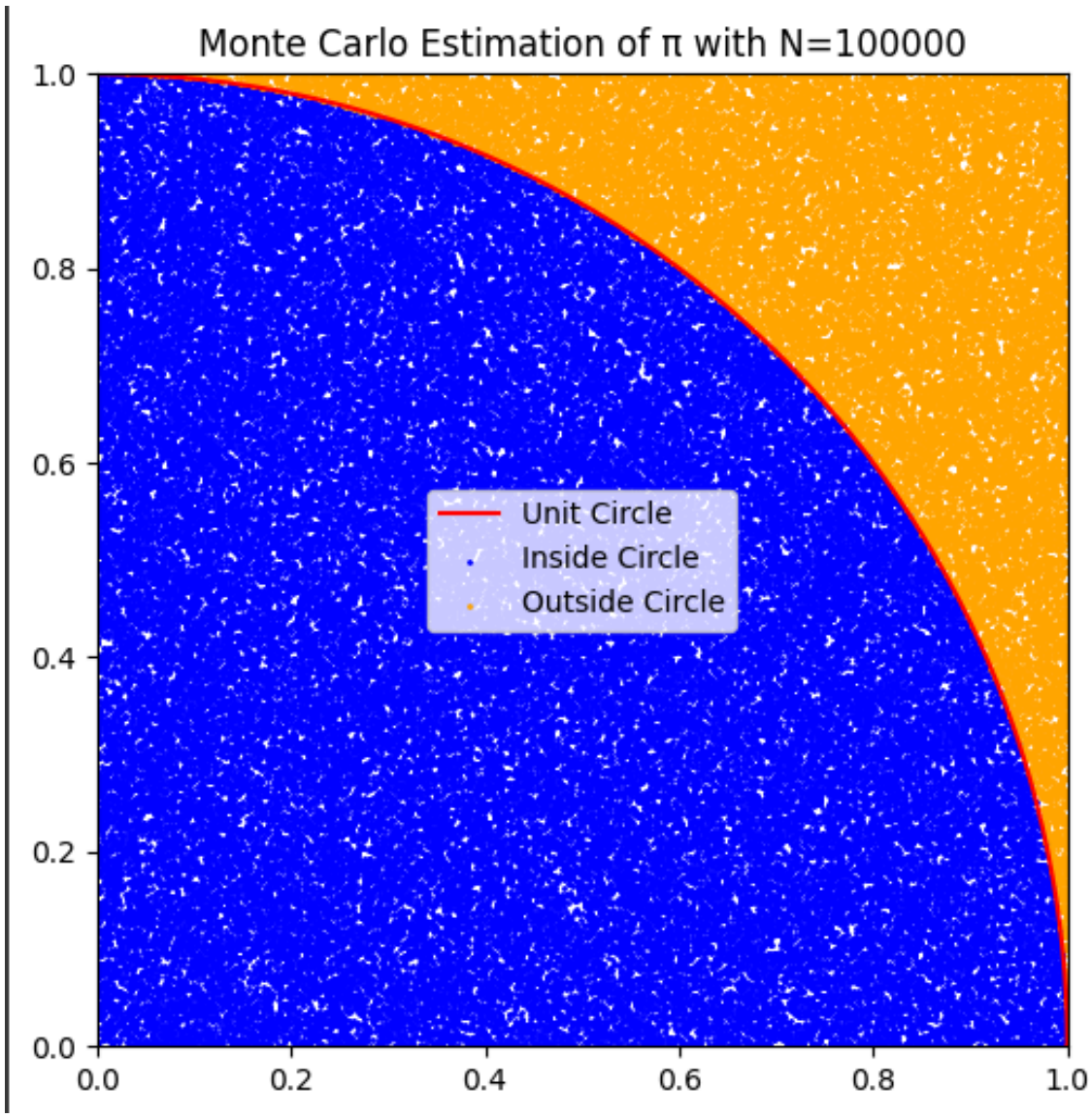
For N = 100000

Estimated π: 3.13608

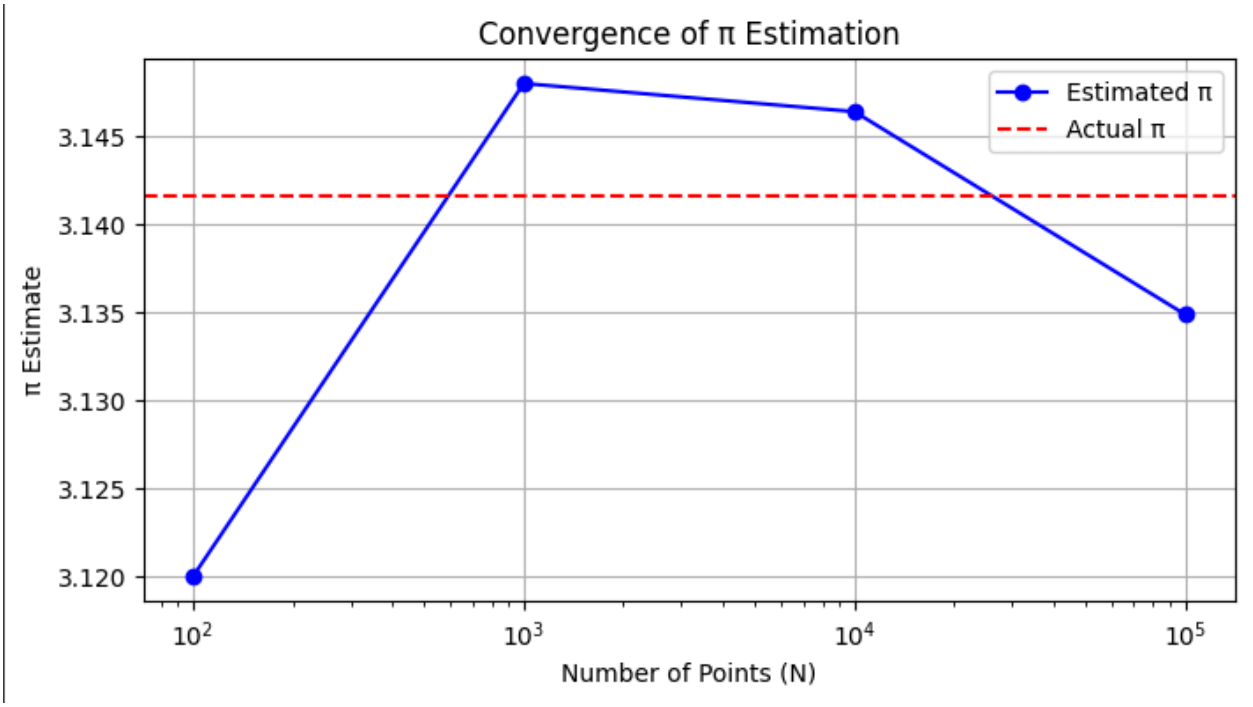
Actual π: 3.141592653589793

Visualizations:

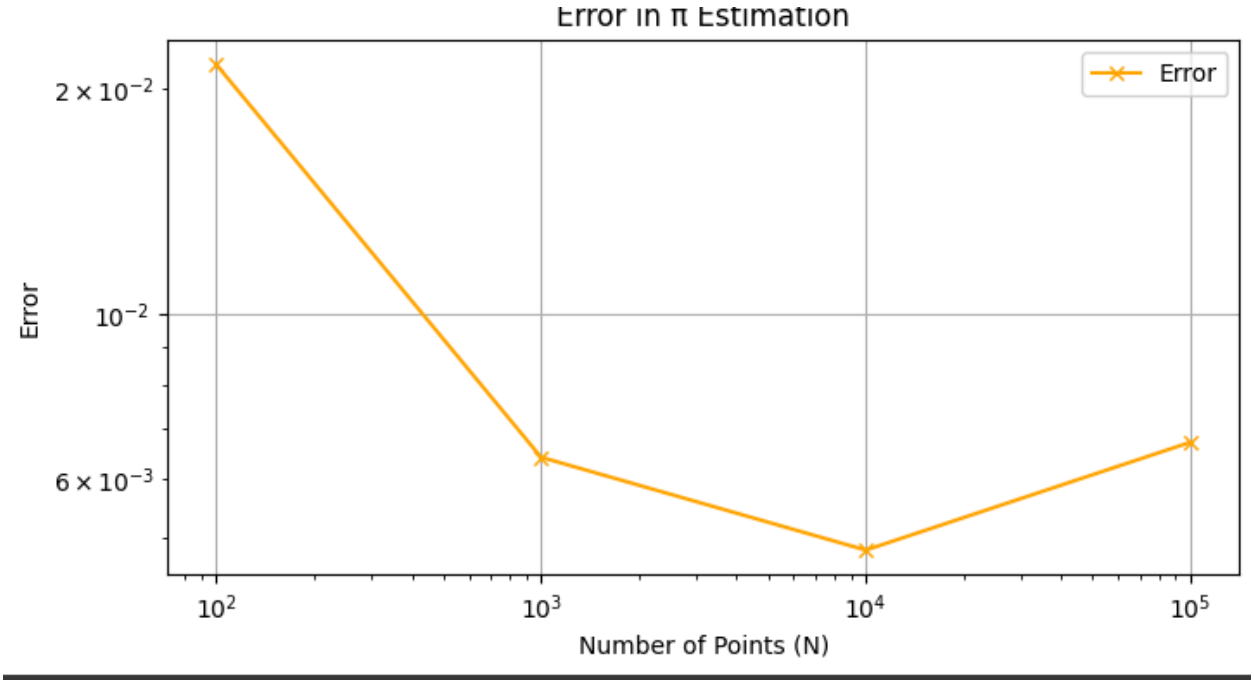
1. Points Inside and Outside the Circle:



2. Convergence of π:



3. Error Analysis:



Task 2: 1D Numerical Integration

Objective: To approximate the definite integral of a function f(x)f(x) over an interval [a,b][a, b] using Monte Carlo integration.

Methodology:

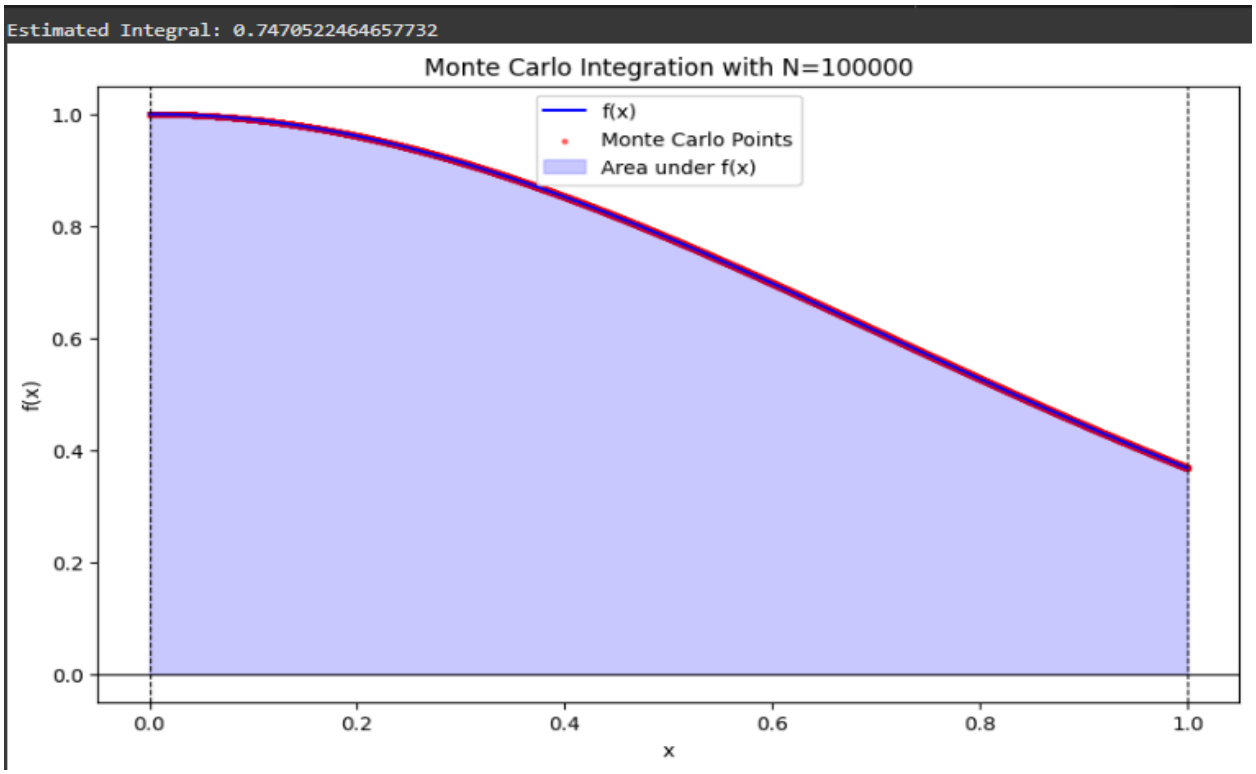
1. Randomly sample N points x uniformly from the interval [a,b].
2. Evaluate the function at each sampled point: f(xi).
3. Estimate the integral using:

$$\int_a^b f(x)\,dx \approx (b-a)\cdot\frac{1}{N}\sum_{i=1}^{N} f(x_i),$$

Example:

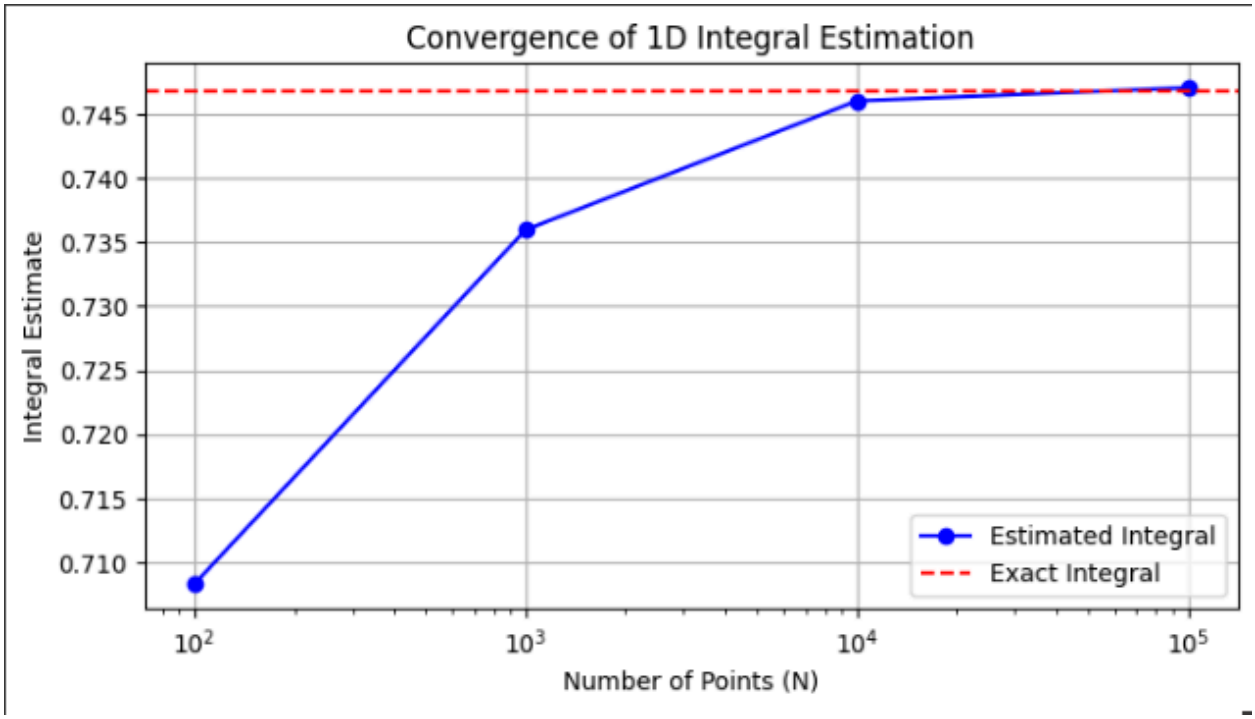
- Function: f(x) = e^{-x^2}
- Integration range: [0,1][0, 1]

Results:

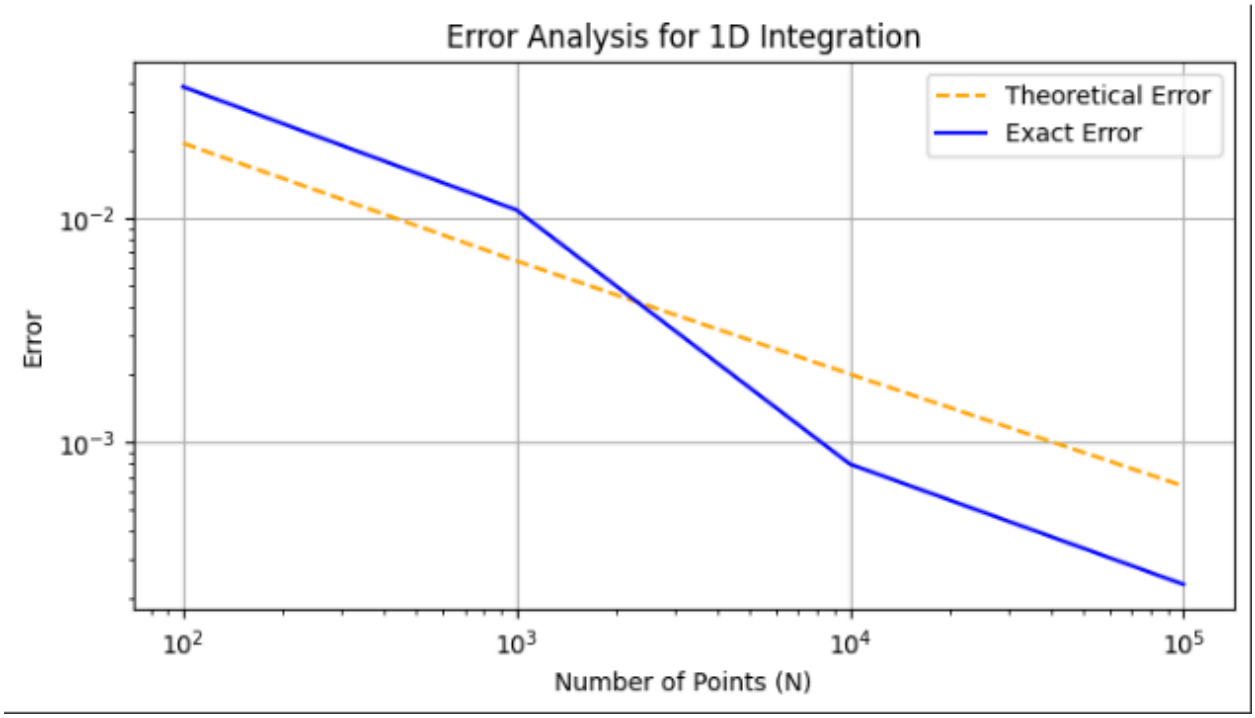


Visualizations:

1. Convergence Plot:



2. Error Analysis:



Task 3: 2D Numerical Integration

Objective: To extend Monte Carlo integration to approximate the integral of a 2D function f(x,y) over a rectangular domain.

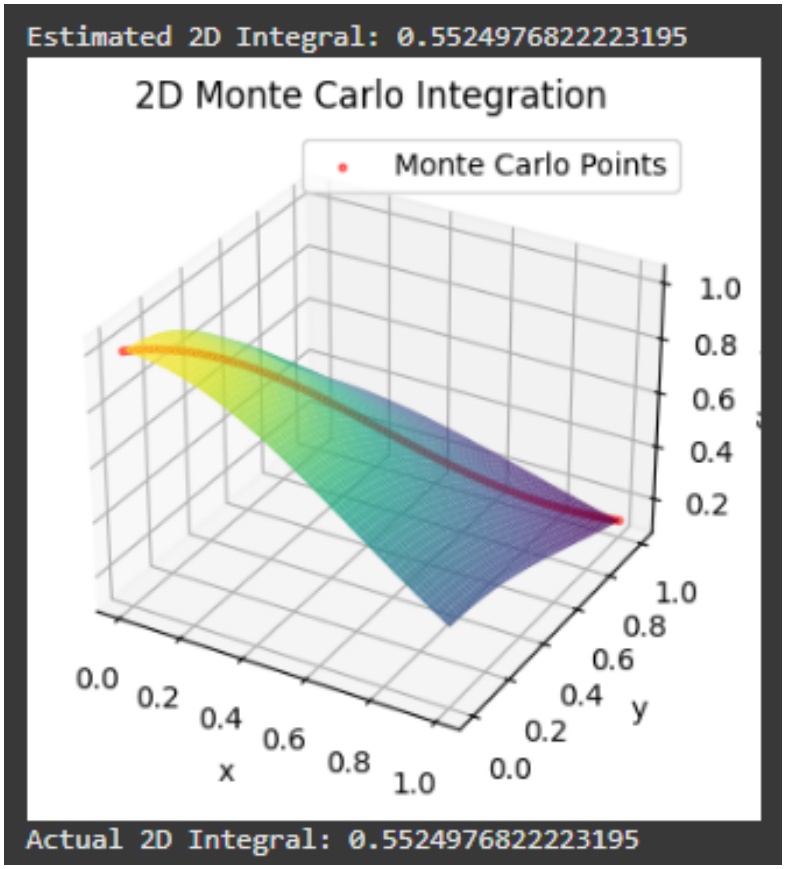
Methodology:

1. Randomly sample N points (x,y) uniformly from the domain [a,b]×[a,b].
2. Evaluate the function at each point: f(xi,yi).
3. Estimate the integral

Example:

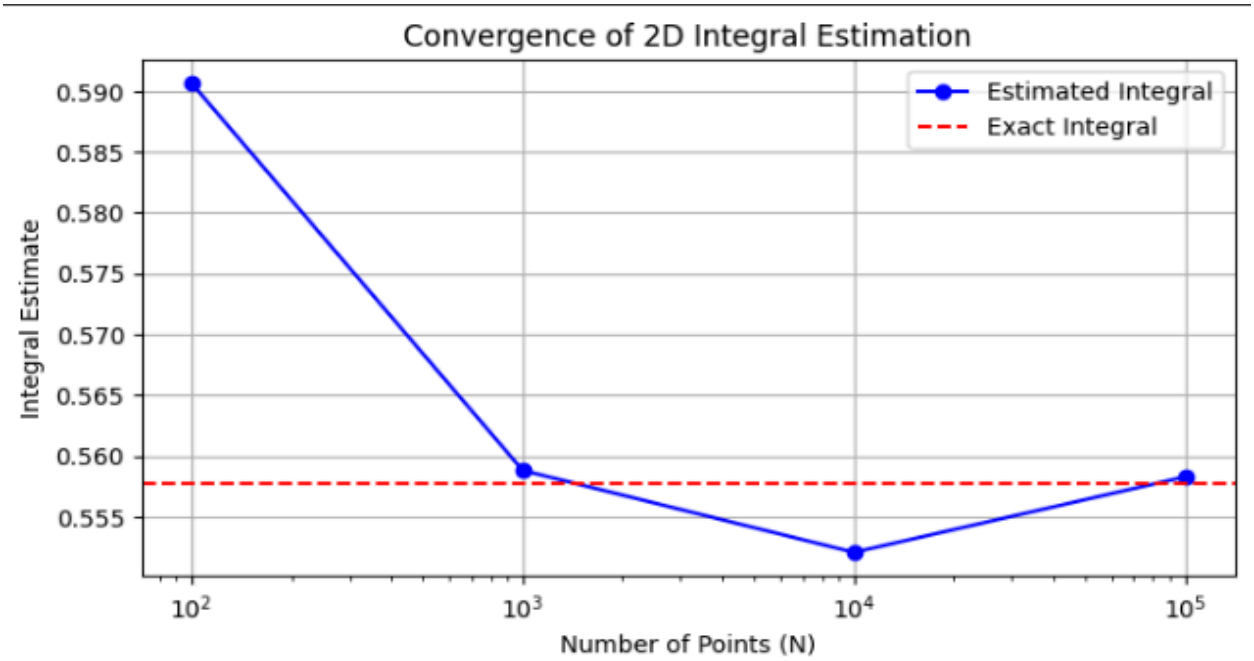
- Function: f(x, y) = e^{-(x^2 + y^2)}
- Domain: [0,1]×[0,1]

Results:

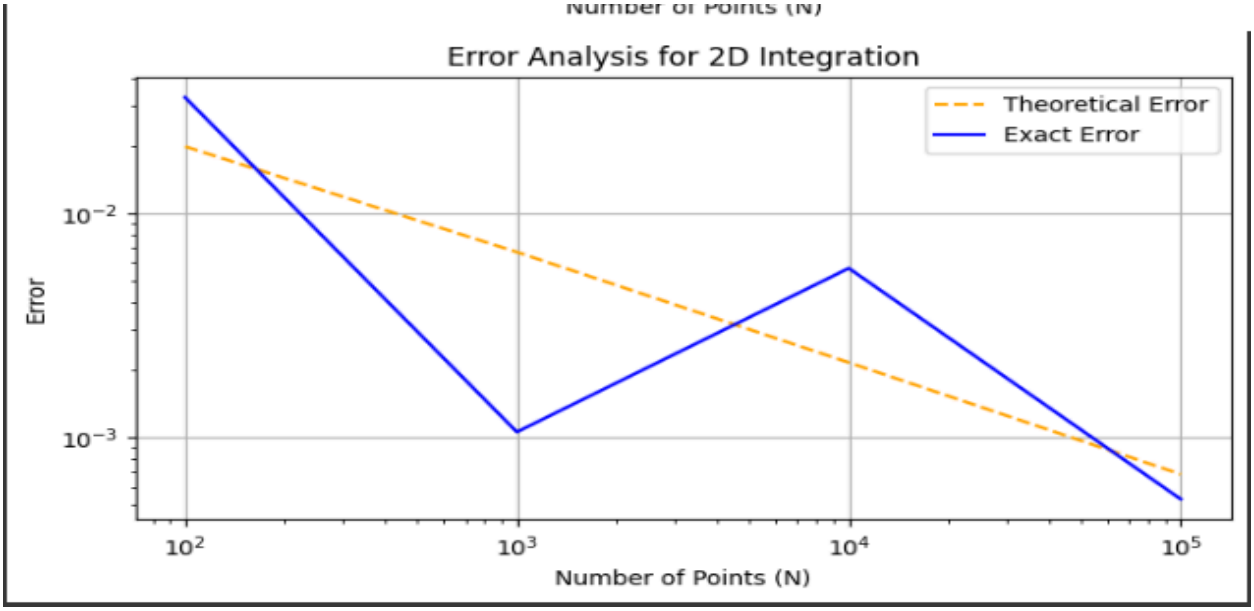


Visualizations:

1. Convergence Plot:



2. Error Analysis:

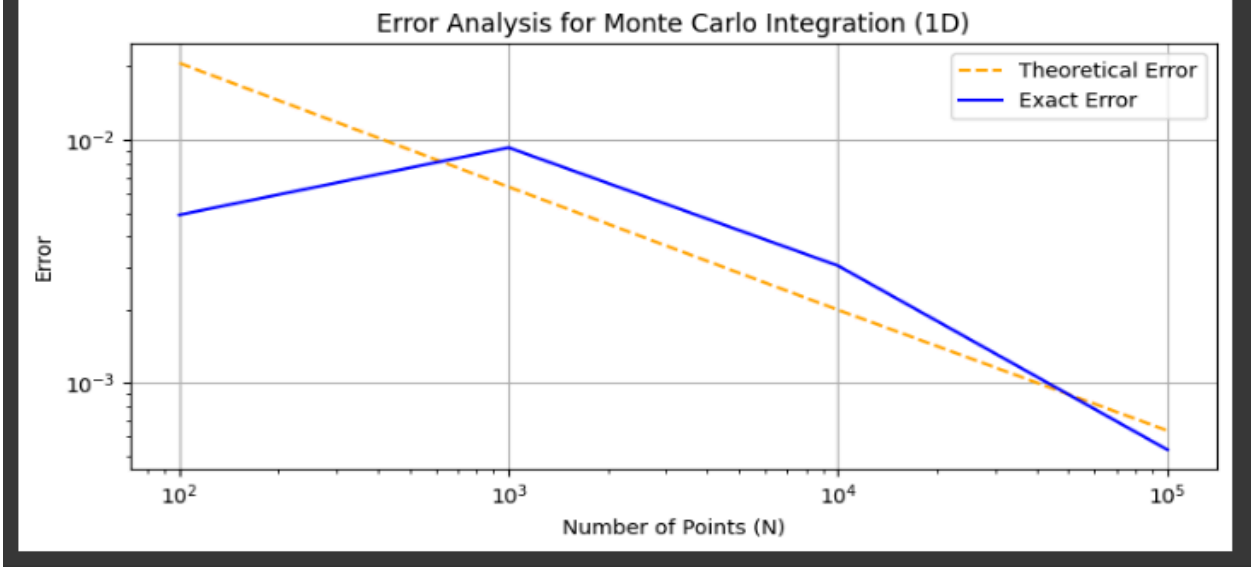


Task 4: Error Analysis

Objective: To analyze how errors in Monte Carlo methods decrease as the number of random points (N) increases.

Results:

```
N = 100: Estimated = 0.751733, Theoretical Error = 0.020668, Exact Error = 0.004908
N = 1000: Estimated = 0.737544, Theoretical Error = 0.006395, Exact Error = 0.009280
N = 10000: Estimated = 0.749856, Theoretical Error = 0.001995, Exact Error = 0.003032
N = 100000: Estimated = 0.746294, Theoretical Error = 0.000636, Exact Error = 0.000530
```



3. Conclusion

Monte Carlo methods provide a powerful framework for estimating mathematical quantities, especially in higher dimensions. The results validate the key properties of Monte Carlo methods:

1. Errors decrease as 1 / sqrt{N}, as predicted by the Central Limit Theorem.
2. Convergence behavior is influenced by the number of samples (N) and the quality of random number generators.
3. Monte Carlo methods are particularly effective for problems in higher dimensions where traditional numerical methods struggle.

4. Discussion

4.1 How the Quality of Random Number Generators Affects Convergence and Accuracy

1. Uniform Distribution:
    - Monte Carlo methods assume uniform sampling across the domain.
    - Poor-quality RNGs may produce uneven or clustered points, causing bias and slower convergence.
2. Independence:
    - Samples must be independent to ensure accurate estimates.
    - Correlated samples lead to systematic errors and increase variance.

3. Dimensionality:
    - In higher dimensions, the quality of RNGs becomes even more critical.
    - Poor RNGs may leave gaps or clusters in the sampling space, affecting accuracy.
4. Examples:
    - High-quality RNGs like PCG (default in NumPy) provide reliable results for most Monte Carlo tasks.
    - Quasi-random generators (e.g., Sobol, Halton) offer better uniformity and faster convergence in low dimensions.

4.2 Trade-Offs Between Computational Cost and Accuracy

1. Accuracy vs. N:
    - Increasing N improves accuracy but with diminishing returns, as the error decreases proportionally to 1/sqrt{N}.
2. Computational Cost vs. N:
    - Computational cost increases with N, as each additional sample requires a function evaluation.
3. Trade-Off:
    - Larger N improves accuracy but at a higher computational cost.
    - The optimal N depends on the problem's required precision and available computational resources.

Resources:

Github:
https://github.com/TigistW/Random-Number-Generators-through-Monte-Carlo-Methods.git