# How many particles does it take to make a plasma?

## Particle-in-Cell Plasma Simulation Project

Kyle Miller, Evan Chmilar

**Physics 512: Computational Physics**, Department of Physics, McGill University

**ABSTRACT**

*Context.* This is a computer simulation project for a computational physics class based on the `python` language.
*Aims.* Provide a brief overview of the subject contained herein (plasma physics and a few particular numerical methods) and cover the important aspects of our code, including an analysis of the results.
*Methods.* A particle-in-cell (PIC) simulation is carried out using `python` with several numerical techniques explored and implemented to show the importance of faster and more efficient methods.
*Results.* Our simulation appears to model simple systems quite well and on reasonable computational time scales.
*Conclusions.* PIC plasma simulations work.

**Key words.** physics – computer simulation – particle-in-cell – plasma physics

## 1. Introduction

In the realm of physics computer simulations, one of the toughest areas is plasma physics. The number of particles, types of interactions, combination of kinetic and collective effects all conspire to make coarse grained simulations less appropriate than in other areas of physics. Plasmas are interesting because they are both practical and found in the majority of astronomical systems (~99% of space is ionized; see Table 1 and Figure 2). This "fourth state of matter" also turns up in particle accelerators, classified military projects, and in an increasing number of commercial applications. For all these reasons (along with the fact that most plasma labs are either top-secret or working on fusion) people have worked on ways to simulate these systems in a way that is accurate and efficient.

Often, in simulations, a fluid approximation approach is taken to smooth over the nitty gritty of particles. This works well in many situations, however, with plasmas, which are generally considered fluids, they often have non-thermal or non-equilibrium energy distributions (see Figure 1). The most fundamental *ab initio* way to model a plasma is by using the 6D distribution function (Vaslov-Maxwell) coupled to Maxwell's Equations in the electromagnetic regime or Poisson's equation in the electrostatic regime. This option is prohibitively expensive computationally and so a type of discretization of the problem is often implemented using single particles—particle-in-cell (PIC)—that are in a sense representative of the underlying physics encoded in the distribution function. Going this route generates a host of problems that need to be taken into account and mitigated (see Section 5 below).

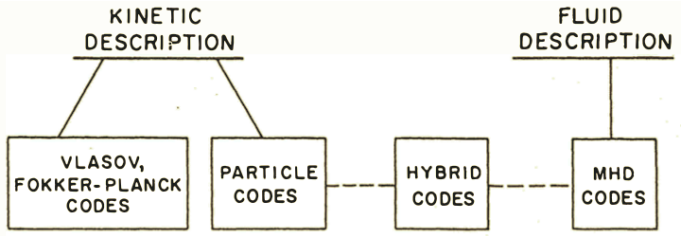


**Fig. 1.** Schematic view of the continuum, from fluid to kinetic-particle, of various plasma simulation types Birdsall (2005).

**Table 1.** Plasma Characteristics

| Name | n | T [K] | $\omega_p$ $[sec^{-1}]$ | $\lambda_D[cm]$ |
|---|---|---|---|---|
| Interstellar Medium | 1 | $10^4$ | $6{\times}10^4$ | $10^3$ |
| Nebula | $10^3$ | $10^4$ | $2{\times}10^6$ | 20 |
| Solar Corona | $10^9$ | $10^6$ | $2{\times}10^9$ | $10^{-1}$ |
| Solar Atmosphere | $10^{14}$ | $10^4$ | $6{\times}10^{11}$ | $10^{-5}$ |
| Warm Plasma | $10^{14}$ | $10^5$ | $6{\times}10^{11}$ | $10^{-4}$ |
| Hot Plasma | $10^{14}$ | $10^6$ | $6{\times}10^{11}$ | $10^{-4}$ |
| Theta pinch | $10^{16}$ | $10^6$ | $7{\times}10^{12}$ | $10^{-5}$ |
| Laser Plasma | $10^{20}$ | $10^6$ | $6{\times}10^{14}$ | $10^{-7}$ |

## 2. It takes 1000 words to say what a PIC is

### 2.1. Plasma and PIC

For a plasma there are a few quantities that are typically used to characterize it. There is the number density of charged particles $n$, which charge neutrality is usually assumed (i.e., $n_e = n_i$). The temperature $T$ is almost always above $10^4$ Kelvin. The more particular quantities are the plasma frequency

$$\omega_p = \left(\frac{4\pi n e^2}{m}\right)^{1/2} \propto n^{1/2}, \tag{1}$$

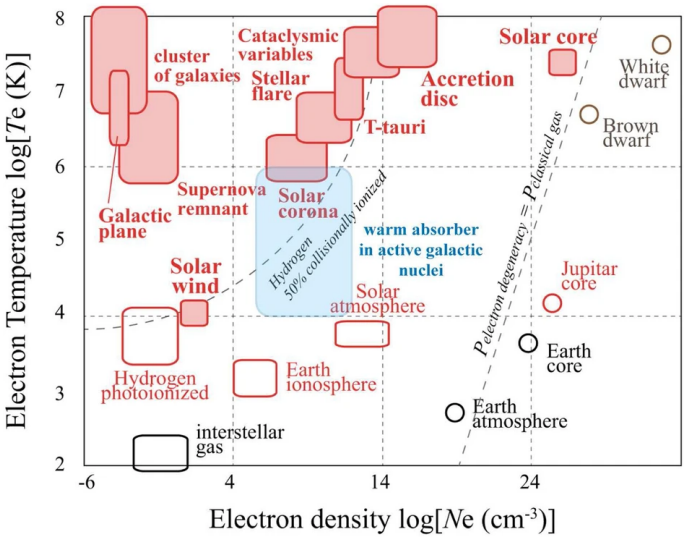


**Fig. 2.** Plot showing various plasmas and their natural systems in a parameter space defined by density and temperature Ezoe (2021).

where $e$ is the charge of an electron and $m$ is the mass of the particle under consideration; and the Debye length

$$\lambda_D = \frac{V_{thermal}}{\omega_p} \propto \left(\frac{T}{n}\right)^{1/2}, \tag{2}$$

where $V_{thermal}$ is the thermal velocity. The plasma frequency $\omega_p$ screens electromagnetic waves of frequency lower than the plasma frequency from being transmitted through the plasma and, in simulations, is usually the fastest temporal scale that needs to be resolved—while the Debye length is the approximate length scale at which collective effects, namely charge screening, begin to happen.

For high-density plasmas, it is typical to use magnetohydrodynamics (MHD), which is essentially a fluid approximation. However, for low-density plasmas, the PIC method provides the best way forward (see Figure 2 for a compendium of plasmas and Figure 1 for various types of simulations). If you are going to simulate plasma as particles—and without using the fundamental distribution function—then you might encounter problems with point-particles. The Coulomb forces essential to the long-range interactions of a plasma can become unwieldy with infinitesimally small particles: consider, for example, two electrons speeding towards each other—once they are close enough the Coulomb force will wildly diverge! To avoid these types of issues the PIC method basically makes the particles into composite particles (also called computational particles or macroparticles); they each represent a little slice of phase space.

The big picture idea of the PIC algorithm is to split up certain operations into a cycle that utilizes a grid where each square element of the grid is a cell where the particles can reside—hence the name particle-in-cell. The first step is to convert particle locations to *weighted* grid points, then solve for the electric potential and electric field, next use those quantities to calculate the forces on the particles due to the Lorentz force

$$\mathbf{F} = (v \times \mathbf{B}) + q\mathbf{E}, \tag{3}$$

where $q$ is the charge of the particle, $v$ is the velocity, and in the case of electrostatics the magnetic field $\mathbf{B}$ is zero; finally, the force is used to solve the equations of motion of the particle based on Newton's second law $\mathbf{F} = m\mathbf{a}$. This is then repeated *ad infinitium* unless there is a stop condition (maybe a limited amount of computer time on a supercomputer); see Figure 3. The time stepping is generally carried out in a leap-frog manner, which is shown schematically in Figure 4.

### 2.2. The two stream instability in two dimensions

To simulate two streams interacting is a great example that shows the usefulness of the PIC method.[1] While this could, it seems, be modeled using two fluids interacting a problem arises when the two streams (viz. fluids) moving at two different velocities start to interact—other velocities begin to emerge. This essentially bifurcates the problem with each new velocity field needing to be its own fluid! Modeling these streams as particles, on the other hand, provides a much more natural way for the system to evolve because each particle can have its own velocity (that being said, remember that these are representative macroparticles). Simulating this classic example will be our first way of showing the *tour de force* that is PIC.[2]

The two stream instability can be solved in the electrostatic paradigm. So, starting with the simpler case of only electrostatic interactions (i.e., no magnetic fields), the Poisson equation

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0} \tag{4}$$

—where $\phi$ is the scalar electric potential, $\rho$ is the charge density, $\nabla^2$ is the Laplacian [why isn't it called the curvature operator (?!) there's more lightning where there's more curvature, anyways, moving on], and $\epsilon_0$ is the "permittivity of free space" a sad result of not using the more elegant Gaussian units—can be solved to then obtain the electric field.[3] There are multiple ways to approach this differential equation Lapenta (2015). Perhaps the simplest way is to implement a finite difference method, while a more optimized algorithm could use the Fast Fourier Transform (FFT) to solve it. Nominally, a finite difference method will have a computational complexity of $O(N^{2d})$ where the factor $d$ represents the dimension of the grid while any FFT-based solution will be order $O(N \log N)$.[4]

For the finite difference implementation, in two dimensions, the electric potential is found by computing

$$\nabla^2\phi = \sum_i \frac{d^2\phi(x_{i,j})}{dx_i^2} \approx \frac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{(\Delta x_i)^2} + \frac{\phi_{i,j-1} - 2\phi_{i,j} + \phi_{i,j+1}}{(\Delta x_j)^2} \tag{5}$$

where the indices $i, j$ represents each dimension and index the grid; this can be optimized by a sparse matrix method. The algorithmic complexity should be clear by how the number of operations scale with dimension and size of the grid. Because of the high computational complexity for a two dimensional grid we choose to optimize with the FFT approach to solving the potential. Solving the Poisson equation in this way means taking the Fourier transform of the equation

[1] https://en.wikipedia.org/wiki/Two-stream_instability.

[2] This problem and the approach taken is inspired by the one dimensional walk-through given here: https://medium.com/swlh/create-your-own-plasma-pic-simulation-with-python-39145c66578b.

[3] The charge density in the PIC method will be found by "scattering" the charges into the grid.

[4] The scaling for finite differences with the factor $d$ would be $O(N^2)$ for 1D, $O(N^4)$ for 2D, *et cetera*.

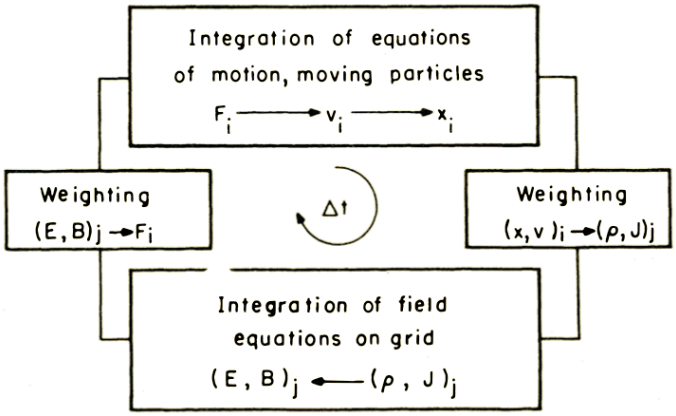


**Fig. 3.** Schematic outline of the PIC algorithm from Birdsall (2005).

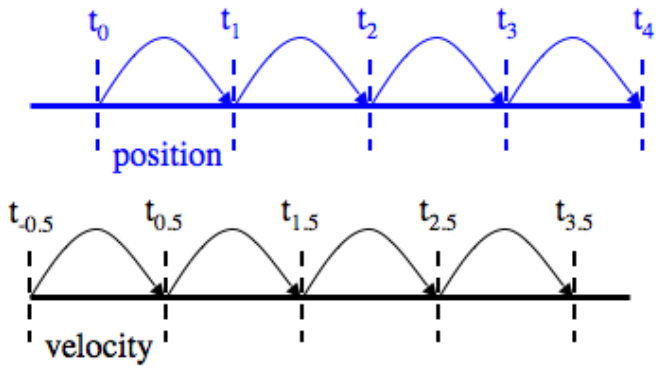


**Fig. 4.** Schematic outline of a leap-frog time stepping scheme Anonymous (2010). Note the position and velocity steps are $\pi/2$ out of phase.

$$\mathcal{F}\left(\nabla^2\phi = -\frac{\rho}{\epsilon_0} = f\right) \rightarrow -k^2\hat{\phi} = \hat{f} \tag{6}$$

where $k$ is the spatial frequency of the grid and $f$ is a shorthand for $\rho/\epsilon_0$. This is, of course, the analytic version. For our discrete case the Fourier expansion of the potential results in

$$\phi(x,y) = \sum_{p,q=-\infty}^{\infty} \hat{\phi_{pq}} e^{-j2\pi\frac{x}{L_x}} e^{-j2\pi\frac{y}{L_y}} \tag{7}$$

where $L_i$ is the length of the grid in the $i$ dimension, $\hat{\phi_{pq}}$ is the coefficient, and $p, q$ index the Fourier components.[5] Once this is substituted back into the Poisson equation the much-easier-to-solve algebraic equation

$$\left[\left(-j2\pi\frac{p}{L_x}\right)^2 + \left(-j2\pi\frac{q}{L_y}\right)^2\right]\hat{\phi_{pq}} = \hat{f} \tag{8}$$

is obtained. This can be solved by dividing the pre-computed charge density by the factor in square brackets.

Once the electric field has been calculated the next step (as per Figure 3) is to find the force using the electrostatic version of Equation 3

[5] For a more detailed version of this mathematical reasoning, please see: https://math.stackexchange.com/questions/1809871/solve-poisson-equation-using-fft

$$\mathbf{F} = q\mathbf{E}, \tag{9}$$

which can then used to solve the equations of motion for the particles. Considering only electrostatic forces reduces the equations of motion to

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i \quad \text{and} \quad m_i\frac{d\mathbf{v}_i}{dt} = m_i\mathbf{a}_i = -q_i\mathbf{E}(r_i), \tag{10}$$

where $v_i$ is the velocity of each particle, $r_i$ the location, $m_i$ the mass, $q_i$ the charge, and $a_i$ is the acceleration. This is done for every particle in the simulation. Once all the particles have been moved by the "particle pusher" they will be in their new locations and the cycle can begin again!

### *2.3. High N is where the wild things (plasma instabilities) are*

With a working PIC algorithm in hand it is possible to start simulating plasmas. Starting with low density will probably be a little disappointing visually. And attempting to put in a lot of particles, in two dimensions, with a finite difference method will probably take longer than the age of the universe to solve. So, if you want to see some fireworks *per se* then take note. Assuming you have gone over Table 1 and succeeded in implementing an FFT solver you might have been able to extrapolate that high densities result in more interesting types of plasma.

The standard way to characterize the plasma, rather than the menagerie of physical characteristics that can be listed, is something called *the plasma (coupling) parameter*. Generalizing the Debye length (Equation 2) to three dimensions results in something called (you guessed it) the Debye cube and the number of particles per Debye cube

$$N_D = n\lambda_D^3, \tag{11}$$

is pretty straight forward; and it is related to the average distance between particles $a = n^{-1/3}$. In plasmas there are competitive effects from thermal interactions (collisions) and electromagnetic interactions (screening/scattering and collective effects). Thermal interactions dominate when the temperature or kinetic energy (i.e., velocities) is high $E_{thermal} = k_BT$ while electromagnetic interactions take over when the electric potential and magnetic field are strong $E_\phi = e^2/4\pi\epsilon_0 a$. By taking the ratio of these two energies the plasma (coupling) parameter is rendered

$$\Lambda = \frac{E_{thermal}}{E_\phi} = \frac{4\pi\epsilon_0 a k_B T}{e^2} = 4\pi N_D^{2/3} \tag{12}$$

where the Debye cube and average distance between particles were used to simplify the expression into the right hand side (the order unity numbers in front can change if you want to get more into the statistical mechanics; see Gurnett (2017)); note that in the electromagnetic case there is the related $\beta$ plasma parameter where the magnetic energy density $\mathbf{B}^2/2\mu_0$ is used in place of the electric potential.

In conclusion, for plasmas with $\Lambda \gg 1$ the plasma is "weakly coupled" and mostly thermal motions; conversely, when $\Lambda \ll 1$ the plasma is "strongly coupled" and exhibits more collective effects. When the plasma is "strongly coupled" it will have less collisions and will therefore have less ability to become unstable. (When magnetic fields are involved the analysis becomes a

little more complicated because there are magnetic instabilities that can crop up.) However, for the electrostatic case under study with our PIC algorithm it is safe to say that only when $N$ is high will there be any instabilities. Moreover, it shouldn't be too hard to figure out the value of $\Lambda$ when instabilities start to happen.[6]

## 3. There's a lot of bits in this git

The implementation of the algorithm outlined above has been carried out in a `jupyter` notebook, which has been uploaded to a repository on GitHub.[7] The code has been implemented in a functional programming style using script-like execution. A few of the subroutines are described below.

The particle-in-cell algorithm used in this program comes in four steps; computing charge density, computing the electric potential, computing the electric field, and updating the position and velocity of each particle.

The first step is performed in the `getAccel` function. A particle's position is counted on the four nearest grid points, which are then binned to a two-dimensional array using `scipy.stats.binned_statistic_2d`. The weight of a particle to each of the four nearest grid points is determined by the distance of the particle from each grid point (see Figure 5). As there is only one species of particle in this simulation, this two-dimensional array of grid points is taken to be the charge density. Because simulating a plasma where each particle represents a single electron would be prohibitively taxing computationally (where there would be on the order of Avogadro's number of particles), each particle in the simulation is actually a "macroparticle" representing a number of actual particles set by the *specific weight*. For example, if the *specific weight* of each particle was set to 100, each simulation particle would be given charge 100, to represent 100 actual particles.

The second step, computing the electric potential, is performed by the `phi_compute` function. A Fast Fourier transform (FFT) is performed using `scipy.fft.fft2` on the charge density. The Fourier transform of the electric potential is then computed in Fourier space, and finally the inverse Fourier transform is performed using `scipy.fft.ifft2` to compute the electric potential in real space. FFT methods work well in this simulation because the boundaries are all periodic.

The third step is performed in the `getAccel` function. The electric field is computed at each grid point using `numpy.grad`, which uses central differencing to compute the derivative. The acceleration is given by taking the negative of the electric field, as the charge (and mass, for that matter) of each particle has been set to one in this simulation.

The fourth and final step is updating the position and velocity of each particle using the leap frog method. This is performed in the main loop of the program, using the acceleration computed with `getAccel`.

## 4. Taste testing the numerical recipes

Our code was tested by comparing it with other examples in the literature. Most common was a *phase space* plot with position on the $x$-axis and velocity on the $y$-axis. This type of plot with a characteristic "swirl" of the two stream instability is shown in Figure 6. By manually varying the number density of particles

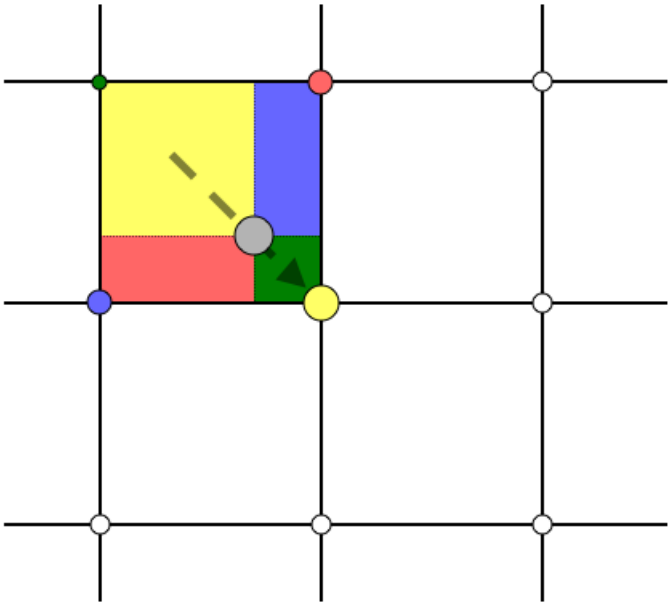

**Fig. 5.** Schematic view of the "scatter" grid weighting process Anonymous (2010).

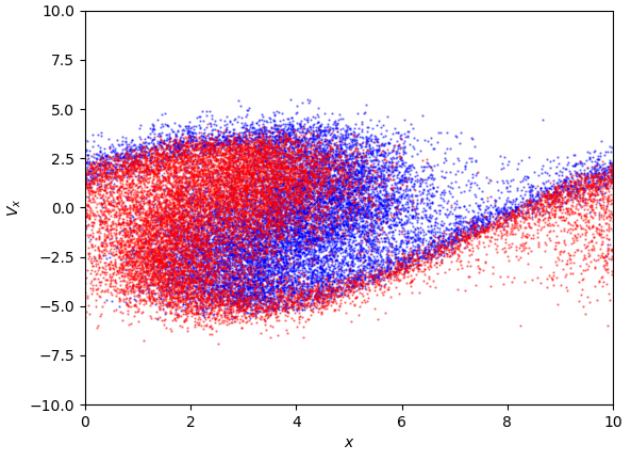


**Fig. 6.** Plot showing characteristic "swirl" of the two stream instability.

the onset of the instability could be measured. It required about 1 to 4 million particles to obtain this condition, with much more or less particles resulting in asymptotic cases that would probably be uninteresting to a plasma physicist.

The results of timing the finite difference and FFT methods are shown in Table 2. This test had to be run with a paltry two particle population because otherwise the finite difference method took an exorbitant amount of time. These results confirm the computational complexity analysis set forth and covered above in Section 2. The FFT electric potential was also compared with a true analytic point particle; see Section 5 and Figure 8. More related issues are addressed in Section 6.

## 5. Enter the Brillouin zone: the computational plasma crystal lattice

The PIC method is classified as a particle-mesh algorithm. By subjecting the particles, which are once again macroparticles, to

[6] This is similar to the Reynolds number in fluid mechanics, which can be used to determine if a fluid will be in the diffusive, laminar regime or the turbulent regime.

[7] https://github.com/evalch3/512-PIC-Project

**Table 2.** Algorithm timing.

| Name | n | steps | $t$ [$sec$] |
|---|---|---|---|
| FFT | 2 | 100 | 2.28 |
| Finite Difference | 2 | 100 | $1.56 \times 10^3$ |

a grid they are, in effect, being put into a crystal lattice.[8] This means that when the FFTs described are being performed the particles' modes in $k$-space will not be fully sampled. This spatial Fourier filter is important. The smallest scale or $k$-mode that needs to be sampled correctly in a plasma is on the order of the Debye length (Equation 2). A good proxy for this—holding temperature constant—could be $n^{-1/2}$; which is also related to the fundamental time-frequency of the simulation: the plasma frequency (Equation 1), it's approximately the inverse!

If the grid is not sufficiently fine it can cause aliasing, which generates numerical noise, heating, and in the worst case destabilization of the simulation. The Brillouin zone or (primitive) cell that the particles can find them selves in should uphold the Nyquist sampling criterion:

$$-\frac{k_{grid}}{2} \leq k \leq \frac{k_{grid}}{2} \tag{13}$$

where $k_{grid} \equiv 2\pi/\Delta x$ is the *grid wave number*. These effects would be hard to extract from the simulation but should be visible by varying the grid size with respect to the particle density. The rectilinear shape of the grid will introduce a top-hat-like window function, which Fourier transforms into a *sinc* function—this is what is ultimately spreading stuff around (i.e., "spectral leakage"). It will tend toward the analytic limit in Equation 6 as the grid spacing goes to zero.[9]

Along with this numerical error comes another numerical nuance introduced by the finite difference used to obtain the electric field. This will numerically damp high $k$ modes. By running the simulation with varying size grids these two effects can be seen. A "scale factor" is used to make the size of the grid bigger while the number of sampling points is held constant. For larger grids the sampling will be more sparse and there should be much more numerical errors, specifically there should be more aliasing with any numerical dampening from the finite difference being swamped by spectral leakage.

To extract this information the coefficients in Equation 7 were captured and summed with a weighting function that highlighted the high $k$ modes. The results are shown in Figure 7. From the plot it can be seen that as the grid size gets larger (i.e., the value of $k_{grid}$ gets smaller) the long term behavior of the plot becomes less damped because of more aliasing. For the medium resolution grids the high $k$ modes quasi-oscillate and do not systematically disappear, but for the highest resolution grid (with the highest $k_{grid}$ value) the numerical dampening is the most apparent. It seems the high $k$ modes have gotten lost in the burgeoning Brillouin zone, with no alias to go by.

[8] Incidentally, there are certain types of plasma that can generate crystal lattice-like behavior! See, for example, this quick introduction: https://sites.baylor.edu/eva_kostadinova/2019/05/10/__trashed-2__trashed/

[9] Page 18 of Birdsall (2005) covers this in more depth.

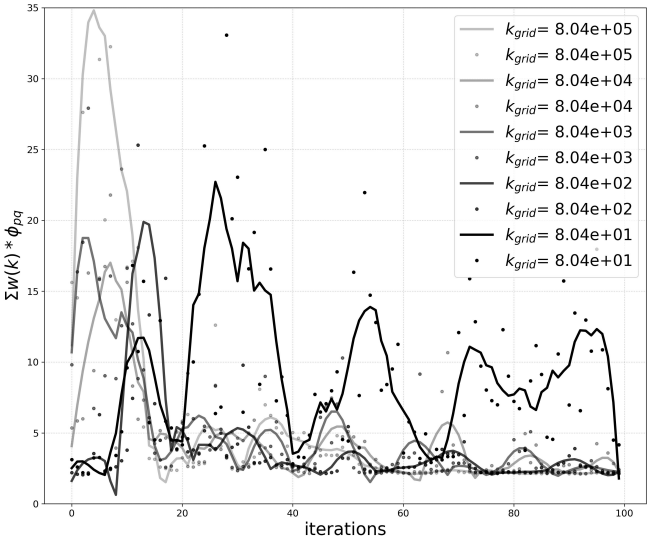


**Fig. 7.** Each run is shown with actual data points and a smoothed representative version (made using a Savitzky–Golay filter). The highest $k$ over-resolves $\lambda_D$ while the lowest does not resolve $\lambda_D$.

## 6. Visions of a bigger phase space

There's the age old question. *What could be better?* This is a great venturing off point into the abyss of plasma physics. Only so much can be done with a laptop, so there are limitations. If the computational processing power required (and/or patience) is neglected then the list of things that could be done with this foundation is quite long.

First, generalizing this simulation to three dimensions (3D) would be a relatively straight forward next step, although visualizing the results will become increasingly more difficult. In the current format, a simulation of the ionosphere reacting to a solar eclipse was the original long-term objective—and this could probably be realized in some capacity. One way would be to have source and sink functions, with the sink functions modeling the recombination of electrons and ions in the shadow of the moon. This would probably not recreate the bow shock observed during the real eclipse because that might require more attention to address things like supersonic effects.[10] Part in parcel to this would also be implementing different species of particles with different masses and charges.

Assuming the magnetic field was negligible, in the case above, and the full Maxwell equations didn't have to be incorporated, then the next step would be to add them to the PIC algorithm. Using the full equations is the *sine qua non* of plasma physics because without them there isn't any Alfén waves and so on and so forth. It would require solving another field and more finite differences in the equations of motion. It's doable but a large undertaking and might push the limits of laptop simulations for an interesting number of particles. Besides essentially making the phase space larger and using more computing resources to see more fantastic plasma physics there are a few more practical concerns.

There was interesting structure that could have been explored in more depth that seemed to appear at longer time scales, which is shown in Figure 9. Before reading too far into it a window function (a "particle shape" in the plasma literature) should be used to reduce aliasing effects. A better, less *ad hoc* scheme

[10] https://phys.org/news/2018-01-solar-eclipse-earth-atmosphere.html

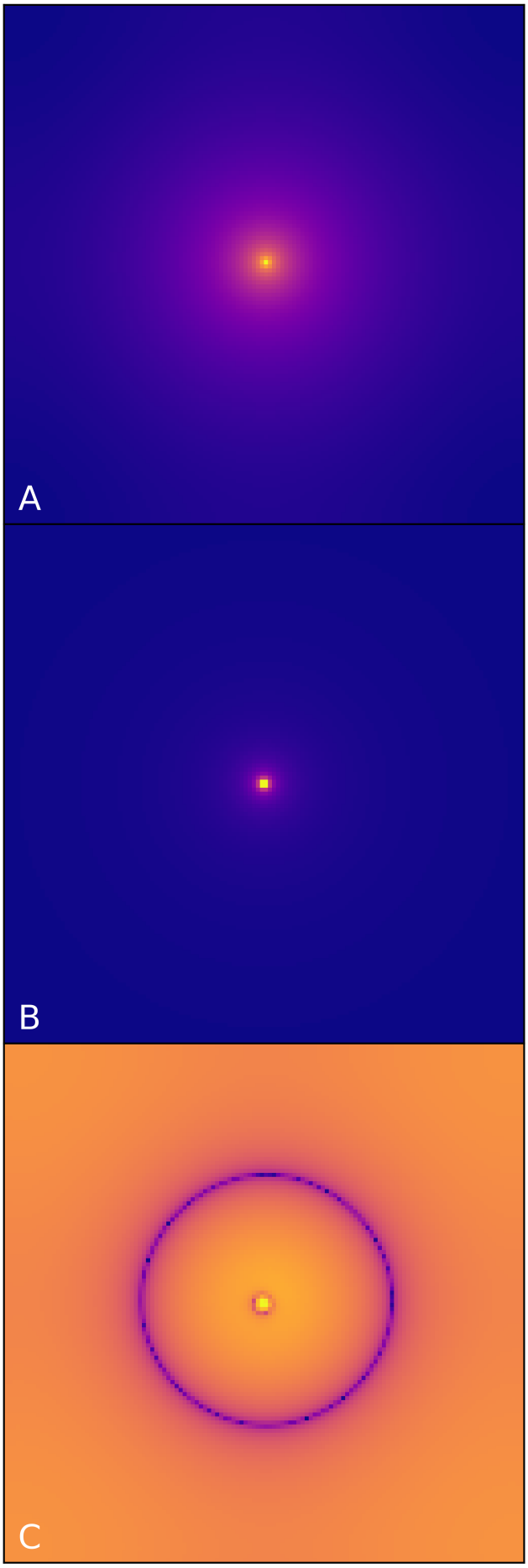


**Fig. 8.** Plot comparing the FFT potential for a single particle (i.e., a macroparticle) with an analytic result. In plot **A** there is a single macroparticle potential, which looks Gaussian; plot **B** shows the analytic single particle with the well-known $1/r$ potential; and plot **C** is a difference image (**A**-**B**) shown in logscale with a *sinc*-like structure.

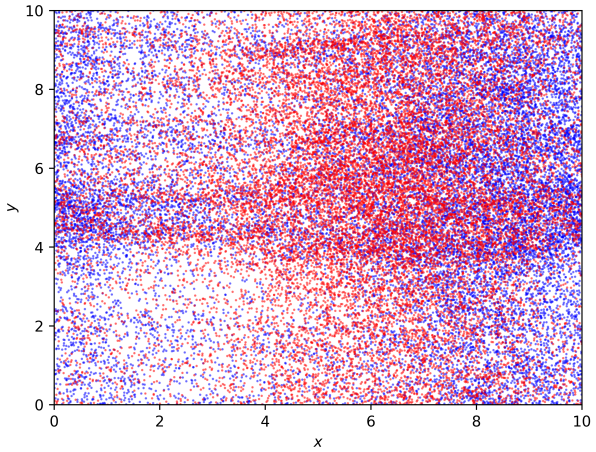


**Fig. 9.** Plot showing undetermined structure on long time scales.

for extracting telemetry from the simulation would be good—enabling one to plot the velocity distribution and compare it to a Maxwellian, for example. Ideally, this would be visualized in an aesthetically appealing dashboard format with a button to stop the simulation if things go awry. Lastly, plasma *in situ* is generally smooth in appearance—it looks like silky energy dancing around; there might be a way to go from a very granular particle scatter plot to a more natural looking representation of the simulation by an unknown "beautification process" that is probably explained elsewhere, at length, in the literature.

## 7. Conclusion

In conclusion, this was a grand romp through the shallow waters of the abyss. The two stream instability was successfully simulated in two dimensions. When taking specific weight into account, the two stream instability was clearly apparent for approximately 1 to 4 million particles.[11] This is only the beginning. There is so much more to explore, in fact, there's a whole textbook Birdsall (2005) and then, at last, there's the vast uncharted *terra incognita* of plasma physics.

## 8. Statement of Contributions

Majority of code written by Evan. Majority of report written by Kyle. Both helped each other with both. The topic was a joint and unanimous choice.

## References

Lapenta, G. 2015, Kinetic Plasma Simulation: Particle In Cell Method: 10.13140/RG.2.1.3319.2801.

Ezoe, Y., Ohashi, T. & Mitsuda, K. High-resolution X-ray spectroscopy of astrophysical plasmas with X-ray microcalorimeters. Rev. Mod. Plasma Phys. 5, 4 (2021). https://doi.org/10.1007/s41614-021-00052-2.

Anonymous, nice and informative website about the plasma particle-in-cell method. https://www.particleincell.com/2010/es-pic-method/.

Spitkovsky's online lecture at the Institute for Advanced Study about particle-in-cell method. https://www.youtube.com/watch?v=I09QeVDoEZY

Gurnett, D. A. and Bhattacharjee, A. *Introduction to Plasma Physics*, (Cambridge 2017).

Birdsall, C. K. and Langdon, A. B. *Plasma Physics via Computer Simulation*, (Taylor & Francis 2005).

[11] For fewer and more particles (holding the grid size constant) the two stream instability was not seen because $\lambda_D$ is not resolved for high $n$.