Computational Physics Final Project

Electrostatic Particle-In-Cell Method

Erica Kotta

December 16, 2016

1 Intro to Particle-In-Cell Method

A plasma is a gas whose atoms, due to certain conditions (high temperature, strong external EM fields, etc) have been ionized. Although the net charge of the plasma as a whole is generally zero, the constituent particles now carry a (positive or negative) charge. These charge-carrying particles now will interact not only with the external EM fields but will create their own fields, which will in turn affect their dynamics as well.
To completely and accurately describe plasma dynamics entails calculating the force $F = q\left[\vec{E} + (\vec{v} \times \vec{B})\right]$ *between each and every pair of particles.* This would mean, for a simulation of N particles, N^2 calculations for one time step. Even in a very simplified model of a million particles, the simulation becomes quite impossible.

Now, however, consider one particle–say, an electron, in this sea of particles in the plasma. It will attract positive ions (let's just say they are simple protons, for simplicity) and repel other electrons. The charge of this electron becomes effectively shielded by the surrounding protons, and the force felt by other particles in the area falls exponentially at a distance called the Debye length, defined by

$$\lambda_{De} = \sqrt{\frac{\epsilon_0 k_B T_e}{q^2 n_0}} \tag{1}$$

where ϵ_0 is the permittivity of a vacuum, k_B is the Boltzmann constant, T_e the electron temperature, and n_0 is the plasma (electron) density.
When we are working with spatial scales larger than the Debye length, the dynamics are not dominated by the interactions of closest neighboring pairs, and we start to see the collective behavior of particles.

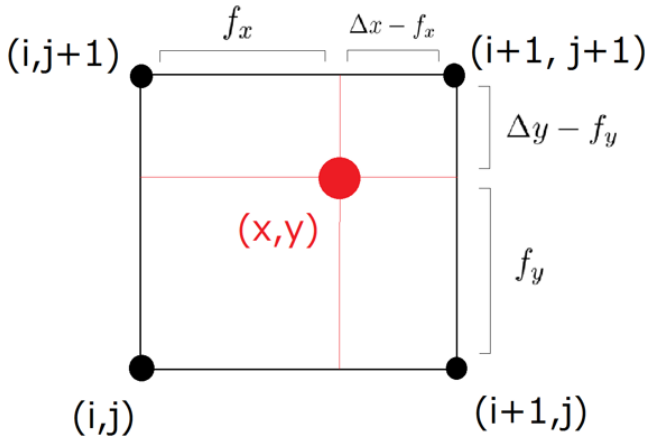
Furthermore, if we consider spatial scales larger than λ_{De} we can also use the concept of *superparticles*, which is simply a clump of many (usually around the ballpark of 10^6) individual particles. A superparticle will behave similarly to an individual particle.

Equipped with these simplifications, we start to see that plasma simulation can in fact be modelled. Here I introduce the Particle-In-Cell (PIC) method. It uses the concept of superparticles along with the neat trick of approximating their charges and thus EM fields at the points of a grid, and using these grid EM values to move the particles one step at a time. Below I describe the basic structure of the PIC code I wrote for a simple 2-D plasma in the electrostatic case (ignoring all B-fields).

Disclaimer: I should warn that this is a *very* qualitative and simplified program; I use toy negative superparticles of total charge -1.0 and mass 0.1, and positive superparticles of total charge +1.0 and mass 1.0. Epsilon is also set to 1.0. This code is light years away from being a tool for studying plasma physics, and instead is simply a demonstration of the basics of the PIC method. Qualitative, unitless graphs in the final section simply show that the code is not overwhelmed by non-physics.

2 Interpolating Particle Charges to the Grid

Let's look at a (2-D) portion of the plasma. We section each dimension into N slices (Nx, Ny), each slice $(\Delta x, \Delta y)$ set to λ_{De} (this ensures we are working at an appropriate scale). Now rather than keeping track of the individual superparticle positions, we weigh the charge to the four grid points surrounding it according to the figure:



Grid point (i,j) gets assigned charge proportional to $\dfrac{(\Delta x - f_x) \times (\Delta y - f_y)}{\Delta x \Delta y}$,

point (i+1,j) gets $\frac{(\Delta x - f_x) \times (f_y)}{\Delta x \Delta y}$, etc. We do this for all grid points, with the boundary and corner points getting doubled overall (as the area contributing to these points is a half of the others; corner points get doubled twice).

3 Using FFT to calculate scalar potential V

With the charges distributed to the grid, we can now approximate the scalar potential field at the grid points from the relationship:

$$\frac{\partial^2 V_{ij}}{\partial x^2} + \frac{\partial^2 V_{ij}}{\partial y^2} = -\frac{1}{\epsilon_0}\rho_{ij} \tag{2}$$

which can be discretized to

$$\frac{(V_{i+1,j} + V_{i-1,j} - 4V_{i,j} + V_{i,j+1} + V_{i,j-1})}{\Delta x \Delta y} = -\frac{1}{\epsilon_0}\rho_{ij} \tag{3}$$

We can solve for ϕ using the Jacobian method as outlined in the course textbook using the 5-point stencil. If however we can assume periodicity in both directions (which we can in our case, if we are looking at a small chunk in the middle of a huge cloud of plasma), it is we can use the Fourier method, which is much faster. The 2-D FT of V_{ij} and ρ_{ij} are (using i^* to indicate $\sqrt{-1}$ differently from index i and setting $N_x = N_y = N$):

$$\begin{aligned} V_{mn} &= \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \exp\left\{\frac{2i^*\pi(mi+nj)}{N}\right\} V_{ij} \\ \rho_{mn} &= \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \exp\left\{\frac{2i^*\pi(mi+nj)}{N}\right\} \rho_{ij} \end{aligned} \tag{4}$$

which inverse-transform to

$$\begin{aligned} V_{ij} &= \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} \exp\left\{\frac{2i^*\pi(-im-jn)}{N}\right\} V_{mn} \\ \rho_{ij} &= \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} \exp\left\{\frac{2i^*\pi(-im-jn)}{N}\right\} \rho_{mn} \end{aligned} \tag{5}$$

Plugging the FT expressions of V_{ij} and ρ_{ij} into the discretized equation, dividing both sides by $\frac{1}{N}e^{2i^*\pi(-im-jn)/N}$ and looking at one (m,n) point at a time (so taking out the $\sum$'s), we get

$$V_{mn} = \frac{\rho_{mn}(\Delta x \Delta y)}{\exp\left\{\frac{-2i\pi m}{N}\right\} + \exp\left\{\frac{2i\pi m}{N}\right\} - 4 + \exp\left\{\frac{-2i\pi n}{N}\right\} + \exp\left\{\frac{2i\pi n}{N}\right\}} \tag{6}$$

and inverse-FT-ing these V_{mn} values gives us back V_{ij}. [1] These steps are implemented in the code using numpy's rfft2 and irfft2 functions.

4 Calculating E-field

Using the newly calculated values V_{ij}, we can now calculate the electric field. Its true definition is the (negative) gradient of the potential; in the code we approximate this with finite differencing (central method), finding the Ex-field and Ey-field separately:

$$\begin{aligned} E_{x,ij} &= -\frac{V_{i,j+1} - V_{i,j-1}}{2\Delta x} \\ E_{y,ij} &= -\frac{V_{i+1,j} - V_{i-1,j}}{2\Delta y}. \end{aligned} \tag{7}$$

and at the boundaries include lines in the code to use forward/backward differencing divided by $(1 \times \Delta x, y)$ to account for the lack of point on the other side. Then we interpolate the field at the *grid points* back onto the *particles*. This is done in the exact same way as interpolating the particle charge onto the grid. In my code function for weighing the particles onto the grid, I stored and returned the (i,j) points and the corresponding weights, so just call these back in the function for this step.

Below is an example plot showing the charge density (plotted using imshow), the calculated potential (the contour lines), and the resulting E-field (the quiver plot).

[1] Note that due to Python's default indexing of arrays versus plot points, a specified x-value from plot (x,y) will correspond to the j-value of grid point (i,j), and vice versa.

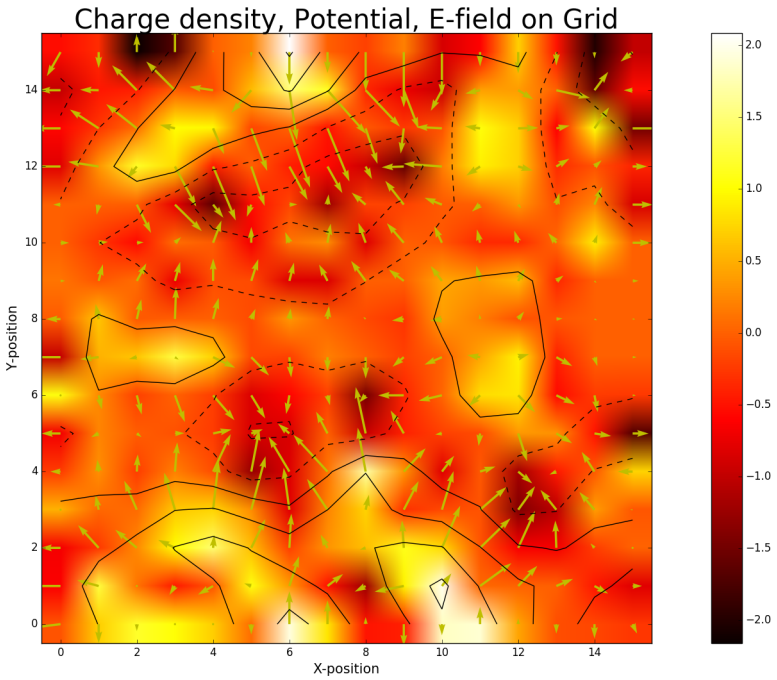


Figure 1: Density plot = Charge distribution; Contour lines = Scalar potential; Vector field = E-field

Though this is a very qualitative plot (see disclaimer at end of Section 1), you can see that the results make sense: the potential lines follow the charge grid as expected, with the calculated E-field vector field pointing in the correct directions and relative magnitudes along the potential gradient.

5 Advancing the Particles

Now we have the strength of the electric field in the x- and y-direction for each particle in our section of the plasma, and will use the Leap-Frog method to advance each particle by one Δt at a time. I chose this method because it is (supposedly) energy-conserving.
The Courant-Friedrichs-Lewy condition for numerical stability in two dimensions is given by the formula

$$\Delta t \leq \frac{CFL}{v_{x,max}/\Delta x + v_{y,max}/\Delta y} \tag{8}$$

where CFL is a constant less than 1. For plasmas, an equivalent condition is given by $\Delta t \leq \dfrac{0.2}{\omega_p}$ where ω_p is given by $\dfrac{q}{m}E_{max}$ and is the maximum Δv for a

given step. I only consider the values corresponding to the electrons here; their smaller relative mass means that they respond the most quickly to the fields, and thus impose the greatest restrictions on the time step.

The particles are advanced in the code in the simplest way:

$$\begin{aligned} \vec{v}_{new} &= \vec{v}_{old} + \Delta t\Big(\frac{q\vec{E}}{m}\Big) \\ \vec{x}_{new} &= \vec{x}_{old} + \Delta t(\vec{v}_{new}) \end{aligned} \tag{9}$$

where $\vec{v}_{old}$ and $\vec{v}_{new}$ are the velocities at $t - \dfrac{\Delta t}{2}$ and $t + \dfrac{\Delta t}{2}$, and $\vec{x}_{old}$ and $\vec{x}_{new}$ are the positions at t and $t + \Delta t$, respectively. At $t = 0$ only, the initial velocity vector is decelerated by half a time step to set up the stagger. (The velocities are also accelerated by half a time step for each loop to un-stagger the positions and velocities for use in calculations of energy, etc.)

One way to check, at least qualitatively, that the simulated dynamics are not completely non-physical is to plot the velocity distribution of the particles at certain times and check that they have a Maxwellian shape.
Starting with particles distributed randomly throughout the area and with zero velocity, I plotted a histogram of the velocity distribution (in x-direction) at different times. The distribution almost immediately formed the Maxwellian shape, and maintained the shape for all consecutive steps.

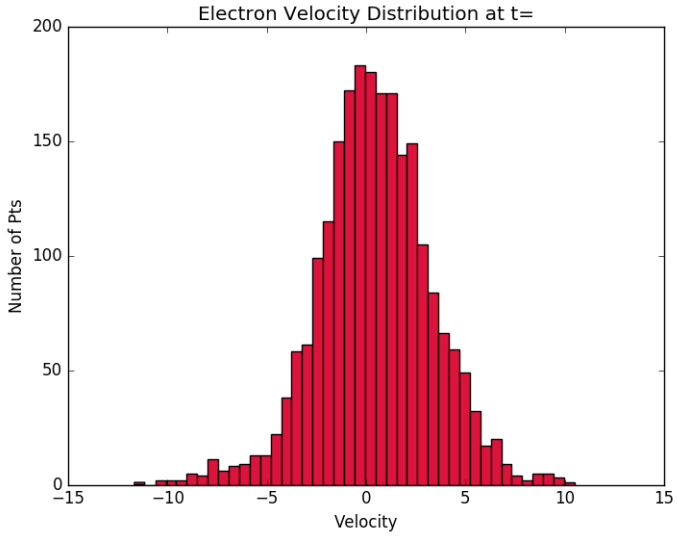


Figure 2: Maxwellian shape apparent by first time iteration, maintains shape for all t.

Another sanity check is to plot a density plot of the position vs velocity. During equilibrium, the plot should be centered around v=0 and uniformly distributed position-wise.

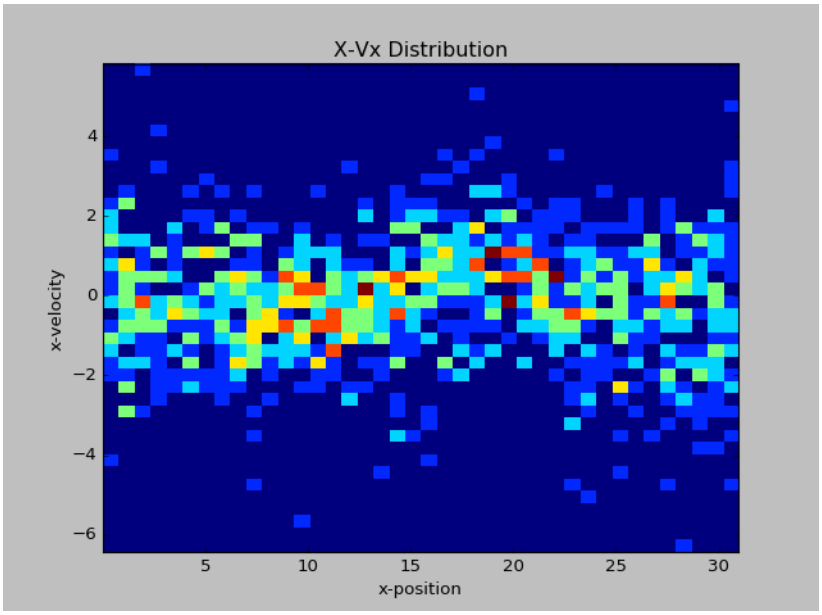
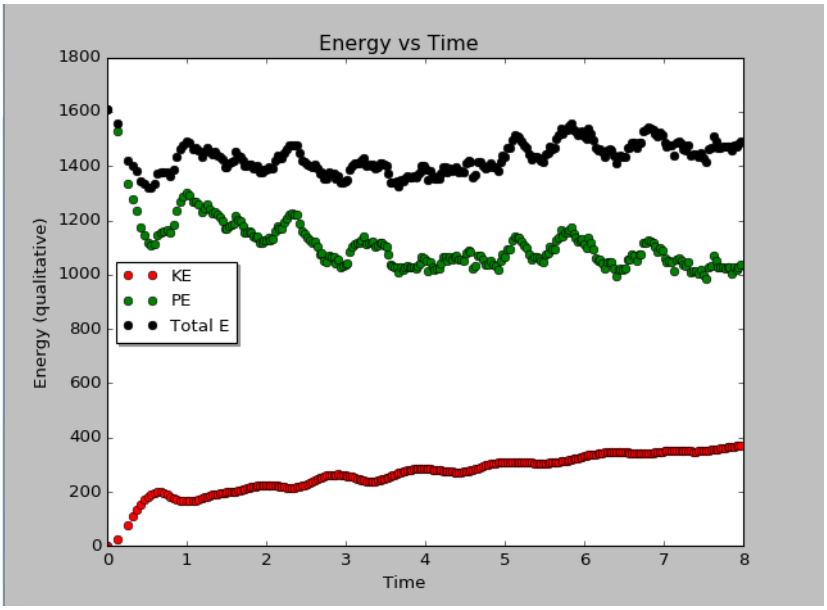


Figure 3: Particle velocities centered around v=0, distributed evenly in position.

Lastly, a plot of energy vs time shows how the code conserves energy. Though the graph shows that total energy is not precisely conserved and seems to gradually increase over time, there at least is an apparent anti-correlation between the kinetic and potential energy.



5.1 References

References

[1] C K Birdsall A B Langdon. *Plasma Physics Via Computer Simulation.* Adam Hilger, 1991.

[2] James J Y Hsu. *Visual and Computational Plasma Physics.* World Scientific Publishing, 2015.

[3] The Electrostatic Particle In Cell (ES-PIC) Method
`https://www.particleincell.com/2010/es-pic-method/`