Random Number Generation and Simulation

Exercise 8

Area estimation using Monte Carlo method

Author
Cesare De Cal

Professor
Annie Cuyt

Assistant Professor
Ferre Knaepkens

December 5, 2019

1 Introduction

The exercise asks to approximate the area of the figure defined by

$$\begin{cases} 1 \leq x \leq 3 \\ -1 \leq y \leq 4 \\ x^3 + y^3 \leq 29 \\ y \geq e^x - 2 \end{cases}$$

using the Monte Carlo method, which in Mathematics means solving a problem using random numbers.

2 Tools

To solve this exercise, I've used C and the following libraries:

- C
- C Math Library
- Intel Math Kernel Library (more specifically, the Vector Statistical Library)
- OpenMP

I've used the following Intel MKL routines:

- `vslNewStream(&stream, brng, seed)`
- `vslLeapfrogStream(stream, k, nstreams)`
- `vsRngUniform(method, stream, nrRandomNumbers, array, start, end)`
- `vslDeleteStream(&streamToDelete)`

Even though it wasn't required for this exercise, I've used OpenMP to support multi-threading and make the computation more efficient. I've used the following OpenMP methods and procedures:

- `omp_get_max_threads()`
- `#pragma omp parallel private(nrOfThreads, threadID)`
- `omp_get_thread_num()`

To get started with the exercise, I have used the template file `template.c` provided on the website. To make the code more clear, I've also wrote my own function `isInsideArea(x,y)` which checks if a given pair of coordinates (x, y) is inside the area drawn by the system of inequalities.

To compile and run the code on my own machine with `gcc`, I created a Makefile based on the compiler options and link line specified here:
`https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor`.

3 Computation

On a high level picture, the Monte Carlo method in this exercise works by generating random (x, y) coordinates in the rectangle formed by

$$\begin{cases} 1 \leq x \leq 3 \\ -1 \leq y \leq 4 \end{cases}$$

And then checks if the points satisfy the following inequalities using the `isInsideArea(x,y)` function I wrote.

$$\begin{cases} x^3 + y^3 \leq 29 \\ y \geq e^x - 2 \end{cases}$$

I define two constants `LOOPS` and `N`, which represent the number of iterations to compute and the numbers of points to generate for each iteration respectively. There's also a seed used to generate random numbers.

The area is given by the ratio of the points inside the system to all the random points `N`. This process is repeated `LOOPS` times, and in each iteration an area is calculated. At the end of the iterations, I calculate the average of all the areas calculated inside the loops.

To make this happen, I first initialize the threads using the `VSL_BRNG_MCG59` random number generator. Then I create two independent streams of uniformly distributed random numbers with `vdRngUniform` in the $(1, 3)$ and $(-1, 4)$ intervals for x and y respectively. If the points are inside the figure, I increase a local variable which keeps count of the points inside the figure. At the end of the iteration, I calculate the area by dividing the count by `LOOPS`.

As mentioned before, the average is taken out of all the areas calculated in the iterations. The result is $7.581675111111076e-02$ with `N` and `LOOPS` equal to 30000 and seed equal to -87654321.

The area of the rectangle is defined by

$$A_{\text{rectangle}} = \text{base} \times \text{height} = 2 \times 5 = 10$$

The area of the figure is then

$$A_{\text{figure}} = A_{\text{rectangle}} \times \frac{\text{number of points inside object}}{\text{total number of points in rectangle}}$$

$$A_{\text{figure}} = 10 \times 7.581675111111076e-02 = 7.581675111111076e-01$$

To find the area mathematically, I've used Maple to calcolate the following integral:

$$\int_1^a (\sqrt[3]{29 - x^3} - e^x + 2)dx \approx 7.581218821150386e-01$$

with $x = a$ point of intersection between the two curves. To get the result above I computed $a = 1.593743361313601$ and then performed numerical integration.

4 Plots

This is a sample plot showing the behavior of the Monte Carlo technique.

5 Observations

I can observe that the Monte Carlo method is pretty accurate if we compare the mathematical solution with the results obtained through the Intel MKL library. I can also observe that (at least in this experiment) reasonable precision is attained with only a moderate number of random numbers (1000) that with 1000 iterations leads to an average area of $7.692400000000001e-02$.