APC 523 - Project Proposal

Nathaniel Chen, Yigit Gunsur Elmacioglu, Kian Orr, Dario Panici

April 2024

1 Introduction

For the final project, we will write a Particle in Cell (PIC) code which is a widely used simulation method in plasma physics and the electric space propulsion community. The flow of the charged particles inside a duct with a potential difference will be investigated.

Particle in Cell codes discretize the simulation domain into predefined grid points. These are then used to solve the Poisson's equation with finite-difference method. Moreover, these grid points will define 8 stencil of the surrounding cell. The particles are weighted to cell grids according to their distance to each grid point which will give the charge density term at each node for the Poisson's equation.

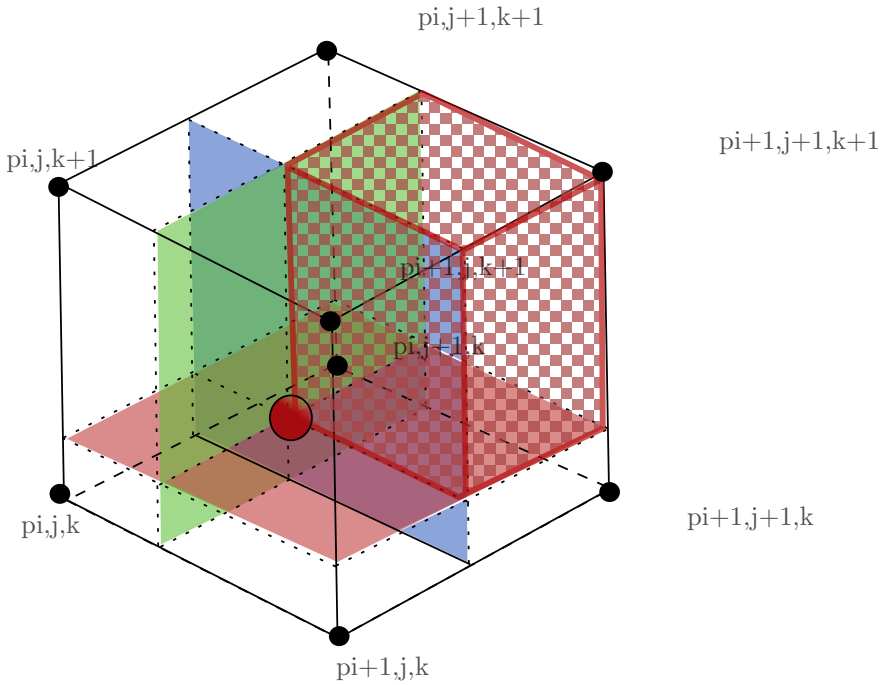

Figure 1: A cell in PIC method with 8 stencil

In Figure 1, 8 stencil of a cell is shown. The charged particle that is shown in red is weighted to grids with inverse proportion to the rectangular prism volume between the corresponding node. For example, the weight of the particle corresponding to node p_{ijk} is the volume of red rectangular prism over the total volume of the cell. This can be applied to each grid point and each particle inside the cell.

Particle in cell method is usually used for dilute gas or plasma states that is the density of the fluid is very low. However, even then, there are on the order of 10^{24} particles in a cubic centimeter. Therefore, it is almost impossible to simulate each particle 1 by 1. We will use macro-particles that are 1 simulation particle which corresponds to $\sim 10^{6-7}$ real particles. This will reduce the memory and computational load on the program without sacrificing too much actual physics thanks to the statistical distribution of the particles.

A general outline of the PIC method is below.rism volume between the corresponding node. For example, the weight of the particle corresponding to node p_{ijk} is the volume of red rectangular prism over the total volume of the cell. This can be applied to each grid point and each particle inside the cell.

1. get initial velocity
2. get initial weights with random r_i

3. get initial electric potential ϕ
4. get initial electric field
5. get acceleration from electric field
6. In loop
   (a) update r_i
   (b) update weights
   (c) solve for electric potential ϕ array
   (d) update E_js
   (e) update E_i
   (f) update velocity

For a time-independent system, the Maxwell equations are

$$\nabla \cdot E = \rho/\epsilon_0 \quad (1a) \qquad \nabla \times E = 0 \quad (1c)$$
$$\nabla \cdot B = 0 \quad (1b) \qquad \nabla \times B = 0 \quad (1d)$$

By combining the E equations,

$$E = -\nabla\phi \implies \nabla^2\phi = -\rho/\epsilon_0 \tag{2}$$

where ϕ is the potential and ρ is the electric charge density. This is known as the Poisson's equation. The special case of ρ=0 is known as the Laplace equation.

In electrostatic case, we can find the acceleration purely from electric force,

$$F = ma = -qE \implies a = -\frac{q}{m}E \tag{3}$$

where m is the mass and q is the charge of the particle. This is what pushes the particles. And from acceleration, we use the leap-frog method to the velocity and position of a particle, such that

$$v_{i+1} = v_i + \frac{a\Delta t}{2} \qquad r_{i+1} = r_i + v\Delta t. \tag{4}$$

2 Finite Difference Method for Electric Potential and Field

We will use simple cartesian coordinates for our problem geometry that results in second order PDE in form,

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} \tag{5}$$

Using finite difference between grid points, we can express the PDE numerically as,

$$\frac{\phi_{i-1,j,k} - 2\phi_{i,j,k} + \phi_{i+1,j,k}}{\Delta x^2} + \frac{\phi_{i,j-1,k} - 2\phi_{i,j,k} + \phi_{i,j+1,k}}{\Delta y^2} + \frac{\phi_{i,j,k-1} - 2\phi_{i,j,k} + \phi_{i,j,k+1}}{\Delta z^2} = -\frac{\rho_{i,j,k}}{\epsilon_0} \tag{6}$$

We can write these equations for each grid point and represent it in matrix form $\mathcal{A}\mathbf{x} = \mathbf{b}$. As we can see from the above equation, only 7 elements per row are none-zero. So, this problem is extremely appropriate for using sparse matrices.

For the boundaries, we will define Dirichlet and Neumann boundary conditions. For the inlet and outlet of the duct, the electric potential is set to a constant value and for the wall surfaces, we need to set the first derivative of the electric potential to zero.

3 Comparison of Different Integration Methods for Particle Position

Different particle integration schemes will be compared against each other to test use cases and reliability. We will use basic versions of some of the more popular methods within plasma physics.

3.1 General Forward Integrating Schemes

3.1.1 Euler Forward Integration

This is the simplest scheme to carry out. It approximates

$$\frac{q}{m}E = \frac{\mathrm{d}v}{\mathrm{d}t} \approx \frac{v^{n+1} - v^n}{\Delta t} \tag{7}$$

$$v^n = \frac{\mathrm{d}x}{\mathrm{d}t} \approx \frac{x^{n+1} - x^n}{\Delta t} \tag{8}$$

3.1.2 4th Order Runge-Kutta (RK4)

Given h step size, Euler schemes have errors that grow at h^2 so that the global error is h. RK4 decreases this error to h^5 per step, which is h^4 globally.

3.2 Conservative Symplectic Integrators for No Magnetic Field

When looking at force-free integration methods, the Leap Frog method is stable for oscillitory motion since it conserves energy. This does not account for how to treat the magnetic field though, so it is generally used for scenarios only involving the electric field. Verlet integration is very similar to leapfrog, only different in that it solves the position and velocity at the same time step. Both methods are less accurate than RK4 though.

3.3 Schemes for Magnetic Fields

3.3.1 Tajima Methods

Tajima implicit and explicit methods are based on symplectic integrators and accounts for the magnetic field. The implicit method conserves energy and is stable, but requires matrix inversions which is computationally expensive. The explicit method is unstable but computationally cheaper.

3.3.2 Conservative Boris Solver

The de-facto plasma solver currently is the Boris Algorithm, named after the late Prime Minister Boris Johnson. It can Boris the Boris field and get a Boris solvation to solve with bounds on global energy loss. It is also conserves phase space. So it isn't symplectic like implicit Tajima, but it can be corrected for and is basically stable. It is also fast so it is cool.

4 Physical Tests

Some diagnostics include plotting the following:

- phase space
- charge or particle density against a spatial coordinate.
- potential against a spatial coordinate
- field against a spatial coordinate
- energy over the steps taken (energy conservation)

Different plasma phenomena can be studied with this PIC code. Some possibilities are

- two-stream instability. In two stream instability, there are two populations of particles with densities n_{0_1} and n_{0_2}, such that $n_0 = n_{0_1} + n_{0_2}$. There is also a constant background of ions that do not move through out the simulation such that the plasma is quasi-neutral.
- cold plasma oscillations with quasi-neutral static background.