CS4540 - Operating Systems
Assignment #4

Introduction

[… A] *simulation* is a fictitious representation of reality, a *Monte Carlo method* is a technique that can be used to solve a mathematical or statistical problem, and a *Monte Carlo simulation* uses *repeated* sampling to determine the properties of some phenomenon (or behavior). Examples:

- *Simulation*: Drawing *one* pseudo-random uniform variable from the interval [0, 1] can be used to simulate the tossing of a coin: If the value is less than or equal to 0.50 designate the outcome as heads, but if the value is greater than 0.50 designate the outcome as tails. This is a simulation, but not a Monte Carlo simulation.
- *Monte Carlo method*: Pouring out a box of coins on a table, and then computing the ratio of coins that land heads versus tails is a Monte Carlo method of determining the behavior of repeated coin tosses, but it is not a simulation.
- *Monte Carlo simulation*: Drawing *a large number* of pseudo-random uniform variables from the interval [0, 1], and assigning values less than or equal to 0.50 as heads and greater than 0.50 as tails, is a *Monte Carlo simulation* of the behavior of repeatedly tossing a coin.

[http://en.wikipedia.org/wiki/Monte_Carlo_method]

A Monte Carlo simulation can be used as an interesting way of evaluating integrals, especially when obtaining a closed-form solution is infeasible.

The algorithm to estimate the value of the integral $\int_a^b f(x)\,dx$ using a Monte Carlo simulation works as follows:

1) Given the interval [a, b], determine the maximum value f_{max} that f(x) assumes within this interval. In Figure 1, f_{max} = f(b).
2) Generate a series of random points (x, y) that fall within the rectangle. In other words, x must be within the interval [a, b], and y must be within the interval [0, f_{max}]. Some of these random points will fall below the f(x) curve (green dots in Figure 1), and some will fall above the f(x) curve (red dots).
3) The value of the integral is approximated by the total area of the rectangle multiplied by the fraction of the number of points that fall under the f(x) curve.

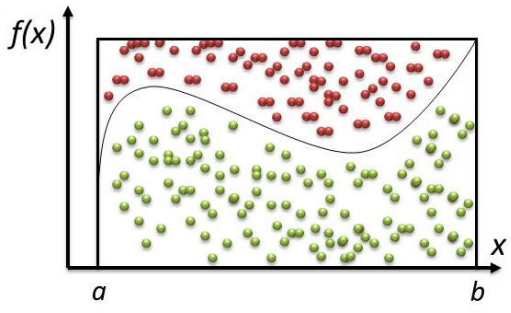

Figure 1: The basic idea of Monte Carlo integration.

4) The greater the number of random points used in this Monte Carlo simulation the more accurate is the estimated integral value.

Problem Specification

Implement a program using Monte Carlo simulation for estimating the following integral:

$$\int_5^{12} x^2\,dx$$

The program must use ten (10) threads, where each thread generates 10,000 random points.

Show the estimates of the value of the integral as calculated individually by each thread, as well as the estimate obtained by the parent process using information it gets from all threads. Make sure the above results are displayed in a nice, clear form.

Extra Credit (10%)

We stated above the claim that *the greater the number of points the more accurate is the estimated integral value*.

Run experiments to support or reject the claim. Do not exceed 1 million points per thread.

Use one or more appropriate tables to store individual and "collective" estimates. Draw one or more diagrams showing *accuracy* of the estimate as a function of the total number of points. (You can calculate the exact value of the integral; then *accuracy* is simply the deviation of the estimate from this exact value.)

Provide a brief discussion with your arguments supporting or rejecting the claim.

SLC Report Requirements

For each program write a full SOFTWARE LIFE CYCLE (SLC) report (analogous to the SLC report presented in class).

Please note that your reporting job is easier due to the following simplifications:

1) If appropriate, the PROBLEM SPECIFICATION section (Step 1) might include just a copy of the given problem(s).
2) The PROGRAM STRUCTURE DESIGN section (Step 2) will have a few modules to name (Substep *2.1. Modules and Their Basic Structure*), and few modules to provide pseudocode for (Substep *2.2.Pseudocode for the Modules*).
3) The sections for RISK ANALYSIS (Step 3), VERIFICATION (Step 4), REFINING THE PROGRAM (Step 7), PRODUCTION (Step 8) and MAINTENANCE (Step 9) can include just 1-2 sentences (analogous in the SLC report presented in class).
4) The CODING section (Step 5) should have the appropriate number of code refinement levels.
5) Due to the simple structures of the programs, the TESTING section (Step 6) should be rather easy.
6) Using a mono-spaced fonts like (`Courier New`) for source code adds a significant value for code readability and quality. Please use this type of font only for the source code in your report.

Coding, Running and Submission Requirements

1) Follow *C Code Style Guide* and the proper programming style it requires, including comments, blank lines, indentations, spaces, etc.
2) All programs must be compiled and executed on Ubuntu running within the Oracle VirtualBox.
3) Remember about *Assignment Submission Instructions* (guidelines), to be followed for each program of this assignment.
4) In addition to what *Assignment Submission Instructions* require, provide a *makefile*, and, if

needed a *README* file explaining how to compile and run your program.

Submission Checklist

For each program, you need to submit:

1) the SLC report (including 9 SLC steps, with as many pseudocode and code refinements as needed);
2) *makefile*, and, if needed a README file explaining how to compile and run the program;
3) the files created by the *script* command (including program output to the terminal, if any);
4) the output files (if any in addition to the output to the terminal);
5) if you have worked on the extra-credit portion of this assignment, provide a *separate file* with these extra results.

Submit your complete assignment package (all files) via Elearning as a zip file. Use `hw<number>_<LastName>.zip` as the format for the name of the zipped file (e.g., `hw1_Smith.zip`)

-------- Good luck! -----