Laboratory # 4

Goals

The goals of this lab are:

1. To give you experience with a nonlinear code.
2. To give you experience with multiple parallelism in a single code.
3. To give you experience writing your own parallel communication routine from scratch.

Background

This lab revolves around the variable `Qval`, which is a 1-D array in the LaplacianOnGrid class. `Qval[p]` is the number of ions associated with the point/cell in the mesh with the natural node number `p`. It is proportional to the number of particles associated with that point. `Qval` is used in the right-hand side of the equation for voltage. In the `FormLS` routine, the right-hand side vector `b` is populated with it and the previous time-step's value for voltage (`phi`). The lines of code that do that are shown below:

```
rLOOP b[r] = -( -c1*Qval[r] + c2*phi[r] );
```

It is important to remember that the impact of each particle is distributed to the four nodes surrounding it, as shown in the left part of Figure 1. The in-class lecture notes from 2024-09-26 gives the interpolation method for computing `Qval` for each point/cell. Those notes are posted in Canvas.

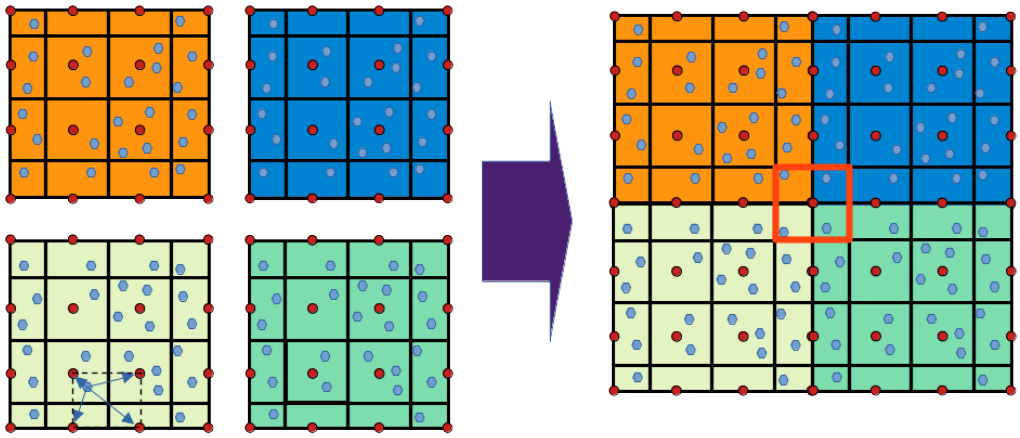

Figure 1: **Left:** The arrows show how each particle is distributed to the four nodes around it. **Right:** The four mashes have nodes in common at their shared boundary. `Qval` must be summed for those nodes. The node outlined in red has four particles associated with it, one on each PE.

The problem with `Qval` is on the process boundary. As shown in the right part of the figure, the four meshes, which are on different PEs, have nodes that overlap at their process boundaries. Each of those nodes on the boundaries have part of their cell in one PE and part on one or more PEs. You will need to sum `Qval` on all nodes on PE boundaries. This will be accomplished in the `PEsum` routine, which will be part of `mpiInfo`.

During Lab: Team Development

1. This entire lab reduces to one problem: Write the routine `void PEsum( VD &field )` in `mpiInfo.h`. It is the only part of the code you must write. Its purpose is to sum `Qval` at the PE boundaries. Study this lab, the code, and the lecture notes and then, as a team, construct a parallel communication plan.
2. Together, implement that plan in `void PEsum`.
3. **Verification Testing:** Before running the application problem (next bullet), modify the code to perform a four PE verification test. In `LaplacianOnGrid`, force each PE to populate `Qval` with 1 on PE0, 2 on PE1, 3 on PE2, and 4 on PE3. i.e., set `Qval[p] = myPE + 1`. Plot the result using `pc_qval` in gnuplot. This verification problem will prove whether or not you are summing values correctly on the process boundaries. Note, carefully, the value at the very center where all four processes touch. It should have the value 10 (1 + 2 + 3 + 4). Include the plot of the verification test in your report.
4. Once you have verified that `Qval` is being summed correctly on the process boundaries, remove your code that artificially sets `Qval` in the previous bullet. Demonstrate that the parallel code works on four processors (2x2) on this application problem, running the case in the `./run` file, in the `./codes/esPIC/src` directory. The `writePlotCmd_py3.py` script is provided in the `src` directory. It writes `pc_qval` and `pc_voltage`, both of which contain gnuplot commands for generating sequential plots, like a movie. Include the last frames from those two plot sequences in your report. For your interest, the `unset key` command is handy in gnuplot for turning off the legend. Also, `set view 0 0` allows for a direct overhead view of the particles.

If you are unable to get the verification problem working, proceed with the debugging techniques described in previous labs.

After Lab: Group Lab Report

For this lab, submit a joint report with your team. In other words, all members of the team submit the same PDF file.

1. In the main body of the report:
   (a) An illustrated description of the parallel design.
   (b) The verification problem results on four processes. If the code did not work, describe the debugging procedure you followed.
   (c) The team's results for the application problem.
2. In Appendix A, include your code for `PEsum`.

Resources

- `https://www.particleincell.com/2010/es-pic-method/`

Grading Rubric

Component	Expectations	Weight
Post-Lab Report		
Main Body	Illustrated parallel design	20%
Main Body	Verification test results or debugging process	20%
Main Body	Application problem	20%
Self-evaluation	One to two paragraphs	10%
Appendix A	Code listing for `PEsum` with in-line comments	20%