**Al-Mustaqbal University**
**College of Science**

# Monte Carlo Simulation

Asst. Lec Hadi Salah

Lecture 6

## Introduction

Monte Carlo experiments are a class of computational algorithms that rely on repeated random sampling to determine the properties of some phenomenon (or behavior). Monte Carlo methods are often used in computer simulations of physical and mathematical systems.

Monte Carlo methods are especially useful for simulating systems with many coupled degrees of freedom, such as fluids, disordered materials, and strongly coupled solids. They are widely used in mathematics, for example to evaluate multidimensional definite integrals with complicated boundary conditions.

Before the Monte Carlo method was developed, simulations tested a previously understood deterministic problem and statistical sampling was used to estimate uncertainties in the simulations. Monte Carlo simulations invert this approach, solving deterministic problems using a probabilistic approach.

## 1 What is Monte Carlo Simulation?

- A numerical approach that uses random sampling to estimate quantities (probabilities, integrals, risks).
- Idea: run many random experiments and take the mean or frequency as an estimator.
- Practical when analytic / closed-form solutions are hard or impossible.

## 2 Why Use Monte Carlo?

- Handles complex, multi-variable, non-linear models.
- Does not require a closed-form formula.

- Accuracy improves as sample size $N$ grows.
- Highly flexible: same logic works across domains.
- Useful when systems have uncertainty and provides probabilistic results.
- Simple to implement with computers.
- Helps in decision making under uncertainty.

## 3 History

- Named after the Monte Carlo Casino in Monaco.
- Developed during the 1940s by scientists working on nuclear projects.
- Now used widely in finance, engineering, physics, and medicine.

## 4 Analytic vs. Monte Carlo

- For simple problems (coin / dice), analytic probability is exact and easy.
- Many real systems are complex: high dimensions, correlations, stochastic dynamics.
- Monte Carlo provides workable approximations without closed-form math.

## 5 Basic Concept

1. Define the model and uncertain parameters.
2. Generate random values for the uncertain variables.
3. Run simulations repeatedly.
4. Analyze the output distribution.

## 6 Mathematical Idea

Let $X$ be a random input and $f(X)$ a quantity of interest. We are often interested in the expectation

$$\mathbb{E}[f(X)].$$

If we draw $N$ independent samples $x_1, x_2, \ldots, x_N$ from the distribution of $X$, then the Monte Carlo estimator is

$$\mathbb{E}[f(X)] \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i).$$

As $N$ increases, this estimator converges (by the Law of Large Numbers) to the true expected value.

## 7 Example: Estimating $\pi$

Consider a circle inscribed in a unit square. Given that the circle and the square have a ratio of areas equal to $\pi/4$, the value of $\pi$ can be approximated using a Monte Carlo method as follows.

### Geometric Idea

- Take a unit square $[0,1] \times [0,1]$.
- Inside it, draw a quarter circle of radius 1 centered at $(0,0)$.
- The area of the unit square is
$$A_{\text{square}} = 1.$$
- The area of a full circle of radius 1 is $\pi$. The quarter circle therefore has area
$$A_{\text{circle}} = \frac{\pi}{4}.$$
- Hence
$$\frac{A_{\text{circle}}}{A_{\text{square}}} = \frac{\pi/4}{1} = \frac{\pi}{4}.$$

If we throw random points uniformly in the unit square, the probability that a point lies inside the quarter circle is exactly the ratio of the two areas:

$$\mathbb{P}\big(x^2 + y^2 \leq 1\big) = \frac{\pi}{4}.$$

Thus

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

### Practical Steps

1. Generate $N$ random points $(x, y)$ uniformly in the unit square.
2. Count the points with $x^2 + y^2 \leq 1$ (inside the quarter circle); let this number be $K$.
3. Use the estimator
$$\hat{\pi} = 4 \times \frac{K}{N}.$$
Convergence improves as $N$ becomes large.

### Python Code

Listing 1: Monte Carlo estimation of $\pi$

```
import numpy as np
import math

n = 200
Nin = 0
```

```
for _ in range(n + 1):
    x = np.random.random()
    y = np.random.random()
    r = x * x + y * y
    if r <= 1.0:
        Nin += 1

p = 4 * Nin / n
e = abs((math.pi - p) / math.pi)

print(f"pi␣approximation:␣{p}")
print(f"relative␣error:␣{e}")
```

## Illustration

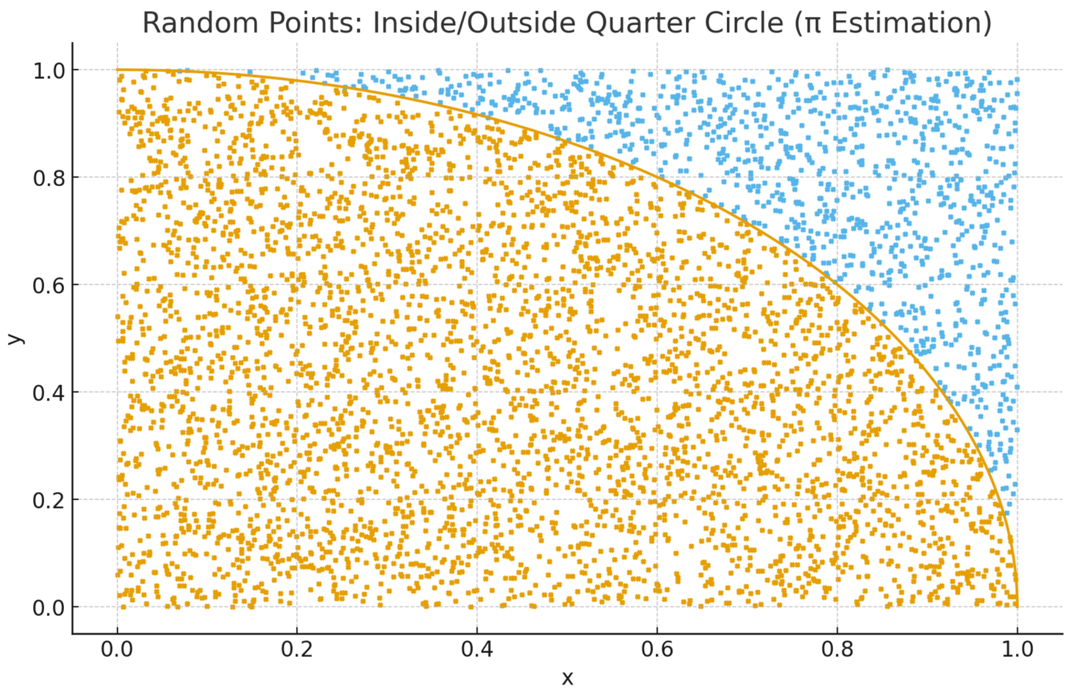

Figure 1: Random points inside and outside the quarter circle for estimating $\pi$.

## 8 Monte Carlo Integration

Let $X$ be a random variable with probability density function (pdf) $p_X(x)$ (or simply $p(x)$) over the interval $[a,b]$. If $f(x)$ is a function of this random variable, then the expected value of $f(X)$ is

$$\mathbb{E}\{f(X)\} = \int_a^b f(x)\,p(x)\,dx.$$

### Uniform Case

If $X$ is uniformly distributed over $(0,1)$, then its pdf is $p(x) = 1$ on $[0,1]$ and

$$\mathbb{E}\{f(X)\} = \int_0^1 f(x)\,dx.$$

We can approximate this integral using Monte Carlo by drawing $N$ independent samples $x_1, \ldots, x_N$ from the uniform distribution on $(0,1)$ and computing

$$\mathbb{E}\{f(X)\} \approx \frac{1}{N}\sum_{k=1}^{N} f(x_k),$$

where $f(x_k)$ is a realization of $f(X)$. This technique can be more efficient than numerical integration, especially for multi-dimensional integrals.

## 9 Example: One-Dimensional Integral

### Problem

Find

$$g = \int_1^3 (y^2 + y)\,dy$$

using Monte Carlo simulation.

### Exact Solution

The exact answer is

$$g = \int_1^3 (y^2 + y)\,dy = \left[\frac{y^3}{3} + \frac{y^2}{2}\right]_1^3 = \frac{38}{3} \approx 12.67.$$

### Transform to Uniform(0,1)

We map the interval $[1,3]$ to $[0,1]$ using a linear transformation.

Let two points be $P_1 = (0,1)$ and $P_2 = (1,3)$.

The slope is

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3-1}{1-0} = 2.$$

Thus the relationship between $x$ and $y$ is

$$y = 2x + 1, \qquad \text{or inversely} \quad x = \frac{y-1}{2},$$

where $x \in (0,1)$ is uniform.

Then

$$g = \int_1^3 (y^2 + y)\,dy = \int_0^1 ((2x+1)^2 + (2x+1)) \cdot 2\,dx.$$

Simplifying:

$$(2x+1)^2 + (2x+1) = 4x^2 + 4x + 1 + 2x + 1 = 4x^2 + 6x + 2,$$

so

$$g = 2\int_0^1 (4x^2 + 6x + 2)\,dx = 4\int_0^1 (2x^2 + 3x + 1)\,dx = 4\,\mathbb{E}\{2X^2 + 3X + 1\}.$$

Equivalently,

$$g = 4\,\mathbb{E}\{2X^2 + 3X\} + 4.$$

## Monte Carlo Estimator

Using $N$ samples $x_1, \dots, x_N$ from Uniform$(0, 1)$, we estimate

$$g \approx 4\left(\frac{1}{N}\sum_{k=1}^{N}(2x_k^2 + 3x_k)\right) + 4.$$

## Python Simulation

Listing 2: Monte Carlo integration example

```
import numpy as np

N = 10000
x = np.random.rand(N, 1)    # Uniform(0,1) samples
g = 4 * np.mean(2 * x**2 + 3 * x) + 4

print(f"Estimated␣value:␣{g}")
e = abs(12.67 - g)
print(f"relative␣error:␣{e}")
```

Sample output:

```
Estimated value: 12.47121841004683
relative error: 0.1987815899531693
```

# Homework: Monte Carlo Simulation Example

## Question

Use the Monte Carlo simulation method to estimate the following double integral:

$$g = \int_0^1 \int_0^1 (4x^2 y + y^2)\,dx\,dy.$$

- Write Python code that uses random sampling over the unit square $(x, y) \in [0, 1] \times [0, 1]$.
- Compute an estimate of $g$ and compare it with the analytical value.
- Calculate the Mean Squared Error (MSE) of your estimator.

## Analytical Solution

The exact value of this integral is

$$g = 1.$$