Homework #3

ME 471/571

1. (**Jacobi Method**) Solve the problem

$$u''(x) = -(2\pi)^2 \cos(2\pi x)$$

on $[0,1]$ subject to $u(0) = u(1) = 1$. You will solve this problem on a mesh with N subintervals of width $h = 1/N$. Choosing $N = 2^m$ for integer m, we distribute the mesh to $p = 2^r$ processors, each with $M = N/p = 2^{m-r}$ intervals (or "cells") per processor. On each processor, we can define a subdomain D_k as

$$D_k = [s_k, s_{k+1}], \qquad k = 0, 1, \ldots, p-1$$
$$s_k = k2^{m-r}h$$

We can discretize (1) on either a cell-centered mesh, or a node centered mesh. You will do both implementations for this assignment.

Cell-centered mesh. On each processor domain D_k, we can discretize (1) as

$$\frac{u^k_{j-1} - 2u^k_j - u^k_{j+1}}{h^2} = f(x^k_j), \qquad j = 1, 2, \ldots, M$$

where cells centers $x^k_j = s_k + (j - 0.5)h$ and $u^k_j \approx u(x^k_j)$. For internal domain boundaries, values u^k_{-1} and u^k_{M+1} must be communicated between processors.

On domains D_0 and D_{p-1}, the corresponding discrete equations are supplemented with discretized boundary conditions, given by

$$\frac{u^0_0 + u^0_1}{2} = u(0), \qquad \frac{u^{p-1}_M + u^{p-1}_{M+1}}{2} = u(1)$$

These ghost cell values should be set at each iteration of the Jacobi method.

Node-centered mesh. On each processor domain D_k, we can replace the continuous problem (1) with the discrete approximation

$$\frac{u^k_{j-1} - 2u^k_j - u^k_{j+1}}{h^2} = f(x^k_j), \qquad j = 0, 2, \ldots, M$$

where $x^k_j = s_k + jh$. For internal domain boundaries, values u^k_{-1} and u^k_{M+1} must be communicated between processors. On boundary processors $k = 0$ and $k = p-1$, this equation must be modified to take into account boundary conditions $u^0_0 = u(0)$ and $u^{p-1}_M = u(1)$. In the node based configuration, the solution and equation at node s_k ($k = 1, 2, \ldots, p-1$) are duplicated on processors $k-1$ and k.

Solving the system. The task on this assignment is to solve (1) using the Jacobi iterative method on both the cell-centered and node-centered meshes. For each implementation, use a tolerance $\tau = 1 \times 10^{-10}$, and iterate until the approximate solutions $\mathbf{u}_k$ satisfies

$$\| \mathbf{u}^{k+1} - \mathbf{u}^k \|_\infty < \tau.$$

where $\| \mathbf{v} \|_\infty = \max_i |v_i|$

(a) Convince yourself that for fixed N, you get the essentially identical results on $1, 2, 4, 8, 16$ processors.

(b) Show that you can reproduce the Jacobi results for both the cell centered and node centered mesh shown in Figure 1. Your results should be independent of the processor count.

(c) Verify the accuracy of your results by solving (1) on a mesh with $N = 32, 64, 128, 256, 512, 1024$ subintervals. Plot the errors for each value of N or create a table to show that you are getting second order accuracy.

(d) For the node-centered approximation, prove that your implementation guarantees that $u_M^{k-1} = u_0^k$ at all internal processor boundaries.

Present your results in a Jupyter notebook and provide any explanatory text. Use Pandas to present your results in tables or plots.

2. (**Gauss-Seidel.**) Use the Gauss-Seidel iterative method to solve the problem presented in Problem 1. To parallelize the algorithm, use "red-black" ordering to update solution values. As with problem Problem 1, you will implement the Gauss-Seidel method on both cell-centered and node-centered meshes.

   (a) Show that for fixed N, you get the essentially identical results on $1, 2, 4, 8, 16$ processors.

   (b) Show that you can reproduce the Gauss-Seidel results for both the cell centered and node centered mesh shown in Figure 1. Your results should be independent of the processor count.

   (c) Verify the accuracy of your results by solving (1) on the range of N used in your Jacobi code. You should observe second order accuracy.

3. (**Conjugate Gradient Method**) Implement the Conjugate Gradient (CG) Method on either the cell-centered or node-centered mesh (your choice).

   (a) Because the CG method converges so much faster for this problem than the Jacobi or Gauss-Seidel methods, we can solve much larger problems than are reasonable with the other methods. Demonstrate this by solving for $N = 2^p$, $p = 5, 6, \ldots, 17$. Plot your results, or create a table to show that you are getting second order accuracy.

   (b) Present scaling results on $1, 2, 4, 8, 16$ processors for the largest values of N.

```
------------------------------------------------------------
                    Node-centered mesh
          N = 128; tol = 1.00e-10; kmax = 100000
------------------------------------------------------------
                  Method    Iter. #    |uk-ukp1|        Error
------------------------------------------------------------
                  Jacobi      51284   9.9984e-11   4.0156e-04
       Gauss-Seidel (RB)      24494   9.9952e-11   4.0148e-04
 Conjugate Gradient (CG)         65   2.6868e-15   4.0164e-04

------------------------------------------------------------
                    Cell-centered mesh
          N = 128; tol = 1.00e-10; kmax = 100000
------------------------------------------------------------
                  Method    Iter. #    |uk-ukp1|        Error
------------------------------------------------------------
                  Jacobi      46678   9.9997e-11   3.0119e-04
       Gauss-Seidel (RB)      24491   9.9954e-11   3.0118e-04
 Conjugate Gradient (CG)         65   2.9357e-15   3.0118e-04
```

Figure 1: Sample output from the Jacobi, Gauss-Seidel and conjugate gradient methods. These results are essentially independent of the number of processors used. The error (shown in the last column) is the error in the discrete approximation, i.e. $e(x) = \| U - u(x) \|$, where U is the computed solution and $u(x)$ is the true solution to the problem. This error should satisfy $e(x) \sim O(h^2)$.