Monte Carlo Simulation

Étienne Long, Xixuan Li

Contents

1 Introduction

The research paper explores the application of Monte Carlo simulation, a computational method employed for the evaluation or approximation of definite integrals that challenge direct integration. It entails the generation of random points within the interval bounds, applying the function f, and computing the area of the integral as $f(x) \times (b - a)$. Subsequently, the average of these computed areas is calculated to yield an approximation of the integral value. The research's theoretical foundation is based on an examination of Chapter 24.2 in Larry Wasserman's "All of Statistics". Practical implementation is demonstrated using Python supplemented with NumPy to illustrate examples drawn from the textbook. Furthermore, the simulation is applied to estimate the value of π by evaluating the integral of the function $\sqrt{x^2 - 1}$ over the interval 0 to 1, representing the area of a semicircle with a 1-unit radius, and subsequently multiplying this computed area by 4.

2 Example 24.1

For the example 24.1, we approximate the value of the definite integral $h(x) = x^3$ by randomly selecting N points within the bounds. We evaluate the function $h(x)$ at each point and compute the area of the integral as $f(x) \times (b - a)$, where $f(x)$ is the function value and $b - a$ is the interval width. Finally, we calculate the average of these areas.

1. Generate N points from the function $h(x) = x^3$ from a to b.

```
import numpy as np
import random
import sympy as sy #used to find the exact integral value

#define a, b, N and h(x)
a=0
b=1
N=100
def h(x):
  return x**3
```

2. Find

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N} w(X_i)$$

where $w(X_i) = X_i(b-a)$

```
sum=0
for i in range(N):
  sum+=h(random.uniform(a,b))
sum = (b-a)*sum
print(sum)
Ihat=1/N*sum
print(Ihat)
```

We get Î = 0.2417854893547572. Next, we calculate for the percentage error.

```
x = sy.Symbol("x")
integralvalue=sy.integrate(h(x),(x,a,b))
print(integralvalue)
percenterror = abs((Ihat-integralvalue)*100/integralvalue)
print(Percenterror,"%")
```

Given that the precise integral of x^3 from 0 to 1 is 0.25 and our Monte Carlo simulation yields a percentage error of 3.28580425809712%, it demonstrates a reasonably close approximation to the true value. It's important to recognize that due to the random selection of points in a uniform distribution, each trial will produce a different sum. Consequently, the percentage error may not consistently remain at 1.61%.

3. Explore the outcomes when altering the value of N. Repeat these procedures for $N = 1000$, 10000, 100000, and 1000000.

```
N=10
for i in range(5):
  N*=10
  print("N is:",N)
  sum=0
  for i in range(N):
    sum=sum+h(random.uniform(a,b))
  sum = (b-a)*sum
  Ihat=1/N*sum
  print("I hat is:",Ihat)
  percenterror = abs((Ihat-integralvalue)*100/integralvalue)
```

```
print("Percentage error :",percenterror,"%")
print()
```

We obtain the following results:

- For $N = 100$:
  - Estimated integral ($\hat{I}$): 0.23644516706604016
  - Percentage error: 5.42193317358394%
- For $N = 1000$:
  - Estimated integral ($\hat{I}$): 0.24527452206910852
  - Percentage error: 1.89019117235659%
- For $N = 10000$:
  - Estimated integral ($\hat{I}$): 0.24596707185253205
  - Percentage error: 1.61317125898718%
- For $N = 100000$:
  - Estimated integral ($\hat{I}$): 0.24868493708934788
  - Percentage error: 0.526025164260846%
- For $N = 1000000$:
  - Estimated integral ($\hat{I}$): 0.2499858421952935
  - Percentage error: 0.00566312188260421%

Hence, we observe that as the sample size grows, the estimation becomes increasingly precise, approaching the exact value of 0.25.

4. Switch up the value of b for experimentation while maintaining N at 1000000.

```
N=1000000
b=0
for i in range(5):
  b+=1
  sum=0
  for i in range(N):
    sum+=h(random.uniform(a,b))
  sum = (b-a)*sum
  Ihat=1/N*sum
  print("I hat from",a,"to",b,"is:",Ihat)
  integralvalue=sy.integrate(h(x),(x,a,b))
  print("actual integral value is:",integralvalue)
  percenterror = abs((Ihat-integralvalue)*100/integralvalue)
  print("Percentage error:",percenterror,"%")
  print()
```

We obtain the following results:

- For x from 0 to 1:
  - Estimated integral ($\hat{I}$): 0.25005418143917824
  - Actual integral value: $\frac{1}{4}$
  - Percentage error: 0.0216725756712943%
- For x from 0 to 2:
  - Estimated integral ($\hat{I}$): 3.994489353952493
  - Actual integral value: 4
  - Percentage error: 0.137766151187679%
- For x from 0 to 3:
  - Estimated integral ($\hat{I}$): 20.23748653417094
  - Actual integral value: $\frac{81}{4}$
  - Percentage error: 0.0617948929830079%
- For x from 0 to 4:
  - Estimated integral ($\hat{I}$): 63.935650182073815
  - Actual integral value: 64
  - Percentage error: 0.100546590509665%
- For x from 0 to 5:
  - Estimated integral ($\hat{I}$): 156.19112166907306
  - Actual integral value: $\frac{625}{4}$
  - Percentage error: 0.0376821317932445%

Observe the variation in percentage error as it correlates with the bounds of the integral. Although the difference may seem insignificant, it's worth noting since we're dealing with an extensive number of random points within a narrow range of interval differences.

3 Example 24.2

The objective is to approximate the value of the cumulative distribution function (CDF) of the standard normal distribution at a specific point s. This is achieved by employing a method wherein N random points are sampled from the standard normal probability density function (PDF). For each sampled point that is smaller than s, a counter is incremented by 1. Eventually, the total count is divided by N to yield an estimation of the CDF value at s. This process, which can also be expressed as (number of observations $\leq s$) $\times \frac{1}{N}$, provides a practical means of approximating the desired CDF value.

1. Generate N points from the standard normal distribution $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ and define N and s. We also define a function $h(x)$ such that $h(x) = 1$ If $s < x$, then $h(x) = 0$ if $x \geq s$.

```
from scipy.stats import norm
N=100
s=1
def h2(x):
  if x<s:
    return 1
  if x>=s:
    return 0
```

3. Find

$$\hat{I} = \frac{1}{N} \sum_i h(X_i)$$

and compare with the exact value of Φ

```
sum=0
N=100
for i in range(N):
  sum+=h2(np.random.standard_normal())

Ihat2=1/N*sum
print("Estimated Phi:", Ihat2)
print("Exact Phi", norm.cdf(s))
percenterror2=abs((Ihat2-norm.cdf(s))*100/norm.cdf(s))
print("Percentage error:", percenterror2,"%")
```

We obtain the following results:

- Estimated Φ: 0.86
- Exact Φ: 0.8413447460685429
- Percentage error: 2.217313891675179%

4. Explore the effects of varying the value of N and repeat the steps for N ranging from 100 to $1,000,000$.

```
N=10
print("Exact Phi:", norm.cdf(s))
print()
for i in range(5):
  N*=10
  sum=0
  print("N is:",N)
  for j in range(N):
    sum+=h2(np.random.standard_normal())
  Ihat2=1/N*sum
  print("Estimated Phi:", Ihat2)
  percenterror2=abs((Ihat2-norm.cdf(s))*100/norm.cdf(s))
  print("Percentage error:",percenterror2,"%")
  print()
```

We obtain the following results:

- Exact Φ: 0.8413447460685429
- For $N = 100$:
  - Estimated Φ: 0.8300000000000001
  - Percentage error: 1.3484063603599918%
- For $N = 1000$:
  - Estimated Φ: 0.833
  - Percentage error: 0.9918343351564866%
- For $N = 10000$:
  - Estimated Φ: 0.8385
  - Percentage error: 0.3381189556166963%
- For $N = 100000$:
  - Estimated Φ: 0.84267
  - Percentage error: 0.157516159416194 68%
- For $N = 1000000$:
  - Estimated Φ: 0.842063
  - Percentage error: 0.08536975298334665%

5. Experiment with different values of s, maintaining $N = 1000000$.

```
N=1000000
s=-3

print()
for i in range(5):
  s+=1
  sum=0
  print("s is:",s)
  print("Exact Phi:", norm.cdf(s))
  for j in range(N):
    sum+=h2(np.random.standard_normal())
  Ihat2=1/N*sum
  print("Estimated Phi:", Ihat2)
  percenterror2=abs((Ihat2-norm.cdf(s))*100/norm.cdf(s))
  print("Percentage error is:",percenterror2,"%")
  print()
```

We obtain the following results:

- $s = -2$:
  - Exact Φ: 0.022750131948179195
  - Estimated Φ: 0.022421

- Percentage error: 1.4467254472585018%

- $s = -1$:
  - Exact Φ: 0.15865525393145707
  - Estimated Φ: 0.15791
  - Percentage error: 0.46973164328932165%
- $s = 0$:
  - Exact Φ: 0.5
  - Estimated Φ: 0.5007159999999999
  - Percentage error: 0.14319999999998778%
- $s = 1$:
  - Exact Φ: 0.8413447460685429
  - Estimated Φ: 0.841375
  - Percentage error: 0.0035959018700038926%
- $s = 2$:
  - Exact Φ: 0.9772498680518208
  - Estimated Φ: 0.977294
  - Percentage error: 0.0045159328869681095%

4 π Estimation

The goal is to estimate the value of π. This is achieved by determining the area of a quarter circle with a radius of 1 and subsequently multiplying the result by 4.

1. Define $h(x) = \sqrt{x^2 - 1}$.

```
def h3(x):
  return (1-x**2)**0.5
```

2. Repeat steps from Example 24.1.

```
a=0
b=1
N=10
for i in range(5):
  N*=10
  print("N is:",N)
  sum=0
  for i in range(N):
    sum+=h3(random.uniform(a,b))
  sum = (b-a)*sum
  Ihat3=1/N*sum
  print("Estimated value of pi is:",4*Ihat3)
```

```python
percenterror3=abs((np.pi-4*Ihat3)*100/np.pi)
print("Percentage error is:", percenterror3,"%")
print()
```

We obtain the following results:

- For $N = 100$:
  - Estimated value of π: 3.0260493064940888
  - Percentage error: 3.677858966332787%
- For $N = 1000$:
  - Estimated value of π: 3.10650850718489
  - Percentage error: 1.1167630649000189%
- For $N = 10000$:
  - Estimated value of π: 3.1617304493320293
  - Percentage error: 0.6410059470703622%
- For $N = 100000$:
  - Estimated value of π: 3.1424282426756522
  - Percentage error: 0.026597626681623517%
- For $N = 1000000$:
  - Estimated value of π: 3.1416425386144136
  - Percentage error: 0.0015878896509209154%

5 Conclusion

In summary, Monte Carlo simulations prove effective when N reaches a significant magnitude. With N exceeding 10,000, the percentage errors consistently remained below 1% across all three scenarios. However, counter-intuitive outcomes occasionally arise. Increasing N by a factor of 10 may paradoxically result in larger percentage errors. This phenomenon occurs because while higher N increases the likelihood of converging to the actual value, randomness still plays a significant role.

MARIANOPOLIS COLLEGE

SCIENCE PROGRAM: OBJECTIVES & GOALS

1. SCIENTIFIC VOCABULARY
- To understand the technical and scientific vocabulary and use it correctly in oral and written communication

2. LAB SKILLS AND INSTRUMENTATION
- To be able to understand and use scientific instruments correctly in the laboratory, and present experimental data for analysis

3. GRAPHICAL REPRESENTATION
- To represent data and results graphically and be able to understand and interpret graphical information

4. OBSERVATION AND ANALYSIS
- To observe and gather data
- To make and test inferences based on the data, and confirm and evaluate the conclusions drawn

5. INDUCTIVE AND DEDUCTIVE REASONING
- To be able to reason from the particular to the general (inductive) and from the general to the particular (deductive)

6. USE OF MATHEMATICAL TOOLS
- To recognize the problems that can benefit from the use of calculus, algebra or other areas of mathematics
- To identify which technique is best suited for a given situation
- To apply the technique correctly

7. PROBLEM SOLVING APPROACH
- To identify relevant variables
- To break a problem into simpler components
- To choose the sequence to follow in solving the individual components
- To reach a conclusion

8. USE OF DATA PROCESSING TECHNOLOGY
- To be at ease with the use of computers to format reports and papers
- To gather and analyze data using spreadsheet software
- To use mathematical software

9. LOGICAL REASONING
- To be able to think through a situation using a systematic approach.
- To understand the difference between a hypothesis and a conclusion

10. ORAL COMMUNICATION
- To make an oral presentation of scientific material

11. WRITTEN COMMUNICATION
- To be able to read and understand scientific material
- To make a written presentation of scientific material

12. AUTONOMOUS WORK
- To develop independent study skills

13. TEAM WORK
- To develop the ability to cooperate with other individuals in a leadership, collaborative or supportive role
- To show respect for the other members of the team

14. SCIENCE AND SOCIETY
- To be sensitized to the implications of some scientific concepts, discoveries and theories to everyday life and environment, in a non-judgmental manner

15. PERSONAL SYSTEM OF VALUES
- To form one's own judgment on contemporary scientific issues
- To compare favorable versus unfavorable consequences of the implementation of scientific and technological developments

16. HISTORICAL CONTEXT
- To be familiar with the time frame and the state of culture at the time of a scientific discovery
- To identify the questions researchers were trying to answer when proposing new theories

17. INTELLECTUAL CURIOSITY
- To demonstrate intellectual curiosity
- To appreciate natural phenomena

18. CRITICAL THINKING
- To evaluate a theory, a result or any information on the basis of logic, knowledge, experience and common sense

19. DEVELOPMENT OF INTUITION
- To develop insight leading to an appropriate approach to new concepts and to problem-solving

20. STIMULATION OF CREATIVITY
- To develop inventiveness, inquisitiveness and originality

21. CAPACITY FOR ABSTRACT THOUGHT
- To develop the ability to conceptualize and visualize a situation or idea

22. STRENGTHS AND LIMITATIONS OF SCIENTIFIC KNOWLEDGE
- To understand the context in which a theory is valid, and recognize its limitations

23. APPLY KNOWLEDGE TO NEW SITUATIONS
- To extract knowledge from prior experience and transfer it to a new setting

24. **INTEGRATIVE ACTIVITIES**
- To establish links among two or more fields of study

MARIANOPOLIS COLLEGE

SCIENCE PROGRAM ÉPREUVE SYNTHÈSE

SELF-EVALUATION FORM ON PROGRAM GOALS

NAME (*Please print*) Étienne Long ID # 2230158

ASSESSMENT ACTIVITY: Monte Carlo Simulations

SUPERVISING TEACHER: M. François Charette COURSE #: 201-LCW-05

DATE: May 2 2024

In the following table identify which Science Program Goals were relevant to your *Épreuve Synthèse* activity, and give a brief explanation or description in what way it met these goals. (Refer to the list of definitions of the Program Goals.)

PROGRAM GOALS	**BRIEF EXPLANATION OF HOW YOUR ACTIVITY CONTRIBUTED TO MEETING THE PROGRAM GOALS**
1 Learning and use of scientific vocabulary	We have used scientific vocabulary in this Épreuve, using vocabulary we learned in our Prob&Stats class to explain our procedings in the Épreuve.
2 Acquisition and use of laboratory skills	
3 Use of graphical representation	
4 Ability to observe and analyze	

5 Inductive/deductive reasoning	
6 Use of mathematical tools	
7 Problem solving	We have demonstrated problem solving skills. Initially, the task given by the Épreuve seemed complicated, but by approaching it step by step we were able to effectively complete it.
8 Data processing technology	We were able to effectively use technology. Firstly, we used coding to simulate our math, and then we used Latex to present it in a aesthetic way.
9 Logical reasoning	
10 Oral communication	
11 Written communication	We were able to demonstrate written communication skills. Firstly, we have understood the textbook by reading it, then we were able to present our findings by giving a clean presentation of our procedures.
12 Autonomous work	
13 Team work	I have worked in a team of two. Me and my partner demonstrated our communicating and organisation skills by producing this Épreuve Synthèse.
14 Awareness of science and society	

15 Development of a personal set of values	
16 Appreciation of historical context of science	
17 Stimulation of intellectual curiosity	
18 Development and use of critical thinking abilities	
19 Development of intuition	
20 Stimulation of creativity	
21 Development of abstract thought	
22 Appreciation of the strengths/limitations of science	We have appreciated the strengths and limitations of our subject, the Monte Carlo simulations. We know that it is useful to compute hard definite integrals, but also it might take a big N to find a value close to the actual value.
23 Application of knowledge to new situations	We have applied our knowledge learned in Prob&Stats to a new situation, the Monte Carlo simulation. We used what we learned in class to approach this Épreuve.
24 Integrative activities	We have integrated multiple fields of study in this Épreuve, notably math (Prob&Stats) and coding (in Python). We were able to use both effectively to produce this Épreuve.

COMMENTS ABOUT THE OVERALL EXPERIENCE (Attach a separate sheet)

Very nice experience overall, to be able to tackle a complex problem such as Monte Carlo simulations and to apply something else (coding) to it, having fun finally being able to solve it and to work in a team with a great partner.

MARIANOPOLIS COLLEGE SCIENCE

PROGRAM ÉPREUVE SYNTHÈSE

SELF-EVALUATION FORM ON PROGRAM GOALS

NAME (*Please print)*: Xixuan Li

ID #: 2232324

ASSESSMENT ACTIVITY: Monte Carlo Simulation

SUPERVISING TEACHER: Professor François Charette

COURSE #: Probability and Statistics

DATE: 02/05/2024

In the following table identify which Science Program Goals were relevant to your *Épreuve Synthèse* activity, and give a brief explanation or description in what way it met these goals. (Refer to the list of definitions of the Program Goals.)

PROGRAM GOALS	**BRIEF EXPLANATION OF HOW YOUR ACTIVITY CONTRIBUTED TO MEETING THE PROGRAM GOALS**
1 Learning and use of scientific vocabulary	By reading the textbook "All of Statistics", I got better at explaining complex ideas using clear scientific language when talking about the method and results. Discussing and writing about my findings also improved how I use scientific words.
2 Acquisition and use of laboratory skills	
3 Use of graphical representation	

4 Ability to observe and analyze	
5 Inductive/deductive reasoning	
6 Use of mathematical tools	
7 Problem solving	
8 Data processing technology	Python served as mathematical software for various numerical computations. In Google Colab, I used Python libraries like SciPy for mathematical functions and NumPy to generate random values in a uniform distribution.
9 Logical reasoning	
10 Oral communication	
11 Written communication	I improved my written communication by researching scientific material and presenting my findings. For instance, I read scientific papers to understand Monte Carlo methods. Then, I wrote with my teammate a concise research presentation, detailing our methodology and results.
12 Autonomous work	When I encountered errors in my code, I relied on online tutorials and Python documentation to find solutions. Additionally, managing my time effectively between learning Python syntax, understanding Monte Carlo methods and coding simulations helped me develop strong autonomous work habits.

13 Team work	By working with a teammate, I developed cooperative skills by assuming both leadership and supportive roles. For example, I took the lead in organizing our project timeline. Meanwhile, during coding sessions, I collaborated with my teammate by sharing ideas and solving issues together. This partnership allowed us to effectively combine our strengths, resulting in a successful project outcome.
14 Awareness of science and society	

15 Development of a personal set of values	
16 Appreciation of historical context of science	
17 Stimulation of intellectual curiosity	
18 Development and use of critical thinking abilities	The project on Monte Carlo simulation allows me to gain the ability to evaluate theories, results and information based on data analysis. For example, when analyzing the outcomes of the simulations, I applied logical reasoning to interpret the results in the context of the underlying theory. This process enabled me to make informed judgments and conclusions based on a thorough evaluation of the simulated data.
19 Development of intuition	
20 Stimulation of creativity	

21 Development of abstract thought	
22 Appreciation of the strengths/limitations of science	
23 Application of knowledge to new situations	While I previously learned integration methods in calculus 2 to find areas under curves, I now employed Monte Carlo techniques to estimate those areas probabilistically. This shift allowed me to solve complex integration problems by generating random samples and computing probabilities, showcasing a novel application of my mathematical knowledge in a new context.
24 Integrative activities	In my Monte Carlo project, I connected coding with probability concepts, merging computer science and mathematics. For example, I used Python to estimate π with random sampling. By integrating probability theory into coding, I used algorithms generating random numbers based on distributions, showing how these fields can work together to solve problems.

COMMENTS ABOUT THE OVERALL EXPERIENCE

Overall, I found the project enjoyable and valuable. It provided a practical link between classroom theory and real-world applications. For instance, applying probability concepts to coding in Monte Carlo simulation illustrated how academic knowledge translates into practical problem-solving skills. Additionally, seeing how different disciplines, such as mathematics and computer science, intersected in this project was particularly enlightening.