HW 2: MPI Programming

The assignment is a classic example of an $N$-body particle method. We will simulate $N$ particles moving under the force of gravity according to $\mathbf{F} = m \mathbf{a}$. Particles have equal mass, and are initially either randomly distributed or placed along a circle. At every time step the forces on each particle due to all the other particles is computed, and the velocity and position of each particle is calculated at the new time step. Using the first-order forward Euler method, this looks like

$$
\mathbf{F}^k_j = \sum_{i \neq j} \frac{G m_i m_j}{d_{ij}^2} \mathbf{n}_{ji}
$$

(1)

$$
\mathbf{a}^k_j = \mathbf{F}^k_j / m_j
$$

(2)

$$
\mathbf{v}^{k+1}_j = \mathbf{v}^k_j + \Delta t \cdot \mathbf{a}^k_j
$$

(3)

$$
\mathbf{x}^{k+1}_j = \mathbf{x}^k_j + \Delta t \cdot \mathbf{v}^k_j
$$

(4)

Here the positions, velocities, etc, are vectors in two dimensions; $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is the distance between particles $i$ and $j$; and $\mathbf{n}_{ji} = (x_i - x_j)/d_{ij}$ is the unit normal pointing from particle $j$ to $i$ (i.e. the direction the force is acting in).

We provide an initial working — but inefficient — MPI program. The assignment is to make this more efficient (as described in class), and provide evidence of such by timing both the original and improved version over a range of particles from $N = 100$ to something several orders of magnitude larger. (How high you can go and still run in under 10 minutes depends on your optimizations). Improvements should consist of blocking the communication into fewer number of larger messages, overlapping communication and computation, using asynchronous messages, and any other improvements you can find. (N.B. The numerics are rather rudimentary in this assignment, but that isn’t the point of it).

Your final submissions should include a discussion of the changes you made to the program and the experiments showing performance. You should discuss for what problems sizes parallelization is effective. You should submit results showing speedup curves as well as scaled speedup (where the number of particles per process is held constant, as well as where the work per processor is constant). The initial inefficient code is in the file part.c, and a matlab script to plot results is in plotpart.m.