Karin Metzgar
ASTE 546
HW #4

Sheath1d_class_edited.cpp – Overview? I'm confused a so I'm going to try write out what the code is doing in in my own words to see if that helps.

The goal of the code is to simulate ..

Define the Mesh:
First, we define the simulation mesh to have 41 nodes, with the origin at 0 and ending at a length of 0.1 (meters?). We define the length of each cell as world.dx.

Define conditions of the plasma:
We then define the conditions of the plasma inside the mesh to have 100,000 simulation particles (N) with the desired number density. The number of real particles in the domain volume is defined as the number density * volume (which in this case can be represented by the length since the height and width are both 1). We define the different particle species with the C++ Species classes ions and electrons which each hold the respective mass, charge, particle weight (real particles/simulation particles) and the number of simulation particles.

Establish initial conditions:
To setup the simulation we need to fill the simulation volume/mesh with stationary particles. To do this we loop through all the simulation particles of each species and assign each one a random location and velocity.

Simulate:
Now that the species, mesh, and initial conditions have been defined we will start the simulation.

1. Advance Species (perform for each particle species)
    A. Get particle location / grid index (li)
        i. XtoL is a function that accepts a position value as a double, and subtracts the mesh origin in order to find its relative position and divides by the cell size in order to return the grid logical coordinate.
    B. Calculate electric field at particle position
        i. The electric field is only known at the grid nodes, so we use a function called gather to interpolate the data at the actual particle position between nodes.
        ii. Gather is defined as a function that accepts the grid location and the vector ef of the Electric Field at the nodes. It finds the x-location of the particle relative to the cell nodes (for example if a particle was at location 3.4 it would give di = 0.4) and then returns the interpolated electric field at that location using equations defined in the Gather slides of lecture 3.
    C. Integrate velocity

        i. Species.part[p] is how a specific particle can be defined, so Species.part[p].v is that particles velocity is =+ the species charge * electric field at particle position / species mass * time step
    D. Integrate position
        i. Particle location += particle velocity * time step
    E. Remove particles that should be removed.
        i. If a particle is out of bounds then set the struct members of that particle equal to that of another? Is this particle backfill? And reduce array size
2. Compute Number Density
    A. Reads in species data, world data and the nde / ndi vector and step through each node (size of nd vector), gets the particle location and calculates scatter. Scatter does the opposite of gather and interpolates particle data to the grid.
3. Calculate rho for each node
    A. Rho = ndi * ion charge + nde * electron charge, for each node.
4. Solve Potential for each node
    A. This function takes in the phi vector, rho vector, and world information, and solves Poisson's equation with Dirichlet boundaries using the Thomas algorithm.
5. Compute EF for each node
    A. Brings in EF vector, phi vector, world information and a Boolean to indicate whether it is second order or not
6. Rewind Velocity for each species (leapfrog)
7. Simulation Main Loop (for each time step)
    A. Advance Species
    B. Compute new number density
    C. Compute new rho
    D. Compute new potential
    E. Compute new node EF
    F. Output data for that timestep

Questions:

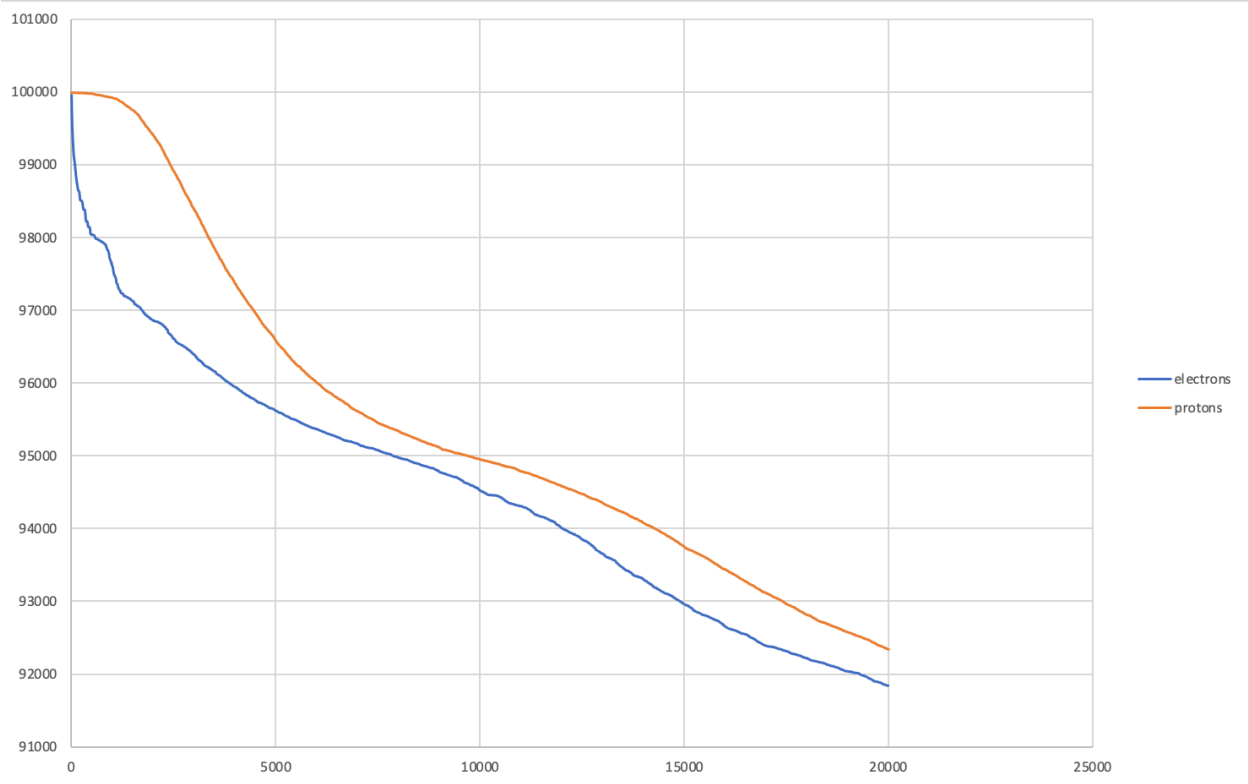
- Where does the vector ef get calculated? I see it get defined with the size at the beginning and when I debug I can see that it contains values but I don't see where those values actually get calculated? Does it get computed with computeEF function? that gets called after the ef vector is used in "advanceSpecies" ? how does that work?
- Same question for the nde and ndi vectors
- I'm confused on the last part of the compute number density function, what does the nd[i] /= world.dx; and the last two lines do?
-

Part 1: Diagnostic File, now add the ability for the code to output the following data to a csv file and plot the number of electrons and number of electrons over time.

a) Log file: diag-1a.csv

ts	time	num_ions	num_electrons
0	0	100000	99956
5	5e-10	100000	99841
10	1e-09	99999	99694
15	1.5e-09	99999	99593

The slope of the lines starts very different but over time they trend together.



b) Kinetic Energy

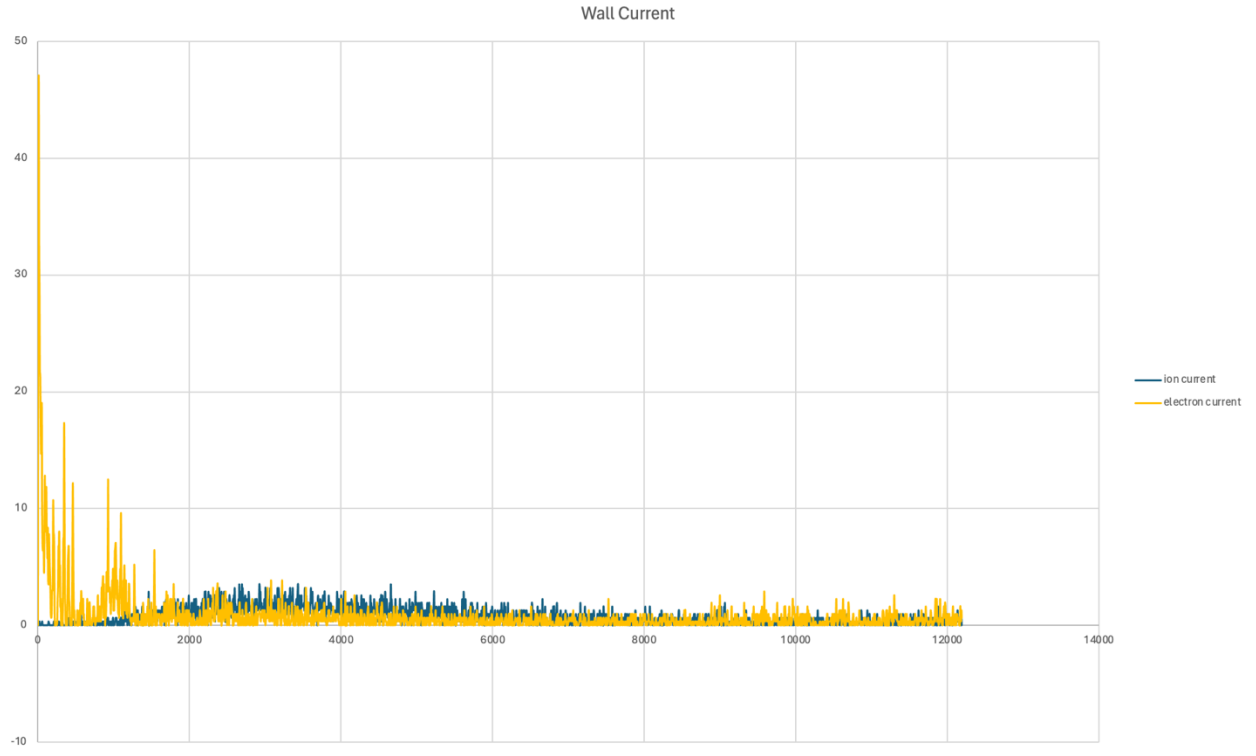
Avg KE Electrons 0.119535 eV
Avg KE Ions,0.00347601 eV

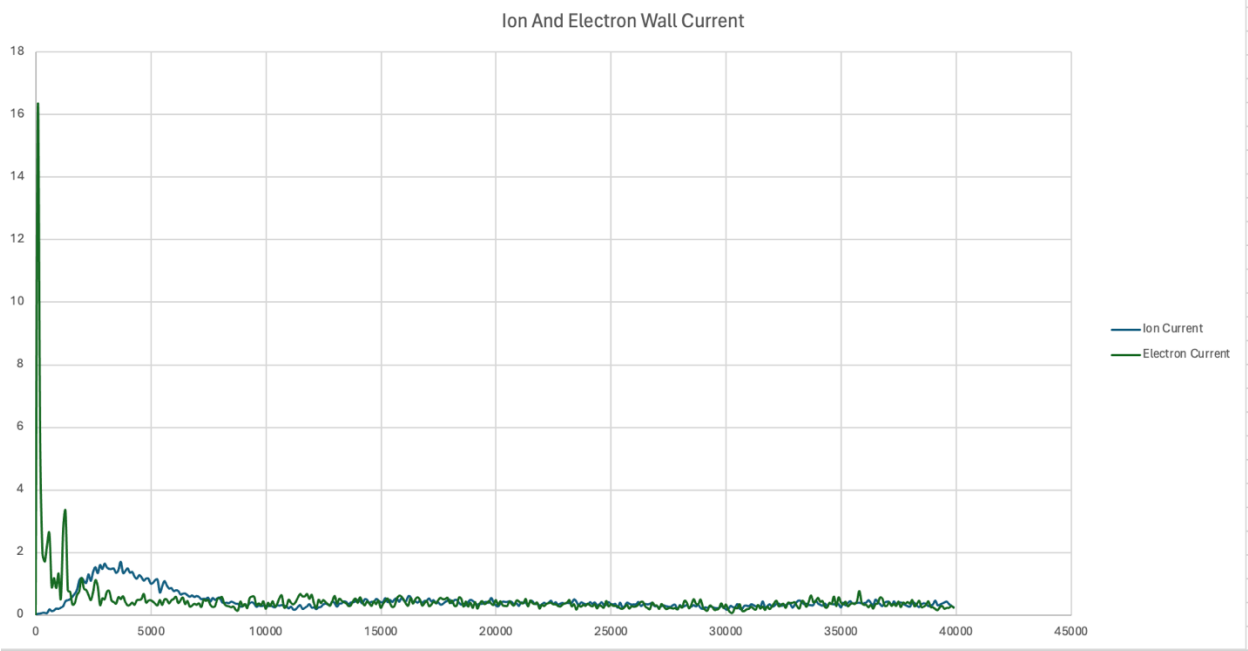
```
//compute average kinetic energy
  double getAveKE() {
    double ave_KE;
    double sum;

        for (int i; i < np ; i++) {
```

```
            sum += part[i].v * part[i].v;
        }

    ave_KE = (0.5 * m * (sum / np) ) / 1.602E-19;
    return ave_KE;
}
```

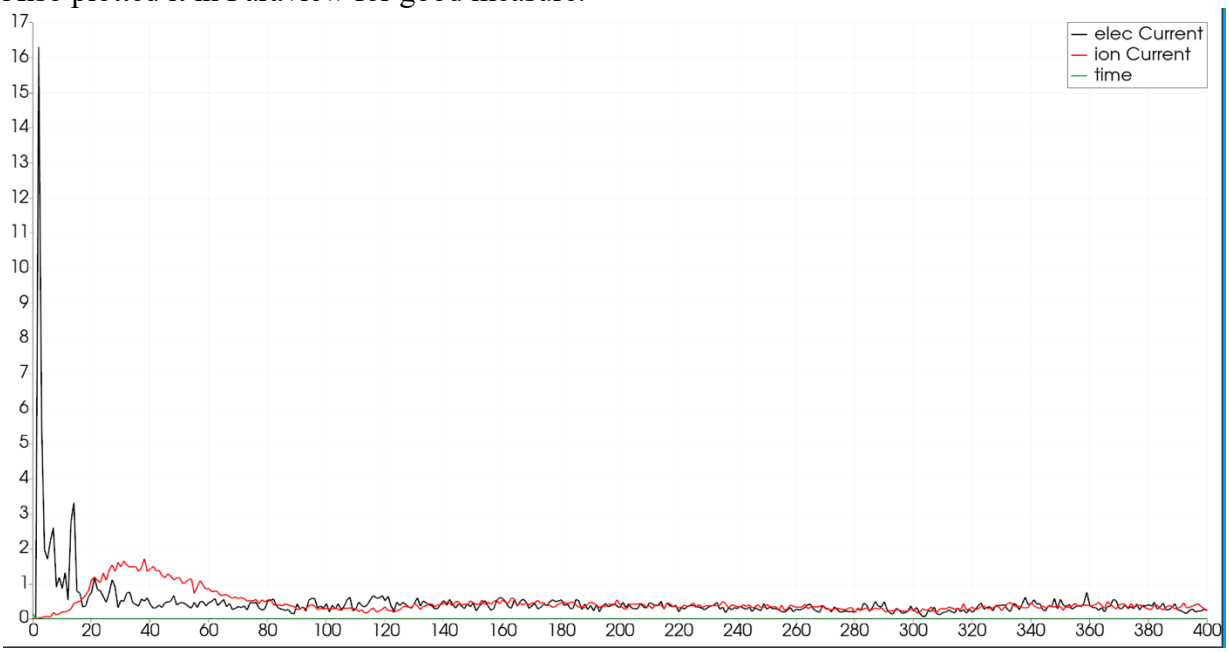
c) Wall Current

Initial run (first 2440 points)
Data: diag_WallCurrent.csv
Using a larger number of simulation particles and less time steps gets rid of some of the noise and clarifies the relationship. In the graph from part A the number of electrons decreases sharply while ions only decrease slightly, then both decrease together. The graph we see here of the Wall current fits into this as the current of electrons being absorbed by the wall is initially high and then decreases over time. While the ion current is near zero at first, then increases (likely in response to the electron current?) and then they seem to reach an equilibrium state.





Also plotted it in Paraview for good measure.



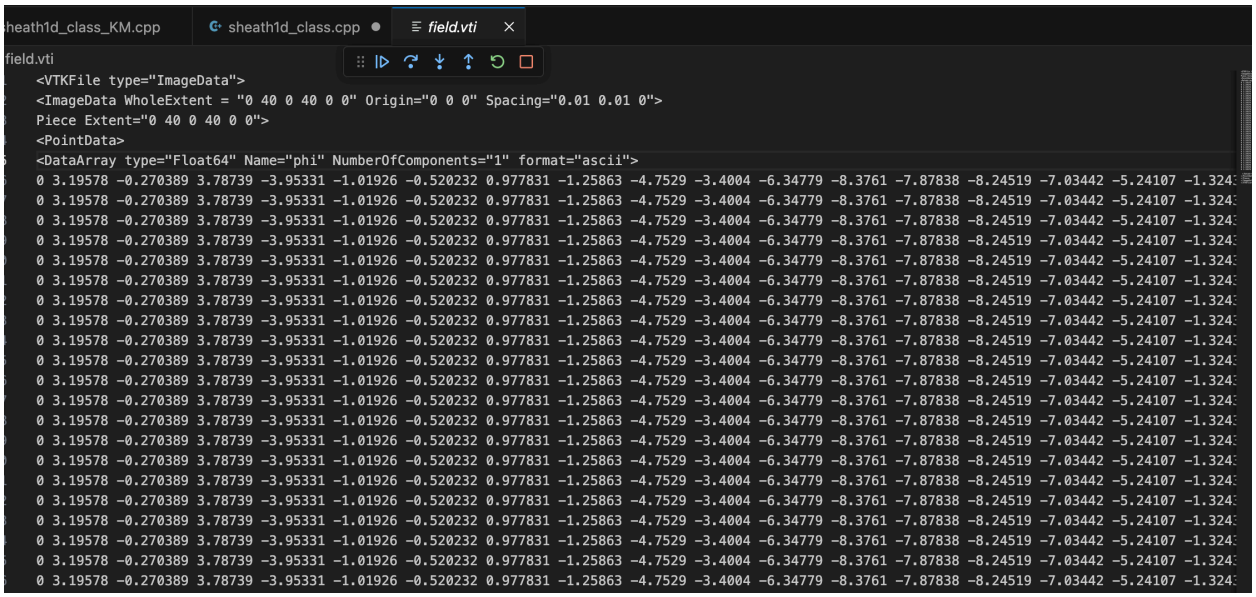
Part 2:

a)Time History Plots

… wow this part is confusing me a lot…

I think I almost got it, but it's producing the transpose of what I want?

Results:

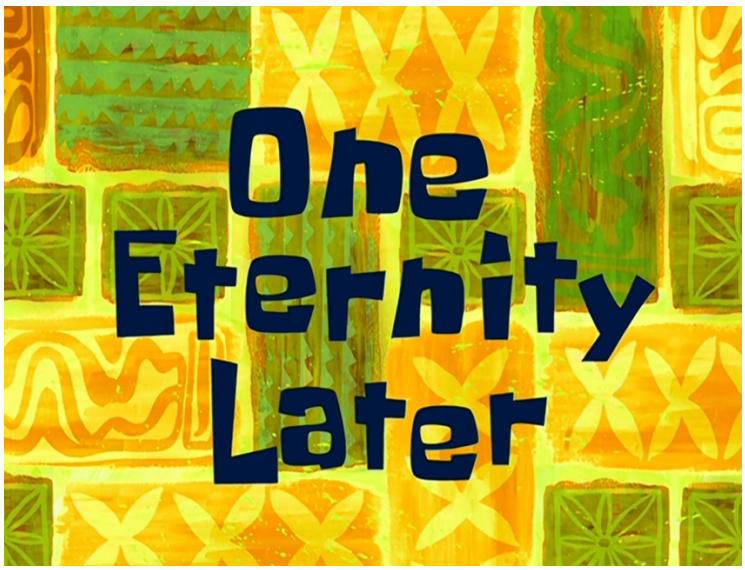


Code:

```
//dimensions
int ni = 0, nj = 0;

bool outputVTI(){
    int width;
    int height;
    int depth;
    width = ni;
    height = nj; //j axis coresponds to time
    depth = 0; //  = 0  with 2D data
    //spacing
    double dx, dy, dz;
    dx = 0.01;
    dy = 0.01;
    dz = 0;

    ofstream out("field.vti");

     if (!out.is_open()) {
        cerr << "Error: Could not open file for writing: field.vti" << endl;
        return false;
    }

    out << "<VTKFile type=\"ImageData\"> \n";
```

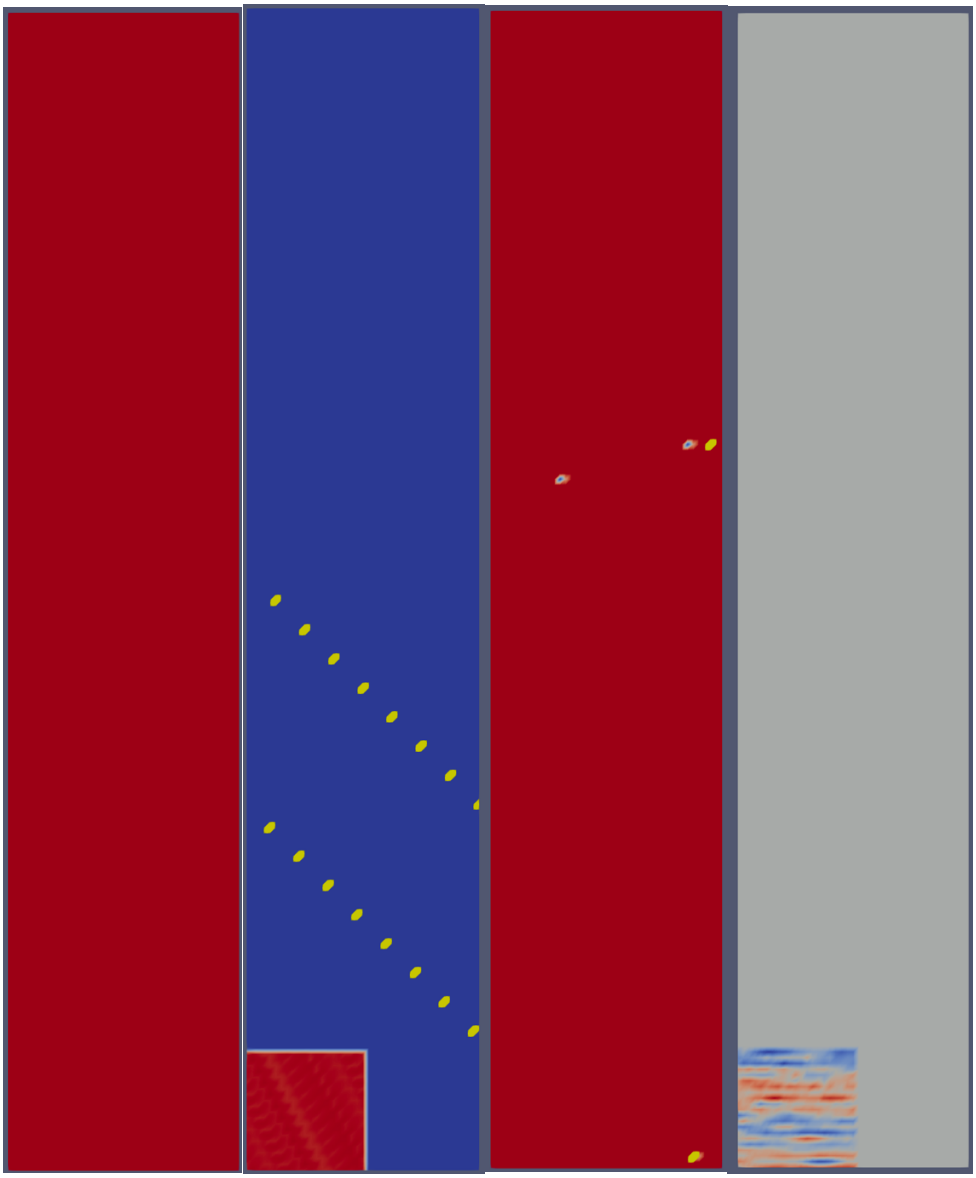
```
        out << "<ImageData WholeExtent = \"0 " << width -1 << " 0 " << height << " 0 " << depth
<< "\" Origin=\"0 0 0\" Spacing=\"" << dx << " " << dy << " " << dz << "\">" << "\n";
        out << "Piece Extent=\"0 " << width -1 << " 0 " << height << " 0 " << depth  << "\">" <<
"\n";
        out << "<PointData> \n";

        //phi
        out << "<DataArray type=\"Float64\" Name=\"phi\" NumberOfComponents=\"1\"
format=\"ascii\">" << "\n";


        for (int j = 0; j < height; j++) {  // Iterate over time steps
            for (int i = 0; i < width; i++) {  // Iterate over positions
            //int u;
            //u = width * height + j * width + i;
            out << phi[i] << " ";
            }
            out << "\n";
        }
```

----trying again .. again ----



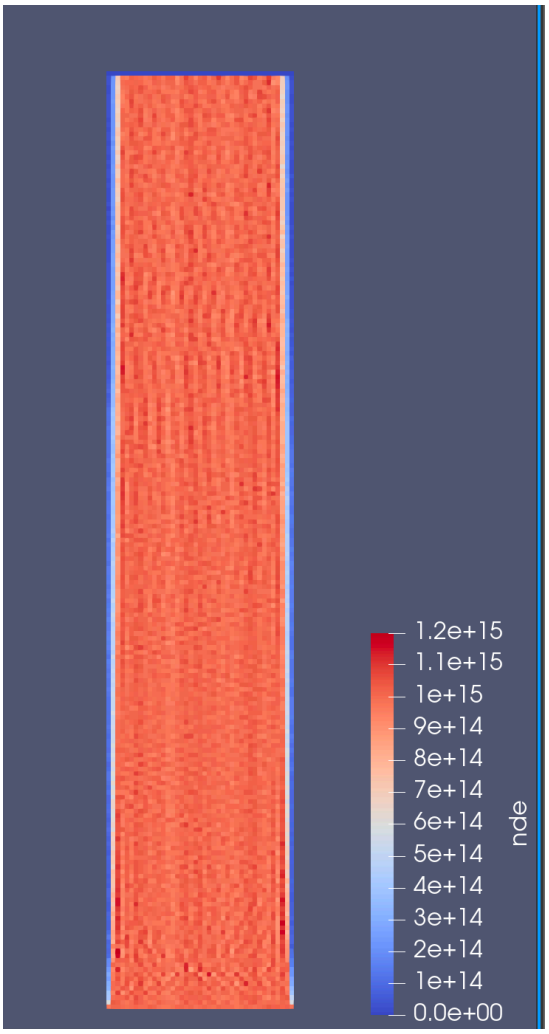
I literally just put what's in the homework description, and realized I was incorrectly defining the Piece Extent which was causing all the errors in Paraview and now I get this… ?

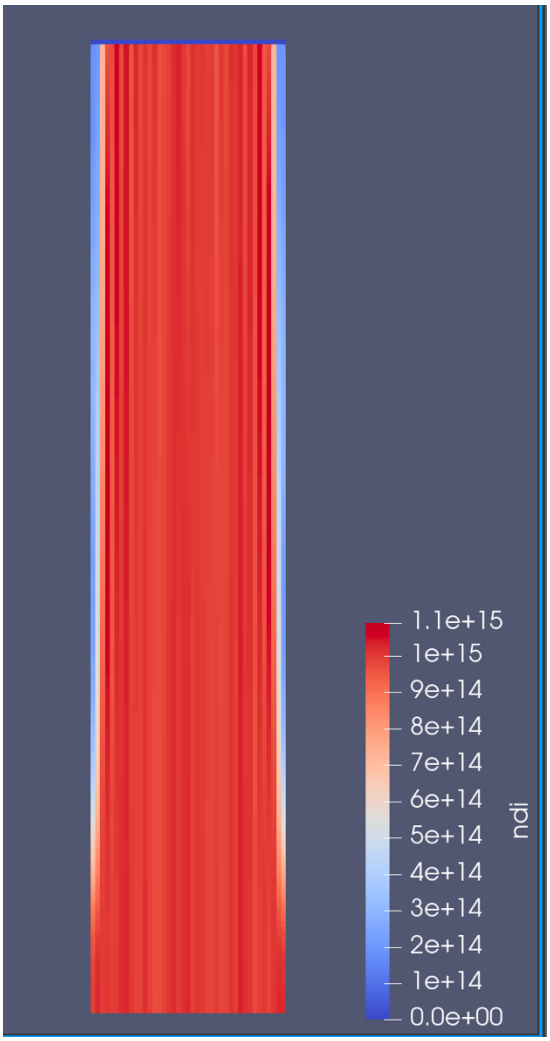
In the order nde, ndi, rho, phi from left to right:

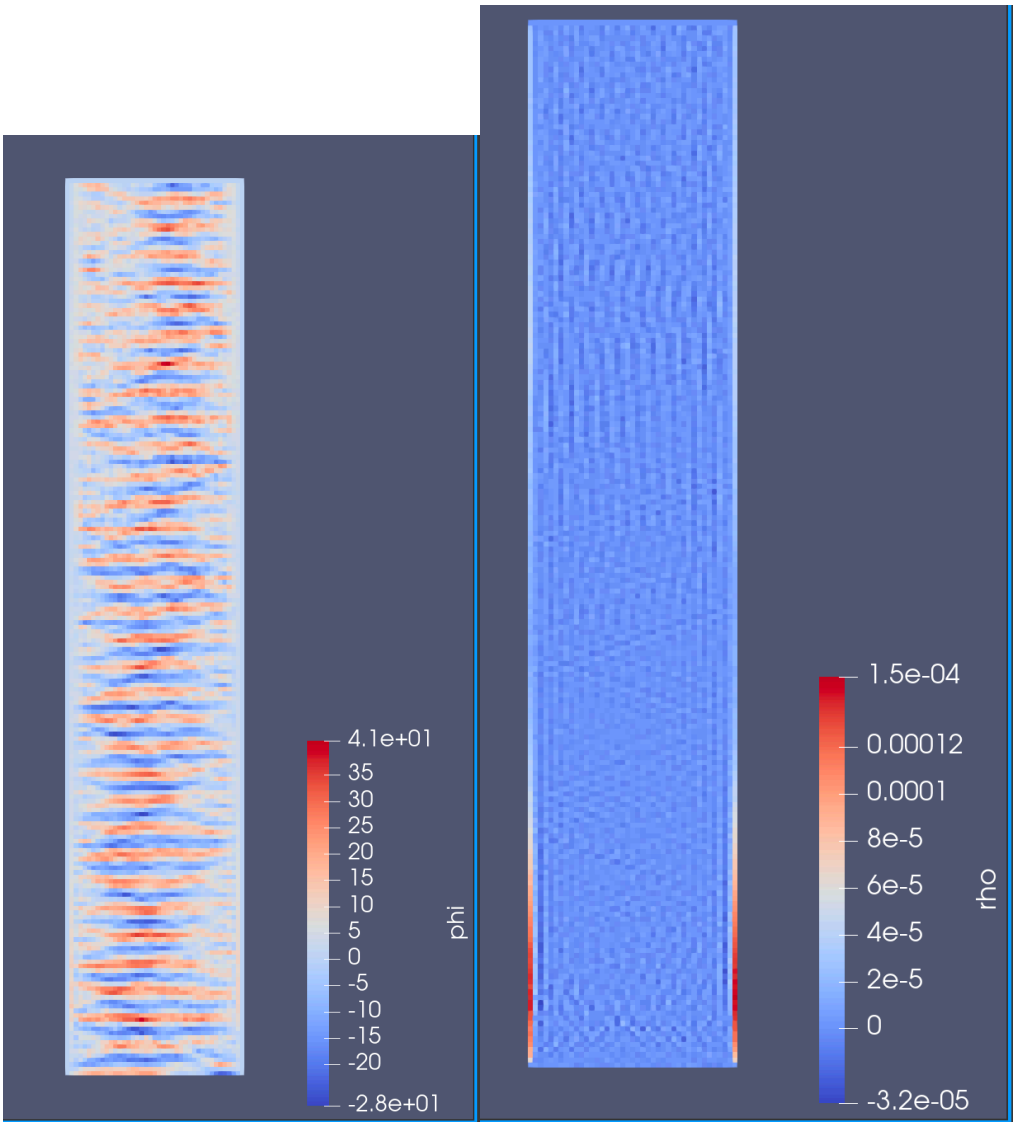


And this definitely doesn't feel entirely correct … or is this the part where you warn Paraview may struggle to plot if I don't scale better?

.. tried on a different computer and yay!


1.2e+15
1.1e+15
1e+15
9e+14
8e+14
7e+14
6e+14
5e+14
4e+14
3e+14
2e+14
1e+14
0.0e+00
nde


1.1e+15
1e+15
9e+14
8e+14
7e+14
6e+14
5e+14
4e+14
3e+14
2e+14
1e+14
0.0e+00
ndi



Data copy code:

```
void addData(dvector &phi, dvector &rho, dvector &ndi, dvector &nde) {

        //do I need to also add if < 0 here?
        if ( j >= nj) {
            cerr << "Out of bounds" << endl;
            return;
        }

        int index = j * ni;
        for (int w = 0; w < ni; w++) {
            this->phi[index + w] = phi[w];
            this->rho[index + w] = rho[w];
```

```
            this->ndi[index + w] = ndi[w];
            this->nde[index + w] = nde[w];
        }

        ++j;
    }

    protected:

    int j =0;


};
```

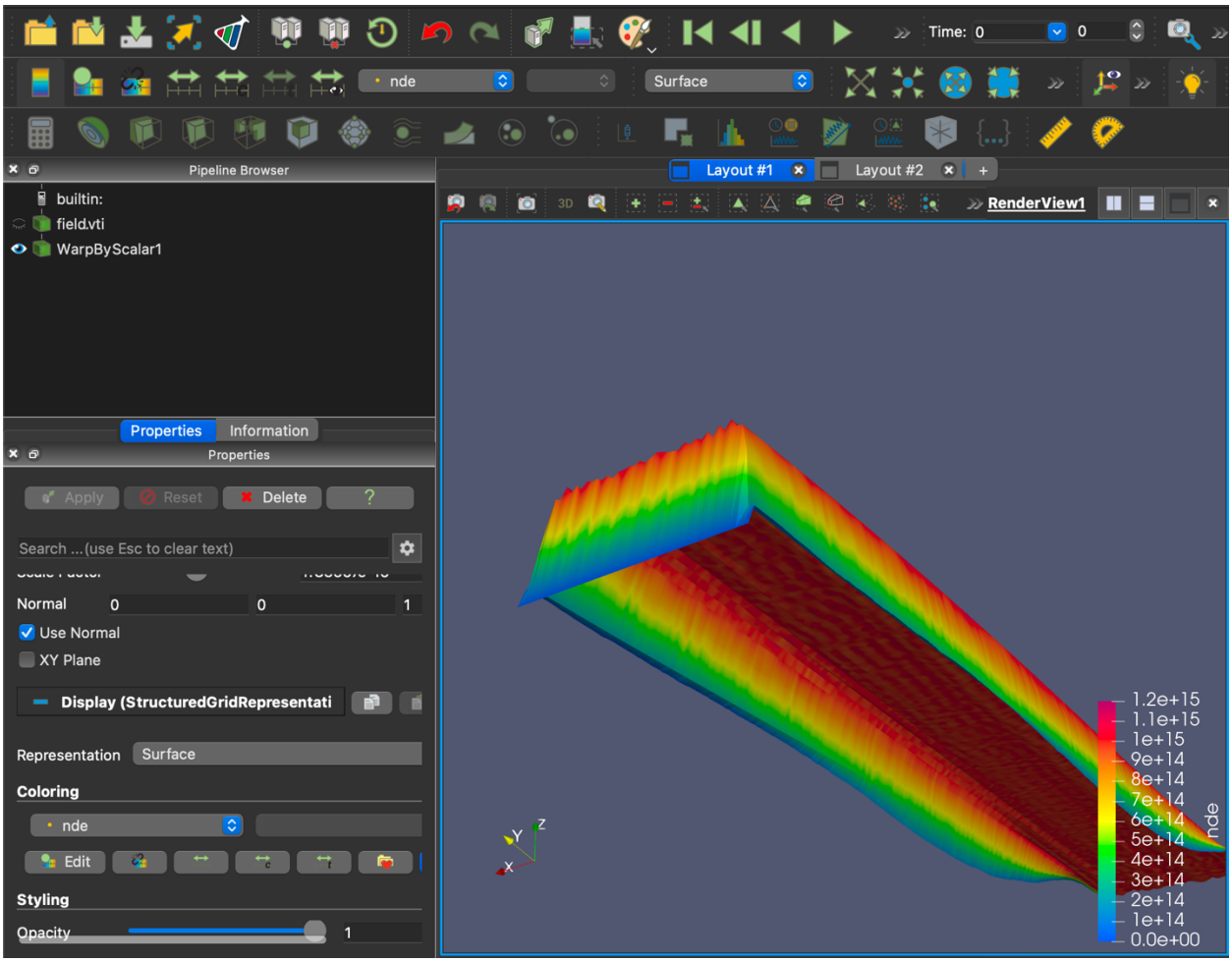
d) Hooks:

```
//added in HW4 2 d)Hooks
    int num_ts = 20000;
    int add_skip = 100;
    Results results(world.ni, num_ts/add_skip); //nj is the number of time entries
```

…

```


        if (world.ts%100==0) {
            double sim_time;
            sim_time = world.ts * world.dt;
            out <<world.ts << "," << sim_time << "," << ions.np << "," <<
ions.getCurrent(world) << "," << eles.np << "," << -eles.getCurrent(world) << "\n";
            results.addData(phi, rho, ndi, nde);
        }
```

e) Testing



f) Cleanup 1
Got the code to compile properly after this.

g) Cleanup 2
Got the code to compile properly after this.

h) Cleanup 3
Have not been able to get the code to output the right data, code is compiling and I think it's just a problem with how I'm attempting to update the output functions, however data is appearing wrong in both the csv's and the vti file won't load into Paraview without throwing errors at the size of phi..
I have tried a few different things but am not sure I understand how to get Results to be able to view the data from ions and eles…

I tried changing all instances of phi to world.phi etc. and nde to eles.den etc., but it didn't like the this->eles.den because Results doesn't have a member "eles" so I figured just the second part was that .. etc.etc. no dice.

```
class Results {
    public:
    //constructor
    Results(World &world, int nj) : ni(world.ni), nj(nj), world{world} {
```

```
    //allocate bufferes and clear all bytes to 0

    phi = new double[world.ni*nj]; memset(phi,0,world.ni*nj*sizeof(double));
    rho = new double[world.ni*nj]; memset(rho,0,world.ni*nj*sizeof(double));
    ndi = new double[world.ni*nj]; memset(ndi,0,world.ni*nj*sizeof(double));
    nde = new double[world.ni*nj]; memset(nde,0,world.ni*nj*sizeof(double));

}

//destructor
~Results() {
    delete[] phi; phi = nullptr;
    delete[] rho; rho = nullptr;
    delete[] ndi; ndi = nullptr;
    delete[] nde; nde = nullptr;

}

//function decleration
//bool outputVTI();

//databuffers
double *phi = nullptr, *rho = nullptr;
double *ndi = nullptr, *nde = nullptr;

//dimensions
int ni = 0, nj = 0;

bool outputVTI(Species &ions, Species &eles){
    dvector &phi = world.phi;
    dvector &rho = world.rho;
    dvector &ef = world.ef;
    dvector &ndi = ions.den;
    dvector &nde = eles.den;

    int width;
    int height;
    int depth;
    width = ni;
    height = nj; //j axis coresponds to time
    depth = 0; //  = 0  with 2D data
    //spacing
    double dx, dy, dz;
    dx = 0.01;
    dy = 0.01;
    dz = 0;
```

```
        ofstream out("field.vti");

         if (!out.is_open()) {
            cerr << "Error: Could not open file for writing: field.vti" << endl;
            return false;
        }

        out << "<VTKFile type=\"ImageData\"> \n";
        out << "<ImageData WholeExtent = \"0 " << width -1 << " 0 " << height << " 0 "
<< depth  << "\" Origin=\"0 0 0\" Spacing=\"" << dx << " " << dy << " " << dz << "\">"
<< "\n";
        out << "<Piece Extent=\"0 " << width -1 << " 0 " << height-1 << " 0 " << depth
<< "\">" << "\n";
        //out << "<Piece Extent=\"0 " << 20 << " 0 " << 20 << " 0 " << 14  << "\">" <<
"\n";
        out << "<PointData> \n";

        //phi
        out << "<DataArray type=\"Float64\" Name=\"phi\" NumberOfComponents=\"1\"
format=\"ascii\">" << "\n";
        for (int u = 0; u < width *height; u++){
            out << phi[u] << " ";
            //out << "\n";
        }
        out << " \n </DataArray> \n";

        //rho
        out << "<DataArray type=\"Float64\" Name=\"rho\" NumberOfComponents=\"1\"
format=\"ascii\">" << "\n";
        for (int u = 0; u < width *height; u++){
            out << rho[u] << " ";
            //out << "\n";
        }
        out << " \n </DataArray> \n";

        //nde
        out << "<DataArray type=\"Float64\" Name=\"nde\" NumberOfComponents=\"1\"
format=\"ascii\">" << "\n";
        for (int u = 0; u < width *height; u++){
            out << nde[u] << " ";
            //out << "\n";
        }
        out << " \n </DataArray> \n";

        //ndi
        out << "<DataArray type=\"Float64\" Name=\"ndi\" NumberOfComponents=\"1\"
format=\"ascii\">" << "\n";
```

```
        for (int u = 0; u < width *height; u++){
            out << ndi[u] << " ";
            //out << "\n";
        }
        out << " \n </DataArray> \n";


        //closing tags
        out << "</PointData> \n";
        out << "</Piece> \n";
        out << "</ImageData> \n";
        out << "</VTKFile> \n";

        out.close();
        return true;
    }

    //is this right? for step g, cleanup 2?
    void addData(Species &ions, Species &eles) {

        //do I need to also add if < 0 here?
        if ( j >= nj) {
            cerr << "Out of bounds" << endl;
            return;
        }

        int index = j * ni;
        for (int w = 0; w < ni; w++) {
            this->phi[index + w] = world.phi[w];
            this->rho[index + w] = world.rho[w];
            this->ndi[index + w] = ions.den[w];
            this->nde[index + w] = eles.den[w];
        }

        ++j;
    }

    protected:
    World &world; //reference to world
    int j =0;


};
```

i) Mesh Averaged Velocity

Part 3: Particle Scatter Plot