Master de Physique
Faculté de Physique & Ingénierie

Plasma Instabilities with a 1D Particle-In-Cell code

D. Aubert, F Thalmann

Season 2023-24

Abstract

Particle-in-cell designates a large class of simulation code where particles evolve according to self-consistent interactions. Examples include self-gravitating ensemble of particles (such as galaxies, stellar clusters) or plasmas of charged particles. The core philosophy of PIC codes is to describe the studied system in terms of 'particles' that can be followed along time, whereas the force field they create is described in terms of cells and grids : as such the methods goes back an forth between a 'Lagrangian' description (particles) and 'Eulerian' description (grids). The coupling between the two arises via the resolution of a field equation, often the Poisson equation. This project aims at implementing a 1D version of a Plasma PIC and uses it to explore some plamas instabilities.

1 Numerical techniques for Particle in Cell

1.1 Particles

The code will follow the 1D evolution of a set of charged particles (for example electrons) with a mass m and charge q. Each particle is defined by a position $x_p(t)$ and velocity $v_p(t)$ and obey Newton's law of motions with:

$$v_p = \frac{dx_p}{dt}, \tag{1}$$

$$F_p = m\frac{dv_p}{dt}. \tag{2}$$

Since we are limited to 1D, the force felt by a particle is the electric force at the particle position only with :

$$F_p = qE(x_p). \tag{3}$$

1.2 Grids

The force will be described on a 1D 'grid' at fixed positions x_i with $0 \le i \le N-1$. If the domain covers positions between 0 and L, the sampling resolution is $\Delta x = L/N$. The electric force is thus defined at each point via a potential :

$$E_i = -\frac{dV_i}{dx} \tag{4}$$

and the potential at each i position V_i is related to the local charge density ρ_i via the Poisson equation:

$$\frac{d^2V_i}{dx^2} = -\frac{\rho_i}{\epsilon_0} \tag{5}$$

1.3 Particle-In-Cell loop

Having defined the 'particle' and 'grid' quantities, the PIC process can be established. It's simply a time loop where it is assumed that the current particles positions and velocities are known and we want to update these values at the next time step Δt . Here is the detailed sequence of the PIC loop:

1. Knowing the particle positions x_p, compute the density of charge at each i 'grid' position, ρ_i . This can be done using histograms, to count the number of particles in each cell i . As particles will move around, ρ_i will evolve with time.
2. Knowing ρ_i the Poisson Equation can be solved. It can be done using a finite differentation scheme of the Poisson equation :
$$\frac{V_{i+1}+V_{i-1}-2V_i}{\Delta x^2}=-\frac{\rho_i}{\epsilon} \tag{6}$$
The potential V_i can be obtained by solving this equation via the resolution of a linear system of equation or via relaxation techniques. An alternative is to express the Poisson equation in Fourier space :
$$-k^2V_k=-\frac{\rho_k}{\epsilon_0} \tag{7}$$
and solve the equation in Fourier space before reconstructing the potential V_i in real space
3. Knowing V_i, the electric field E_i can be obtained by simple finite differentiation.
4. However what is needed is the electric field at the *particles* positions $E(x_p)$. This can be obtained simply by interpolation of E_i at each x_p using e.g. the `interpolate` function of `numpy`.
5. Knowing the force at each particles positions v_p can be updated and x_p can be updated. This can be done using e.g. a simple Euler update or, better, a Leapfrog.
6. Go back to 1 !!

2 Implementation and parameters

2.1 Sampling

Typical sampling for your implementation of the PIC should be about 10^5 particles and 300 cells for example. The more is always the better, but can be quite demanding in terms of memory and computing time.

2.2 Time step

Ideally your PIC code should have an adaptative time step and in any case, the time step Δt should be such as a particle won't cross more than 1 cell during a time step :

$$\Delta t<\frac{\Delta x}{v_p} \tag{8}$$

In practice, the time step can be computed at each iteration for all the particles and the smallest one is applied to all the particles, ensuring that the above criteria is satisfied.

2.3 Boundaries

The best choice is to assume periodic boundary conditions. It should be taken care of when updating the particle positions (if a particle exits the box on one side, it should go back on the other side : it's easily done using modulus operations), and when solving the Poisson equation and looking for neighbor cells (note that using the Fourier technique it is automatically assumed).

2.4 Initial Conditions

An interesting situation is the so-called 'two-stream' instability : it simply correspond to an initial situation where particles are split into two families. Both families have particles with the same charge, initial positions distributed homogeneously in all the box but with opposite initial velocities. As time advance, the two streams will experience violent mixing.

2.5 Charge density

Note that you will follow P particles with the same charge q and we will assume that there is also a static homogeneous background of opposite charged particles to ensure electrical neutrality. It means that if you count for example n_i particles in cell i with an individual -1 charge , the local charge is given by :

$$\rho_i = (P/N - n_i) \tag{9}$$

where N is the total number of cells.

3 Results

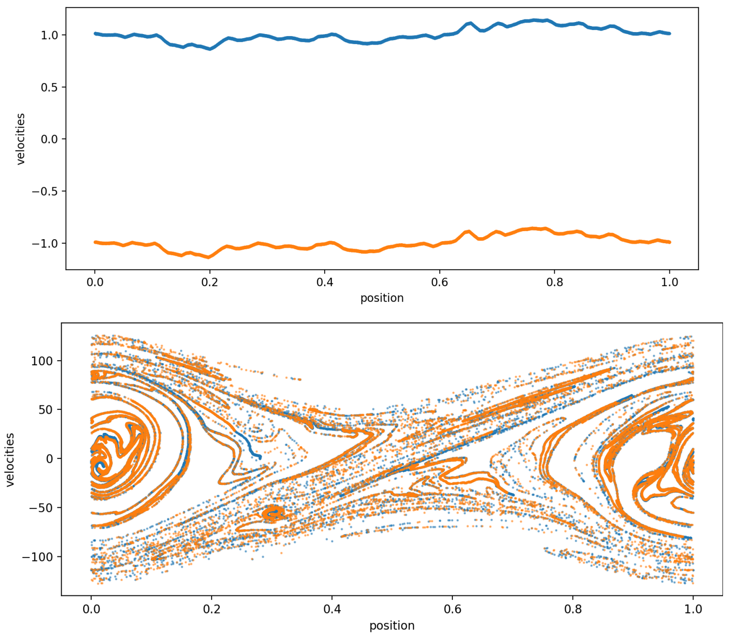

Figure 1: An example of phase space distribution of charged particles after some violent phase mixing. Top : the initial phase space distribution of particles. Bottom: the final situation. The colors designate two families of particles that initally consisted in two beams with initial opposite velocities. It is now impossible to tell them apart.

If you manage to implement your PIC code and run the two-stream problem suggested in the previous section, you should end up having something similar to Fig. 1. Don't hesitate to also display and investigate projections of the phase space distributions, e.g. the distribution of positions of the two families or measure the velocity dispersion and the heating induced by the two stream interactions.

4 And After ?

Once you have a first functional prototype, you should investigate how your choice of implementation may or may not have modified the outcome

- explore the different recipes for the Poisson Solver, eventually they should all agree
- this code can be quite numerically intensive, don't hesitate to look for optimisation routes
- what is the influence of grid and or particle sampling ?
- investigate the different time integration techniques
- and of course, you can set up any experiment you want. For example what if you have different charges, an external field, etc.
- this is very easy to switch to gravitation instead, what is the main difference and maybe you might want to explore 'gravitational' plasmas...