Homework 2

Abhijit Chowdhary

March 13th 2019

Before we begin, this was all ran on a Intel i7-3770 which has speed 3.40GHz, but can turbo up to 3.90GHz. In addition, it has 4 cores and 8 threads to use.

Problem 2

See the files labeled `MMulBlockXX.txt` in the `blocktimings` folder to see the timing for block size `XX` under the parallelized code. In the end, I found that a size of 8 actually was the best for me, since when choosing a block of size 8, given my Cache size, it would fit into it exactly.

Now regarding the order of the loops, I chose them in order to minimize jumping across indices in a, b and c. The index most relevant to that is j, therefore I kept it in the outermost loops, and then aftewards in p, so then I moved that to the second outermost. I was able to see a noticable speedup in changing the order (roughly a factor of $.2s$ before parallelization).

Problem 4

See the folder `laplacetimings` for the timings for $N = 100, 200, \ldots, 700$, and with fixed number of iterations 5000. We experimented with different number of threads, and noticed while there was clear improvement when going from serial to doublly parallelized, we see no where near the scaling we would expect when going to 8 cores. This might be due to the fact that we haven't properly optimized the splitting of work along each core, for more than two core.

I theorize that I could probably speed this up a lot, if I create a new parallel region per largish for loop I encountered. Ideally, I would construct a parallel regions before the iteration counter, then for each for loop, I would just use a `#pragma omp for ~~`. I would have to use barriers to prevent each thread from moving along and making mistakes in the residual computation, but when i did such a thing, I didn't really get very much more performance.