Particle-In-Cell simulations of the two-stream instability

Yoav Heller, Charlotte Orgonasi

Our objectives with this project

In plasma physics, the particle-in-cell (PIC) method is a numerical method which approximates a plasma by a finite number of charged particles.

Our project implements the PIC method step by step for a one-dimensional electrostatic system governed by the Poisson equation. A classical plasma phenomenon, such as the two-stream instability, can then be studied numerically. More specifically, we tried to find back what were the conditions that let to this behaviour, where is considered the interpenetration of two counterstreaming electron beams with equal density and opposite velocity :

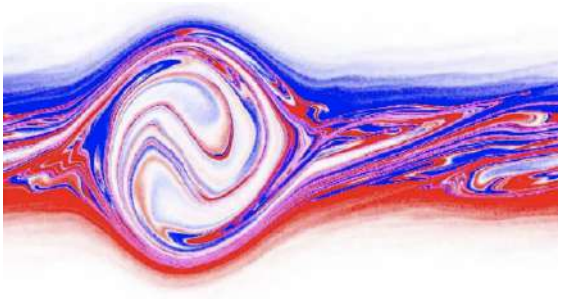


Figure 1. The two-stream instability

Our main objectives for this project were then to :

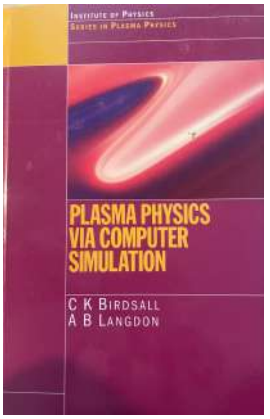
- solve Poisson's equation numerically for the electric field generated by a charged particle distribution
- simulate the behavior of charged particles interacting with an electrostatic field
- find the conditions were in which the picture was taken.

We both chose this topic for our interest in fluid mechanics. Charlotte is doing the PA specialised in nuclear physics and wants to do research in thermohydraulics in the future. Yoav has been studying plasma physics, and currently is doing research for LULI. In the future, he would like to join the global effort to make nuclear fusion a green energy resource.

Table of contents

Theory

A. Principles



Our model consists of charged particles moving about due to the Lorentz force and their own electrostatic field. The physics comes from two parts, the fields produced by the particles and the motion produced by the forces.

For this part, the book *Plasma Physics via Computer Simulation* by C.K. Birdsall and A.B. Langdon was especially helpful.

B. The Physics

The fields are calculated from Maxwell's equations by knowing the positions of all the particles and their velocities.

$$E = -\nabla \phi \quad (1) \quad \text{and} \quad \nabla E = \frac{\rho}{\epsilon} \quad (2)$$

where ρ is the current density and ϵ is the permittivity.

We have an electrostatic problem, therefore the differential equation we have to solve is Poisson's equation:

$$\nabla^2 \phi = -\frac{\rho}{\epsilon} \quad (3)$$

The forces on the particles are found using the electric field : $F = qE$ (4).

We do not use Coulomb's laws because there are too many particles to calculate it one by one.

We proceed discontinuously in time, step by step. At each time, the program solves the field from the particles and then moves the particles. We then repeat this step. The cycle starts at t = 0 with the appropriate initial conditions on positions and velocities (see D.2). The computer runs for any given number of steps.

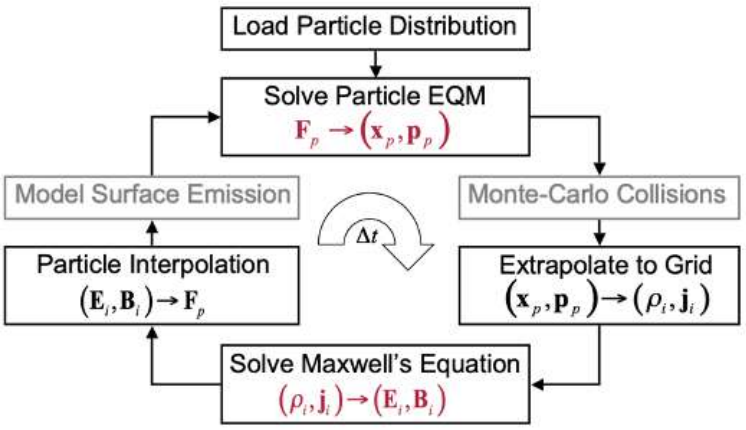


Figure 2. A typical cycle

C. The grid

A grid is set into the plasma region in order to measure charge density. From these we will obtain the electric field at the nods of the grids (which we could store, at each step, in an array). The cells have a width of dx, and have to be smaller than the Debye length because they need to be small enough to resolve a Debye length in order to measure the charge density and thus the electric field.

D. Integration of the equations of motion

D.1. The leap-frog method

One of the fastest and easiest methods we can use to integrate is the leap-frog method. To calculate at each step the positions x_i and velocities v_i we use:

$$m\frac{dv}{dt} = F \quad (5) \quad \text{and} \quad \frac{dx}{dt} = v \quad (6),$$

which give the finite equations: $m\frac{v_{new} - v_{old}}{\Delta t} = F_{old}$ (7)

$$\text{and} \quad \frac{x_{new} - x_{old}}{\Delta t} = v_{new} \quad (8).$$

We then advance v_t and x_t every Δt, but v and x are not at the same time. When x is calculated at t, v is calculated at t - $\frac{\Delta t}{2}$. We took that into account when considering the initial conditions.

D.2 The initial conditions

We start to compute the velocities at $v(-\frac{\Delta t}{2})$. We then have to push $v(0)$ back to $v(-\frac{\Delta t}{2})$ using the force F calculated at t=0.

E. Calculating the densities

It is necessary to calculate the charge density on the grid points to calculate the force applied at the particles. We consider particles to be points, with a gaussian-shaped density, with the full width at half maximum being $2dx$. The density in each cell is the sum of the particles densities that are at this cell.

F. Obtaining the electric field

We now have the densities ρ_i . In order to obtain the electric field in each field, we use ϕ_i . $\rho_i(x)$ and $\phi_i(x)$ both have Fourier transforms, $\rho_i(k)$ and $\phi_i(k)$, where k is the wave vector in the Fournier transform. This allows us to obtain $\phi_i(k)$ from $\rho_i(k)$:

$$\phi_i(k) = \frac{\rho_i(k)}{\epsilon.k^2} \quad (9)$$

The next step is to take the inverse Fourier transform of $\phi_i(k)$ to obtain $\phi_i(x)$ and then $E_i(x)$ using (2).

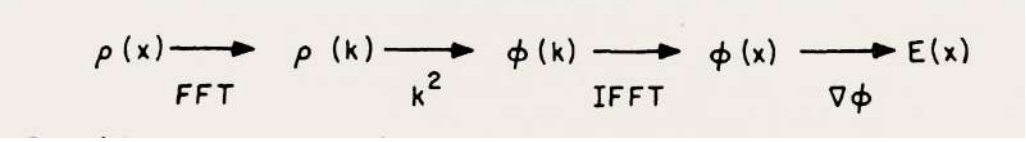


Figure 3. Diagram to obtain E

We can then obtain the field at position x_i: $E_i(x_i) = (\frac{X_{j+1} - x_i}{\Delta x})E_j + (\frac{x_i - X_j}{\Delta x})E_{j+1}$

where *j* is the position of the nearest grid point of x_i.

Our code structure

The code is structured into three modules: production, analysis and visualization.

A. Production

This module runs the simulation and handles the core functionalities of the Particle-in-Cell (PIC) method. The classes defined are Particle, Field, and PicSimulation.

The `Particle` class models individual particles with attributes such as position, velocity, charge, and mass.

The `Field class` represents the electric field and includes a function to calculate the charge density using various weighting methods (square-shaped, triangular-shaped or a gaussian). We finally opted for a gaussian density.

The `PicSimulation class` manages the simulation, initializing particles, solving Poisson's equation to compute the electric field, and advancing the simulation step by step. The data is stored for the analysis and visualization.

B. Analysis

In the Analysis module, the `DataProcessor` class computes the kinetic energy and momentum.

C. Visualization

In the visualization, various diagnostics are printed out at the end of the run, in two forms. The first are in the form of snapshots at particular times, like the density profile and electric field distribution or velocity distributions. The second are in the form of time histories, such as energy versus time.

Results

A. Production and Analysis

We started by implementing the production code, which we then tested, with a few particles. Here is the plot with 20 electrons (10 electrons in each beam), for a hundred steps. At the initial step, the particles are uniformly distributed.

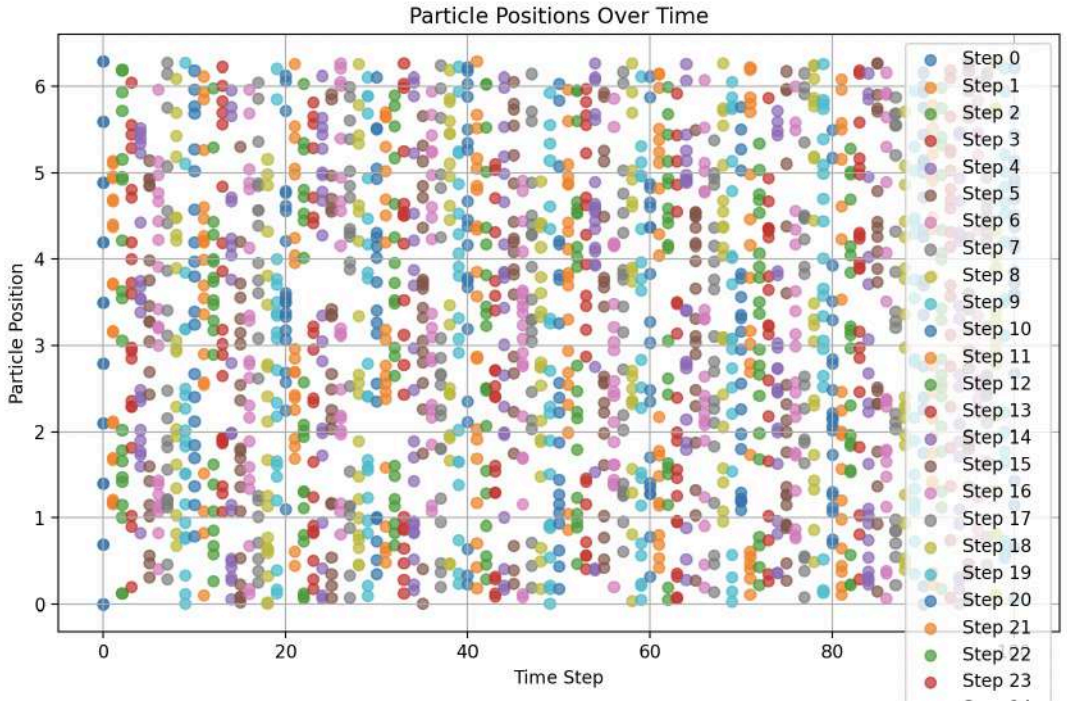


Figure 4. Particles positions as a function of time steps

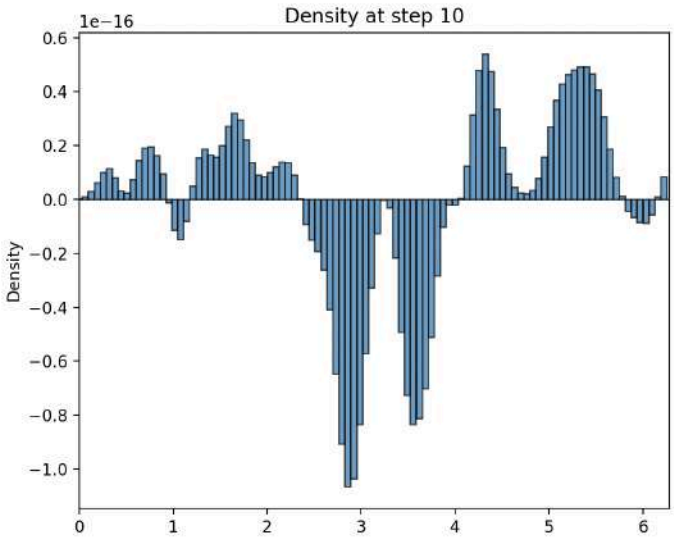
We then tested the analysis, which saves the data from production and the number of steps. It also extracts the data in an excel.

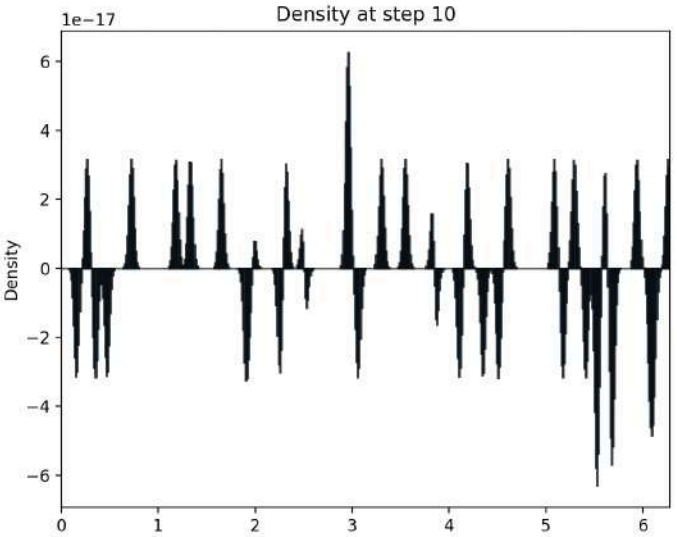
B. Visualization

The visualization code has the goal of plotting the velocity as a function of the positions, like in the picture in page 1. However, before that, we also used it to plot other interesting features to check if our simulation was working.

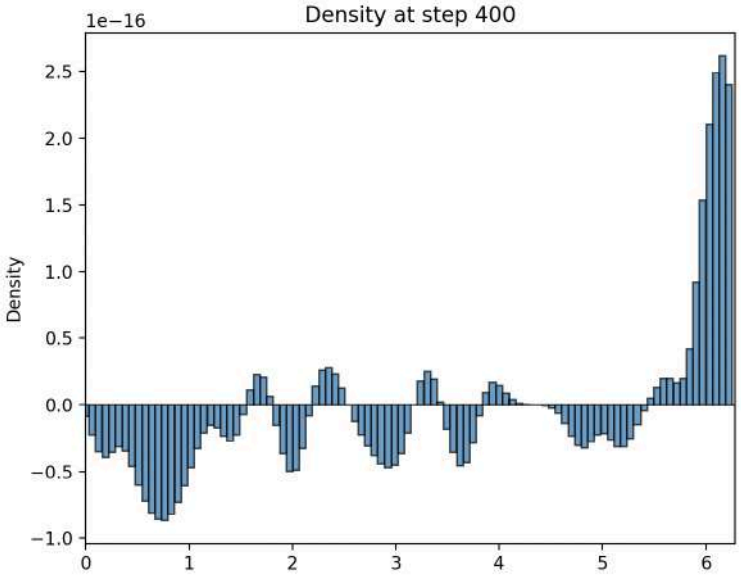
B1. The density

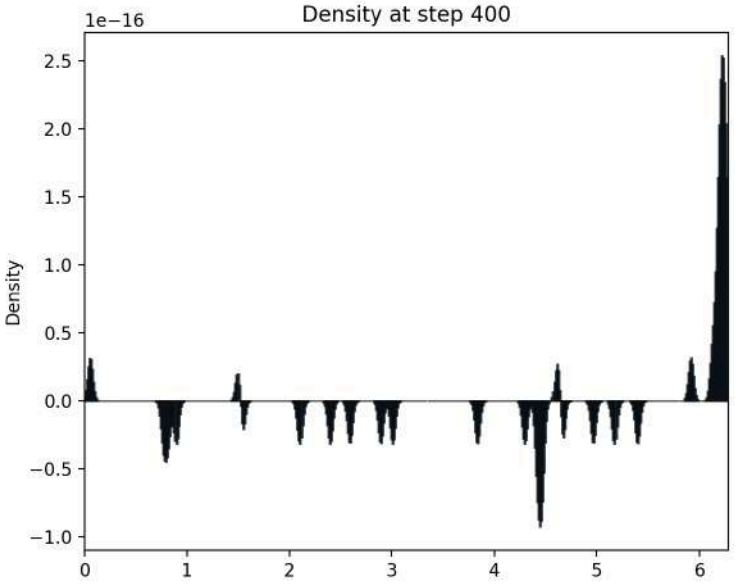
First, to understand where the particles are going, we plotted the density profile, with different numbers of cells and at a different time step. We can see a higher density towards the sides of our box, where the particles are getting quickly ejected.





Figures 5 and 6. Density as a function of the positions, for 200 and 500 cells at step 10.





Figures 7 and 8. Density as a function of the positions, for 200 and 500 cells at step 400.

B2. The velocity-root-mean square

To see how quickly the particles got ejected, we also plotted the velocity-root-mean square $v_{rms} = \sqrt{\frac{1}{N}\sum(v_i - v_i^0)^2}$, where v_i^0 is the initial velocity of the particle i.

The figure shows that the electron's velocity rises rapidly in the time scale. The electrons are accelerated quickly up to large velocities.

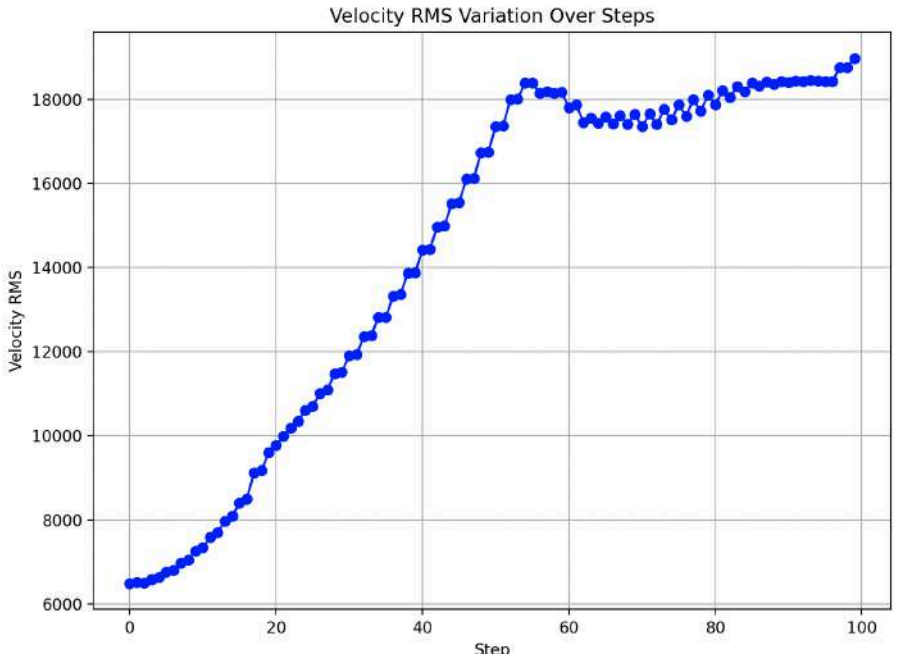
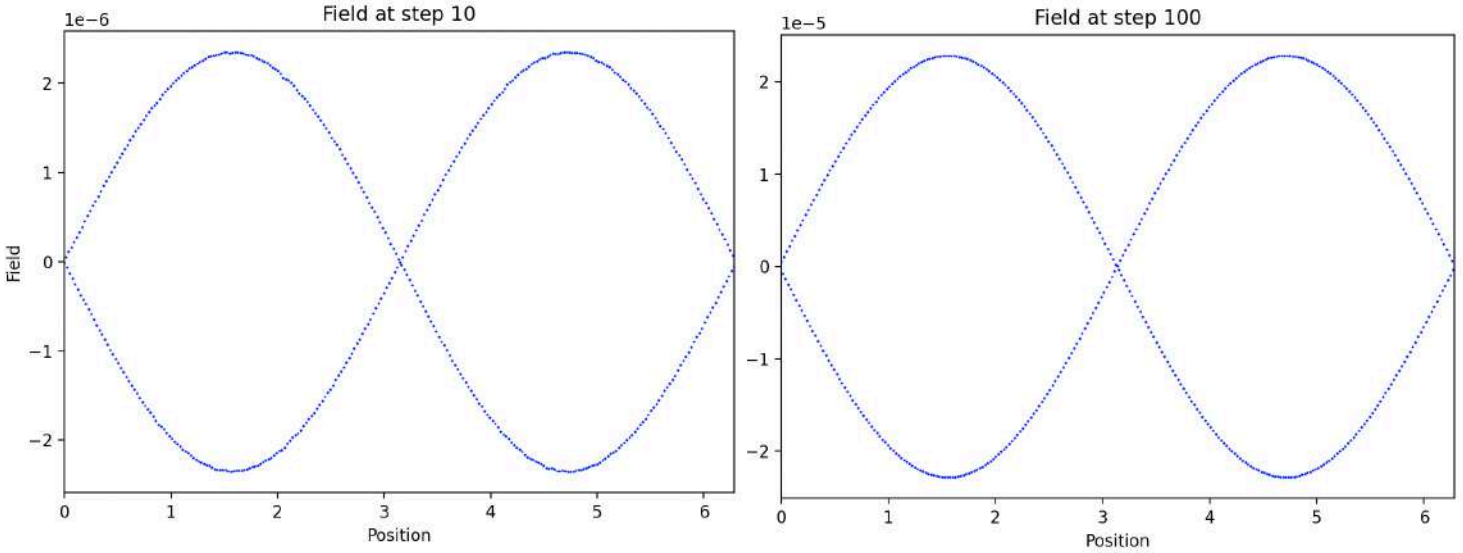


Figure 8. Velocity-root-mean square as a function of the positions for 10 steps.

B3. The electric field

Visualization also plots the electric field as a function of time steps. However, the electric field plot remained the same no matter the variable we put as an input (time steps, number of cells, number of particles). This behavior is hard to explain since the particles do not have the same behaviour when changing the variables. The value of the electric field however grows.

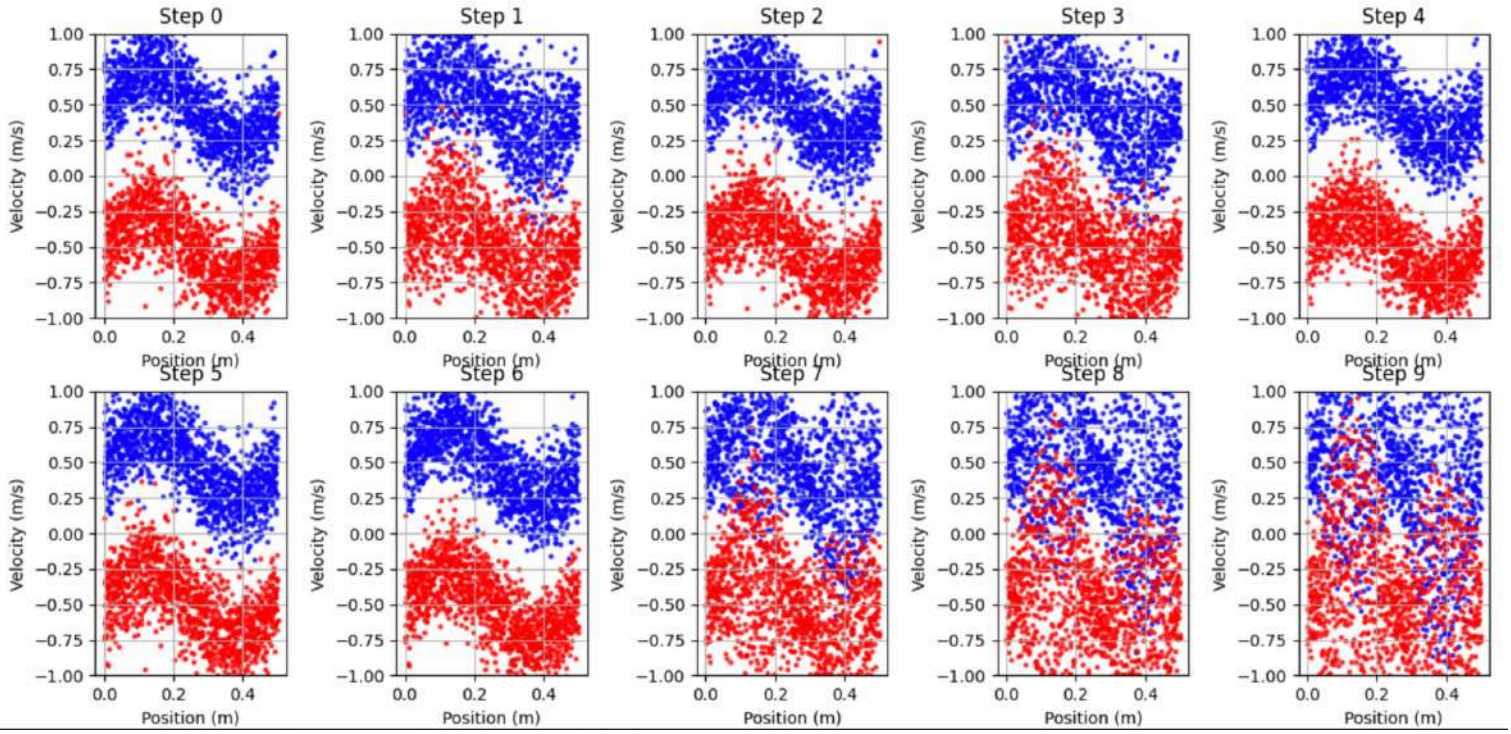


Figures 9 and 10. Electric field as a function of the positions at step 10 and step 100.

B4. The velocity

Finally, visualization plots the velocity as a function of the positions. However, we do not observe the round-eye structure, characteristic of the two-stream instability. Furthermore, we plotted the velocities with different frequencies as a perturbation, but observed the same behaviour.

The two sets of red and blue points represent the different electron beams.



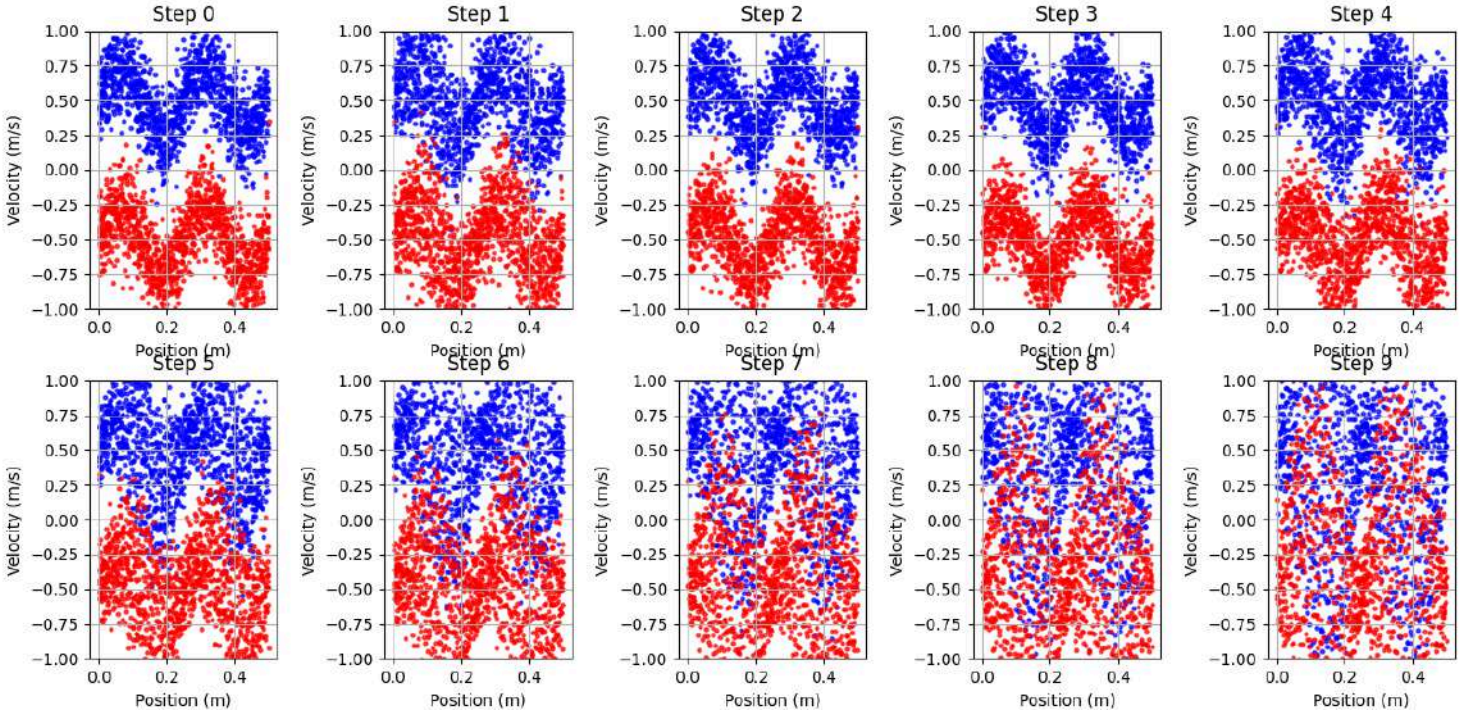


Figure 9 and 10. Velocity as a function of the positions for 10 steps.

Conclusion

Unfortunately, we haven't managed to find back the typical aspect of a two-stream instability. If the two electron beams are merging, they do not form the characteristic round-eye structure. To debug our code, one idea would have been to animate the particles to see the physics evolving, but we did not manage to implement it on time.