EECS 739: Homework 5

Due: Thursday, May 7, 2015 (At the beginning of lecture)

Important Note: No late homework will be accepted since this is the last day of class and I will be handing out solutions in class.

Answer all of the questions listed below. I will choose one problem to grade; it will be worth 20 points. The submission instructions are identical to those for Homework #1 in regards to how to submit code questions vs. non-code questions.

You may work with at most one other student in the course to formulate ideas only; please write the name of the student with whom you worked (if any) at the top of your homework solutions.

Questions:

1. Let
$$f(x, y) = x^3 - 12xy + 8y^3.$$
Classify the critical points of f. For each minimum or maximum, determine if the extremum is a local or global one. **Justify your answer.**

2. Implement Newton's method in C++. Test your algorithm on the Rosenbrock function given by
$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$
using $(x_0, y_0) = (-1.2, 1)$. Run enough iterations to obtain reasonable convergence. Plot the iterates your method generates on top of a contour plot of f. (You can use Matlab or another mathematical software package to generate the plot.)

3. Implement Newton's method in MPI and C++. Test your algorithm on the extended Rosenbrock function given by
$$f(x_1, x_2, x_3, ..., x_n) = \sum_{i=1}^{n-1} \left[100 \left(x_i^2 - x_{i+1} \right)^2 + (x_i - 1)^2 \right].$$
Test your algorithm with numerous random starting points and record to which critical point (if any) the method converges for the case when $n = 100$. Employ reasonable convergence criteria and a maximum update of $\|\Delta x\|_2 < 10n$ which will help with convergence. In addition, consider the method to have not converged if it does not satisfy your convergence criteria within 20 iterations. Here Δx denotes the difference between two successive iterates. Run your algorithm on $1, 2, 4, 8$, and 16 processors. Report the wall clock time and the speedup obtained in each case. **Note: Be sure to use an efficient linear solver!**

4. Implement Newton's method in CUDA and C++. For this question, you should perform the same tests as in the above question. However, instead of running the method on $1, 2, 4, 8$, and 16 processors, run it with different block sizes and numbers of threads. **It is OK to use block sizes and numbers of threads that divide the problem dimension easily.** Report the wall clock time and the speedup obtained in each case. How do the speedups of the MPI and CUDA implementations compare?