Homework #4

ME 471/571

In several of the problems below, you are asked to provide "convincing evidence" that a parallel code is working. Here are some examples of what you might provide.

- Show plots from your serial and parallel results, at the same time value t and show that in the "eyeball norm", they are the same.
- Report the solution to both the serial and parallel runs at a particular location, at a particular time. For example, for the heat equation, you might show the value at a particular (i, j) location after $M = 100$ time steps. For the Forward Euler scheme, the results should be identical, to within machine precision.
- For problems in which you know the true solution, you could plot the errors in both the serial and parallel runs. They should be essentially the same.
- For iterative methods, the number of iterations required to reached a given tolerance on a parallel run should be about the same (within one or two iterations) of those required for the serial code.

1. **Parallel heat solver** Convert your two dimensional serial spiral wave solver to a parallel code. Use the Cartesian communicator to exchange values at parallel boundaries.

   (a) Provide convincing evidence that your parallel code produces the same results as the serial code.

   (b) Run your simulation on 1,2,4,8 and 16 processors and report the strong scaling timing results and efficiency in two separate plots. To get reasonable timing results, make sure that the problem is large enough so that at the higher processor counts, each processor has enough work to do. Also, choose a final time T so that your code on a single processor takes at least a minute.

2. **Poisson Equation (serial)** The Poisson problem can be viewed as a model of a heat diffusion process that has reached an equilibrium state. For this problem, you will solve the two dimensional Poisson problem on a domain $[0, 1] \times [0, 1]$, given by

   $$\nabla^2 u = f$$

   where $f(x, y) = -8\pi^2 \sin(2\pi x) \sin(2\pi y)$, subject to $u = 0$ on the boundary of the domain. The exact solution to this problem is $u(x, y) = \sin(2\pi x) \sin(2\pi y)$.

   Use the Conjugate Gradient algorithm described in class to solve this problem. You may start with the code provided on the course GitHub repository. Specify a tolerance of 1×10^{-12}.

   Report the following

   (a) Create a loglog plot showing how the number of iterations required varies with grid size. Fit your data points to the model

   $$I = CN^{2p} \tag{1}$$

   where I is the iteration count and N is the number of cells on the side of a grid, i.e. each grid is $N \times N$. What value of p do you get?

   (b) Show a plot of your solution u and compare it to the true solution.

   (c) (optional) Provide a loglog plot of the error in your computed solution verses N. To compute the error use a "grid norm" function. For example the grid 1-norm of error is defined as

   $$\|e\|_1 \equiv \|u - u_{exact}\|_1 = \sum_{i,j} |u_{ij} - u(x_i, y_j)| \, \Delta x \Delta y$$

   where $u(x_i, y_j)$ is the exact solution at location (x_i, y_j) and Δx and Δy are the mesh widths in the x and y directions, respectively. Plot a best-fit line through your error and show that the slope is close to -2.

3. **Parallel Conjugate Gradient** Write a parallel version of the Conjugate Gradient algorithm and test it on the Poisson problem from Problem 2. Provide convincing evidence that your parallel implementation is correct.

4. **Spots and Stripes**. Solve the Turing problem using the Backward Euler time stepping scheme. The Turing model is given by

$$\begin{aligned} u_t &= D\delta\nabla^2 u + \alpha u(1-\tau_1 v^2) + v(1-\tau_2 u) \\ v_t &= \delta\nabla^2 v + \beta v\left(1 + \frac{\alpha\tau_1}{\beta}uv\right) + u(\gamma + \tau_2 v) \end{aligned}$$

   where $u(x,y)$ and $v(x,y)$ are initialized to random values in $[-1/2, 1/2]$. The patterns that form depend on the choice of parameters.

   - **Spots:** $\delta = 0.0045$, $D = 0.516$, $\tau_1 = 0.02$, $\tau_2 = 0.2$, $\alpha = 0.899$, $\beta = -0.91$, $\gamma = -\alpha$.
   - **Spots:** $\delta = 0.0021$, $D = 0.516$, $\tau_1 = 3.5$, $\tau_2 = 0$, $\alpha = 0.899$, $\beta = -0.91$, $\gamma = -\alpha$.

   Use no-flux boundary conditions on the boundary of your domain. As a suggestion, you can use a domain $[-1,1] \times [-1,1]$, although this domain size may depend on your choice of parameters. You should run your simulation long enough so that you see the final quasi-steady pattern form.