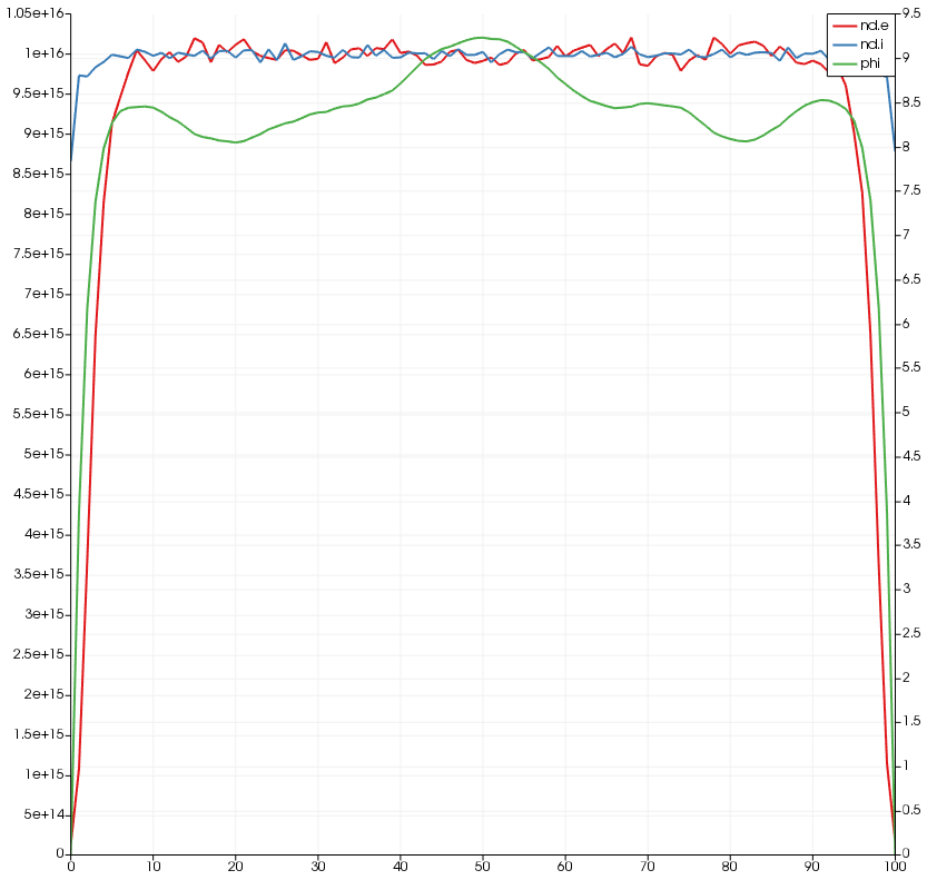
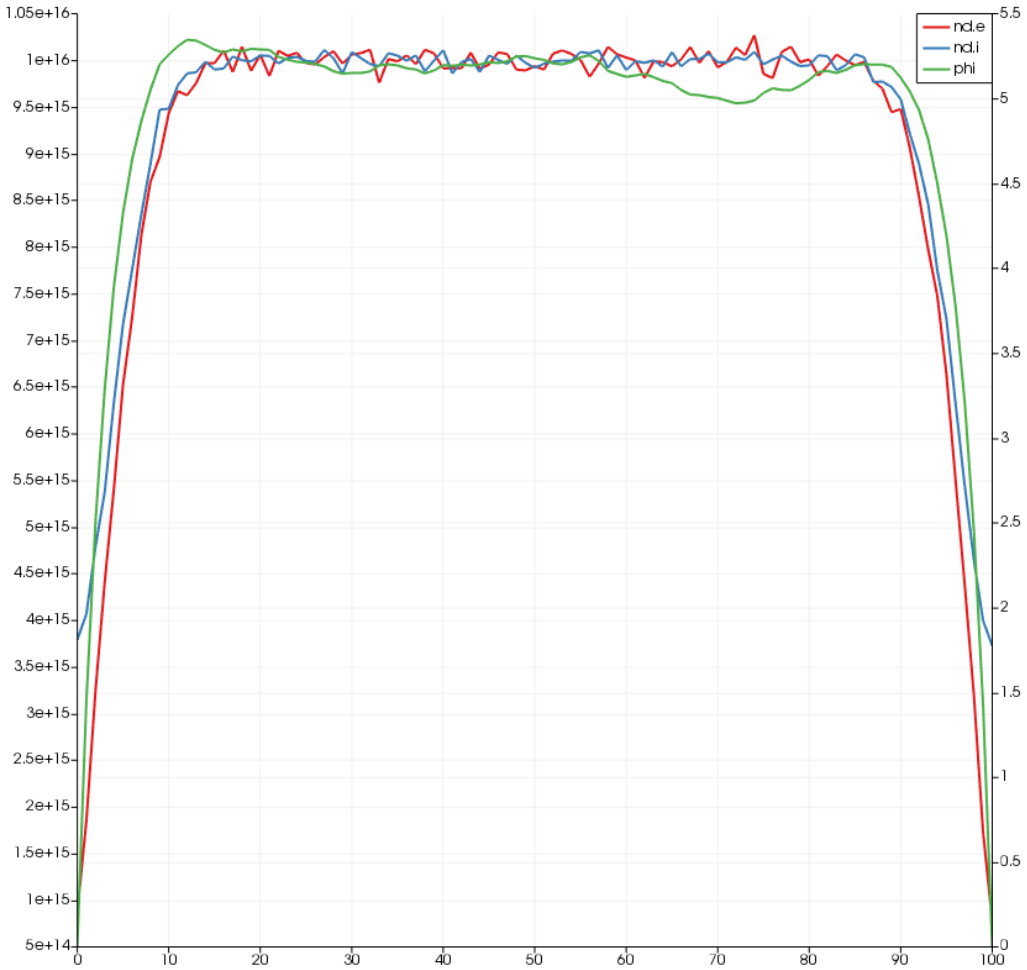
ASTE 404
HW9
Nikita Persikov

Problem 1:

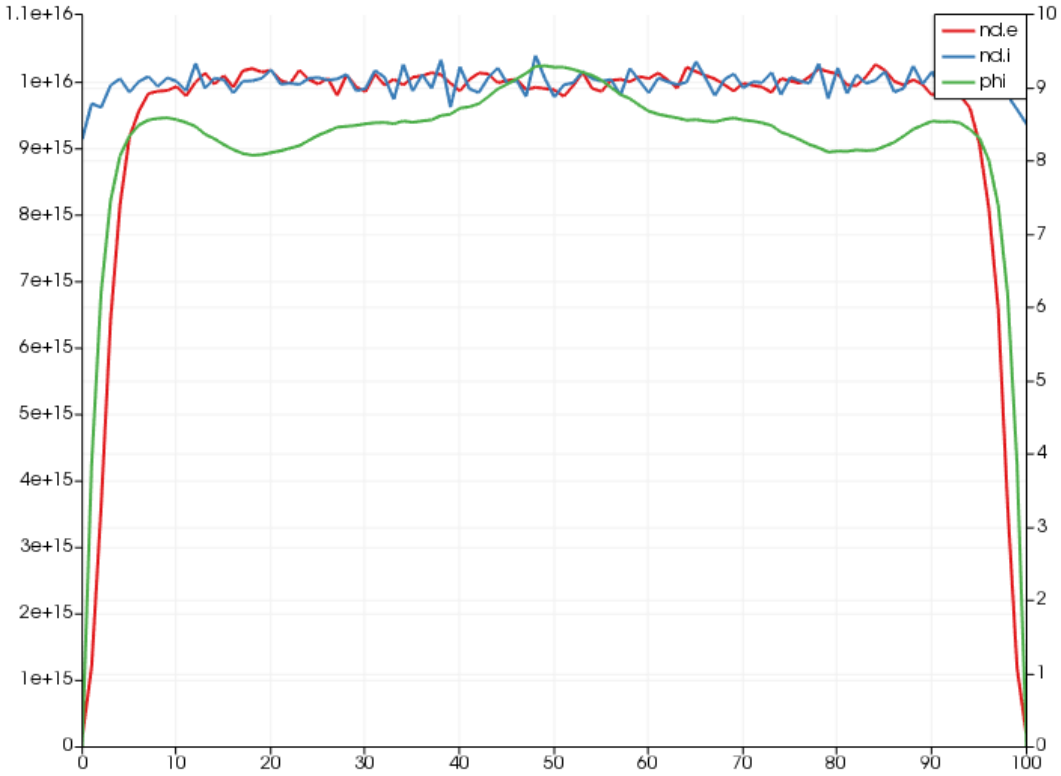
500,000 particles, 1,000 iterations: (time per time step: 42.1221 ms)



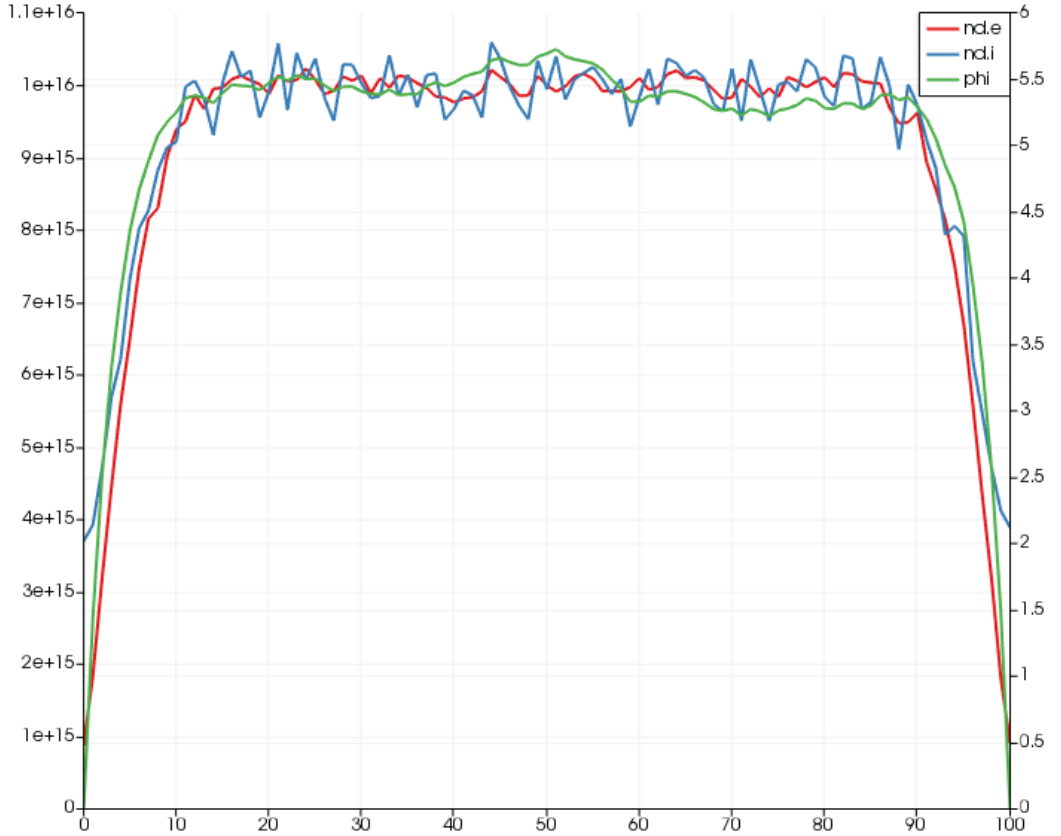
500,000 particles, 10,000 iterations:



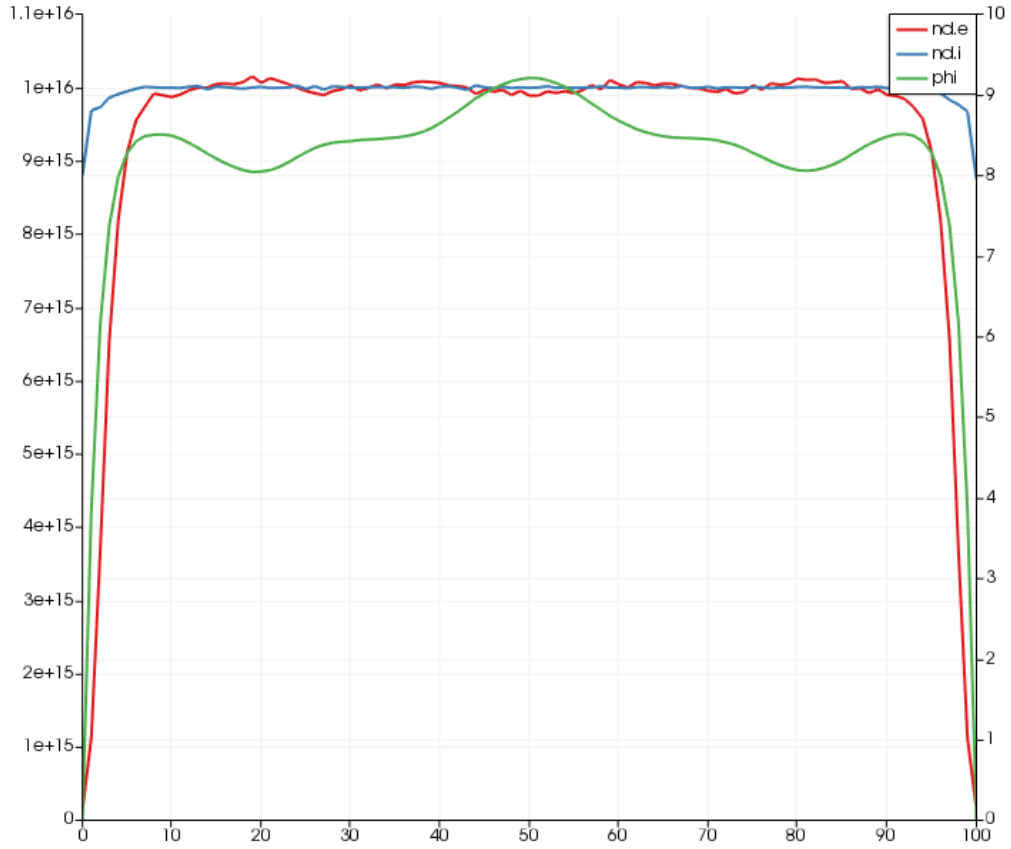
50,000 particles, 1,000 iterations: (time per time step: 23.6692 ms)



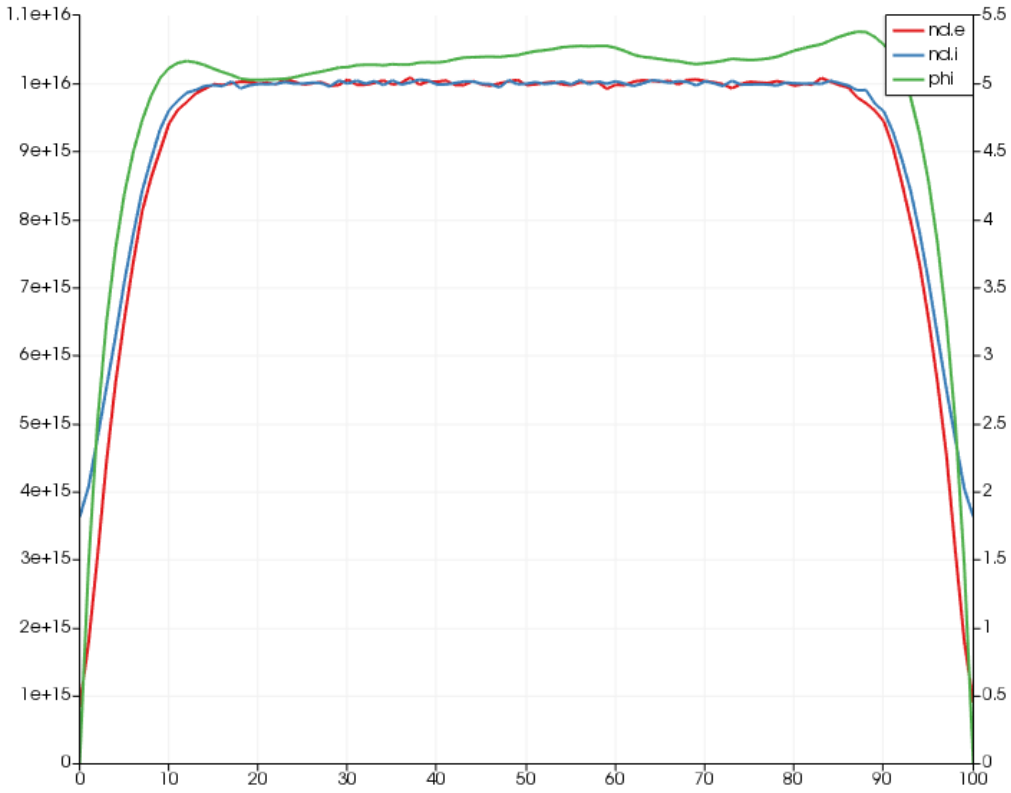
50,000 particles, 10,000 iterations:



5,000,000 particles, 1,000 iterations: (time per time step: 341.068 ms)



5,000,000 particles, 10,000 iterations:



Problem 1 Part 4: The code has a variable for number of ions and another for number of electrons. The values are the same by default, which implies that the ions are assumed to be singly charged.

To implement doubly and triply charged ions, I would do the following:

1. I would have three variables: one for singly, one for doubly, and one for triply charged ions. The number of electrons would be the sum of number of singly charged ions, double the number of doubly charged ions, and triple the number of triply charged ions. The number of particles would just be the sum of the numbers of ions.
2. Another step is to implement an "ions"-like structure for all three types of ions, since the "ions" structure uses the charge of one proton, while the other two types of ions will have twice or thrice the charge. Wherever "ions" is implemented, structures for the other two types of ions should also be used.
3. The loop that loads electrons and ions should load the right number of doubly and triply charged ions as well as singly charged ones.
4. ComputeRho should also use different structures for ions with different charges.

To make sure the mass ratio is true to reality, the mass can be changed in the structure for each ion. All that is needed is a separate struct for each ion where the charge and mass is specified. The correct NUM_IONS would have to be used for each species.

Since hardcoding these structs can be repetitive, all the fields of the Species structs can be set using arrays/Vectors instead of single values. The "part" field would need to become a 2D array or a List/Vector/Map of arrays (a Map may make it easier to find the list of particles for a Species by name). Some of the existing functions that use Species structs can be modified to loop through all values in the variables that used to be single values (for example, ComputeRho should just take in one Species struct, and instead of directly adding "ions" and "electrons" densities, it should loop through all densities in the Species struct and add them together). Functions like addParticle would need to have an argument for species name so the code knows where to add the particle.

Problem 1 Part 5:

Question 1: Why is #define used instead of const and data point to define constants?

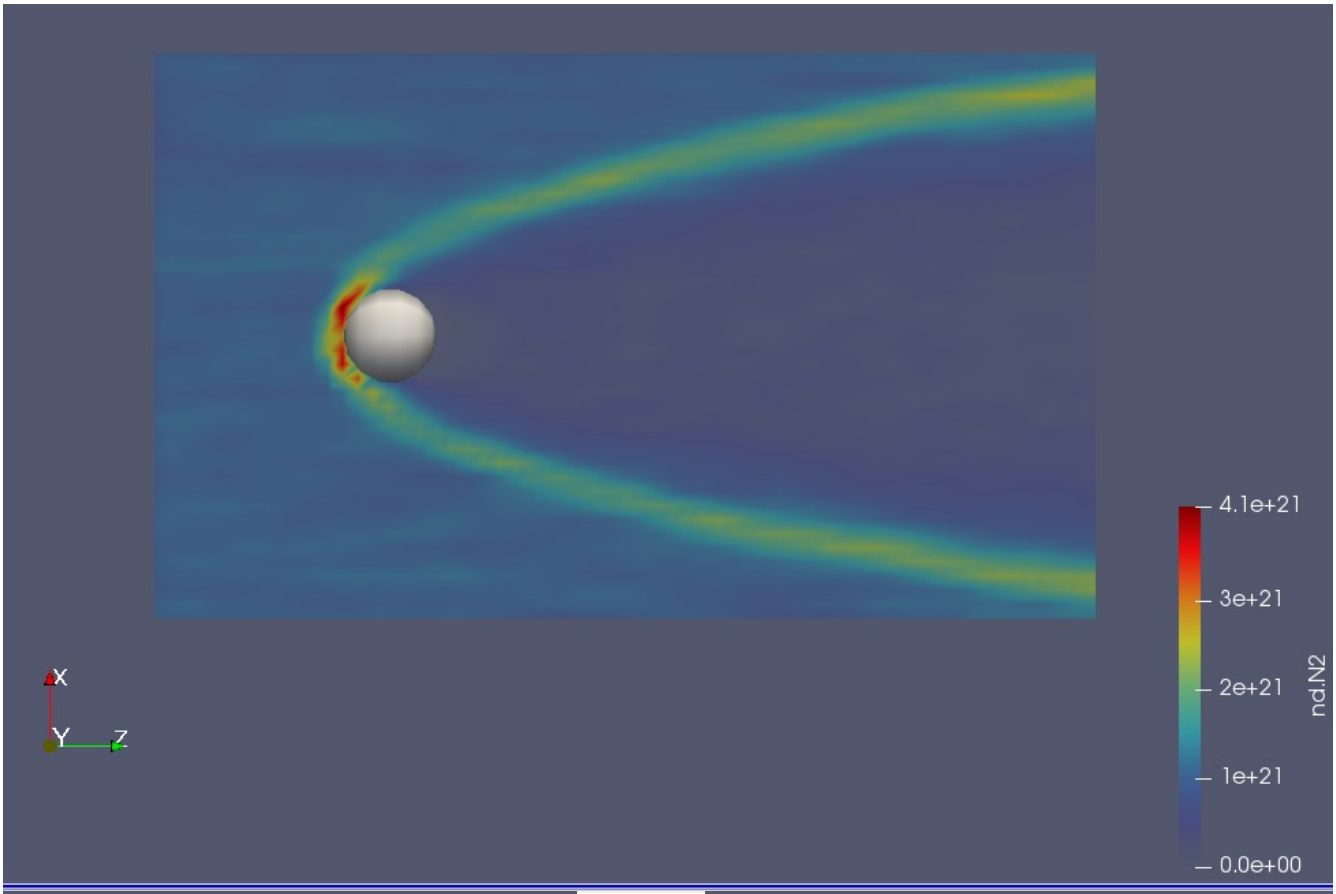
Question 2: Is it realistic to assume that there are no neutral particles in any given plasma, or is plasma usually only ions and electrons?

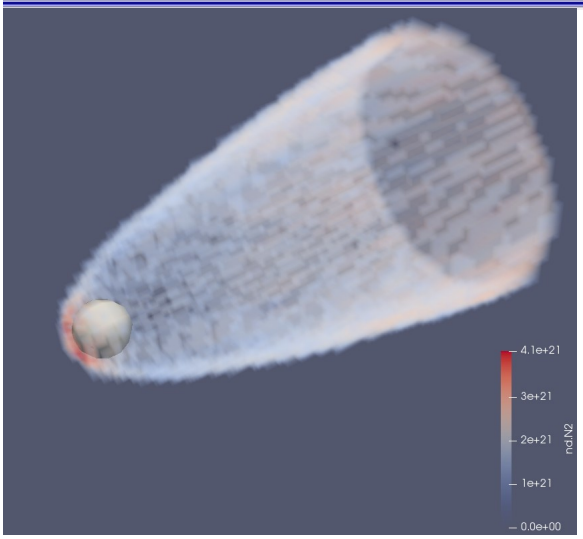
Question 3: Why is recombination not considered? Is it realistic to assume that recombination has negligible effects on the plasma simulation? Shouldn't ions and electrons be combining into a neutral particle at least part of the time? Would that slow down the simulation?

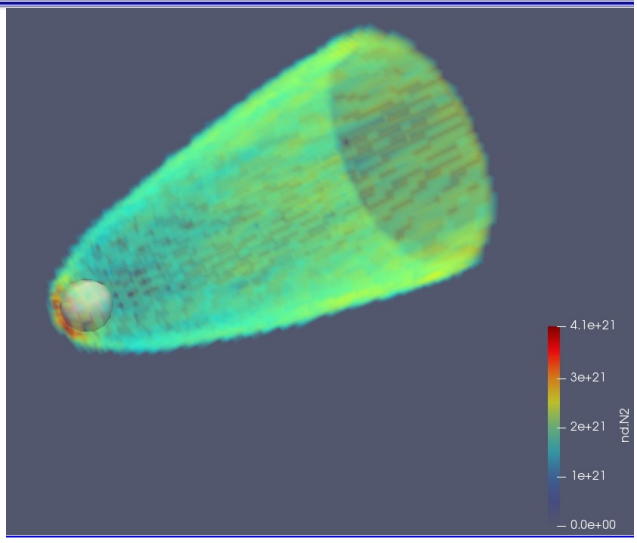
Problem 2:

Parts 1 and 2: Code has been compiled. Results are below:

Run Time:

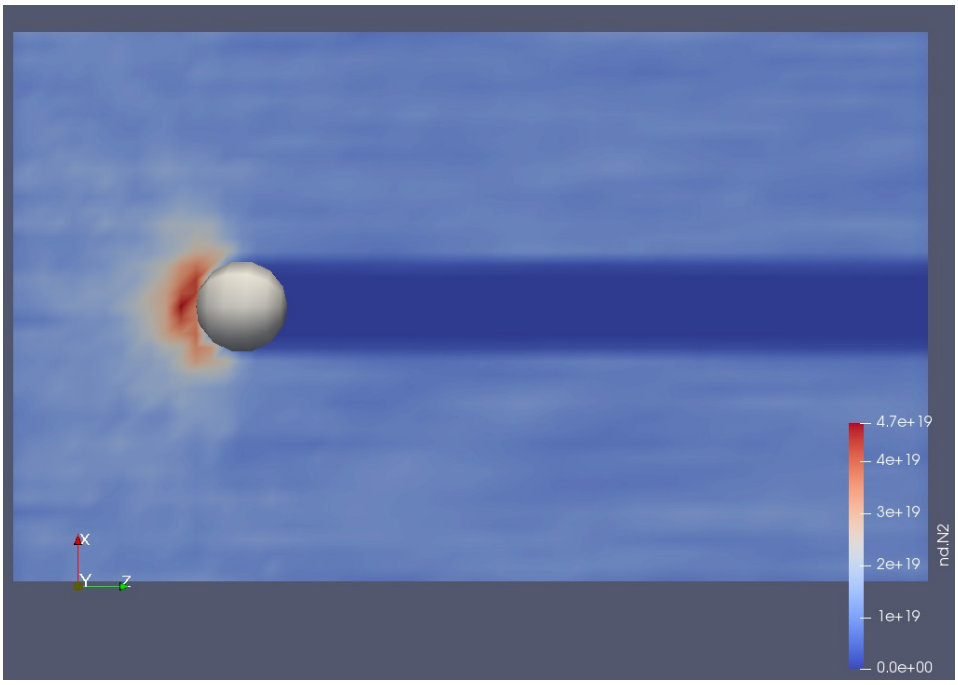

4.1e+21
3e+21
2e+21
1e+21
0.0e+00
nd.N2
X
Y
Z


4.1e+21
3e+21
2e+21
1e+21
0.0e+00
nd.N2


4.1e+21
3e+21
2e+21
1e+21
0.0e+00
nd.N2

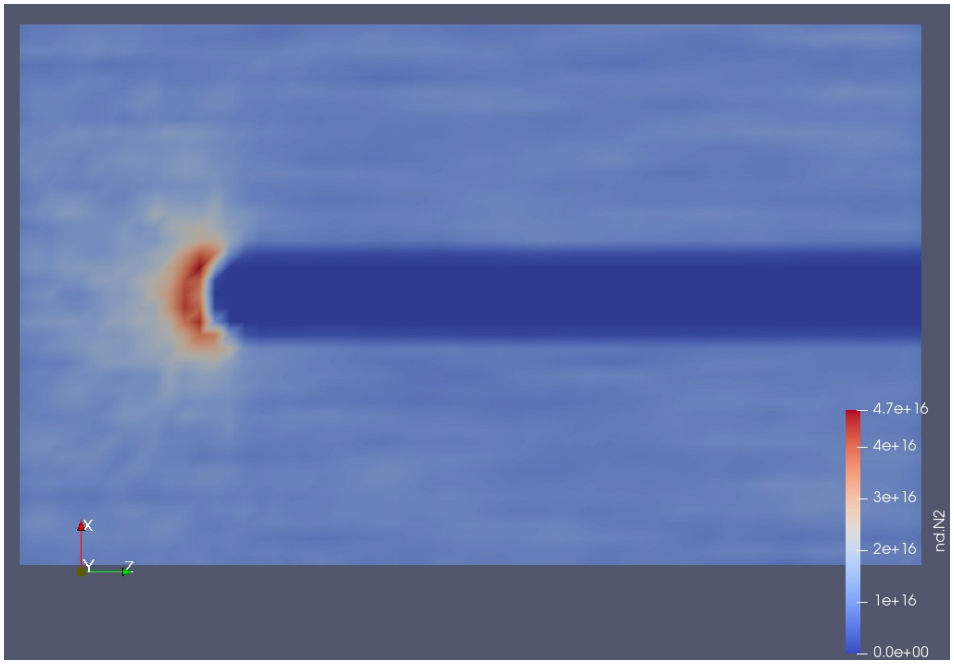
Part 3:

When running the simulation for a number density of 1e19, the plot looks like this:



In this case, the structure of the bow shock is almost completely wrong. It does not seem to capture the parabolic shape at all.

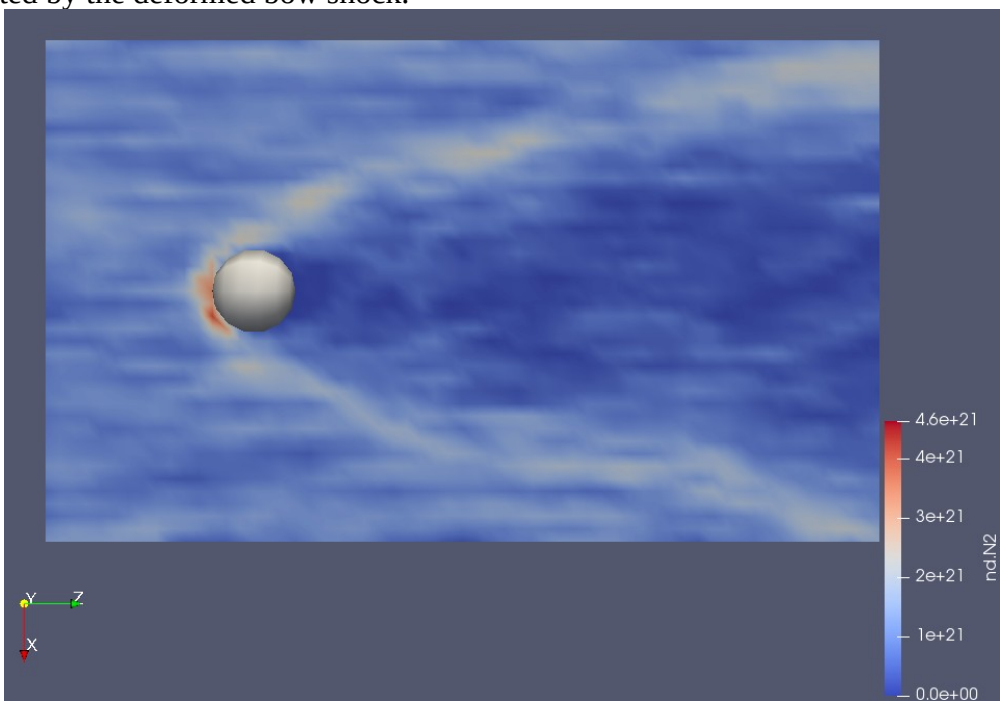
For a number density of 1e16, the flow looks like this:



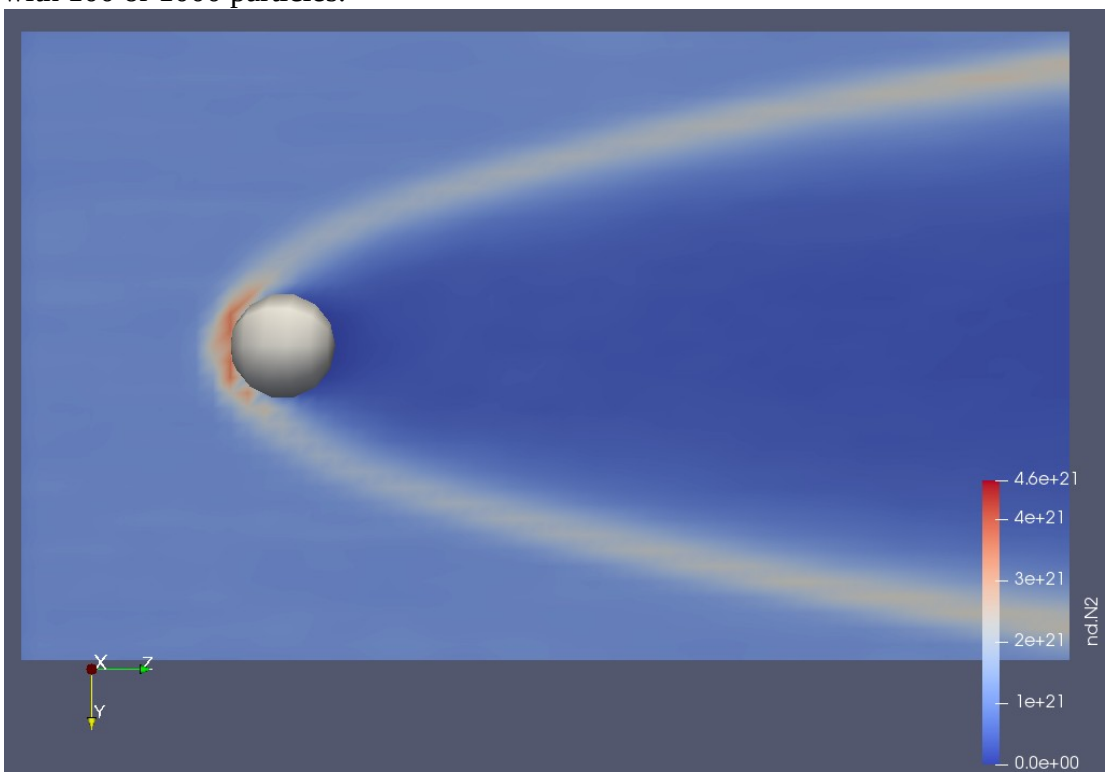
In this case, the error is the same: the bow shock is not calculated correctly. This simulation looks almost exactly the same as the one with a number density of 1e19, however the color legend shows that the number densities of the simulation are different.

Part 4:

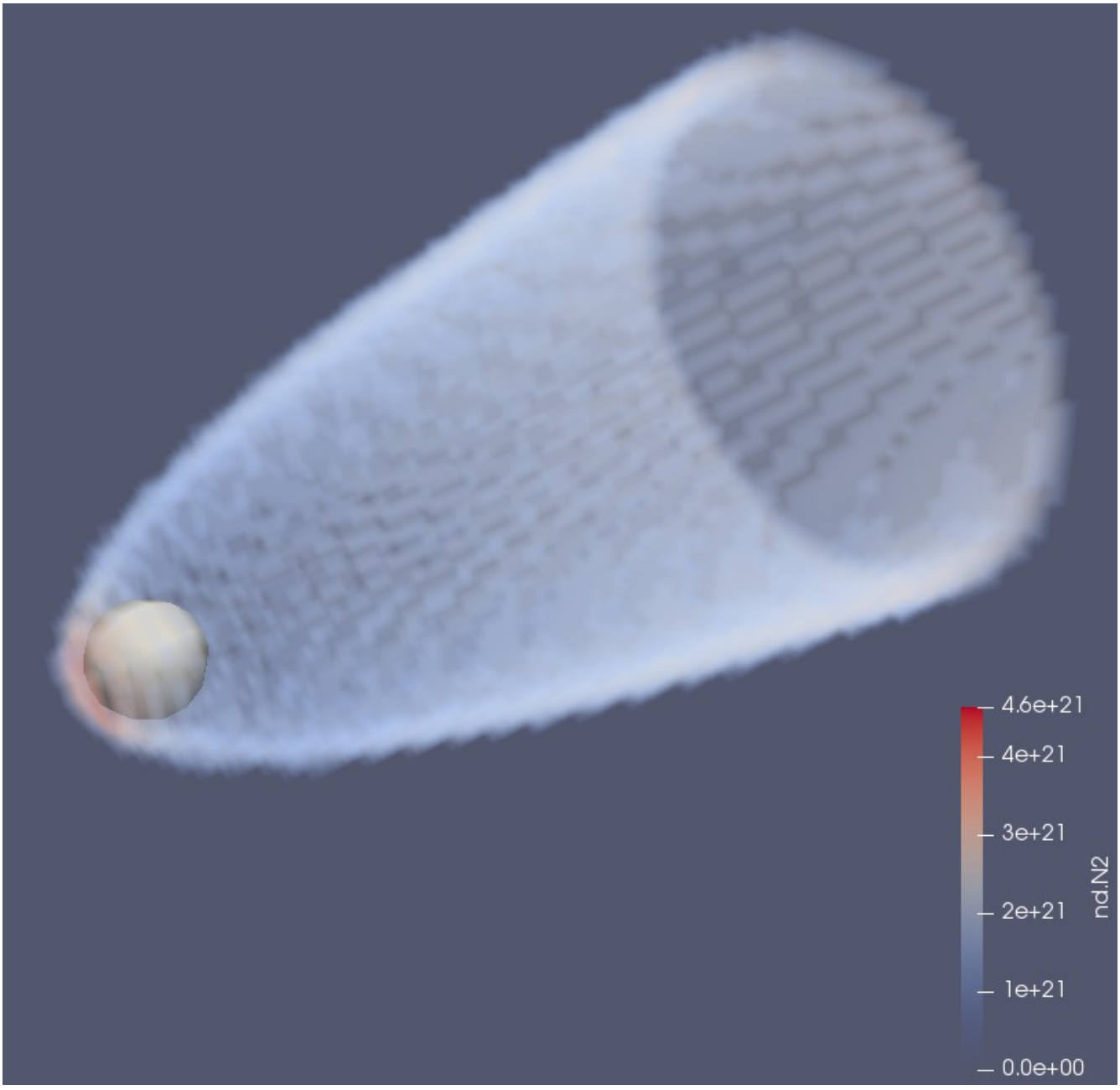
For 100 particles, the simulation took **245.68 seconds**. The bow shock appears to have significant noise, indicated by the deformed bow shock.



For 10000 particles, the simulation took **20081.9 seconds**. This image appears much smoother than the runs with 100 or 1000 particles.



An isometric view of the simulation with 10000 particles.



Part 5:

Box and Collision Check Changes:

First, the velocity would have to be set to zero and the sphere would be removed. The World code has to be updated to create a box instead of a sphere. A collision calculation function will have to be created for the box, in which case a collision will count when a particle is on the surface or outside of it, rather than the way it was done for a sphere, where a collision involved a particle on or inside the sphere. The box would have to be sized such that the mesh domain has some area outside of the box included in the simulation (this is the 5 cm described in the problem). When checking for a collision with the inside of the box, a hole position also needs to be specified. If the particle passed through this hole, a collision will not be counted. The way to know if a particle passed through the hole rather than a section of wall near the hole is to linearly interpolate using the current point's velocity. Using geometry, the line can be extended to the plane that has the face with the hole. If the point on the line at that plane is within the area of the hole, then the particle does not collide, and instead flies out. That criteria can be used to create a new if statement for the collision check function. As before, particles that leave the system will be removed from the simulation.

Physics Changes:

Every cell now needs to have a pressure. In the beginning, all cells inside will start with 1 atm, and all cells outside the box will start with 0 atm. Pressure creates a force on an area. A force is only generated if there is a pressure imbalance. Cells with differing pressures need to be compared so their forces can be calculated. Instead of comparing all cells to each other, there can be a list that stores cell coordinates of cells that are known to have different pressures from at least one adjacent cell. When the simulation is set up, this list will contain the layer of P = 1 atm cells just inside the box hole. In each iteration, each cell in the list will have its pressure compared to each adjacent cell to find the pressure difference. The pressure difference will be multiplied by the cell face area to find the force. Using the mass of the simulation particles and the time step, the acceleration and change in velocity can be found. This will be applied to each particle. All adjacent cells that were interacted with will have their coordinates added to the list of cells to update in the next simulation iteration. In addition, the pressures of each cell will need to be recalculated. As particles leave a cell, the cell's number density changes. Assuming a constant temperature, the new pressure can be easily calculated using the ideal gas law:

Ideal gas law: $PV = nRT = NkT$

Dividing the right side by volume gets P=nkT, where n is the number density. This way, the next step will be run with updated densities.