HOMEWORK1: Exercises for Monte Carlo Methods

Student ID: 21307210   Student Name: 傅祉珏

Lectured by 梁上松, Sun Yat-sen University

1 Exercise 1

1.1 Question

The Monte Carlo method can be used to generate an approximate value of π. The figure below shows a unit square with a quarter of a circle inscribed. The area of the square is 1 and the area of the quarter circle is $\pi/4$. Write a script to generate random points that are distributed uniformly in the unit square. The ratio between the number of points that fall inside the circle (red points) and the total number of points thrown (red and green points) gives an approximation to the value of $\pi/4$. This process is a Monte Carlo simulation approximating π. Let N be the total number of points thrown. When $N = 50, 100, 200, 300, 500, 1000, 5000$, what are the estimated π values, respectively? For each N, repeat the throwing process 100 times, and report the mean and variance. Record the means and the corresponding variances in a table.

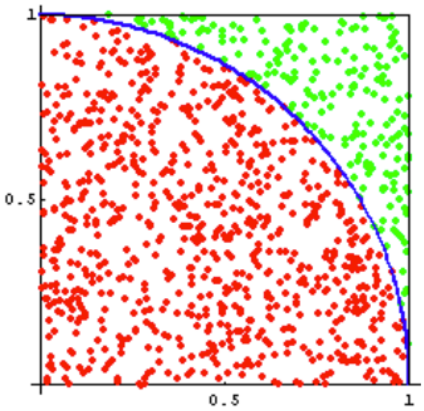

图 1: Using the Monte Carlo method to estimate π

1.2 Answer

To use the Monte Carlo method to estimate the value of π, we first need to understand the basic geometric relationship. Inside a unit square, there is an inscribed quarter circle with a radius of 1. The area of the unit square is 1 square unit, while the area of the quarter circle is $\pi/4$ square units. By randomly generating points within the unit square and calculating the proportion of those points that fall inside the quarter circle, we can estimate the value of π.

Specifically, we randomly generate N points that are uniformly distributed within the unit square. For each point, we check whether its coordinates satisfy the equation of the circle $x^2+y^2 \leq 1$. If it does, the point is inside the quarter circle; otherwise, it is outside. By calculating the ratio of the number of points that fall inside the circle to the total number of points N, we can obtain an approximate value for $\pi/4$. Since this ratio multiplied by 4 gives an approximation of π, we can use this ratio to estimate π.

Following this concept, the code has been written, and the resulting code is shown in `Exercises1.py`. Based on the requirements of the code, the following results can be obtained.

表 1: Exercises 1 Answer Table

Index	N	Means	Variances
0	50	3.122400	0.054458
1	100	3.128800	0.025107
2	200	3.147600	0.015110
3	300	3.138133	0.010537
4	500	3.144560	0.005675
5	1000	3.141560	0.002618
6	5000	3.144552	0.000533

2 Exercise 2

2.1 Question

We are now trying to integrate the another function by Monte Carlo method:

$$\int_0^1 x^3 dx$$

A simple analytic solution exists here: $\int_0^1 x^3 dx = 1/4$. If you compute this integration using Monte Carlo method, what distribution do you use to sample x? How good do you get when $N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100$, respectively? For each N, repeat the Monte Carlo process 100 times, and report the mean and variance of the integrate in a table.

2.2 Answer

To integrate the function $f(x) = x^3$ over the interval $[0, 1]$ using the Monte Carlo method, we can follow these steps:

1. Sampling Distribution

   Since we are integrating over the interval $[0, 1]$, we will sample x uniformly from this interval. This means we can use a uniform distribution $U(0, 1)$.

2. Monte Carlo Integration

   The Monte Carlo estimate for the integral is given by:

   $$I \approx \frac{b-a}{N} \sum_{i=1}^{N} f(x_i)$$

   where $[a, b] = [0, 1]$ and x_i are uniformly sampled points in the interval.

3. Performing the Simulation

   We will repeat the Monte Carlo process for $N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100$ and perform 100 trials for each N.

4. Calculating Mean and Variance

   For each N, we will calculate the mean and variance of the estimates from the 100 trials.

Following this concept, the code has been written, and the resulting code is shown in `Exercises2.py`. Based on the requirements of the code, the following results can be obtained.

表 2: Exercises 2 Answer Table

Index	N	Means	Variances
0	5	0.245029	0.015738
1	10	0.259467	0.008685
2	20	0.257500	0.004301
3	30	0.248446	0.002323
4	40	0.246228	0.002289
5	50	0.252243	0.001301
6	60	0.250549	0.001021
7	70	0.246137	0.001154
8	80	0.248234	0.000905
9	100	0.247640	0.000868

3 Exercise 3

3.1 Question

We are now trying to integrate a more difficult function by Monte Carlo method that may not be analytically computed:

$$\int_{x=2}^{4} \int_{y=-1}^{1} f(x, y) = \frac{y^2 \times e^{-y^2} + x^4 \times e^{-x^2}}{x \times e^{-x^2}}$$

Can you compute the above integration analytically? If you compute this integration using Monte Carlo method, what distribution do you use to sample (x, y)? How good do you get when the sample sizes are $N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100, 200$ respectively?

For each N, repeat the Monte Carlo process 100 times, and report the mean and variance of the integrate.

3.2 Answer

First, we need to determine if the function can be integrated analytically over the specified limits. Given the complexity of $f(x, y)$, it's challenging to perform this integration directly. The integrand involves exponential terms and polynomial terms, making it unlikely that a simple closed form exists.

The Monte Carlo method involves randomly sampling points from the region of integration and estimating the integral based on the average value of the function at those points. In this case, Since x is constrained to the interval $[2, 4]$ and y to $[-1, 1]$, we can use a uniform distribution to sample x and y: For x, sample from $U(2, 4)$; for y, sample from $U(-1, 1)$.

For each N, repeat the sampling process 100 times, generating N samples of (x, y) each time, and calculate the average value of $f(x, y)$ over these samples. Then estimate the integral by multiplying the average value by the area of the region, which is 4. Finally, compute the mean and variance of the estimated integral values from the 100 repetitions.

Following this concept, the code has been written, and the resulting code is shown in `Exercises3.py`. Based on the requirements of the code, the following results can be obtained.

表 3: Exercises 3 Answer Table

Index	N	Means	Variances
0	5	95200.743354	1.989497e+10
1	10	112027.570910	1.024562e+10
2	20	103470.545021	4.807861e+09
3	30	114248.092622	4.681874e+09
4	40	113754.545521	3.144835e+09
5	50	106303.688565	1.668617e+09
6	60	113812.281106	2.470408e+09
7	70	114186.319649	1.696168e+09
8	80	113507.130148	1.631517e+09
9	100	115568.852197	1.197880e+09
10	200	113505.642425	7.224751e+08

4 Exercise 4

4.1 Question

An ant is trying to get from point A to point B in a grid. The coordinates of point A is $(1, 1)$ (this is top left corner), and the coordinates of point B is (n, n) (this is bottom right corner, n is the size of the grid).

Once the ant starts moving, there are four options, it can go left, right, up or down (no diagonal movement allowed). If any of these four options satisfy the following:

(a) The new point should still be within the boundaries of the $n \times n$ grid

(b) Only the center point $(4, 4)$ is allowed to be visited zero, one or two times, while the remainder points should not be visited previously (are allowed to be visited zero or one time).

If P is the probability of the ant reaching point B for a 7×7 grid, use Monte Carlo simulation to compute P. Pick the answer closest to P in value (assume $20,000$ simulations are sufficient enough to compute P).

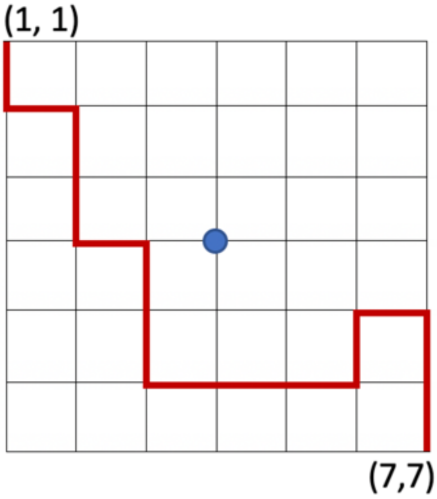

图 2: An ant is trying to get from point $A(1, 1)$ to point $B(7, 7)$ in a grid

4.2 Answer

To estimate the probability P of the ant reaching point B in a 7×7 grid using Monte Carlo simulation, we first set the grid range from $(1, 1)$ to $(7, 7)$, where the center point $(4, 4)$ can be visited up to two times, while other points can only be visited once. The ant can move in four directions: left, right, up, or down, while ensuring that movements remain within the grid boundaries. The simulation continues until the ant reaches $(7, 7)$ or can no longer move without violating the visiting rules. We will conduct a total of $20,000$ simulations and count the number of successful trials, i.e., the times the ant reaches $(7, 7)$. Finally, we calculate P as the ratio of successful trials to the total number of trials.

Following this concept, the code has been written, and the resulting code is shown in `Exercises4.py`. Based on the requirements of the code, the probability of the ant reaching point B for a 7×7 grid is $25.5\% \pm 0.5\%$.

5 Exercises 5

5.1 Question

Given a system made of discrete components with known reliability, what is the reliability of the overall system? For example, suppose we have a system that can be described with the following high-level diagram in 图 3.

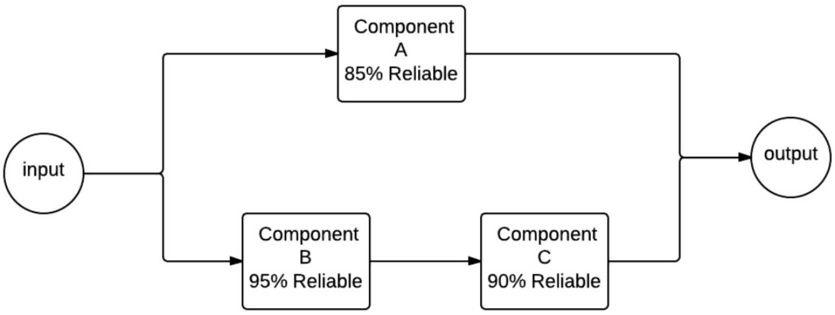

图 3: A system made of discrete components

When given an input to the system, that input flows through component A or through components B and C, each of which has a certain reliability of correctness. Probability theory tells us the following:

$$reliability_{BC} = 0.95 \times 0.90 = 0.855$$
$$reliability_{A} = 0.85$$

And the overall reliability of the system is:

$$\begin{aligned} reliability_{sys} &= 1.0 - [(1.0 - 0.85) \times (1.0 - 0.855)] \\ &= 0.97825 \end{aligned}$$

Create a simulation of this system where half the time the input travels through component A. To simulate its reliability, generate a number between 0 and 1. If the number is 0.85 or below, component A succeeded, and the system works. The other half of the time, the input would travel on the lower half of the diagram. To simulate this, you will generate two numbers between 0 and 1. If the number for component B is less than 0.95 and the number for component C is less than 0.90, then the system also succeeds. Run many trials to see if you converge on the same reliability as predicted by probability theory.

5.2 Answer

Based on the research question's requirements, this study uses conditional statements to filter randomly generated number sequences by applying specified conditions. The success count is then divided by the total number of attempts to calculate the final probability value. Following this concept, the code has been written, and the resulting code is shown in `Exercises5.py`. However, in the experiment, if half of the data passes the first stage and the other half passes the second stage, the probability obtained through programming is 85.25%±0.25%, which significantly differs from the theoretical calculated value of 0.97825%.

Through mathematical expression and analysis, this study identifies that the original calculation method incorrectly considered the probability of data failing both A and B and C, rather than the success probability of data passing A or B and C. Consequently, the code was revised to classify any data passing A or B and C as a success. After this correction, the probability calculation result was $97.80\% \pm 0.10\%$, which statistically aligns with the theoretical calculated value.