Particle-in-cell simulations

Kirit Makwana

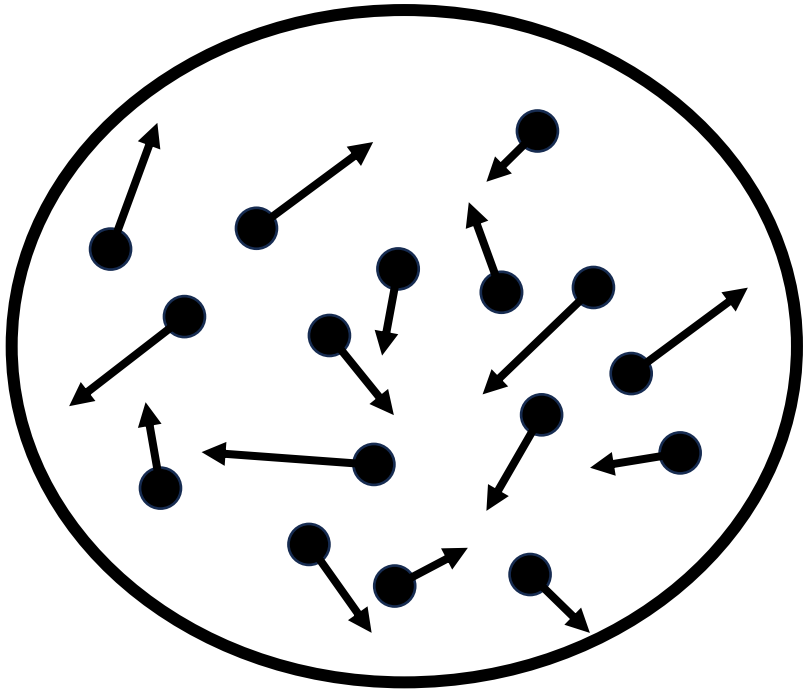
Dept. of Physics

IIT Hyderabad

Fluid treatment

- Consider matter made up of many particles colliding with each other
- If the mean free path of these particles is much smaller compared to the length scale of the system - this is the regime of very low Knudsen number, $K_n = \lambda/L$ – where λ is the mean free path and L is the typical size of the system

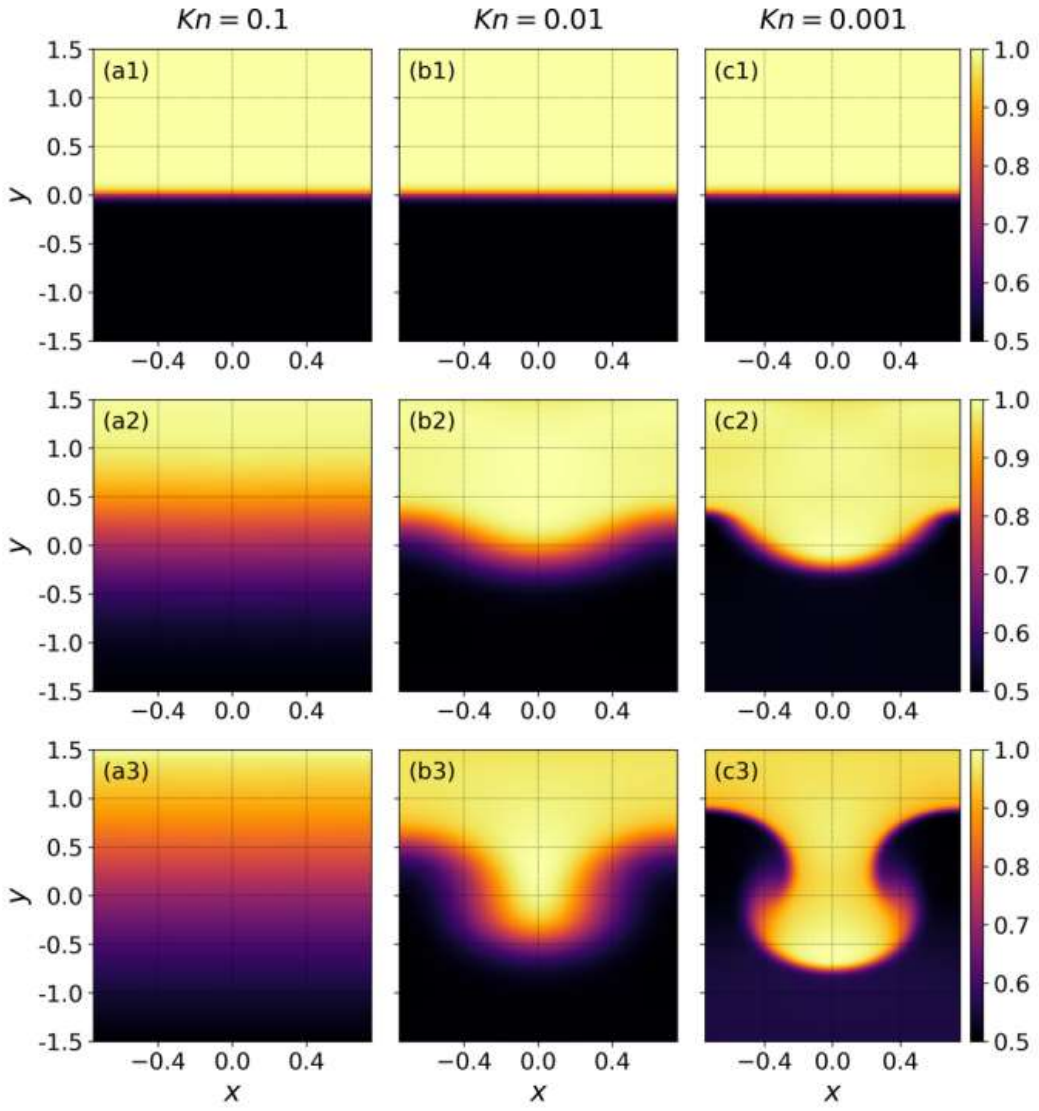
- In this regime it is possible to construct an infinitesimal volume which contains very many matter particles – such an infinitesimal volume can be thought of as a fluid element with smooth, continuous physical properties likes density, velocity, pressure, temperature, etc.



Particle treatment

- When the length scale of interest is comparable or smaller than the mean free path, the fluid (continuum) approximation is not a good approximation
- Motion of dust particles in a rarified medium, like in upper atmosphere, has a large mean free path, ex. volcanic ash
- In biological systems, flow of water molecules through small channels and pores follows molecular dynamics, it does not look like a smooth flow in a pipe
- Electrical discharge consists of flow of electrons which shows disruptions not typically expected in a fluid flow
- Electrical discharge is an example of a plasma

Rayleigh-Taylor instability variation with Kn



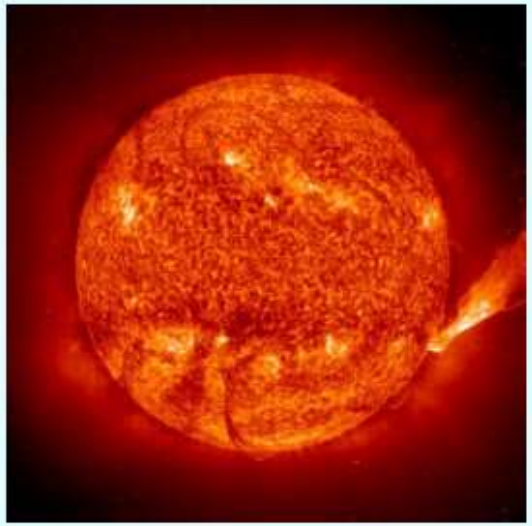
- A particle code run with a varying mean free path – i.e. varying Knudsen number
- For high Kn, there is no Rayleigh-Taylor instability, simply diffusion of the particles
- As collisionality increases, RT instability sets in like we see in typical fluids

Rodman, et al., Phys. Rev. E, 105, 065209 (2022)

Plasmas

- Particle treatment is very useful in simulating hot ionized gases – known as plasmas
- In this state the matter is made up of positively charged ions and negatively charged electrons
- In many cases, plasmas are “collisionless” – meaning that the Knudsen number is very high, or the mean free path is much larger than system length scale
- One more complication is the fact that plasma interact by electromagnetic interactions which are long-range interactions
- This gives rise to collective wave-particle interactions – these cannot be captured by fluid models

Examples of plasmas



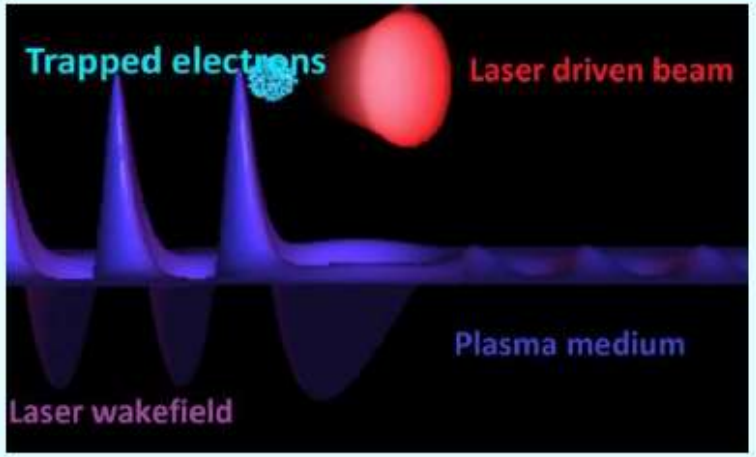
The Sun *credit:NASA*

Interstellar Medium *credit: HST, SST, Chandra*

Solar Corona *credit:APOD*

Aurora *credit:APOD*

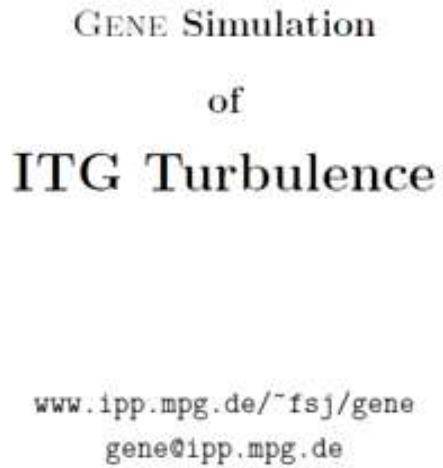
Tokamak plasma *credit:MAST UK*



Laser plasma acceleration *credit:U Liverpool*

Fusion plasmas

- Plasmas have a potential application of energy generation through the process of nuclear fusion – fusion of hydrogen isotopes into helium, along with release of energy
- In order for this energy production process to occur, we require to keep this plasma confined for a long enough time so that it can overcome the repulsion between the positively charged hydrogen ions
- However, many particle-scale instabilities spoil this confinement faster than simple fluid analysis predicts



Credit: M. J. Pueschel & GENE team, IPP, Germany

Equations of motion of charged particles

- Consider a set of N charged parties, with charge q and position x_i
- The equation of motion of these particles is then (assuming only electric fields)

$$m\frac{d^2\boldsymbol{x}_i}{dt^2} = \boldsymbol{F}_i = q\boldsymbol{E}_i$$

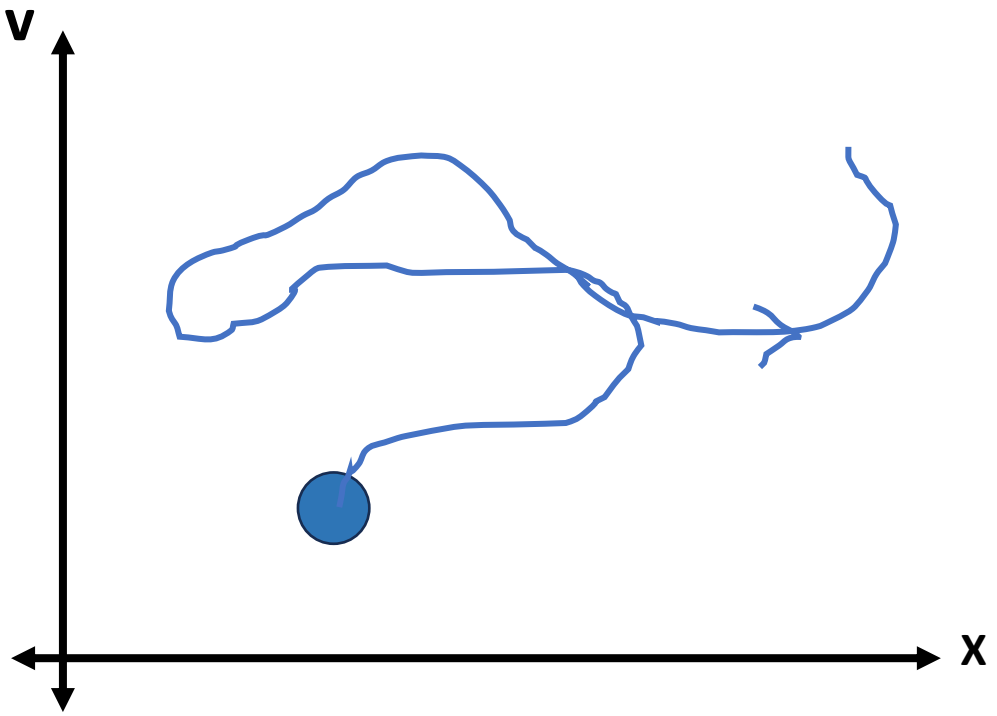
$$\boldsymbol{E}_i = \sum_{j\neq i} \frac{kq^2(\boldsymbol{x}_i - \boldsymbol{x}_j)}{|\boldsymbol{x}_i - \boldsymbol{x}_j|^3}$$

- To solve for each particle, the force has to be calculated from every other particle
- The order of operations is $\mathcal{O}(N^2)$ - computationally improbable for something like 10^10 particles

Phase space

- In order to make this computationally feasible, we need to introduce some form of averaging of the force
- This can be done in phase space which is a 6N dimensional space – there are N particles and each particle has 3 position co-ordinates and 3 velocity co-ordinates

- At any given point of time, the system of N particles is located at a single point in this phase space
- As time advances, the system moves along in this phase space



Single particle equation of motion

- The general equation of motion of a charged particle in electromagnetic force field is

$$\frac{d\boldsymbol{x}_i}{dt} = \boldsymbol{v}_i$$

$$\frac{d\boldsymbol{v}_i}{dt} = \frac{q_i}{m_i}[\boldsymbol{E}(\boldsymbol{x}_i) + \boldsymbol{v}_i \times \boldsymbol{B}(\boldsymbol{x}_i)] \equiv \boldsymbol{a}_i$$

- Here E_i and B_i are the values of the electric and magnetic fields at the location of the i-th particle
- Rigorously, they are the fields produced by all the other particles at that location, but if somehow we can find a smooth field due to all the particles once, then it will be easy to calculate it for all the particles

Equation of motion of system in phase space

- Instead of a point in phase space, lets represent the system as a fuzzy ball centered at some point in phase-space, with a probability density surrounding that point
- The change in probability of finding the system at a given point in phase space is then denoted as

$$df = \frac{\partial f}{\partial t}dt + \sum_{i=1}^{N}\left[\frac{\partial f}{\partial \boldsymbol{x}_i}\cdot d\boldsymbol{x}_i + \frac{\partial f}{\partial \boldsymbol{v}_i}\cdot d\boldsymbol{v}_i\right] = 0$$

$$\frac{\partial f}{\partial t} + \sum_{i=1}^{N}\left[\boldsymbol{v}_i\cdot\frac{\partial f}{\partial \boldsymbol{x}_i} + \boldsymbol{a}_i\cdot\frac{\partial f}{\partial \boldsymbol{v}_i}\right] = 0$$

$$\frac{Df}{Dt} = 0$$

-> convective derivative in phase space showing conservation of probability density

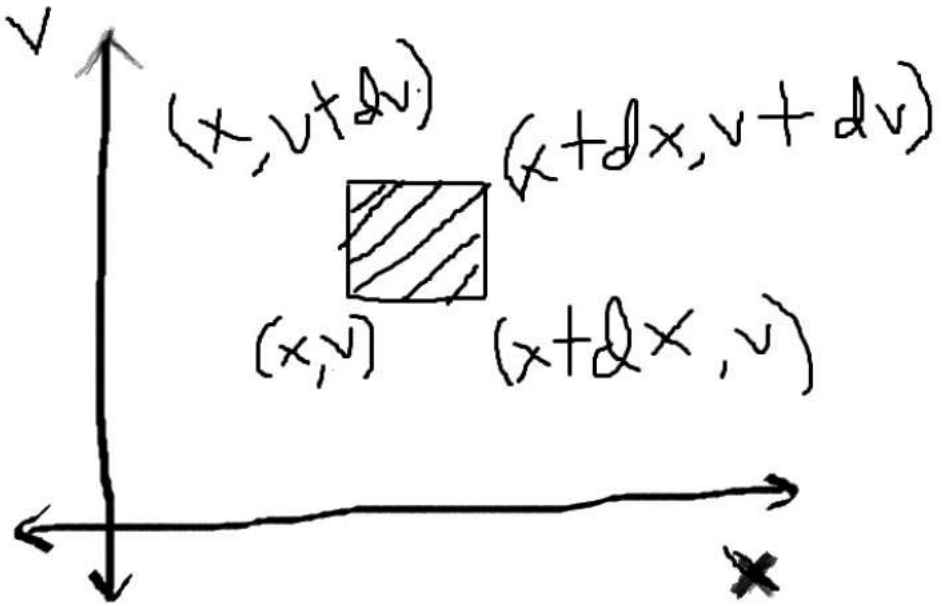
Single particle probability density

- Instead of talking about all particles, we can instead define a single-particle phase space, of just 6 dimensions
- We then define a probability density f, which is the probability of finding a particle in some region of this phase space

- The probability of finding a particle in this shaded region of phase space is

$$f(x,v)dxdv$$

- This averaging is over scales smaller than collisional length scale



Vlasov-Maxwell equations

- It can be shown that this single particle distribution function satisfies the so-called Vlasov-Maxwell equations given below in the absence of two-particle interactions
- This nicely does the job of making a smooth electromagnetic field from particle distribution

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \frac{\partial f_s}{\partial \mathbf{x}} + \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_s}{\partial \mathbf{v}} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \qquad \epsilon_0 \nabla \cdot \mathbf{E} = \rho$$

$$\nabla \times \mathbf{B} = \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mu_0 \mathbf{j} \qquad \nabla \cdot \mathbf{B} = 0$$

$$\rho(\mathbf{x}, t) = \sum_s q_s \int_{\mathbb{V}} f_s(\mathbf{x}, \mathbf{v}, t) d\mathbf{v} \qquad \mathbf{j}(\mathbf{x}, t) = \sum_s q_s \int_{\mathbb{V}} \mathbf{v} f_s(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}$$

G. Lapenta / Journal of Computational Physics 231 (2012) 795–821

Solving Vlasov-Maxwell equations

- The Vlasov-Maxwell equations are integro-differential equations in 7 dimensions (3 position, 3 velocity, and 1 time)
- They are complicated by the fact that the integrals are often times not well-defined, they need to be handled properly by means of complex integration techniques
- Solving 3D PDE's also requires significant computing power
- Adding 3 more dimensions makes the computing much more challenging
- A way to make this simpler is to sample the velocity space by particles, rather than solving the velocity space with some Eulerian grid
- This is the Particle-in-Cell method

Discretization of distribution function

- The distribution function is thought to be made up of pseudo (quasi) particles

$$f_s(\mathbf{x}, \mathbf{v}, t) = \sum_p f_p(\mathbf{x}, \mathbf{v}, t)$$

$$f_p(\mathbf{x}, \mathbf{v}, t) = N_p S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p(t)) S_{\mathbf{v}}(\mathbf{v} - \mathbf{v}_p(t))$$

- S_x and S_v are the shapes of the quasi-particle in position and velocity space
- N_p is the number of real particles that one quasi-particle corresponds to
- x_p and v_p are the position and velocity of the quasi-particle in phase space

Shape functions

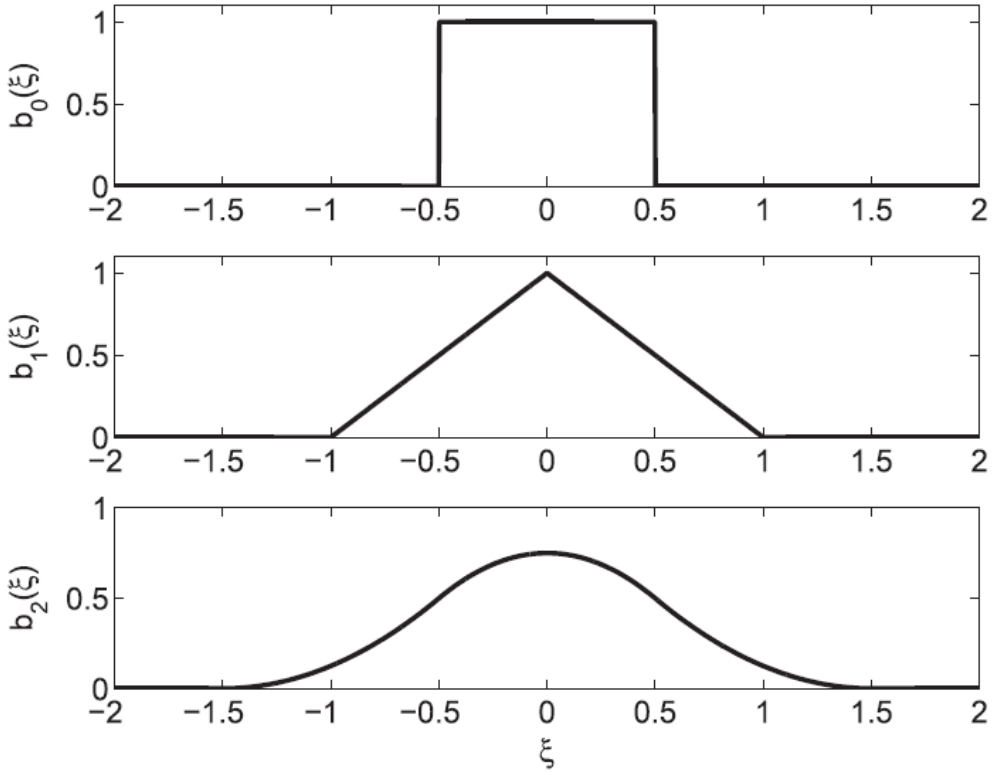
- In velocity space, just choose a Dirac-delta function

$$S_{\mathbf{v}}(\mathbf{v}-\mathbf{v}_p)=\delta(v_x-v_{xp})\delta(v_y-v_{yp})\delta(v_z-v_{zp})$$

$$S_{\mathbf{x}}(\mathbf{x}-\mathbf{x}_p)=\frac{1}{\Delta x_p\Delta y_p\Delta z_p}b_l\left(\frac{x-x_p}{\Delta x_p}\right)b_l\left(\frac{y-y_p}{\Delta y_p}\right)b_l\left(\frac{z-z_p}{\Delta z_p}\right)$$

- B_l are some spline functions that define the shape of the super-particle in real space
- x_p is the central position of this particle shape and v_xp is its velocity
- Integrating out this shape function over velocity and position space should return unity, i.e., probability is unity of finding the particle somewhere

Cloud-in-cell



- Typically codes use the zeroth spline – that means its density is constant inside the shape, and zero outside

Equations of motion

- Substituting the distribution function into Vlasov equation gives

$$\frac{\partial f_p}{\partial t} + \mathbf{v} \cdot \frac{\partial f_p}{\partial \mathbf{x}} + \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_p}{\partial \mathbf{v}} = 0$$

- Further calculation simply gives the equation of motion of a normal particle

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p$$

$$\frac{d\mathbf{v}_p}{dt} = \frac{q_s}{m_s}(\mathbf{E}_p + \mathbf{v}_p \times \mathbf{B}_p)$$

- Ep and Bp is found from the shape function

$$\mathbf{E}_p = \int S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p)\mathbf{E}(\mathbf{x})d\mathbf{x} \qquad \mathbf{B}_p = \int S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p)\mathbf{B}(\mathbf{x})d\mathbf{x}$$

Calculating charge density

- The main computational simplification of particle-in-cell method is that the fields are solved on a grid instead of all particles
- For ex., if we need to solve the electric field, we can solve the Poisson equation

$$\boldsymbol{E} = -\nabla\phi; \qquad \nabla \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0}$$

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

- In order to solve this, we need to get the charge density ρ on the grid points

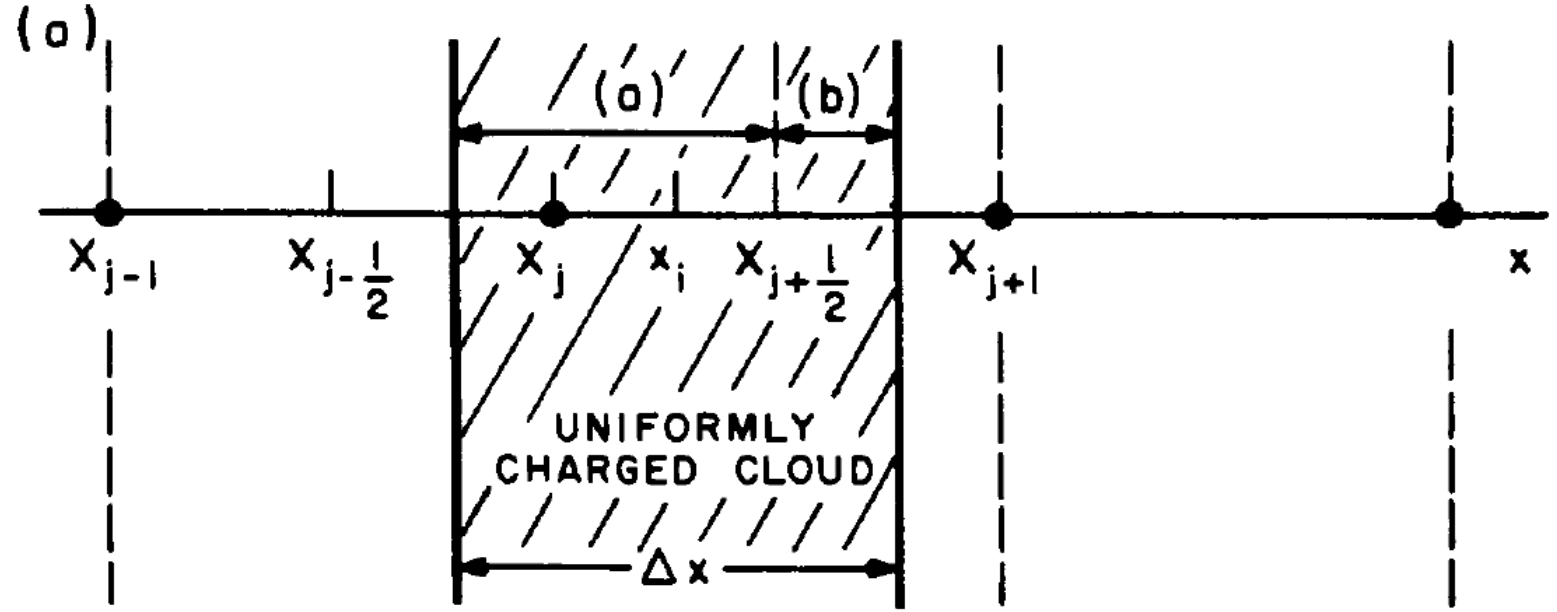
Calculating charge density

- The charge density at a grid point is the volume average of the charge in the controlling volume of that grid point

$$\rho_g = \sum_s \frac{q_s}{V_g} \int_{\mathbb{V}} \int_{V_g} f_s \, d\mathbf{v} \, d\mathbf{x} \implies \rho_g = \sum_p \frac{1 q_s}{V_g} \int_{V_g} S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p) d\mathbf{x}$$

- Consider the cloud-in-cell method where the particle shape is zeroth order b-spline
- Let the grid spacing be Δx , then typically the size of the particle is also taken as Δx
- The control volume of the grid point x_j extends from $x_j - (\Delta \mathrm{x}/2)$ to $x_j + (\Delta \mathrm{x}/2)$
- Let the position of the i-th particle be x_i

Particle weighting



- The contribution of this particle to the charge density on the grids is

$$\rho_j = \frac{q_s}{\Delta x}[\Delta x - (x_i - x_j)] \qquad \rho_{j+1} = \frac{q_s}{\Delta x}[x_i - x_j]$$

- What will be the charge density if first order b-spline is used?

Current density calculation

- The magnetic field can be calculated by using the following equation

$$\nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J} + \mu_0 \epsilon_0 \frac{\partial \boldsymbol{E}}{\partial t}$$

- For this we need to calculate the current density

$$\mathbf{j}_g = \sum_s \frac{q_s}{V_g} \int_{\mathbb{V}} \int_{V_g} \mathbf{v} f_s \, d\mathbf{v} \, d\mathbf{x} \quad \Longrightarrow \quad \mathbf{j}_g = \sum_p \mathbf{v}_p \frac{1 q_s}{V_g} \int_{V_g} S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p) d\mathbf{x}$$

- This is same as the charge density calculation except that the velocity of the particle has to be multiplied

Motion of particle in electric field

- To get the acceleration of the particle, we need to calculate the total force on it

$$\frac{d\mathbf{v}_p}{dt} = \frac{q_s}{m_s}(\mathbf{E}_p + \mathbf{v}_p \times \mathbf{B}_p) \qquad \mathbf{E}_p = \int S_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_p)\mathbf{E}(\mathbf{x})d\mathbf{x}$$

- But we know the value of E only on the grid points. How to get E_p?
- We assume that if particle i is between grid points j and j+1, then the electric field varies linearly in this region
- So the electric field becomes

$$E(x) = \frac{(E_{j+1} - E_j)}{\Delta x}(x - x_j) + E_j$$

Electric field interpolation

- Now we can get the E_p by doing the integration

$$E_p(x_i) = \frac{1}{\Delta x} \int_{x_i - \frac{\Delta x}{2}}^{x_i + \frac{\Delta x}{2}} E(x) dx$$

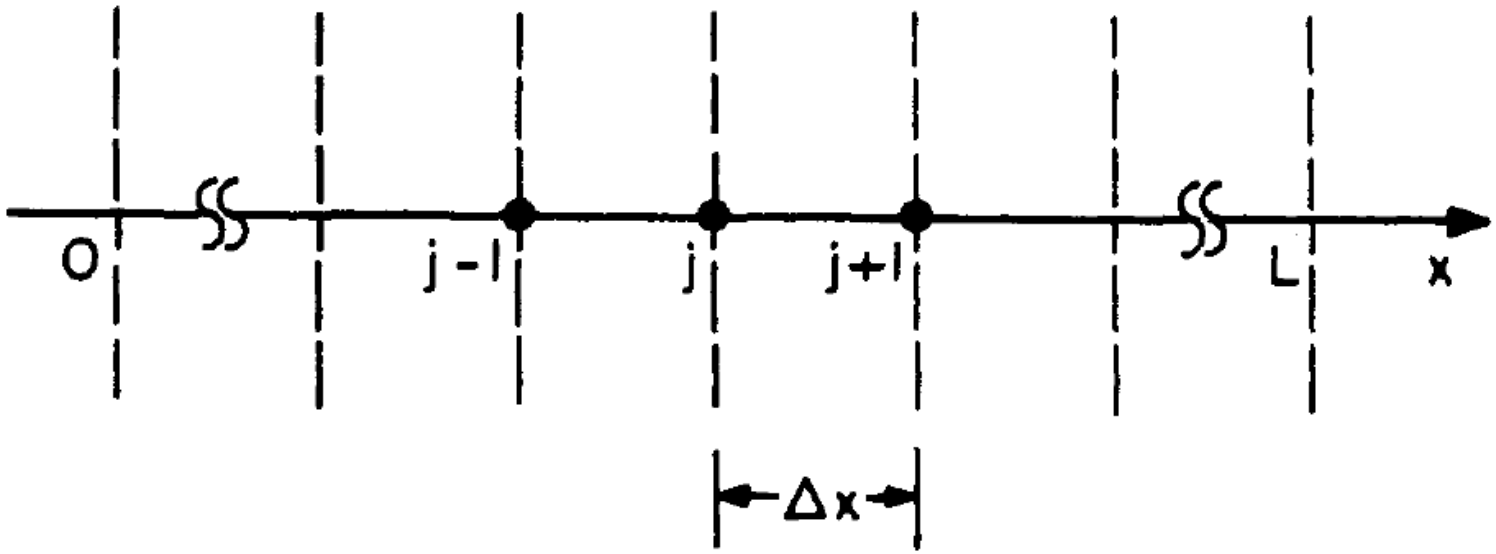
- Substituting the expression for the electric field gives

$$E_p(x_i) = E_{j+1} \frac{(x_i - x_j)}{\Delta x} + E_j \frac{(\Delta x - (x_i - x_j))}{\Delta x}$$

- This is simply the linear interpolation of the electric field at the grid points to the position of the particle. This is the case for the cloud-in-cell particles

Solving Maxwell's equations

- The electric and magnetic fields are located on the grid points



- To solve for the electric field, we need to calculate the charge density
- This is done by taking into account the cloud of the particle

Solving for electric field

- In electrostatic case, electric field can be calculated by solving the Poisson equation

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0} \quad \text{or} \quad \frac{\partial^2\phi}{\partial x^2} = -\frac{\rho}{\epsilon_0}$$

- It can solved by finite difference method

$$\frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{(\Delta x)^2} = -\frac{\rho_j}{\epsilon_0}$$

- Or by doing Fourier transform

$$\rho(x) \xrightarrow[\text{FFT}]{} \rho(k) \xrightarrow[k^2]{} \phi(k) \xrightarrow[\text{IFFT}]{} \phi(x) \xrightarrow[\nabla\phi]{} E(x)$$

Euler method

- Now all these parts can be put together to solve for the full particle distribution function $f(x, v, t)$
- Given a set of initial conditions with particle positions, we can first calculate the charge and current densities
- These can then be used to solve for the electric and magnetic fields
- Using these fields the particles can be moved to their next position in phase space

$$\frac{x_i^{n+1} - x_i^n}{\Delta t} = v_i^n \qquad \frac{v_i^{n+1} - v_i^n}{\Delta t} = \frac{q_i}{m_i}[E_i^n + v_i^n \times B_i^n]$$

- Variable at time step n are used to advance to time step n+1
- What is the order of accuracy of this method?

Leap-Frog method

- The particles are advanced by this method

$$\frac{\mathbf{x}_p^{n+1} - \mathbf{x}_p^n}{\Delta t} = \mathbf{v}_p^{n+1/2}$$

$$\frac{\mathbf{v}_p^{n+1/2} - \mathbf{v}_p^{n-1/2}}{\Delta t} = \frac{q_s}{m_s}\mathbf{E}_p(x_p^n) + \frac{q_s}{m_s}\left(\frac{\mathbf{v}_p^{n+1/2} + \mathbf{v}_p^{n-1/2}}{2}\right) \times \mathbf{B}_p\left(x_p^n\right)$$

- This involves the same number of steps as the Euler method
- Only difference is that the velocity variable is evaluated at half-integer steps
- This makes the accuracy second-order

Leap-frog method

- The first velocity step is taken by an explicit Euler method of half time-step

$$\frac{\mathbf{v}_p^{1/2} - \mathbf{v}_p^0}{\Delta t/2} = \frac{q_s}{m_s}\mathbf{E}_p(x_p^0) + \frac{q_s}{m_s}\left(\frac{\mathbf{v}_p^{1/2} + \mathbf{v}_p^0}{2}\right) \times \mathbf{B}_p\left(x_p^0\right)$$

- This is an implicit equation, but it can be solved algebraically
- The initial electric and magnetic fields can be given as an input
- The full velocity step is also an implicit equation

Boris method

- There is a geometric way to solve the velocity equation by considering

$$\mathbf{v}'_{\text{old}} = \mathbf{v}_{t-\Delta t/2} - \frac{\mathbf{E}\times\mathbf{B}}{B^2}$$

$$\mathbf{v}'_{\text{new}} = \mathbf{v}_{t+\Delta t/2} - \frac{\mathbf{E}\times\mathbf{B}}{B^2}$$

- Then we get the relation

$$\frac{\mathbf{v}'_{\text{new}} - \mathbf{v}'_{\text{old}}}{\Delta t} = \frac{q}{m}\left[\mathbf{E}_{\parallel} + \frac{\mathbf{v}'_{\text{new}} + \mathbf{v}'_{\text{old}}}{2}\times\mathbf{B}\right]$$

- Split the velocity in perpendicular and parallel components w.r.t. B vector

$$\boldsymbol{v}_{new} = \boldsymbol{v}^{\parallel}_{new} + \boldsymbol{v}^{\perp}_{new}; \qquad \boldsymbol{v}_{old} = \boldsymbol{v}^{\parallel}_{old} + \boldsymbol{v}^{\perp}_{old}$$

Plasma Physics via Computer simulation – Birdsall & Langdon - IOP Press

Boris method

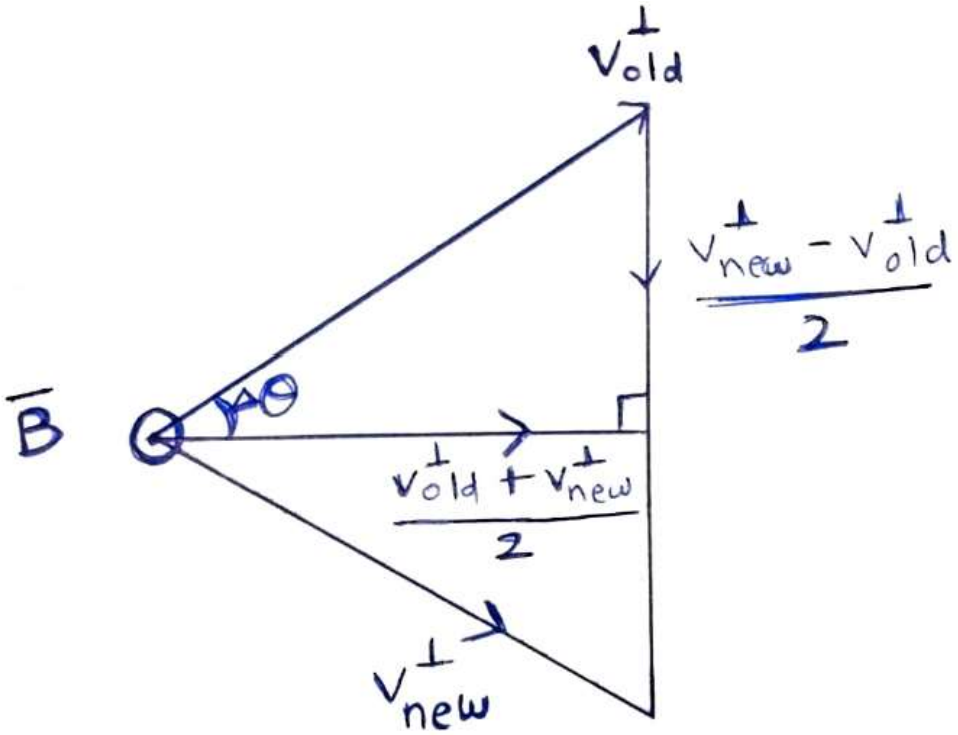
- The parallel velocity simply becomes

$$\boldsymbol{v}_{new}^{\parallel} = \boldsymbol{v}_{old}^{\parallel} + \frac{q}{m}\Delta t E_{\parallel}$$

- The perpendicular velocity simply becomes rotated by an angle

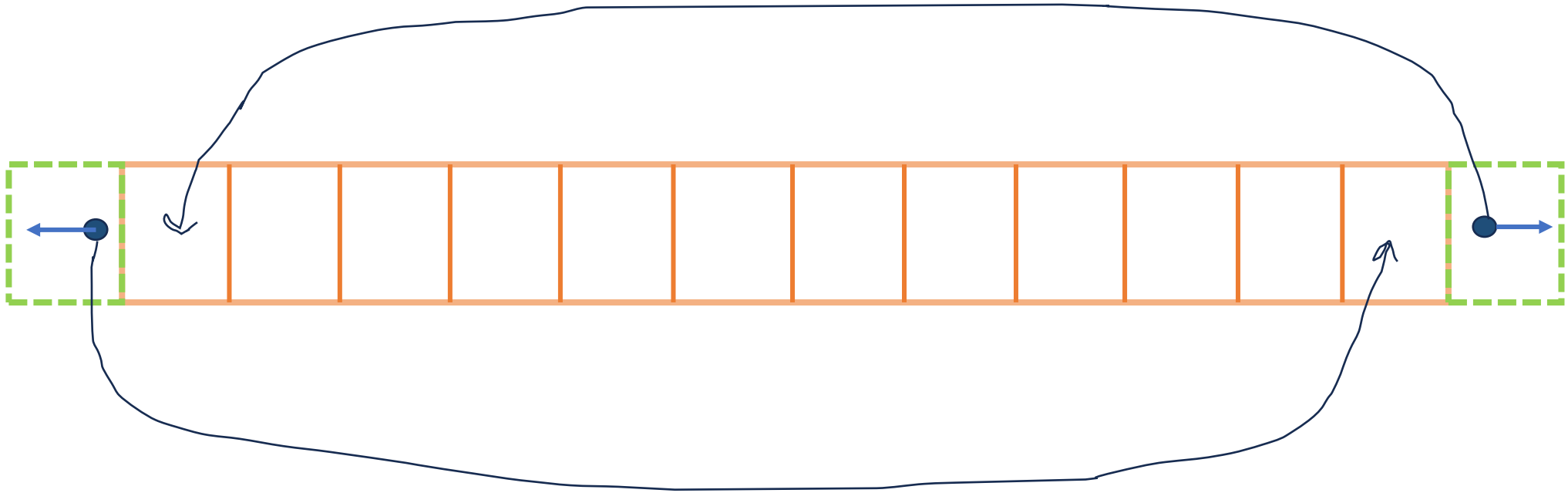
$$2\Delta\theta \approx \frac{qB\Delta t}{m}$$

- These operations are easier and faster with built-in vector and matric multiplication operations, rather than direct matrix inversion method

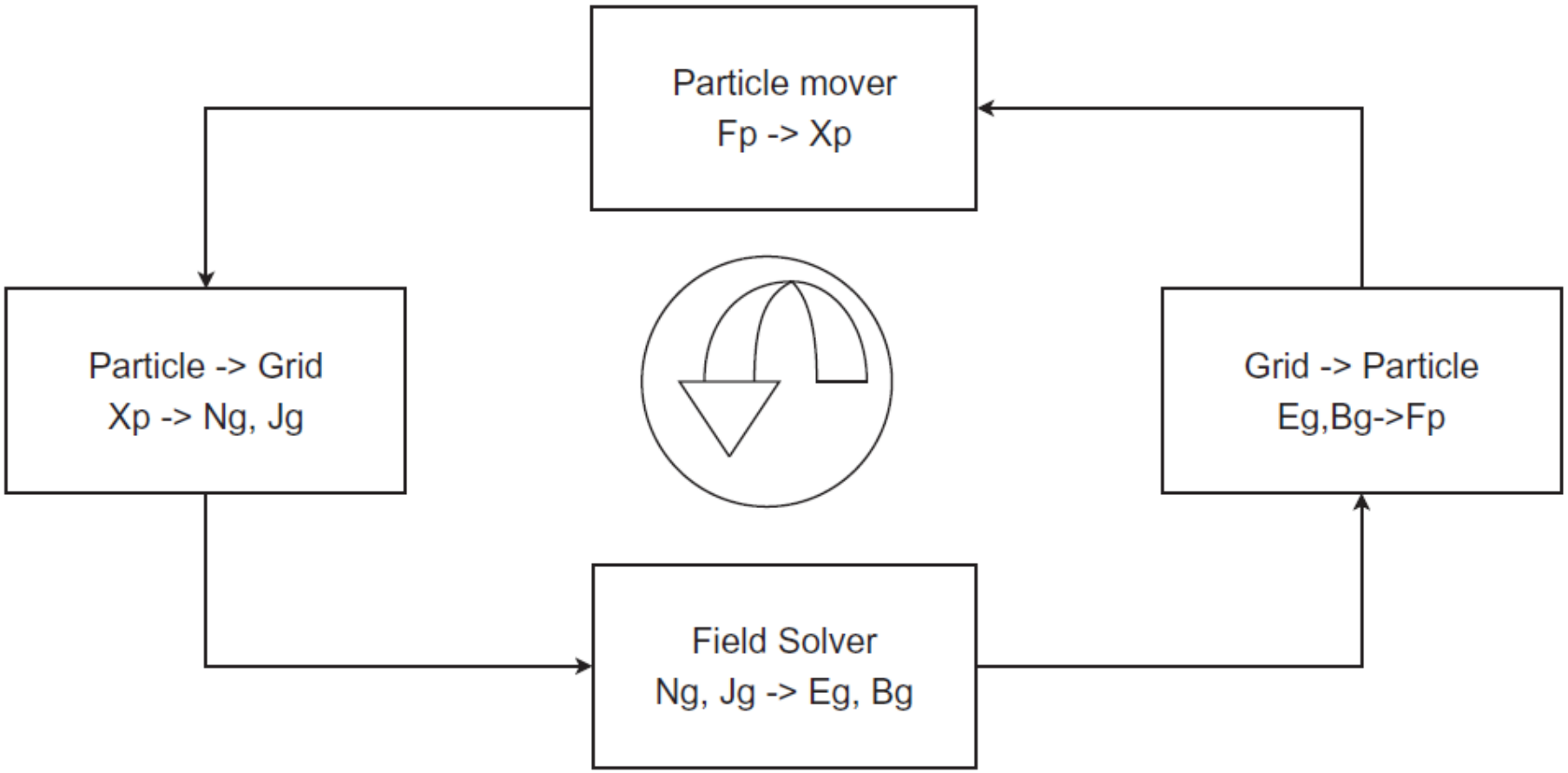


Periodic boundary conditions

- We can assume periodic boundary conditions for both fields and particles



Time loop


Particle mover
Fp -> Xp
Particle -> Grid
Xp -> Ng, Jg
Grid -> Particle
Eg,Bg->Fp
Field Solver
Ng, Jg -> Eg, Bg

General outline of the hands-on code

- Initialize particles of both positive and negative charges
- Calculate initial charge density and then calculate the electric field (ignore magnetic field this assignment)
- Take the half step for particle velocity (n=1/2)
- Start time loop
  - Advance particles to next position (n+1)
  - Get the new charge density (n+1)
  - Get the new electric field (n+1)
  - Get the next half velocity (n+3/2)
  - Output data if we are at output interval
  - Redo the loop
- End

Code compilation and running

- Get starter code from Github
- module load gcc
- module load fftw/3.3.5
- module load Anaconda3
- Create directory “output” which will store the output files
- `g++ -o run.exe pic_code.cpp -lfftw3`
- ~~srun --nodes=1 --ntasks-per-node=1 --time=01:00:00 --pty bash –l~~
- ./run.exe
- Analyze data by using “python analysis.py”

Step 1 – function init

- First step is to code up the initial conditions
- Initialize the code with box size of 1 unit, 100 particles per cell (nppc), total 100 cells (nc), mass of proton as 1 and mass of electron as 1/50 (i.e., mass ratio is $\frac{m_i}{m_e} = 50$)
- Distribute the particles uniformly throughout the box
- Take output interval as every 10 steps
- Initialize ParArray with position, charge, and mass. Velocity of each particle is in velocity_x array– normalize charge and mass of particle with nppc
- Give a sinusoidal perturbation to the charge of electrons

$$q = q_e + \delta q \sin\left(\frac{2\pi x}{L}\right)$$

- Check by printing out a few particles position and charge

Step 2 – calculate charge density (calc_chargre_density)

- Loop over all the particles
- Locate their nearest neighbor grid points
- Deposit charge on the grid points using the formulae shown before
- Convert to charge density (divide by Δx)
- Arrays “pcharge” and “echarge” store the positive and negative charge densities respectively
- The last part is required due to the periodic boundary conditions
- Verify by plotting the charge density

Step 3 – Electric field calculation (E_cal)

- Forward transform of charge density is already given in the code
- Calculate the array of ‘k’ values
- Using the ‘k’ values and the FT of charge density, calculate the electric field in k-space
- Do inverse Fourier transform of E-field in k-space to get E-field in real space
- Last part contains some normalization of the Fourier transform to get the physical values and storing in array
- Verify by plotting the electric field. It is analytically solvable for our initial charge density setup

Step 4 – Particle update (half_velocity, particle_push)

- Function half_velocity
  - Interpolate electric field to particle position
  - Update the velocity_x array to the ½ time step
  - This function will be called only once before start of the time loop
- Function particle_pusher
  - Update the particle position and particle velocity (pay careful attention to the order in which they are updated)
  - Apply periodic boundary conditions to particles which leave the box
- This can be tested for a simple particle moving in zero electric field with constant velocity

Plasma oscillation test

- All these functions are put together in the function ‘main’
- Look carefully and understand the way the time advance works
- Given these initial conditions, check how the electron density evolves
- It should show a well known feature known as plasma oscillations
- The theoretically expected angular frequency of these oscillations is

$$\omega_{pe} = \sqrt{\frac{n_0 e^2}{m_e}}; \qquad T = \frac{2\pi}{\omega_{pe}}$$

- n_0 is the number density of electrons, e is the charge, and m_e is mass of electrons, T is the time period of oscillation
- Validate whether your simulation produces the expected time period?

Further investigations

- How does the behaviour change if you change nppc, nc, mass ratio?
- How does the error behave with these changes?
- Can you parallelize the particle loops with OpenMP?