EECS 739: Homework 3
Due: April 18, 2017 (at 11am)

Questions:

1. (30 points) Consider solving the following 3D boundary value problem given by:

$$\nabla^2 u(x,y,z) = \frac{\partial^2 u}{\partial x^2}(x,y,z) + \frac{\partial^2 u}{\partial y^2}(x,y,z) + \frac{\partial^2 u}{\partial z^2}(x,y,z) = f(x,y,z)$$

for $(x,y,z) \in R = \{(x,y,z) |\ a < x < b, c < y < d, e < z < l\}$, and $u(x,y,z) = g(x,y,z)$ for (x,y,z) on ∂R.

Note this corresponds to solving the 3D version of Poisson's equation on a rectangular prism with Dirichlet boundary conditions.

For this question, show how to discretize the geometric domain and the PDE using the same low-order finite-difference approximations for each partial derivative as we used in lecture for the corresponding 2D problem. Once you apply the finite-difference method, you will end up with a linear system of equations. Give the linear system of equations in matrix form.

For this problem you should first perform the derivation for the abstract problem. Then, you should substitute in the specific parameters shown below in order to yield a matrix and right-hand side vector with numbers in them.

Parameters to use:
Source term :$f(x,y,z) = 0$
BC$'$s :$g(x,y,z) = 1$ on R
Domain $R : a = 0, b = 1; c = 0, d = 1; e = 0, h = 1$
Number of points in each dimension : $m = 5,\ n = 4,\ q = 3$.

Comment on the banded matrix structure which you observe.

2. (30 points) Implement a parallel numerical PDE solver in MPI and C/C++ for use in solving the 3D elliptic boundary value problem given in (1). In particular, your implementation must include a parallel iterative linear solver which is based on the Gauss-Seidel method. **Hint: Consider how the parallel red-black Gauss-Seidel method generalizes from 2D to 3D, and develop a parallel implementation of the 3D generalization.** Your implementation should make use of *nonblocking* MPI commands whenever possible. For example, these commands may include MPI_Isend, MPI_Irecv, MPI_Test, MPI_Wait, etcetera. Specify your choice of p, the number of processors, as it relates to the other parameters.

For this MPI code, it is acceptable to hard code f and g (to avoid writing code to parse functions given as input). However, a, b, c, d, e, l, m, n, and q should be input as parameters. In addition, p should be specified in your PBS script.

Then use your code to solve the problem in (1) with the following parameters:
Source term :$f(x,y,z) = 0$
BC$'$s :$g(x,y,z) = 1$ on R
Domain $R : a = 0, b = 1; c = 0, d = 1; e = 0, h = 1$
Number of points in each dimension : $m = 100, n = 100, q = 100$.

Use the SLURM cluster to run your code. Comment on the convergence of your method.

3. (30 points) Perform strong and weak scaling experiments on the full range of 1-64 processors on the SLURM cluster available through the EECS 739 reservation. For your experiments, you should use the same source term, boundary conditions, and domain as specified above. However, you will need to determine how to intelligently set m, n, and q for the various experiments based on the problem size requirements; this is particularly true for the weak scaling study.

   Hint: Problem size here pertains to the complexity of the computations based on the matrix size. There is a relationship between this and the mesh size. Once you know the desired matrix size, you can compute the desired mesh size. Note here that sparsity will also make the problem size calculations more interesting than those we performed in class for dense matrices. Do NOT simply assume the matrices are dense for the purposes of these calculations. In addition, be sure to apply meaningful convergence criteria across the set of runs (within an experiment).

   Plot the speedups vs. p for the various numbers of processors used in each experiment. Comment on the observed scaling properties of the parallel iterative linear solver.

   Note: Within reason, you will need to work with large matrix problems to obtain good scaling results. However, please be kind to your fellow EECS 739 students and remember that the cluster is a shared resource!