

PHYS 511: Computational Modeling and Simulation - Fall 2016

Assignment #5, due Friday November 25, before class

Introduction to parallel computing with OpenMP

1. Take the program written for assignment #3, which solved the stationary heat equation, and parallelize it with OpenMP. You can take either your own code (if it was ok and you received a high score for that assignment) or you can simply use the code posted on the class webpage as a possible solution example. Also, change the stopping criterion from 0.00001°C to just 0.001°C as here we are interested in parallelization itself rather than generating converged results. Name this new parallel program `as05a.f90`. Run the program first with one thread (use can do it by typing `export OMP_NUM_THREADS=1` in the terminal before running the program) and then with the number of threads equal to the number of cores in your computer `export OMP_NUM_THREADS=2` (depending on the CPU model in your computer you may be able to use 2, 4, or even more threads efficiently). Measure time by preceding the execution with command `time`, e.g.

```
time ./myprog
```

This will execute your program and at the same time print the total execution time on the screen when the program finishes. For cleanliness of the experiment make sure you do not load your computer with other tasks (such as heavy internet browsing – a browser may take CPU resources even if running in the background) when the program is executed. You might also repeat the execution a couple of times and make sure the timings are consistent (they might be scattered a little bit – that is ok).

Do you see any speed up when you use more threads? What is the speed up factor? It is very likely that you will get only a fraction of the perfect speed up factor (e.g. 4.0 if you use four threads) you might have naively expected. Report your timings and comment on what you think might have been the reason for not-so-perfect speed up.

For the curious ones, you can run the program with the number of threads that exceeds the number of logical cores in your CPU. See what happens.

2. Write a Fortran (or C) program that evaluates the following 5D integral

$$I = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x_1, x_2, x_3, x_4, x_5) dx_1 dx_2 dx_3 dx_4 dx_5,$$

where the integrand is given by

$$f(x_1, x_2, x_3, x_4, x_5) = \arctan \left(\sqrt{1 + x_1^2 + 2x_2^2 + 3x_3^2 + 4x_4^2 + 5x_5^2} \right) e^{-x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_5^2}.$$

To do so, use the rectangle method. That is split the total integration interval into small 5D rectangles (rectangular boxes) and sum the function values evaluated in the center of each rectangular box by the 5D volume of this small box. Use a grid of $n = 50$ points along each direction (x_1, x_2, x_3, x_4 , and x_5). The integrand vanishes at infinity, so you can replace the limits of the integration with some finite values. Make a good educated guess what would be a reasonable (semi-optimal) value for those limits. The denser the points the higher is the accuracy of the rectangle method (smaller integration error). At the same time, if the finite integration limits are too small it may introduce a significant error, too.

Name this program `as05b.f90`, parallelize it with OpenMP and measure the execution time when running it with a single and multiple threads (just like you did in task 1). Make observations, report your timings, and give comments. Also, report the value of the integral. Make sure the result is sensible and does not change with the number of threads used (apart from a possible small rounding error). Upload all relevant files in directory `as05` in your google drive directory shared with the instructor.