Spring 2015: Advanced Topics in Numerical Analysis: High Performance Computing
Assignment 2 (due April 13, 2015)

Handing in your homework: Please hand in your homework as described in Assignment #1, i.e., by creating a git repository containing a Makefile, and by sending me the path to that repo. The git repository `https://github.com/NYU-HPC15/homework3.git` contains the examples needed for this homework. For an overview over OpenMP, you can either use the material I pointed to in class or the official documentation for the OpenMP Standard 4.0.[1]

1. **Finding OpenMP bugs.** The above repository contains five OpenMP problems that contain bugs. Try to find these bugs and fix them. Add a short comment to the code describing what was wrong and how you fixed the problem. Add the solutions to your repository using the naming convention `omp_solved{2,...}.c`, and provide a Makefile to compile the fixed example problems.

2. **OpenMP version of Jacobi/Gauss-Seidel smoothing.** Write an OpenMP version of the Jacobi and Gauss-Seidel smoothers for the problem discussion in Assignment #0. As we have seen, the update for the i-th point in the Gauss-Seidel smoother depends on the updated $(i-1)$-th point. This dependence makes it difficult to parallelize the Gauss-Seidel algorithm. As a remedy, we consider a variant of Gauss-Seidel, which uses *red-black coloring* of the unknowns. In the simple one-dimensional case we target here, this amounts to "color" points x_i with even index i as red and points with odd i as black. Then, each Gauss-Seidel iteration is broken into two sweeps: first, one updates all red and then all the black points (using the already updated red points). The point updates in the red and black sweeps are independent from each other and can easily be parallelized[2].

   - Write OpenMP implementations of the Jacobi and the Gauss-Seidel method, and call them `jacobi-omp.c` and `gs-omp.c`.
   - Run your codes on Stampede. To request one node with shared memory, use `#SBATCH -n 16` and specify the number of OpenMP threads by setting the environment variable `OMP_NUM_THREADS` in the job script before calling the executable. I recommend to use Intel Compilers on Stampede, as discussed in the user guide[3].
   - Report timings you find on Stampede for different numbers of threads.

[1] `http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf`

[2] Coloring schemes to increase the concurrency also exist in higher dimensions, see for instance `http://www.cfm.brown.edu/people/gk/chap7/node18.html`. Depending on the discretization and the dimension of the problem, one might require more than two colors to ensure that updates become independent from each other and allow for parallelism. Efficient coloring for unstructured meshes with as little colors as possible is an interesting research question.

[3] `https://portal.tacc.utexas.edu/user-guides/stampede#appdev-compiling-openmp`