ME 701 – Development of Computer Applications In Mechanical Engineering
Homework 5 – More C++, Fortran, and Python – Due 11/03/2014

Instructions: For this homework, please include a single PDF file with any of the written or plotted deliverables. This will make grading a lot easier.

Problem 1 – Graphical User Interfaces

In class, we are using a pretty straightforward idea—a function evaluator—to motivate several features of GUI's and PyQt. Your job is to expand the in-class exercises in the following ways:

1. During the first day, we developed a function/value/output GUI. Replace the function box with a drop-down box of built-in functions.
2. Extend the GUI to handle array-valued inputs, i.e., allow the user to enter `1, 2, 3` for x. Even better (and, likely, useful) would be to allow users to enter arrays in the form `np.linspace(0, 1, 10)`. Doing so requires a bit more parsing, but it's still pretty straightforward.
3. During the third day, we saw how to plug into matplotlib in order to embed plots in our GUI. Your job is to create a function plotter by combining the results from days 1–3.

I encourage you to use git (+ GitHub) on these to keep on your version control skills (I've haven't reinforced these and some other tools learned early on as well as I had hoped...)

Deliverables: Your well-documented code, the final, self-contained result of days 1–3, and a screen shot of your GUI for $f(x) = sin(x)$ and $x \in [0, \pi]$ (with at least 100 points).

Problem 2 – Shared Memory Parallelism

Use OpenMP in either C++ or Fortran to write a parallel, 2-D midpoint integration routine. Test the scheme with enough points to require about 1 minute to finish with 1 thread. Then, do the same for 2–$2 \times N$ threads, where N is the number of cores on your machine. If you have only a single-core machine, let me know, and I will get you access on a machine with more cores.

Deliverables: Your well-documented code and a table of the the parallel speedups, i.e., t_1/t_n, where t_n is the computational time for n threads.

Problem 3 – Parallelism via MPI

Use MPI in C++, Fortran, or Python to write a 2-D midpoint integration routine. Test the scheme with enough points to require about 1 minute to finish with 1 process. Then, do the same for 2–$2 \times N$ process, where N is the number of cores on your machine. If you have only a single-core machine, let me know, and I will get you access on a machine with more cores.

Deliverables: Your well-documented code and a table of the the parallel speedups, i.e., t_1/t_n, where t_n is the computational time for n threads.