Introduction to Modern C++

Homework 2 : PRNGs, Monte-Carlo Integration, and Markov Chains

1 Context

In many applications, where an exact solution is not feasible, *randomized algorithms* can provide fast approximations. In fact, for most problems, approximate solutions is the best that we can get and thus randomized algorithms are often the only option available.

A *randomized* algorithm is an algorithm that has access to a pool of values uniformly distributed in $[0, 1)$, called “pseudo-random” numbers. The goal of this homework is to build and use two famous randomized algorithms.

2 Pseudo-Random Number Generators

Let’s start with the beginning: How to generate pseudo-random numbers? This is one common way: Choose a starting point s_0 (called the “seed”) and apply some function F to it:

$$s_{i+1} \leftarrow F(s_i) \tag{1}$$

Note that the sequence (s_i) is *completely deterministic* and *will be the same everytime* we compute it, unless we change s_0. Further note that this sequence is *completely predictable*. But if F is well chosen, then the s_i are distributed uniformly over $[0, 1)$.

The program of Appendix A shows how to generate pseudo-random numbers in the range $[0, 1)$, using for F a standard PRNG algorithm called Mersenne Twister. Test it and make sure that you always get the same result. Then change the seed to the value `314159` — we will stick to this value for the rest of this work.

Question 1. It is a good idea to use pseudo-random numbers to generate passwords? Why or why not?

Question 2. How would you generate samples t_i that are distributed according to a Gaussian distribution of mean μ and variance σ?

3 Monte-Carlo Integration

In Lab 2 we saw how to approximate an integral using Riemann’s lemma: Using a subdivision of the real line into N pieces, and approaching the integral of our function on these intervals by the area of a rectangle. This works fine for well-behaved, one-dimensional functions. But in practice we often have to deal with multi-dimensional integrals over complicated domains, and Riemann’s method doesn’t scale well.

Question 3. Use Riemann’s approach to compute the integral of this function over $[0, 1] \times [0, 1]$:

$$1_C(x, y) = \begin{cases} 1 & \text{if } x^2 + y^2 \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

(Hint: Use Fubini theorem to integrate over x, then y). Take $N = 1000$ for your computation. How many sums do you have to perform? What is the error vis-à-vis the expected result? (Hint: Make a drawing.)

Question 4. We will now compute the same integral, but Monte-Carlo style: The integral of 1_C is the same thing as the mean of 1_C. Implement the following algorithm:

```
1. $S \leftarrow 0$
2. Repeat $N$ times:
    (a) Choose two numbers $x$ and $y$ uniformly at random in $[0, 1)$
    (b) $S \leftarrow S + 1_C(x, y)$
3. Output $S/N$.
```

Make sure that you understand this algorithm. Take $N = 1000$ for your computation. How many sums do you have to perform? What is the error vis-à-vis the expected result?

Question 5. Use a Monte-Carlo algorithm to compute the volume of a 3-dimensional unit sphere (Hint: $x^2 + y^2 + z^2 = 1$). Compare your result to the mathematical value $4\pi/3$. Compute the (hyper-)volume of a 10-dimensional sphere. Compare your result to the mathematical value $\pi^5/120$.

Question 6. Run your algorithm several times to measure you result's variance. Don't forget to change the seed! (Why?)

3.1 Application: Portfolio evaluation

Consider two stocks A and B, with respective prices $S_A(t)$ and $S_B(t)$. I own α units of A and β units of B, so that by total wealth is $W_t = \alpha S_A(t) + \beta S_B(t)$. I would like to estimate the probability that my portfolio drops by more than 10% at horizon T, i.e.

$$\theta = \mathbb{P}\left(\frac{W_T}{W_0} \leq 0.9\right). \tag{3}$$

If we write L the event that $W_T/W_0 \leq 0.9$, then $\theta = \mathbb{P}(L) = \mathbb{E}[I_L]$ with

$$I_L(S_A, S_B) = \begin{cases} 1 & \text{if } W_T/W_0 \leq 0.9 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Assume that[1]

$$S_A(T) = S_A(0) \exp\left((\mu_A - \sigma_A^2/2)T + \sigma_A B_1(T)\right)$$
$$S_B(T) = S_B(0) \exp\left((\mu_B - \sigma_B^2/2)T + \sigma_B B_2(T)\right)$$

where $\mu_A, \mu_B, \sigma_A, \sigma_B$ are parameters, and $B_1(T), B_2(T) \sim \frac{1}{\sqrt{T}}\mathcal{N}(0, 1)$.

Question 7. Estimate θ using a Monte-Carlo algorithm. Test your code with $T = 0.5$ years, $\mu_A = 0.15$, $\mu_B = 0.12$, $\sigma_A = 0.2$, $\sigma_B = 0.18$, $S_A(0) = \$100$, $S_B(0) = \$75$, and $\alpha = \beta = 100$. (Hint: You can use the code provided in Appendix B to generate B_1 and B_2).

4 Markov Chains

A Markov chain is a process whose future only depends on the present (and not the past). Markov chains are used in finance and economics to model a variety of different phenomena, including asset prices and market crashes.

In this section we use Markov chains for bond credit risk modelling, using real-world data from *Standard & Poor's* (January 2001). Over time, bonds are liable to move from one rating category to another (AAA being the best). This is sometimes referred to as "credit ratings migration". Rating agencies produce from historical data a ratings transition matrix such as Table 1. This table show the probabiltiy of a bond moving from one rating to another during a certain period of time.

In Table 1, the current rating is given by the row: For instance, the probability to become AAA next year, knowing that I am CCC now, is 0.0015.

[1]We say that S_A (resp. S_B) follows a "Geometric Brownian Motion" of drift μ_A (resp. μ_B) and volatility σ_A (resp. σ_B).

Rating	AAA	AA	A	BBB	BB	B	CCC	Default
AAA	0.9366	0.0583	0.0040	0.0009	0.0002	0	0	0
AA	0.0066	0.9172	0.0694	0.0049	0.0006	0.0009	0.0002	0.0002
A	0.0007	0.0225	0.9176	0.0518	0.0049	0.0020	0.0001	0.0004
BBB	0.0003	0.0026	0.0483	0.8924	0.0444	0.0081	0.0016	0.0023
BB	0.0003	0.0006	0.0044	0.0666	0.8323	0.0746	0.0105	0.0107
B	0	0.0010	0.0032	0.0046	0.0572	0.8362	0.0384	0.0594
CCC	0.0015	0	0.0029	0.0088	0.0191	0.1028	0.6123	0.2526
Default	0	0	0	0	0	0	0	1.0000

Table 1: One-year rating transition probabilities.

Question 9. Write a program that simulates the possible evolution of an AAA-rated bond over 10 years. What is the probability that it gets downgraded to CCC in 10 years time? What is the probability that the bond remains AAA-rated for 10 consecutive years?

A Hint for Part 1

```
#include <random>
#include <iostream>

int main() {
  int seed = 42;                                   // Seed
  std::mt19937 gen(seed);                          // Mersenne Twister
  std::uniform_real_distribution<> unif(0, 1);     // Uniform distribution

  for (int n = 0; n < 10; ++n) {
    std::cout << unif(gen) << ' ';
  }
  std::cout << std::endl;
}
```

B Hint for Part 2

```
#include <random>
#include <iostream>

int main() {
  int seed = 42;                                   // Seed
  std::mt19937 gen(seed);                          // Mersenne Twister
  std::normal_distribution<> d(0,1);               // Normal distribution

  std::cout << "Here is a number: " << d(gen) << std::endl;
}
```