CS 5334/4390 Spring 2015 Name ________________________________
Shirley Moore, Instructor
Lab 6
Due April 29

**Parallel Matrix Operations Using MPI**

For this lab assignment, you will use what you have learned about MPI programming to implement and test parallel MPI versions of functions for performing matrix operations on a distributed memory computer.

1. In class we developed a function for performing matrix-vector multiplication using a row-wise 1D block decomposition of the matrix and vector, and we wrote a driver main program to test it. Please develop a function that implements matrix-vector multiplication using a 2D block decomposition of the matrix and vector and write a driver program to test it. For your function, assume that the data have already been distributed with the vector distributed among the processes in the rightmost column of the process grid. The result vector should be distributed in the same manner. Your driver program should do a serial computation to check that your parallel implementation is giving the correct results. You may assume a square matrix and that the number of processes divides the dimension size evenly. (Hint: You will find it easier to program your function if you use a 2D Cartesian topology for the processes and define sub-communicators for the rows and columns of the process grid).

2. A straightforward implementation of matrix-matrix multiplication using a 2D block decomposition of the matrices A, B, and C (assuming we are multiplying A times B to give C) requires storing all the blocks of a row of A and a column of B on all processors. Canon's matrix multiplication algorithm avoids this excessive storage overhead by shifting the blocks of A and B onto the appropriate processor exactly when they are needed to perform the block multiplication. Please develop a function that implements Canon's algorithm and write a driver program to test it. (Hint: You will find your function easier to program if you use a 2D Cartesian topology for the process grid).

3. Construct a Makefile that will build all of your functions and test programs on Stampede. Run your programs and make sure they work correctly. Please upload your source code files and Makefile to a directory named <yourname>-lab6 in your SVN repository.

4. Required for graduate students/extra credit for undergraduates: Construct an analytical performance model for your parallel systems for problems 1 and 2. Do the iso-efficiency analysis and find an asymptotic upper bound on the number of processes that can be used cost optimally for each problem. Measure the runtime of just the computational portions of your programs for various values of the dimension size and the number of processors and compare the experimental and analytical results. In addition to uploading your code to your SVN repository, please also upload a report analyzing your results.