Two Stream Instability Simulation in a Collisionless Plasma

Lujain AlMarhabi
Instructor: Professor Tom Kelley
Project Advisor: Professor Oleg Batishchev
Northeastern University Department of Physics

November 29, 2018

Abstract

The two-stream instability is a rapidly growing collisionless plasma instability arising from small charge imbalances. A local imbalance leads to the acceleration or deceleration of particles in its vicinity, which in turn leads to an even stronger imbalance. One setup that allows to easily observe the instability is two counter-streaming beams of identical charge in a periodic system. The advantage of this configuration is that the generated plasma wave becomes a standing wave, thus allowing to easily observe the formation of the phase space vortices. The goal of this project to simulate the formation of the phase space vortices (instabilities) for a 1D1V non-relativistic electrostatic low-density plasma using PIC and SOR algorithms in Python.

1 Motivation

A plasma is a partially or completely ionized gas. Approximately 99.99% of the visible matter in the universe exists in the state of plasma, as opposed to a solid, fluid, or a gaseous state. Matter assumes a plasma phase if the average velocity of particles in a material achieves an enormous magnitude, hence all matter, if heated to a significantly great temperature, will reside in a plasma state. When a collection of charged particles in a plasma is of low density the plasma is deemed to be "collision-less", as collisions between particles become infrequent.

The extent to which a plasma is ionized (α) is of great concern in many plasma physics, but for the scope of this project, we will assume that $\alpha = 1$, i.e. plasma is approximately fully ionized. The equations to explain the behavior of system of particles in the plasma are extremely complex as each ion and electron are individually charged, with opposite signs. Each ion will be attracted to the electron and vice versa, while ions repelling other ions and electrons repelling other electrons. That is, the current density and charge density generated by single-particle motion feeds back and alters the original applied electric and magnetic field generating a "self-consistent" system. The challenge of calculating and modelling the overall behavior of a plasma is very great indeed. Many of the computational techniques and methods discussed in this paper were designed

specifically for plasma simulations as far back as the 1950's yet these techniques are still used today with a great success.

2 Plasma Physics

We start with the Vlasov-Poisson equation which describes the kinetic collision events of a plasma interaction more accurately than the Boltzmann equation for long range Coulomb effects. Let the plasma be cold, $KT_e = KT_i = 0$, and let be no magnetic field. Therefore, for a 1D with a non-relativistic velocity v, the equation is given by:

$$\frac{\partial f}{\partial t} + \vec{v}.\frac{\partial f}{\partial x} + \frac{qE}{m}.\frac{\partial f}{\partial \vec{v}} = 0 \tag{1}$$

Essentially, this above equation comes from the conservation of particle's density, $f(x, v, t)$, along its trajectory, i.e. $\frac{df}{dt} = 0$. However, for a steady state case, i.e. $\frac{\partial f}{\partial t} = 0$ since it doesn't explicitly depends on time and equation 1 is simplified to:

$$\vec{v}.\frac{\partial f}{\partial x} + \frac{qE}{m}.\frac{\partial f}{\partial \vec{v}} = 0 \tag{2}$$

The motion of each particle is governed by the Newtons second law:

$$\frac{\partial x}{\partial t} = \vec{v}$$

$$\frac{\partial \vec{v}}{\partial t} = \frac{q}{m}(\vec{E})$$

We will attempt to find the x and v that satisfies the Vlasov equation of motion above. The Poisson's equation for a self-consistent electric field:

$$\nabla^2 + \rho = 0 \tag{3}$$

Will be used to calculate the potential field $\Phi$from a charge density ρ.

3 Particle-In-Cell Method

One of the main computational tools for simulating the behavior of collisionless plasmas is the particle-in-cell (PIC) method, which combines finite-difference approximations with interpolation and averaging techniques in order to track computational particles called superparticles that represent the real ions, electrons and neutrals. The basis of this technique is to discretize the system into several small cells with nodes at the intersections.

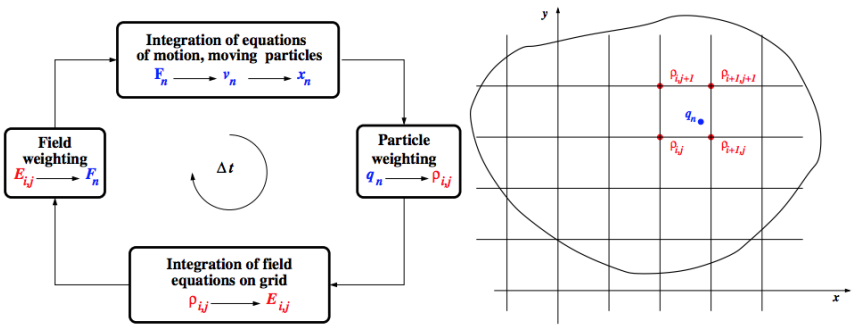


Figure 1: Left: basic cycle of one time iteration. Right: sketch of the relation between particle charge and density mesh.

The PIC method can be simplified into a four-step iteration process that loops many times over the course of the entire calculation as shown in Figure 1.

3.1 Algorithm

The PIC method consists of an initial setup, the main loop, and a final clean-up/results output. All computation happens in the main loop and it consists of the following steps:
1. Compute Charge Density: particle positions are scattered to the grid.
2. Compute Electric Potential: performed by solving the Poisson equation.
3. Compute Electric Field: from the gradient of potential.
4. Move Particles: update velocity and position from Newtons second law. (integrate particle motion through a time step Δt).
5. Generate Particles: sample sources to add new particles.
6. Output: optional, save information on the state of simulation.
7. Repeat: loop iterates until maximum number of time steps is achieved or until simulation reaches steady state.

4 Numerical Methods

4.1 SOR

The successive over-relaxation (SOR) method is an example of a classical iterative method for the approximate solution of a system of linear equations. It solves Poisson's equation by converting the entire problem into a system of linear equations of the matrix form:

$$Ax = b \tag{4}$$

This means it generates a sequence of numbers that update the approximate solutions for a specific problem and is guaranteed to eventually converge on the correct solution depending on the relaxation parameter w. This parameter

depends on the grid spacing, the geometrical shape of the domain, and the type of boundary conditions imposed on it. As long as the condition: $0 < w < 2$ is enforced the method will converge. The only tricky part is choosing the ideal value for w that minimizes the time to convergence. Since the proof is rather involved therefore generally speaking we can use the following formula:

$$w = \frac{2}{1 + \frac{\pi}{N}} \tag{5}$$

Where, N is the number of grid samples (nodes). To see how one can computationally implement this method we show the following derivation. Starting with Poisson's equation:

$$\nabla^2 + \rho = 0$$

Then converting it into a linear system of equations using 4:

$$(Ax)w = wb$$

Using the (L-U) decomposition method we rewrite matrix A as its diagonal (D), upper (U), and lower (L) components:

$$(wD + wU + wL)x = wb$$

After a series of algebra, we obtain an equation for x:

$$x = (D + wL)^{-1}.[wb - [wU + (w-1)D]x]$$

Taking triangular form of $(D + wL)$ and applying forward substitution to find a solution:

$$x = (1-w)x + \frac{w}{a}(b - \sigma)$$

$$x_{new} = (1-w)_{old} + \frac{w}{a}(b - \sigma)$$

$$x_{new} = (1-w)_{old} + \frac{w}{a_{ii}}(b_i - \sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k)$$

σ in above equations is the difference in sums between new and old values (k+1 and k), where k is the current iteration and time step.

Then if we substitute in variables for charge potential and charge density we obtain the following equation:

$$\Phi_i^{k+1} = (1-w)\Phi_i^k + \frac{w}{a_{ii}}(b_i - \sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k \tag{6}$$

Φ_i^{k+1} is the new value of x expressed in its old values a and b and they're matrices in a linear system of equations that are equivalent to A and b in 4. Evidently, Φ_i^k is the old value of x. Finally, this is the form that will be useful to numerically solve Poisson's equation.

4.2 Triangular Matrix and Forward Substitution

In numerical analysis and linear algebra, lower–upper (LU) decomposition or factorization factors a matrix as the product of a lower triangular matrix and an upper triangular matrix. Forward substitution is the process of solving a system of linear algebraic equations (SLAE) on form of equation 4 $Ax = b$. By factoring A as a product of two triangular matrices $A = LU$. Where, L,U are the lower and upper triangular matrices respectively. And we can rewrite the L-U decomposition of matrix A as the following:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} + \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \tag{7}$$

Using forward substitution to solve a matrix equation, a system of equations, for x where a is now lower elements of the lower matrix of A.

Thus,

$$\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{8}$$

Starting with the first terms of a and x, we substitute the values forward. Eventually, we obtain the equation for any x term of the matrix. This outlined below by the following algorithm:

$$x_1 = \frac{b_1}{a11}$$

$$x_2 = \frac{(b_2 - a_{21}x_1)}{a22}$$

$$\vdots$$

This algorithm may be simplified as the following summation:

$$X_m = \frac{b_m - \sum_{i=1}^{m-1} a_{m,i}x_i}{a_{m,m}}$$

This above equation is actually equation 6 before we apply the SOR method to it.

4.3 Finite Difference Method

We use this method to approximate for the Laplace operator which represents matrix A in equation 4. x is the charge potential Φ and matrix b is the charge density ρ.

4.3.1 The Five-Point Star

The first step in applying FDM is to define a mesh, where the electric potential function will be sampled. Letting h be the distance between each sample, the points that lie on the mesh may be defined by: $x_i = ih$, $y_j = jh$

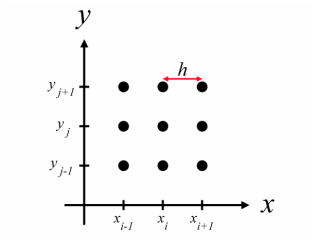


Figure 2: Mesh points for the FDM grid.

In practice, i and j will eventually be used as indices for a matrix of potential samples. We shall therefore replace the spatial coordinates with simple indices by assuming the following convention: $\Phi(i,j) = \Phi(x_i, y_j)$ Similarly, we define the charge density samples along the same mesh by using the $\rho(i,j)$ notation. The next step is to expand the Poisson equation by explicitly showing the partial derivatives in space:

$$\frac{\partial^2 \Phi(i,j)}{\partial x^2} + \frac{\partial^2 \Phi(i,j)}{\partial y^2} = -\rho(i,j) \tag{9}$$

In one dimension, the Laplacian operator can be simplified in fractional form as the following:

$$\frac{\partial^2 \Phi(i,j)}{\partial x^2} \approx \frac{\Phi(i-1,j) + 2\Phi(i,j) + \Phi(i+1,j)}{h^2} \tag{10}$$

Then equation 10 can be expressed in matrix representation with the contributing terms across the diagonal:

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & & \dots & \\ & & & -1 & 2 \end{bmatrix} \tag{11}$$

Each row represents an equation with a shifted indices. In addition, the value of constant a_{ii} in equation 5 can be found using the matrix above. This leads to the value of a_{ii} = -2 as shown in the code.

4.4 Plasma Perturbation

A longitudinal plasma wave manifests itself as a disturbance in the electron density. We can excite such a wave computationally by displacing the initial positions of the particles:

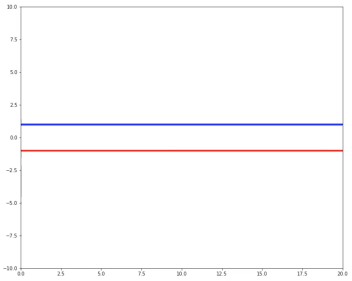
$$x_i = x_{0,i} + \vec{x_i}$$

Where, $x_{0,i} = (i + 0.5)\Delta x$, and, $\vec{x_i} = \Delta x_0 \cos(2\pi \cdot mode \cdot \frac{x_{0,i}}{X})$ X is the grid length and note that for large amplitudes Δx_0, some particles may be displaced across the simulation boundary, so a periodic boundary conditions must be used to avoid 'losing' particles at the edges.

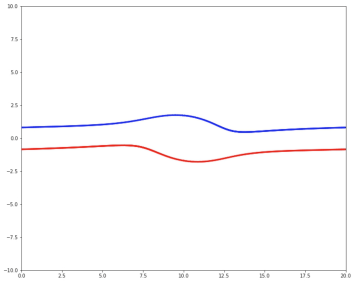
5 Results and Discussion

One of the basic problems of plasma physics is the two-stream instability. It describes the dynamics of two streams of plasma passing close to each other in opposite directions. The dynamics include a fluctuation, vortices, and can generate an oscillating instability over time. Using the following packages in Python3: NumPy, mathplotlib, and ffmpeg we were able to produce a 1D simulation of a basic two stream instability. The behavior shown in the our simulation was found to be match predictions from literature and other previous simulations.

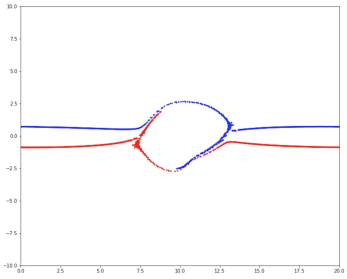
The below figure illustrates the development of the two-stream instability in phase space taken from the first few seconds of the animation. It is found computationally that such configuration is unstable, as after some time the nonlinear development of the system demonstrates the persistence of "holes" in phase space. These "holes" or vortices shown in figure 3 part (c) represent particles that are trapped by oscillating with the plasma frequency about a hole. Other particles are un-trapped and move over each hole.



(a) Two-stream distribution



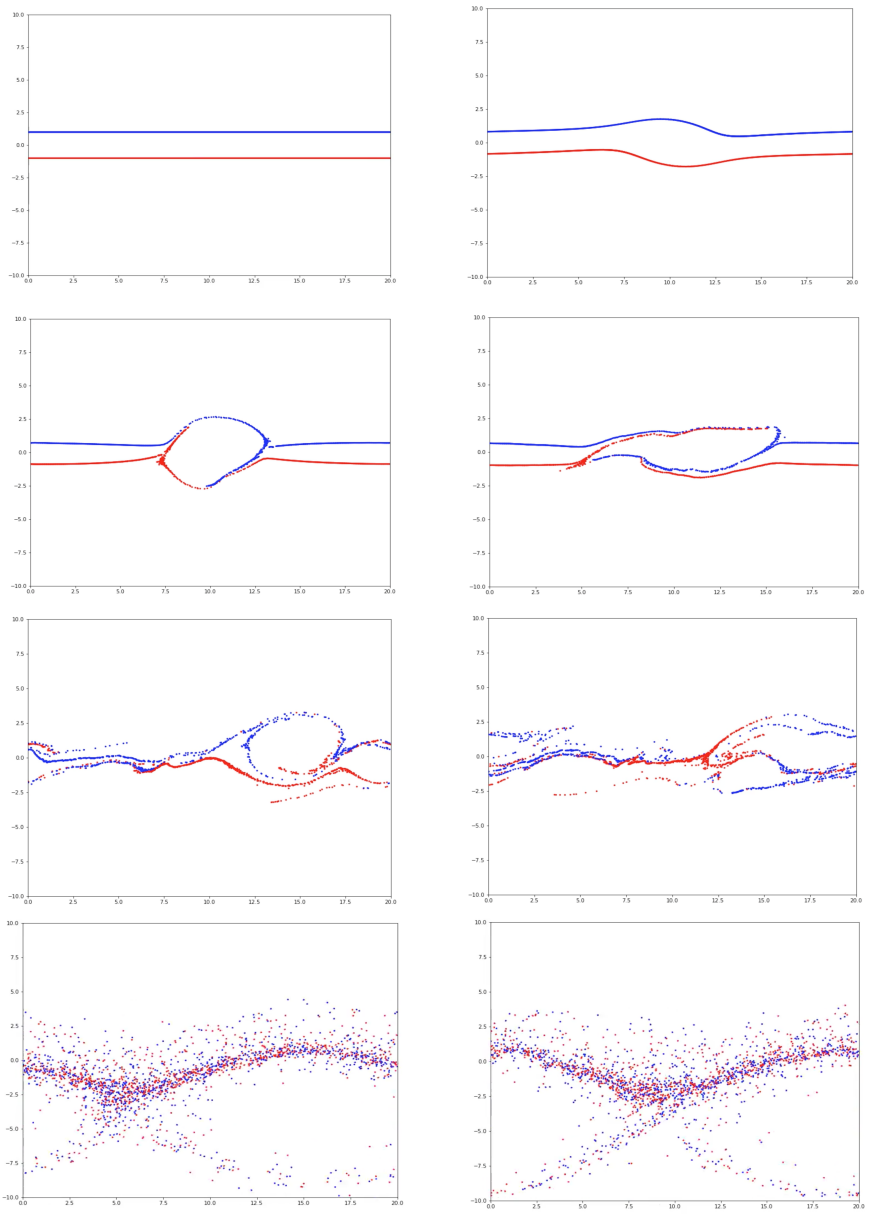
(b) Linear-phase of instability



(c) Non-linear phase-mixing of particles

Figure 3: Development of two-stream instability in phase space

The simulation was performed for 2000 particles with 0.1 perturbation amplitude and 30fps and the following produced set of images below were taken from the animation.



The last two set of images above show an instability that grows over time as expected. These images are from a 17 seconds animation generated from the 500 step PIC code, with 2000 particles, and 100 nodes. To store the data needed for this animation we made one file for each step. Within each file, we recorded the position and velocity of each of the 2000 particles.

6 References

1. P. Gibbon, KU-Leuven FZ-Juelich (2013, November). ESPIC: a simple 1D1V electrostatic PIC code. Retrieved from https://indico.cern.ch
2. Brieda, Lubos (2015, April 29). Two Stream Instability Javascript Simulation. Retrieved from https://www.particleincell.com
3. Nikas, Nicole (2015, October 16). Using the Relaxation Method to solve Poisson's Equation. Retrieved from http://storm.cis.fordham.edu
4. C.K. Birdsall, A.B. Langdon, Plasma Physics via Computer Simulation, McGraw-Hill, New York. See especially Chapters 14, 15, 16 on bounded plasmas.
5. D. Potter, Computational Physics, John Wiley and Sons Ltd, UK.
6. Guio, Patrick (2012, November).PicSim: 5D/6D Particle-In-Cell Plasma Simulation. Retrieved from http://www.ucl.ac.uk