**CX 4010 / CSE 6010**
**Assignment 7**
**Monte Carlo Integration Using OpenMP**
Note that this is an individual, not a team, assignment

**Due Date: 11:59pm on Saturday, November 14**
(extended by 2 days in consideration of the revised date for Assignment 6)
(48-hour grace period through Monday, November 16)
Submit a single zipfile as described herein to Canvas

In this assignment, you will use Monte Carlo integration to evaluate a definite integral, with parallelization using multiple threads under OpenMP.

Monte Carlo integration is a way of calculating an integral by evaluating randomly selected points in a domain to estimate the area under a curve. A typical example is the calculation of $\pi$. Consider a square with corners at (0, 0), (0,1), (1,1), and (1,0) with a quarter of a circle inscribed centered at the origin with radius 1. For a sufficiently large number of points randomly generated within the square, the fraction of points within the quarter-circle should be the same as the ratio of the area of the quarter-circle to the area of the square. In this case, the square will have side length of 1 so its area will be 1, and the quarter-circle will have radius 1 and thus area $\pi/4$. So the fraction of points within the circle should be $\pi/4$, and we can obtain an estimate of $\pi$ by calculating

$$4 * \frac{number\ of\ points\ within\ the\ circle}{total\ number\ of\ points\ within\ the\ square}.$$

In this assignment, you will use a similar approach to estimate the value of $e$. It can be shown (by substitution) that

$$\int_0^1 2xe^{x^2} dx = e - 1.$$

Thus, if we estimate the value of the integral, we can add 1 to that result to obtain an estimate of $e$.

To estimate the integral, you will generate random x- and y-coordinates of $N$ points within a box bounded below by the x-axis and evaluate whether each y-coordinate is below the function evaluated at the corresponding x-value (meaning it is a point under the curve). You should count the number of points that satisfy this condition. Here you will use a box that is bounded by the y-axis and the line x=1 on the sides, by the x-axis below, and by a horizontal line above at a value $yscale$ of your choosing. You should select a value for $yscale$ large enough to be greater than $2xe^{x^2}$ for all values of $x$ between 0 and 1, but not unnecessarily large.

Once you have a count of the number of points, compute an approximation of the integral as

$$\frac{number\ of\ points\ under\ the\ curve}{total\ number\ of\ points} * yscale,$$

where it is necessary to multiply by your selected upper limit $yscale$ to account for the fact that it is not a unit square in this case.

Finally, because the integral evaluates to $e - 1$, you should add 1 to your final approximation of the definite integral.

**OpenMP component:** Monte Carlo algorithms parallelize easily; you can distribute the workload across multiple threads because each thread can independently generate x- and y-coordinates of points within the box for testing. At the end, you will need to calculate the total number of points (across all threads) that are under the curve.

You should test your curve for at least 3 different values of $N$ for a single thread (serial version) and for at least 4 threads (you are encouraged to test using the maximum number of threads possible). The three values of $N$ should be chosen such that they are described by $N_0$, 10 $N_0$, and 100 $N_0$. You should calculate the time to run for the serial and parallel versions for each of the three values of $N$. For the largest value 100 $N_0$, the serial version of the program should take roughly 10 seconds (say, 5-20 seconds) to run.

**Output:** You should compare your result with the value of $e$, which you can calculate as exp(1). However, you should not use this value for any other purpose in your program. You should output results ***to the screen*** for each serial/parallel run and for each value of $N$. In particular, you should list the following.

- Number of threads
- $N$
- Approximation of $e$
- Either the calculated version of $e$ using exp(1), for comparison with the approximation, or the difference (your approximation – exp(1)), appropriately labeled
- The time for the calculation

**Use of PACE ICE:** Although you may choose to perform your main development elsewhere (OpenMP is pretty widely supported), for this assignment we will require that your code compile and run correctly (including the required timing) on PACE-ICE. We highly recommend that you verify that your program runs on this platform and that you verify your ability to log in and copy files back and forth as soon as possible.

**Submission information**

You should submit to Canvas a single zipfile that is named according to your Georgia Tech login—the part that precedes @gatech.edu in your GT email address. To receive full credit, your code must be well structured and documented so that it is easy to understand. Be sure to include comments that explain your code statements and structure.

The zipfile should include the following files:

(1) your code, with the main function in a file called montecarlo.c (submit all files; for this assignment, you may choose to write all code in a single file). If you use more than one file and are using linux or Mac OS, we recommend you use a makefile to compile and run your program, and you should include it if so.

(2) a README text file (not formatted in a word processor, for example) that includes the compiler and operating system you used for compiling and running your code, instructions on how to compile and run your program, and a note that you have verified your code runs on the cluster.

(3) a series of slides composed in PowerPoint or similar software, saved either in PowerPoint or as a PDF and named slides.pptx or slides.pdf, in the following order:

- 1 slide: your name and a brief explanation of how you developed/structured your program. This should not be a recitation of material included in this assignment document but should focus on the main structural and functional elements of your program (e.g., the purpose of any loops you used, the purpose of any if statements you used to change the flow of the program, the purpose of any functions you created, etc.). In other words, under the assumption that the mathematics behind what you are doing is already known, what were the main things you did to translate those requirements into code?
- 1 slide: a description of your testing procedure and evidence of correct operation. Explain why you think your program is correct from these tests.
- 1 slide: a summary and interpretation of your results. Assess whether the runtime of your program scaled with the number of threads, and indicate whether you observed a dependence on the selected value of $N$. Also give your opinion about the usefulness of a Monte Carlo approach for integration.