Chapter 3

Random Numbers

Ali Raisolsadat

School of Mathematical and Computational Sciences
University of Prince Edward Island

Table of contents

Introduction

Introduction to Random Numbers

- Randomness has played an important role for centuries in games, gambling, and statistics.
- Historically, random numbers were generated manually or mechanically (e.g., spinning wheels, rolling dice, shuffling cards, or using random number tables).
- Early statisticians and engineers relied on these methods to perform experiments, simulations, and probability calculations.
- Today, computers generate **pseudo-random numbers**, allowing fast, repeatable, and large-scale simulations.

Historical Example: Buffon’s Needle

- One of the earliest applications of randomness in probability was **Buffon’s Needle problem** (1777), proposed by Georges-Louis Leclerc, Comte de Buffon.
- The experiment involved dropping a needle of length l onto a floor with parallel lines spaced d apart ($l < d$) and estimating π based on the probability that the needle crosses a line.
- Early statisticians used **manual or mechanical randomization techniques** (e.g., physically dropping needles or drawing lots) to perform the experiment.

Objectives

- A fundamental component of any simulation study is the ability to generate **random numbers**.
- These numbers typically represent values of a random variable that is **uniformly distributed** on $(0, 1)$.
- This chapter explains how such numbers are **generated by computers**, including pseudo-random number generation.
- We will introduce the **Monte Carlo method**, which uses random sampling to approximate integrals.

Random Numbers

Pseudo-Random Number Generation

- A **pseudo-random number sequence** consists of deterministically generated values that **appear** to be independent uniform random variables on $(0,1)$.
- The most common method generates numbers starting from an initial value x_0 (the **seed**) and recursively computes successive values x_n for $n \geq 1$ using

$$x_n = a x_{n-1} \mod m \tag{1}$$

  where a and m are positive integers.
- The remainder operation ensures that $x_n \in \{0, 1, \ldots, m-1\}$.

Pseudo-Random Number Generation

- The quantity $\frac{x_n}{m}$ is called a **pseudo-random number** and is taken as an approximation of a uniform $(0, 1)$ random variable.
- This approach is known as the **Multiplicative Congruential Method**.
- Since each x_n assumes one of the values $0, 1, \ldots, m-1$, eventually a value will repeat after at most m iterations.
- Therefore, the constants a and m should be chosen to maximize the length of the sequence before repetition occurs, for any initial seed x_0.

Choosing Constants in the Multiplicative Congruential Method

- In general, the constants a and m should satisfy three criteria:
    1. For any initial seed, the sequence appears to be independent uniform $(0, 1)$ random variables.
    2. The period (number of generated values before repetition) is large.
    3. The values can be computed efficiently on a digital computer.
- A common guideline is to choose m as a large prime number compatible with the computer's word size.
- Example: On a 32-bit machine (with the first bit as a sign bit), the choice $m = 2^{31} - 1$ and $a = 7^5 = 16807$ produces desirable properties.

Mixed Congruential Generators

- Another type of pseudo-random number generator uses the recursion

$$x_n = (ax_{n-1} + c) \mod m \tag{2}$$

  combining both a multiplicative and additive term. These are called **Mixed Congruential Generators**.
- In this case, m is often chosen to match the computer's word length for efficient computation.
- Assuming such a generator, we can produce a sequence of pseudo-random numbers approximating independent uniform $[0, 1]$ random variables.
- Conceptually, we can treat the generator as a ”black box” that returns a random number on request.

Applications of Generated Random Numbers

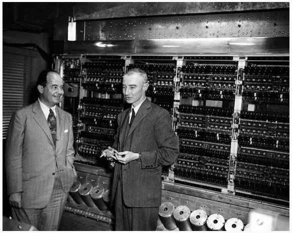
History of the Monte Carlo Method

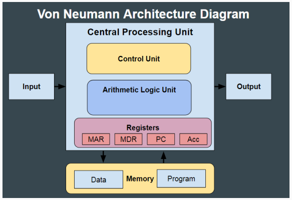
- The Monte Carlo method was developed in the 1940s during research for the Manhattan Project.
- Mathematician Stanislaw Ulam conceived the idea while recovering from an illness, inspired by thinking about solitaire.
- He wondered: *“What is the expected number of winning hands in a game of solitaire?”*
- This question led to the use of repeated random sampling to estimate expectations — the foundation of Monte Carlo simulation.

Stanislaw Ulam (1909–1984)

Early Development and Implementation

- Ulam proposed using random sampling to solve complex problems in neutron diffusion.
- John von Neumann and colleagues formalized the method and implemented it on the ENIAC computer, one of the earliest electronic computers.
- The name *"Monte Carlo"* was inspired by the frequent trips of Ulam's uncle to the casino in Monte Carlo, Monaco.


ENIAC - John von Neumann and Oppenheimer



Von Neumann architecture

Applications and Impact of Monte Carlo (MC) Methods

- Monte Carlo methods are widely used in statistical physics, computational chemistry, statistical inference, genetics, and finance.
- The Metropolis algorithm, a key Monte Carlo technique, was named one of the top algorithms of the 20th century by a committee of mathematicians, computer scientists, and physicists.
- The increasing availability of computational power has greatly expanded the use of Monte Carlo simulations.

Motivation: Queuing Systems

- Consider a newly opened coffee shop where customers arrive and queue to be served. Each customer's order takes a random amount of time to prepare.
- The manager wants to ensure customer satisfaction and prevent employees from working beyond regular hours (9am–5pm).
- Question: On a busy holiday, if the baristas continue serving everyone already in the shop at 5pm, what is the probability they finish all orders by 5:30pm?
- Let X denote the number of customers still in the shop at 5:30pm. The probability of interest is

  $$\mathbb{P}(X = 0) = \mathbb{E}\left[\mathbb{I}(X = 0)\right],$$

  where $\mathbb{I}(\cdot)$ is the indicator function.

Queuing Systems as an Expectation

- Let N be the number of customers in the store at 5:00 pm.
- Let $S_1, \dots, S_N$ be i.i.d. service times (minutes) with pdf $f(s)$.
- We finish by 5:30 pm if

$$\underbrace{S_1 + S_2 + \cdots + S_N}_{\text{total service time}} \le 30$$

- The probability of serving all customers:

$$\mathbb{P}(X = 0) = \sum_{n=0}^{\infty} \mathbb{P}(N = n)\, \mathbb{P}\Big(S_1 + \cdots + S_n \le 30 \,\Big|\, N = n\Big)$$

- For fixed n, the conditional probability can be expressed as an expectation:

$$\begin{aligned}
&\mathbb{P}\Big(S_1 + \cdots + S_n \le 30\Big) \\
&= \mathbb{E}\Big[\mathbb{I}\{S_1 + \cdots + S_n \le 30\}\Big] \\
&= \int_0^{\infty} \cdots \int_0^{\infty} \mathbb{I}\{s_1 + \cdots + s_n \le 30\}\, f(s_1) \cdots f(s_n)\, ds_1 \cdots ds_n
\end{aligned}$$

Motivation: Particle in a Random Medium

- Consider a particle $(X_t)_{t=1,2,\dots}$ moving in space $\Omega = \mathbb{R}^d$ according to a stochastic model (e.g., random walk, diffusion, or Markov chain). Randomness may arise from the environment (”medium”) or from the particle’s motion.
- **Absorption rule:** At each time step t, the particle can be absorbed with probability $1 - G(X_t)$, where $G : \Omega \to [0, 1]$ is the **survival weight** at location X_t.
    - $G(x) \approx 1$: safe location (low absorption risk)
    - $G(x) \approx 0$: high absorption risk (almost surely absorbed)
- The survival probability up to time T is

$$\mathbb{P}(\text{not absorbed at time } T) = \mathbb{E}\Big[\prod_{t=1}^{T} G(X_t)\Big].$$

Multiple Particles in a Random Medium

- Suppose we have n independent particles, each evolving over T time steps according to the stochastic model.
- Let $X_{i,t}$ denote the position of particle i at time t, for $i = 1, \dots, n$ and $t = 1, \dots, T$.
- Let $G(X_{i,t})$ denote the survival weight at that position.
- The joint survival probability for all particles is

$$\mathbb{P}(\text{all particles survive up to } T) = \mathbb{E}\Big[\prod_{i=1}^{n}\prod_{t=1}^{T} G(X_{i,t})\Big].$$

- For continuous positions, this is a $(n \cdot T)$-dimensional integral:

$$\mathbb{E}\Big[\prod_{i=1}^{n}\prod_{t=1}^{T} G(X_{i,t})\Big] = \int \cdots \int \prod_{i=1}^{n}\prod_{t=1}^{T} G(x_{i,t})\, p(x_{1,1}, \dots, x_{n,T})\, dx_{1,1} \cdots dx_{n,T}$$

where $p(\cdot)$ is the joint density of all particle positions.

Using Random Numbers to Evaluate Integrals

- One of the earliest applications of random numbers was in the computation of integrals.
- Let $g(x)$ be a function, and suppose we want to compute

$$\theta = \int_0^1 g(x)\,dx$$

- If $U \sim \text{Uniform}(0,1)$, then

$$\theta = \mathbb{E}[g(U)]$$

- Let $U_1, \ldots, U_k$ be independent and identically distributed (i.i.d.) $\text{Uniform}(0,1)$ random variables.
- Then $g(U_1), \ldots, g(U_k)$ are i.i.d. with mean θ.

Using Random Numbers to Evaluate Integrals

- **Law of Large Numbers (LLN)**: If $Z_1, \dots, Z_n$ are i.i.d. with $\mathbb{E}[|Z_i|] < \infty$, then

$$\frac{1}{n}\sum_{i=1}^{n} Z_i \longrightarrow \mathbb{E}[Z_1] \quad \text{almost surely as } n \to \infty$$

- Therefore, if we
  - Let $Z_i = g(X_i)$ for $i = 1, \dots, k$.
  - Then $Z_1, \dots, Z_k$ are i.i.d. because X_i are i.i.d.
  - Assume $\mathbb{E}[|g(U)|] < \infty$.
  - By the Law of Large Numbers:

$$\hat{\theta}_k = \frac{1}{k}\sum_{i=1}^{k} g(U_i) \longrightarrow \mathbb{E}[g(U)] = \theta \quad \text{as } k \to \infty$$

Monte Carlo for Finite Intervals

- Suppose we want to compute

$$\theta = \int_a^b g(x)\,dx$$

- Transform to a Uniform$(0,1)$ variable:

$$y = \frac{x-a}{b-a} \quad \Rightarrow \quad x = a + (b-a)y, \quad dx = (b-a)dy$$

- Then
$$\theta = \int_0^1 g(a+(b-a)y)\,(b-a)\,dy = \int_0^1 h(y)\,dy$$
where $h(y) = g(a+(b-a)y)\,(b-a)$.
- Generate i.i.d. $U_1, \dots, U_k \sim$ Uniform$(0,1)$ and compute

$$\hat{\theta}_k = \frac{1}{k}\sum_{i=1}^{k} h(U_i)$$

which converges to θ as $k \to \infty$.

Monte Carlo for Semi-Infinite Intervals

- Suppose we want to compute

$$\theta = \int_0^\infty g(x)\,dx$$

- Apply the substitution

$$y = \frac{1}{x+1} \quad \Rightarrow \quad x = \frac{1}{y} - 1, \quad dx = -\frac{1}{y^2}dy$$

- Then

$$\theta = \int_0^1 g\left(\frac{1}{y} - 1\right)\frac{1}{y^2}\,dy = \int_0^1 h(y)\,dy$$

  where $h(y) = \frac{g(1/y-1)}{y^2}$.
- Generate i.i.d. $U_1, \ldots, U_k \sim \text{Uniform}(0,1)$ and compute

$$\hat{\theta}_k = \frac{1}{k}\sum_{i=1}^{k} h(U_i)$$

  which converges to θ as $k \to \infty$.

Monte Carlo for Multidimensional Integrals

- For a function g of n variables, consider

$$\theta = \int_0^1 \cdots \int_0^1 g(x_1, \ldots, x_n)\, dx_1 \cdots dx_n$$

- Key idea: Express θ as an expectation:

$$\theta = \mathbb{E}[g(U_1, \ldots, U_n)]$$

  where $U_1, \ldots, U_n$ are independent Uniform $(0, 1)$ random variables.

- Generate k independent samples:

$$\begin{bmatrix} U_1^1 & \ldots & U_n^1 \\ U_1^2 & \ldots & U_n^2 \\ \vdots & & \vdots \\ U_1^k & \ldots & U_n^k \end{bmatrix}.$$

- Then $g(U_1^i, \ldots, U_n^i)$ are i.i.d. with mean θ, so we can estimate θ as

$$\hat{\theta}_k = \frac{1}{k} \sum_{i=1}^{k} g(U_1^i, \ldots, U_n^i)$$

Generalization: Monte Carlo Estimator

- **Goal:** Approximate the expectation

$$\theta = \mathbb{E}[g(X)] = \int g(x)f(x)\,dx$$

  where $X \sim F$ and f is the corresponding probability density function.

- **Monte Carlo Idea:** Replace the integral with a sample average based on simulated draws

$$X_1, \ldots, X_k \overset{\text{iid}}{\sim} F : \quad \hat{\theta}_k = \frac{1}{k}\sum_{i=1}^{k} g(X_i)$$

- **Connection to Uniforms:** Each X_i can be generated by first drawing

$$U_i \sim \text{Uniform}(0, 1)$$

  and then transforming via the inverse CDF:

$$X_i = F^{-1}(U_i)$$

  This highlights that all random draws ultimately originate from Uniform$(0, 1)$ numbers (more about transformations in the next lecture).

Monte Carlo Algorithm

Algorithm

1. **Generate Uniform random numbers:** Draw

$$U_1, U_2, \ldots, U_k \overset{\text{iid}}{\sim} \text{Uniform}(0,1)$$

2. **Transform to target distribution:** Use the inverse transform method:

$$X_i = F^{-1}(U_i), \quad i = 1, \ldots, k$$

so that $X_i \sim F$, where F is the cumulative distribution function of X.

3. **Evaluate the function:** Compute

$$g(X_i) \quad \text{for each sampled } X_i$$

4. **Compute the Monte Carlo average:**

$$\hat{\theta}_k = \frac{1}{k} \sum_{i=1}^{k} g(X_i)$$

Example 1: Particle in a Random Medium (Star Wars Edition)

SPOILERS

- During the first duel between Luke and Vader on Cloud City, Luke cuts a Tibanna gas pipe to impair Vader's vision, releasing gas into the surrounding area.
- Let $X_{t,i}$ denote the state (e.g., position) of the i-th gas particle at time t, for $t = 1, 2, \dots, T$ and $i = 1, 2, \dots, n$.
- Goal: Simulate multiple particle trajectories to approximate a quantity of interest $\hat{\theta}$ as the gas disperses while Luke fights for the fate of his friends and the galaxy.

Example 1: Particle in a Random Medium (Star Wars Edition)

- Our goal is to compute an expectation of the form:

$$\theta = \mathbb{E}\left[\prod_{t=1}^{T} G(X_t)\right]$$

  where G represents the particle's survival weight (or contribution) at each time step in the medium.

- Using Monte Carlo simulation with n independent particles, the estimator is

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n}\prod_{t=1}^{T} G(X_{t,i})$$

  where $X_{t,i}$ denotes the state of particle i at time t.

Algorithm: Monte Carlo Estimation of Particle in Random Medium (Uniform Transformation)

Algorithm 1 Monte Carlo Estimation of $\theta = \mathbb{E}\left[\prod_{t=1}^{T} G(X_t)\right]$ using Uniform$(0,1)$ numbers

```
1:  Input: Number of particles $n$, number of time steps $T$, function $G$, step size $\sigma$
2:  for $i = 1$ to $n$ do                                        ▷ Loop over particles
3:      Initialize trajectory: $X_{1:T,i} \leftarrow 0$
4:      for $t = 2$ to $T$ do                                    ▷ Loop over time steps
5:          Draw uniform random number: $U \sim$ Uniform$(0,1)$
6:          Transform to normal step via inverse CDF: $s \leftarrow \sigma \cdot \Phi^{-1}(U)$
7:          Update position: $X_{t,i} \leftarrow X_{t-1,i} + s$
8:      end for
9:      Compute particle contribution: $P_i \leftarrow \prod_{t=1}^{T} G(X_{t,i})$
10: end for
11: Output: Monte Carlo estimate:
        $$\hat{\theta} \leftarrow \frac{1}{n}\sum_{i=1}^{n} P_i$$
```

Algorithm: Monte Carlo Estimation of Particle in Random Medium

Algorithm 2 Monte Carlo Estimation of $\theta = \mathbb{E}[\prod_{t=1}^{T} G(X_t)]$

```
 1: Input: Number of particles $n$, number of time steps $T$, function $G$, step size $\sigma$
 2: for $i = 1$ to $n$ do                                          ▷ Loop over particles
 3:     Initialize trajectory: $X_{1:T,i} \leftarrow 0$
 4:     for $t = 2$ to $T$ do                                      ▷ Loop over time steps
 5:         Sample step: $s \sim \mathcal{N}(0, \sigma^2)$
 6:         Update position: $X_{t,i} \leftarrow X_{t-1,i} + s$
 7:     end for
 8:     Compute product for particle $i$: $P_i \leftarrow \prod_{t=1}^{T} G(X_{t,i})$
 9: end for
10: Output: Monte Carlo estimate:
```

$$\hat{\theta} \leftarrow \frac{1}{n} \sum_{i=1}^{n} P_i$$

Monte Carlo Particle Simulation Visualization

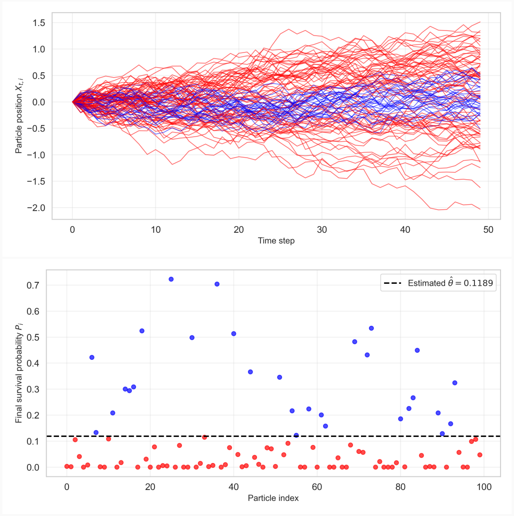


Figure 1: Monte Carlo particle simulation: **Top** shows particle trajectories $X_{t,i}$ colored by final survival probability relative to $\hat{\theta}$ (blue $\geq \hat{\theta}$, red $< \hat{\theta}$). **Bottom** shows final survival probabilities P_i with the dashed line indicating $\hat{\theta}$.

Example 2: Estimating π with Monte Carlo Algorithm

Consider the 2×2 square. Denote this square $\mathcal{S} \subseteq \mathbb{R}^2$.

Now consider a circle (disk) denoted as $\mathcal{D}$ of radius of 1. We start by setting the ratio of the area of a circle to the area of the square that contains it:

$$\frac{\text{Area of Circle}}{\text{Area of Square}} = \frac{\int\int_{\mathcal{D}} dx_1 dx_2}{\int\int_{\mathcal{S}} dx_1 dx_2} = \frac{\pi}{4}.$$

Example 2: Estimating π with Monte Carlo Algorithm

Our goal is to estimate this ratio using simulation. We can write it as:

$$\frac{\int\int_{\mathcal{D}} dx_1 dx_2}{\int\int_{\mathcal{S}} dx_1 dx_2} = \int\int_{\mathcal{S}} \mathbb{I}((x_1, x_2) \in \mathcal{D}) \frac{1}{4} dx_1 dx_2 = \mathbb{E}[\phi(X_1, X_2)] = \theta,$$

where the expectation is taken with respect to the uniform distribution over $\mathcal{S}$, and

$$\phi(X_1, X_2) = \mathbb{I}((X_1, X_2) \in \mathcal{D}).$$

To sample uniformly from the square $\mathcal{S} = (-1, 1) \times (-1, 1)$, we use:

$$X_1 = 2U_1 - 1, \quad X_2 = 2U_2 - 1,$$

where $U_1, U_2 \sim \mathcal{U}(0, 1)$.

Example 2: Estimating π with Monte Carlo Algorithm

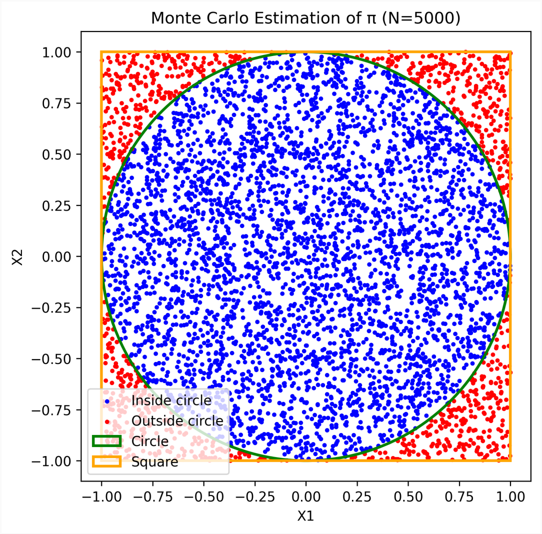


Figure 2: A 2×2 square $\mathcal{S}$ with inscribed disk $\mathcal{D}$ of radius 1 and Monte Carlo samples.

Example 3: Estimating π with Monte Carlo Algorithm

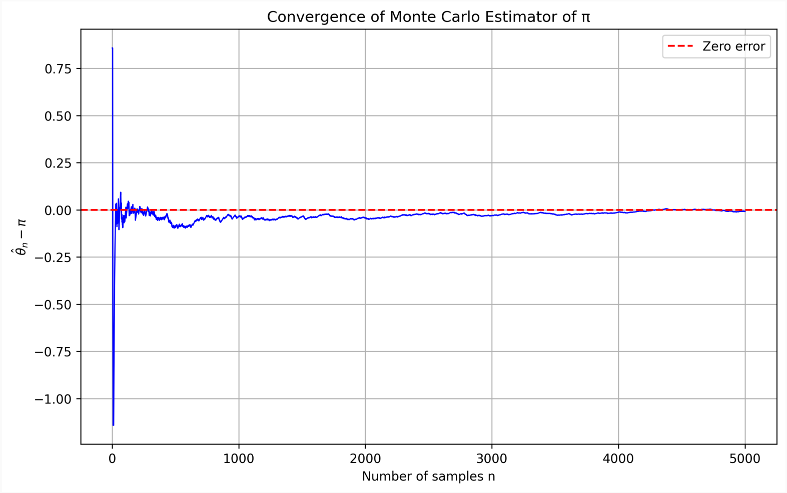


Figure 3: $\hat{\theta}_n - \theta$ as a function of the number of samples n = 5000.

Algorithm: Monte Carlo Estimation of π

Algorithm 3 Monte Carlo estimator for π (square $(-1,1)^2$)

```
1: Input: number of samples $N$
2: Initialize counter $C \leftarrow 0$
3: for $i = 1$ to $N$ do
4:     Sample $U_1, U_2 \sim \mathcal{U}(0,1)$ independently
5:     Set $X_1 \leftarrow 2U_1 - 1$ and $X_2 \leftarrow 2U_2 - 1$    (uniform on $(-1,1)$)
6:     if $X_1^2 + X_2^2 \leq 1$ then
7:         $C \leftarrow C + 1$
8:     end if
9: end for
10: Output: $\hat{\pi} \leftarrow 4 \cdot \frac{C}{N}$
```

Vectorizing the Random Particle Simulation

- In the standard Monte Carlo simulation, particle positions are updated sequentially using nested loops.
- Vectorization replaces explicit loops with array operations, allowing computations for all particles (and/or all time steps) simultaneously.
- Sample all random steps for all particles in a single matrix operation.

$$S = \begin{bmatrix} s_{2,1} & s_{2,2} & \dots & s_{2,n} \\ \vdots & \vdots & & \vdots \\ s_{T,1} & s_{T,2} & \dots & s_{T,n} \end{bmatrix}, \quad s_{t,i} \sim \mathcal{N}(0, \sigma^2)$$

- Compute cumulative sums along the time axis to get particle trajectories.

$$X = \begin{bmatrix} 0 & 0 & \dots & 0 \\ \sum_{k=2}^{2} s_{k,1} & \sum_{k=2}^{2} s_{k,2} & \dots & \sum_{k=2}^{2} s_{k,n} \\ \vdots & \vdots & & \vdots \\ \sum_{k=2}^{T} s_{k,1} & \sum_{k=2}^{T} s_{k,2} & \dots & \sum_{k=2}^{T} s_{k,n} \end{bmatrix}$$

Vectorizing the Random Particle Simulation

- Apply the survival function G across all particles and time steps using element-wise operations.

$$W = G(X) = \begin{bmatrix} G(X_{1,1}) & \dots & G(X_{1,n}) \\ \vdots & & \vdots \\ G(X_{T,1}) & \dots & G(X_{T,n}) \end{bmatrix}$$

- Reduce across particles (e.g., take the product over time and average over particles) to compute the Monte Carlo estimate.

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} \prod_{t=1}^{T} W_{t,i}$$

Complexity Comparison: Loop vs. Vectorized Implementation

- **Loop Implementation:**

  $$\mathcal{O}(nT)$$

  because we update n particles over T time steps using explicit nested loops.
- **Vectorized Implementation:**

  $$\mathcal{O}(nT)$$

  in theory, but avoids Python-level loops and uses highly optimized NumPy (C/Fortran backend). This leads to practical speedups of $10-100\times$.
- Both algorithms scale linearly in n and T, but vectorization minimizes interpreter overhead.

Monte Carlo Simulation Performance

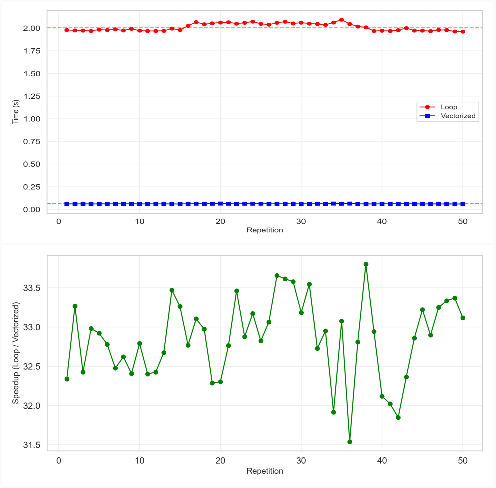


Figure 4: Top: Monte Carlo simulation running times for Loop (red) vs. Vectorized (blue) implementations, with dashed lines showing mean times. Bottom: Speedup achieved by vectorization (green), averaged over repetitions.

Next Lecture

In the next class, we will cover:

- Generating Discrete Random Variables
- The Inverse Transform Method
- Generating Poisson and Binomial Random Variables
- The Acceptance-Rejection Method

”The idea of using random sampling to solve problems that might be deterministic in principle was something new... It all came out of my attempts to calculate the probability of winning a solitaire game by actual simulation.”

Stanislaw Ulam, 1940s

Example 1: Queuing Simulation

- **Independence Assumptions:**
  - Inter-arrival times are independent of service times.
  - Service times are independent across customers and baristas.
- **Arrival Process:**
  - Customers arrive according to a *Poisson process* with rate λ (customers per minute) until 5:00pm.
  - Inter-arrival times T_i are i.i.d. exponential random variables:

$$T_i \sim \text{Exp}(\lambda), \quad i = 1, 2, \ldots$$

- **Simulation Horizon:**
  - Track customer arrivals up to 5:00pm.
  - Continue serving all customers present at 5:00pm until 5:30pm.
- **Outcome of Interest:**
  - $X =$ number of customers remaining in the shop at 5:30pm.
  - $\mathbb{P}(X = 0) =$ probability that all customers are served by 5:30pm.