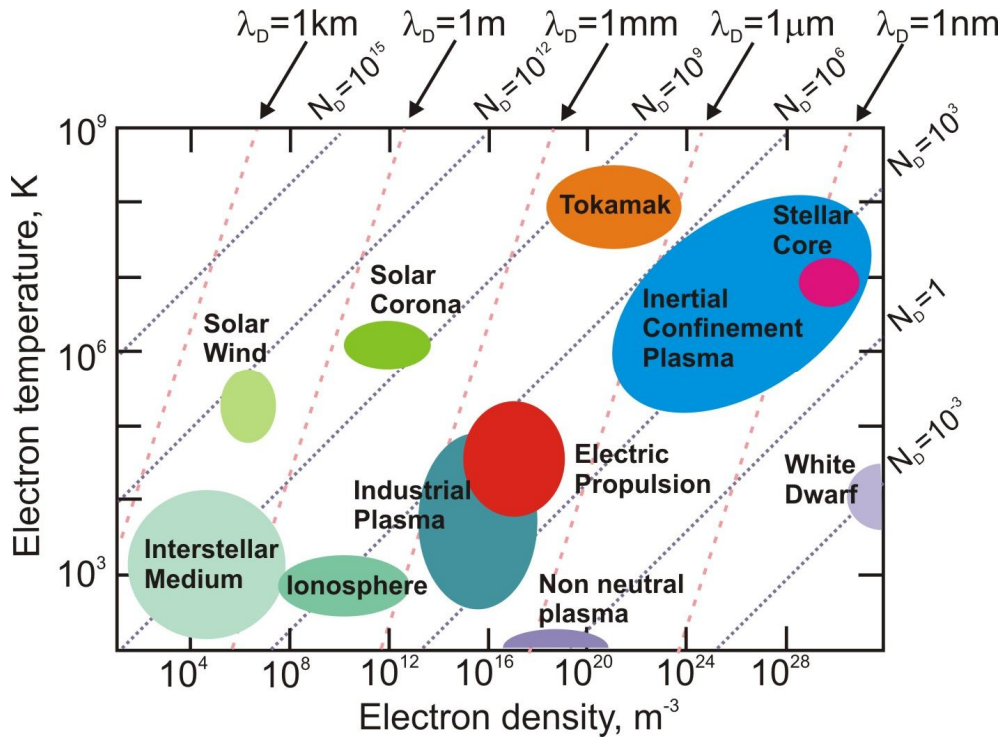
Theoretische Plasmaphysik

Dr. Ninad Joshi
Institut für Theoretische Physik
ninad.joshi@theo.physik.uni-giessen.de

Chapter : Computational Plasma Physics

Plasma Classification

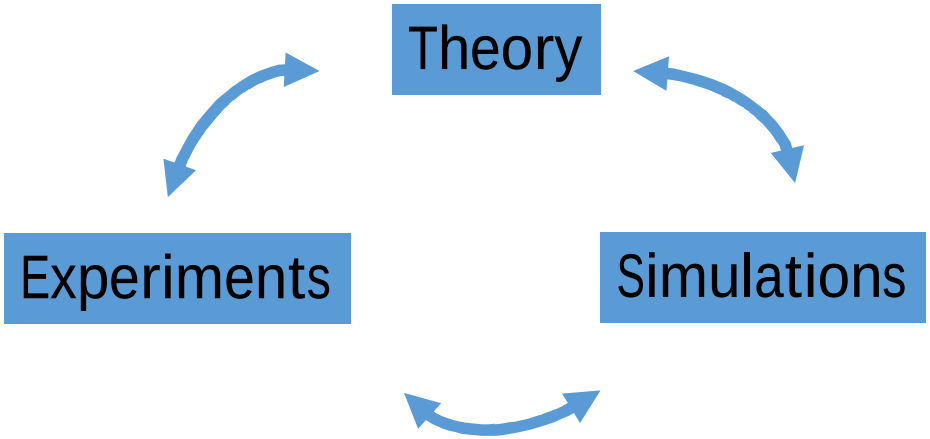
- Different types of classification exists e.g. Cold plasma (non-thermal) – Hot plasma (thermal), Collisional-non collisional, neutral –non neutral, complex
- They are all broadly classified depending on density, temperature and behaviour



Role of computer

- Engineering experiments - designed to predict accurately the behaviour of a device before construction of using expensive technology.
- Inaccessible data experiments - performed to obtain data which can not be measured using experiments
- Theoretical experiments - used as guides for theoretical development. Complex plasma phenomena can

be investigated by limiting subset of process to evaluate its effect.



Plasma Modelling : Hierarchy

3 Level of description Bogoliubov-Born-Green-Kirkwood-Yvon (BBGKY) Hierarchy

- Exact microscopic description
- Kinetic description
- Fluid description

N –body Model

- Plasma or particle beam is made of a number of particles
- Classical or relativistic dynamics : obeys Newton‘s Law

$$\frac{d\gamma m\mathbf{v}}{dt} = \sum \mathbf{F}_{ext} \qquad \frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i$$

- Plasma contains more than 10^{10} particles
- Simulation would be too expensive
- Each operation count would introduce numerical error

Kinetic Model

- Each particle species **s** in plasma is characterized by distribution function f_s(**x**,**v**,t) corresponding to statistical mean repartition in phase space
- Collisionless Boltzmann equation is given as

$$\frac{\partial f}{\partial t} + \mathbf{v}\frac{\partial f}{\partial \mathbf{x}} + \frac{\mathbf{F}}{m} \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

- Inserting Lorenz force term we get Vlasov equation

$$\frac{\partial f}{\partial t} + \mathbf{v}\frac{\partial f}{\partial \mathbf{x}} + \frac{q}{m}\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

- The collisions can be included in Vlasov equation as

$$\frac{\partial f}{\partial t} + \mathbf{v}\frac{\partial f}{\partial \mathbf{x}} + \frac{q}{m}\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) \cdot \frac{\partial f}{\partial \mathbf{v}} = \left(\frac{\partial f}{\partial t}\right)_c$$

- The RHS term in simplest form given as

$$\left(\frac{\partial f}{\partial t}\right)_c = -\left(\frac{f - f_m}{\tau}\right)$$

f_m is distribution function, τ is collision time

Fluid Model

- Due to collisions the system relaxes to Maxwellian distribution ie thermodynamic equilibrium
- In this state plasma can be described as fluid
- Fluid model derived from Vlasov equation are coupled to Maxwell‘s equations

Continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

Cauchy momentum equation

$$\rho \frac{d\mathbf{v}}{dt} = (\mathbf{J} \times \mathbf{B}) - \nabla p$$

Energy equation

$$\frac{d}{dt}\left(\frac{p}{\rho^{\gamma}}\right) = 0$$

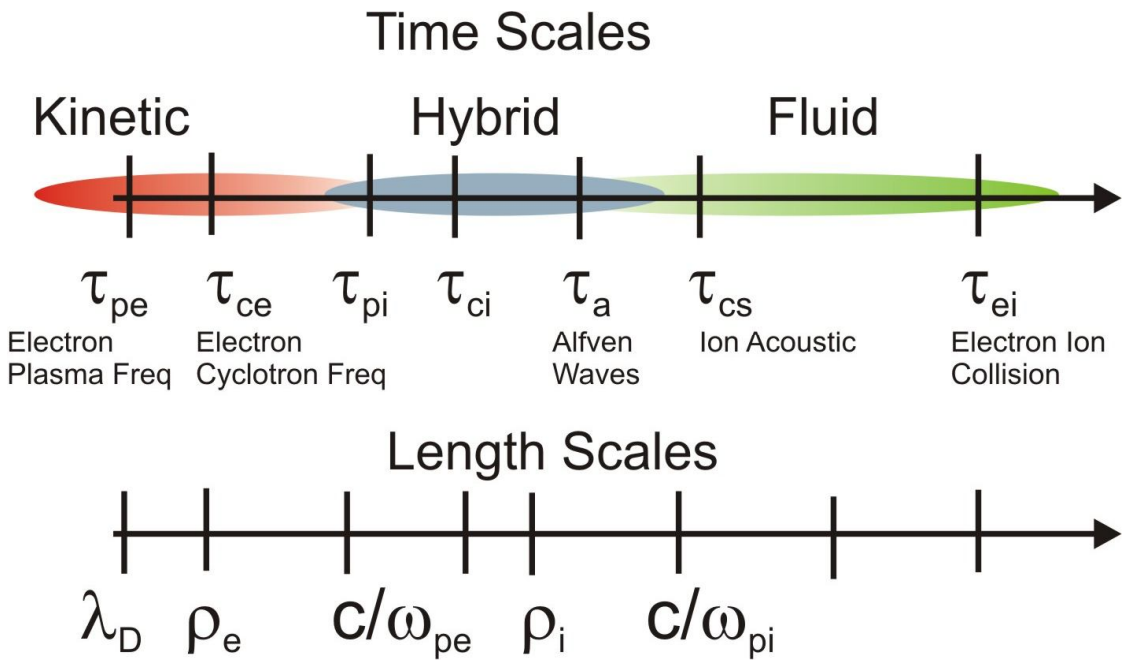
Ohm’s law

$$\mathbf{E} + (\mathbf{v} \times \mathbf{B}) = \eta \mathbf{J}$$

where Γ is ratio specific heat, σ is Plasma conductivity, and $\eta = \sigma^{-1}$

Kinetic and Fluid Models

- The choice of plasma description, either Kinetic or Fluid model is made based on the temporal and spatial scale
- Hybrid models also employed where one of the plasma component is treated as fluid and other kinetic



Single particle simulation

Single particle in uniform electric field

$$\frac{dv}{dt} = \frac{q}{m}E$$

Forward difference method

$$\frac{v_{i+1} - v_i}{\Delta t} = \frac{q}{m}E_i$$

$$\frac{x_{i+1} - x_i}{\Delta t} = v_i$$

Simple oscillatory field

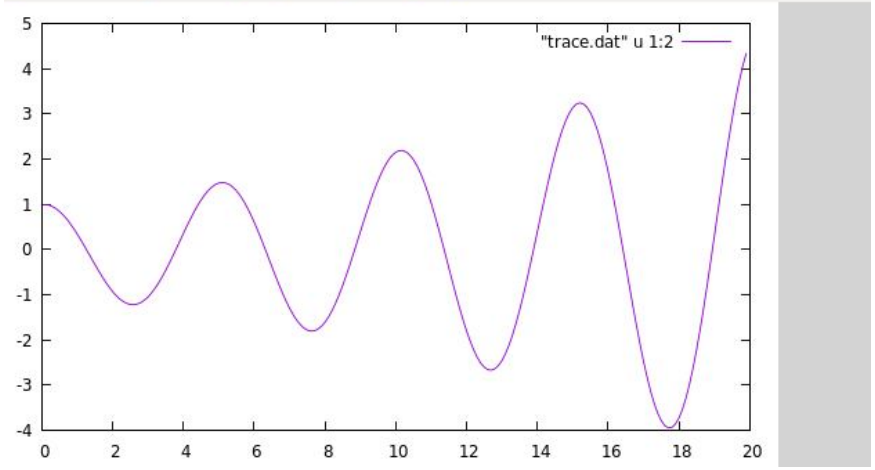
$$E = -(m/q)\omega^2 x$$

Getting started with computer simulation

```
/*main*/
int main()
{
        //load particle
        double m = ME;  //particle mass
        double q = -QE; //particle charge
        double x0 = 1.0;        //initial position
        double v0 = 0;  //stationary
        double E; // Electric field
        double x2,v2;
        double omega = 1.25;
        /*
        simulation parameters
        */
        double dt = 0.1;        //timestep

        //open particle trace file
        ofstream fout("trace.dat");
        if (!fout) {cerr<<"Failed to open file"<<endl;return -1;}

        //time step loop
        for (int it=0;it<200;it++)
        {
                //write trace data
                fout<<it*dt<<"\t"<<x0<<"\t"<<v0<<"\n";

                //compute future position and velocity
                E = -(m/q) * pow(omega,2.0) * x0 ;
                v2 = v0 + (q/m)*E*dt;
                x2 = x0 + v0*dt;
                x0= x2; v0= v2;
        }
```

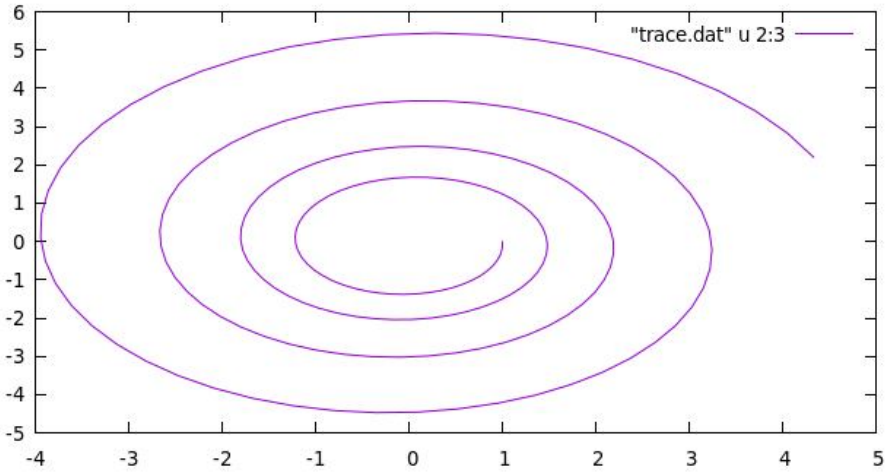
Gnuplot ...

```
(base) joshi@joshi-Satellite-Pro-L650:~/Projects/Lecture$ gnuplot

        G N U P L O T
        Version 5.2 patchlevel 2    last modified 2017-11-01

        Copyright (C) 1986-1993, 1998, 2004, 2007-2017
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help FAQ"
        immediate help:   type "help"  (plot window: hit 'h')

Terminal type is now 'wxt'
gnuplot> plot "trace.dat" u 1:2 w l
```



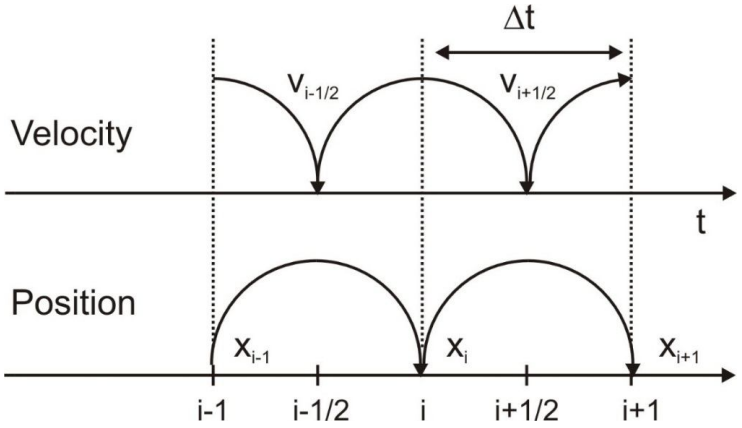
The instability of forward difference

x



Single Particle simulation

Leapfrog



$$\frac{v_{i+1/2} - v_{i-1/2}}{\Delta t} = \frac{q}{m} E_i$$

$$\frac{x_{i+1} - x_i}{\Delta t} = v_{i+1/2}$$

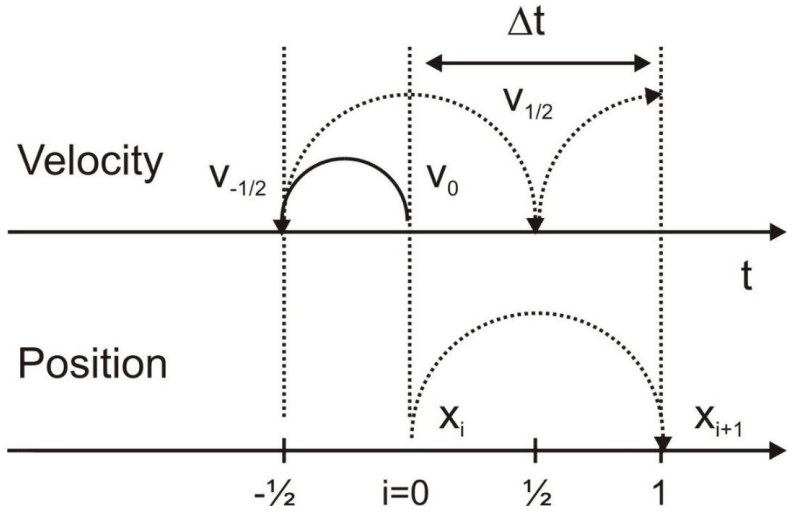
Position advance and synchronization

Ofcourse once the velocity is updated the positions are advanced

$$\frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\Delta t} = \mathbf{v}_{i+1/2}$$

Since **x** and **v** are not evaluated at same time synchronization is necessary

- Initial conditions: typically v is pushed back half a time step
- Evaluating diagnostic quantitites similar synchronization is necessary

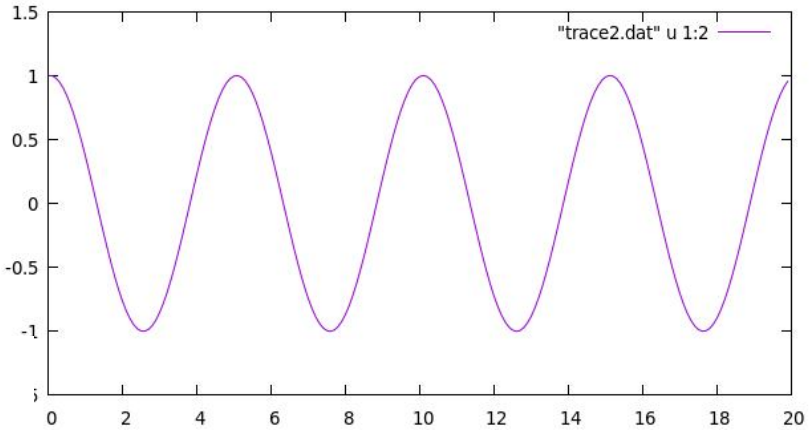


Leap frog with time synchronization

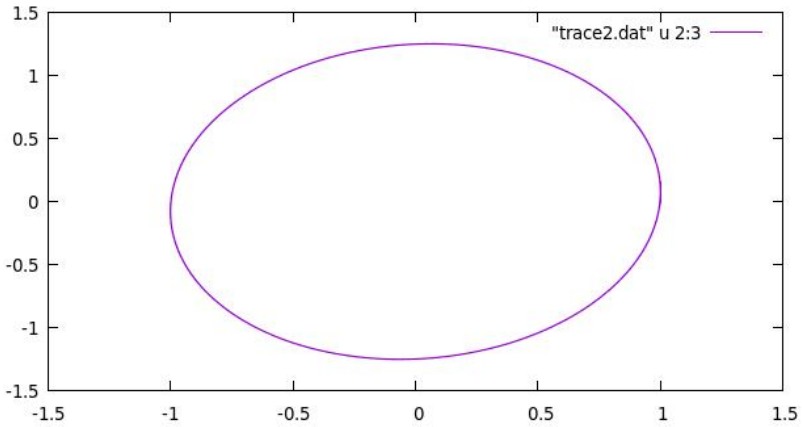
```
|
E = -(m/q) * pow(omega,2.0) * x0 ;
v0 = v0 - (q/m)*E*dt;
```

x

```
//compute future position and velocity
E = -(m/q) * pow(omega,2.0) * x0 ;
v2 = v0 + (q/m)*E*dt;
x2 = x0 + v2*dt;
x0= x2; v0= v2;
```



vx



x

PIC: Mover / Particle Pusher

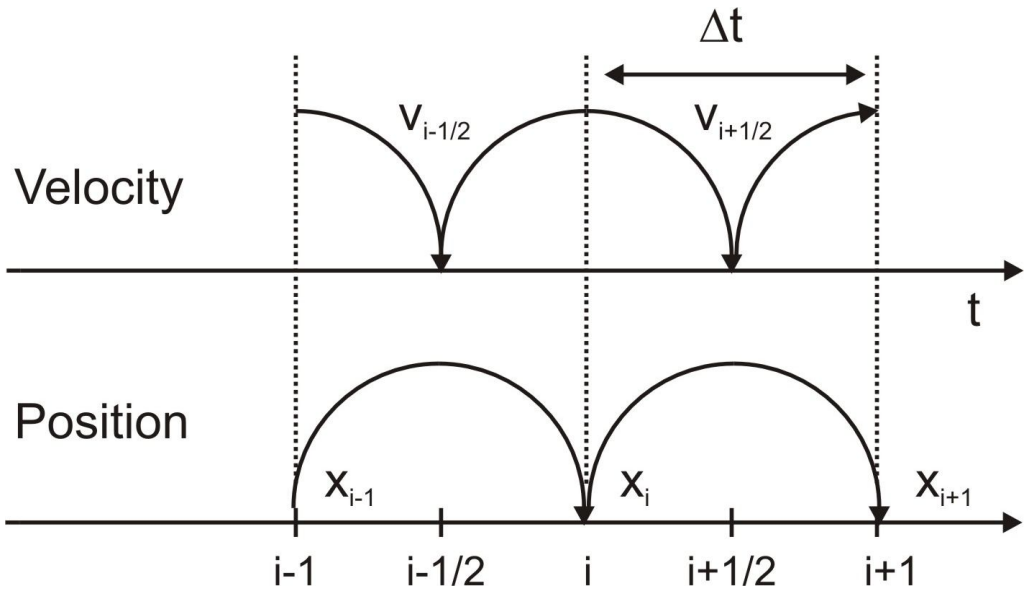
- There are two types of solvers for particle mover Implicit and Explicit
- Explicit method calculate later state of system from current while the implicit method solving the equation involving both the states

Explicit:

$$Y(t + \Delta t) = F(Y(t))$$

Implicit:

$$G\left(Y(t), Y(t + \Delta t)\right) = 0$$



Explicit methods : Leapfrog method

First center difference form was proposed by Buneman

$$\frac{\mathbf{v}_{i+1/2} - \mathbf{v}_{i-1/2}}{\Delta t} = \frac{q}{m}\left[\mathbf{E}_i + \frac{\mathbf{v}_{i+1/2} + \mathbf{v}_{i-1/2}}{2} \times \mathbf{B}_i\right]$$

The scalar components are solved simultaneously

Boris Method : Equations for E and B are separated within half time period

$$\mathbf{v}_{i-1/2} = \mathbf{v}^- - \frac{q\mathbf{E}}{m}\frac{\Delta t}{2}$$

$$\mathbf{v}_{i+1/2} = \mathbf{v}^+ + \frac{q\mathbf{E}}{m}\frac{\Delta t}{2}$$

$$\frac{\mathbf{v}^+ - \mathbf{v}^-}{\Delta t} = \frac{q}{2m}\left(\mathbf{v}^+ + \mathbf{v}^-\right) \times \mathbf{B}$$

Relativistic formula involves more steps to accomodate γ changing during jump

Tajima's Impicit method

This is an example of implicit method, the updated velocity is given as

$$[\mathbf{I} - \mathbf{R}\epsilon]\,\mathbf{v}_{i+1/2} = [I + \mathbf{R}\epsilon]\,\mathbf{v}_{i-1/2} + \frac{q}{m}\mathbf{E}\Delta t$$

$$\mathbf{R} = \frac{1}{B}\begin{bmatrix} 0 & B_z & -B_y \\ -B_z & 0 & B_x \\ B_y & -B_x & 0 \end{bmatrix}$$

where ε = qB/m (Δt/2)

$$\mathbf{v}_{i+1/2} = [\mathbf{I} - \mathbf{R}\epsilon]^{-1}\,[I + \mathbf{R}\epsilon]\,\mathbf{v}_{i-1/2} + [\mathbf{I} - \mathbf{R}\epsilon]^{-1}\,\frac{q}{m}\mathbf{E}\Delta t$$

The solution involved matrix inverse given as

$$[\mathbf{I} - \mathbf{R}\epsilon]^{-1} = \frac{1}{det\,([\mathbf{I} - \mathbf{R}\epsilon])}\begin{bmatrix} 1+\epsilon^2 B_x^2 & \epsilon B_z + \epsilon^2 B_x B_y & -\epsilon B_y + \epsilon^2 B_x B_z \\ -\epsilon B_z + \epsilon^2 B_x B_y & 1+\epsilon^2 B_y^2 & \epsilon B_x + \epsilon^2 B_y B_z \\ \epsilon B_y + \epsilon^2 B_x B_z & -\epsilon B_x + \epsilon^2 B_y B_z & 1+\epsilon^2 B_z^2 \end{bmatrix}$$

The method is computationally heavy.

Boris Implementation

$$\bar{\tau} = \frac{q\Delta t}{2m}\mathbf{B}^t$$

$$\mathbf{s} = \frac{2\bar{\tau}}{1+\tau^2}$$

$$\mathbf{v}^- = \mathbf{v}^{\mathbf{t-1/2}} + \frac{q\Delta t}{2m}\mathbf{E}^t$$

$$\mathbf{v}' = \mathbf{v}^- + \mathbf{v}^- \times \bar{\tau}$$

$$\mathbf{v}^+ = \mathbf{v}^- + \mathbf{v}' \times \mathbf{s}$$

$$\mathbf{v}^{\mathbf{t+1/2}} = \mathbf{v}^+ + \frac{q\Delta t}{2m}\mathbf{E}^t$$

Gyration in Magnetic field

Lets consider the case : particle in **B**-field

Let **E** =0, Bx =0, By =0, Bz = 0.01T

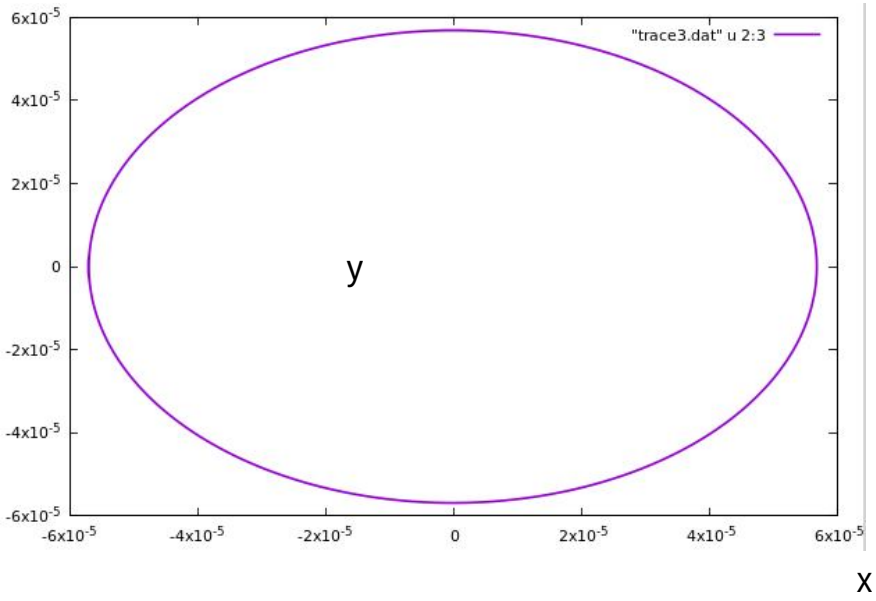
Initial velocity vx = 0, vy = 1e5 m/s, vz =0

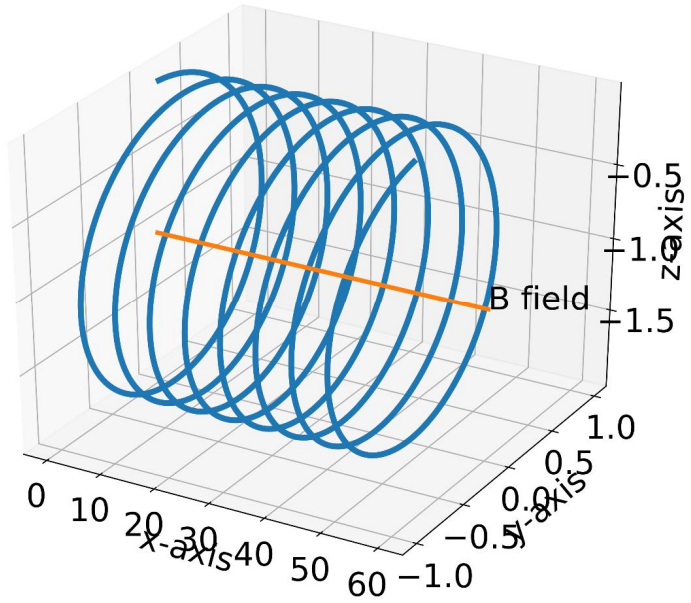
Initial position x= 5.68e-5m, y =0, z =0

Electron

Mass = 9.1e-31 kg

Charge = -1.6e-19 C





Kinetic Model : Particle Method

Vlasov Maxwell system

$$\frac{\partial f}{\partial t} + \mathbf{v}\frac{\partial f}{\partial \mathbf{x}} + \frac{q}{m}\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

$$\begin{aligned} \nabla \cdot \mathbf{E} &= -\frac{\rho}{\epsilon_0} \\ \nabla \cdot \mathbf{B} &= 0 \\ \nabla \times \mathbf{E} &= -\frac{\partial B}{\partial t} \\ \nabla \times \mathbf{B} &= \mu_0\left(\mathbf{J} + \epsilon_0\frac{\partial E}{\partial t}\right) \end{aligned}$$

$$\frac{d\mathbf{v}}{dt} = \frac{q}{m}\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right)$$

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

$$\rho(\mathbf{x}) = \sum_j q_j \delta(\mathbf{x} - \mathbf{x}_j)$$

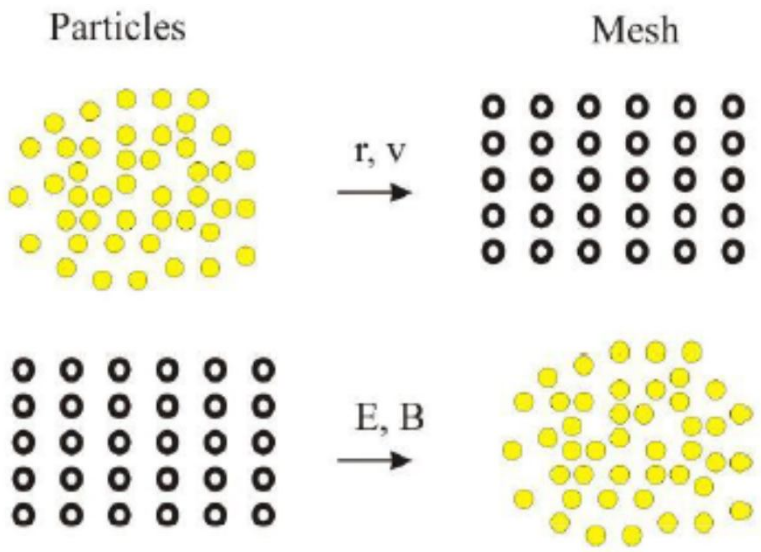
$$\mathbf{J}(\mathbf{x}) = \sum_j q_j \mathbf{v}_j \delta(\mathbf{x} - \mathbf{x}_j)$$

Electric and magnetic fields can be self induced or external from electrodes

Direct solution will be very expensive

Particle in Cell : Concept

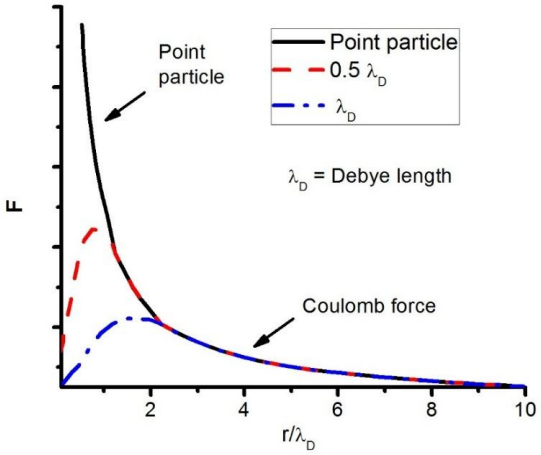
- 6D Vlasov Equations are not practical
- (Re-) intoduce computational particle called super-particle discretize f(**x**,**v**,t)
- Thus particles in Lagrangian frame are tracked in continuous phase space, and fields are computed on Eulerian mesh

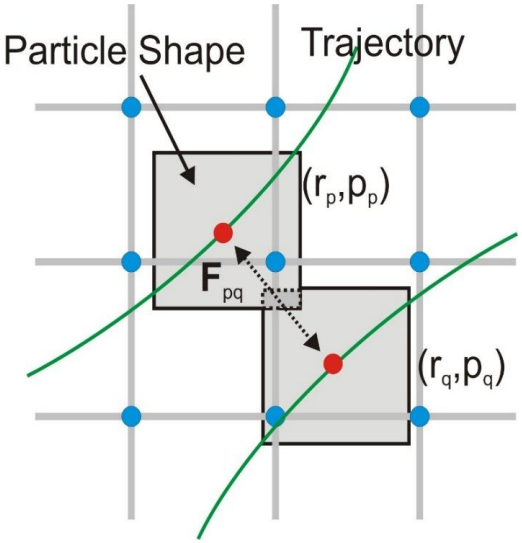


Particle in Cell : Concept

- 6D Vlasov Equations are not practical
- (Re-) intoduce computational particle called super-particle discretize f(**x,v**,t)
- Thus particles in Lagrangian frame are tracked in continuous phase space, and fields are computed on Eulerian mesh

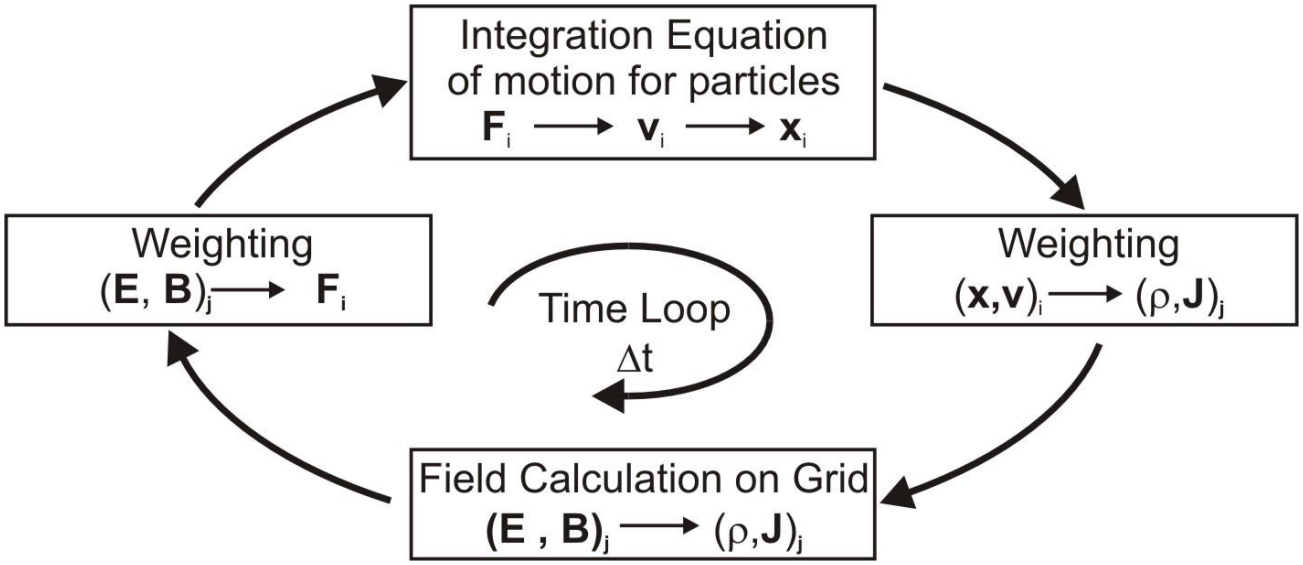
Advantage of finite-size particle is that being finite-sized they interact more weakly than point particle. Thus singularity is avoided as they approach zero distance.





Particle in Cell: The main loop

- Integration of the equation of motion
- Interpolation of charge and current source terms to the field mesh
- Computation of field on grid points
- Interpolation of the fields from mesh to the particle location



PIC: Weighting Methods

Particle in Cell = divide particle quantities like charge in the cell which it is located

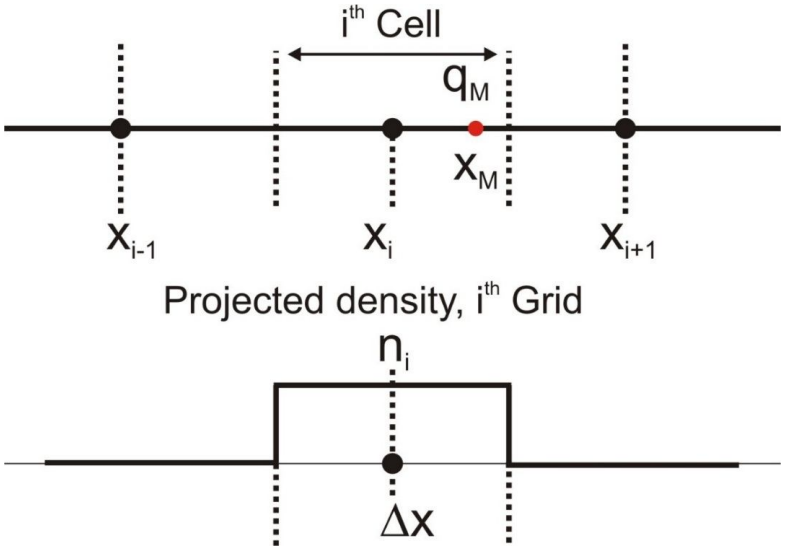
Many methods exist, grouped into 4

- Nearest Grid Point
- Cloud in Cell (CIC)
  - Shaped particles
  - PIC methods, linearly shaped particles
- Multiple
  - Dipole, subtracted dipole, etc.
- Higher order methods
  - Splines
  - K-space cutoff in discrete transform

- Possible hybrid methods also exist.

Weighting: Nearest Grid Point

- Nearest Grid Point assigns charges to nearest grid cell
- Fast and easy to implement in 1D, 2D and 3D
- Noisy
- Deposit macro particle charge
- Similary the current densities can also be deposited $q_M \mathbf{v}_M$



- Charge deposition at ith grid point

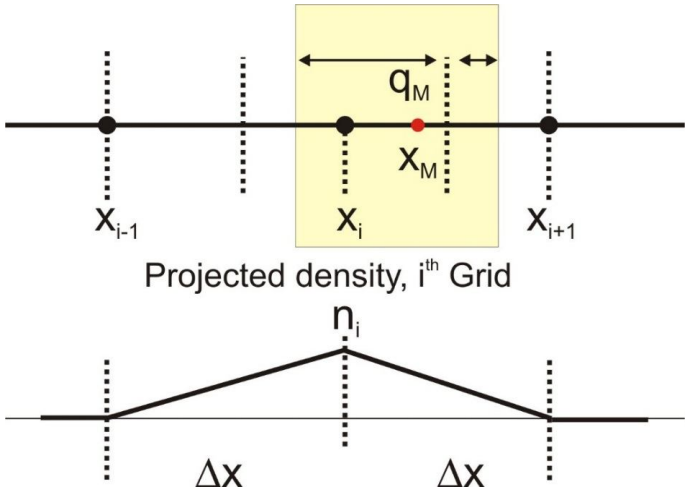
$$q_i = q_M$$

- Field interpolation to Particle

$$\mathbf{E}x\big|_{x=xM} = E_{xi}$$

Weighting: Cloud in Cell

- Cloud in cell is smoother than NGP at the cost of additional computation
- Linear interpolation is used results in triangular shaped particles



Charge deposition at grid points

$$q_i = q_M \frac{x_{i+1} - x_M}{\Delta x}$$

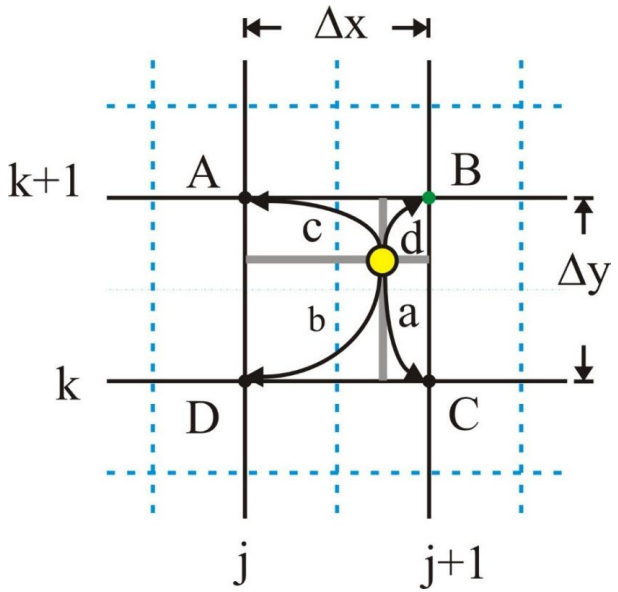
$$q_{i+1} = q_M \frac{x_M - x_i}{\Delta x}$$

Field interpolation to Particle

$$E_x|_{x=x_m} = \left[\frac{x_{i+1} - x_M}{\Delta x}\right] E_i + \left[\frac{x_M - x_i}{\Delta x}\right] E_{i+1}$$

Weighting : Charge division in 2D , Area weighting

- In 2D Cloud in Cell method weighting is accomplished using rectangular „area weighting“ to the nearest grid point
- Procedure can be extended in 3D using volume elements
- Currents can be interpolated similar way



PIC: Field calculation on Grid

Most commonly used methods are catagorized into

- Finite Difference Method (FDM)
  - Oldest of the methods
  - Continuous domain is replaced by discrete grid points
  - Derivatives are approximated with differences between neighbouring grid point values
- Finite Element Method (FEM)
  - Continuous domain is divided into elements
  - PDEs are treated as eigenvalue problem and solution is calculated using basis functions
  - This method is more accurate that FDM at the cost higher computational power
- Spectral Element Method
  - In this case also PDEs are transformed into eigenvalue problem but basis functions are higher order
  - Domain can remain continuous
  - This is very recent method though have problems implementing complex geometries

PIC : Electrostatic field solution

The total electric field E is from externally applied field and space charge

$$\mathbf{E} = \mathbf{E_a} + \mathbf{E_s}$$

The Poisson‘s equation

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

And from Potential we get

$$E = -\nabla\phi$$

Discretized form of Poisson‘ equation in 1D

$$\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} = \frac{\rho_i}{\epsilon_0}$$

$$\begin{bmatrix} -2 & 1 & \cdot & \cdot & & & & 0 \\ \cdot & 1 & 2 & 1 & \cdot & \cdot & & \\ \cdot & & 1 & 2 & 1 & \cdot & \cdot & \\ & & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & 1 & -2 & 1 \\ 0 & & & & \cdot & \cdot & 1 & -2 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \cdot \\ \cdot \\ \phi_{nx-2} \\ \phi_{nx-1} \end{bmatrix} = \frac{\Delta x^2}{\epsilon_0} \begin{bmatrix} \rho_1 + \frac{\epsilon_0}{\Delta x^2} V_l \\ \rho_2 \\ \rho_3 \\ \cdot \\ \cdot \\ \rho_{nx-2} \\ \rho_{nx-1} + \frac{\epsilon_0}{\Delta x^2} V_r \end{bmatrix}$$

Matrix Eqquation of form

$$A\mathbf{x} = b$$

This can be easly extended in 2D and 3D form

Solution for Matrix Equation

- Direct method
  - Fast inversion of Sparse matrx
  - Computes precise solution to problem
  - Example: Gaussian Elimination
- Spectral method
  - Fast Fourier Transform
  - Periodic boundary conditions can be easily applied
- Iterative methods
  - Most preferred method
  - Starting from Initial guess, in successive approximations solution is achieved
  - Example: Jacobi, Gauss-Seidel, Successive Over Relaxation, Multigrid

PIC : Electrodynamic field solution

- The electrodynamic part is solved using inhomogeneous wave equation also known as Telegraphers equation

$$\nabla^2 - \mu_0\epsilon_0\frac{\partial^2}{\partial t^2}\mathbf{E}(\mathbf{x},t) = \mu_0\frac{\partial}{\partial t}\mathbf{j}(\mathbf{x},t)$$

- Assuming harmonically varying fields and currents, we write in complex form,

$$\begin{aligned}\mathbf{E}(\mathbf{x},t) &= \tilde{\mathbf{E}}(\mathbf{x})e^{i\omega t}\\ \mathbf{B}(\mathbf{x},t) &= \tilde{\mathbf{B}}(\mathbf{x})e^{i\omega t}\\ \mathbf{j}(\mathbf{x},t) &= \tilde{\mathbf{j}}(\mathbf{x})e^{i\omega t}\end{aligned}$$

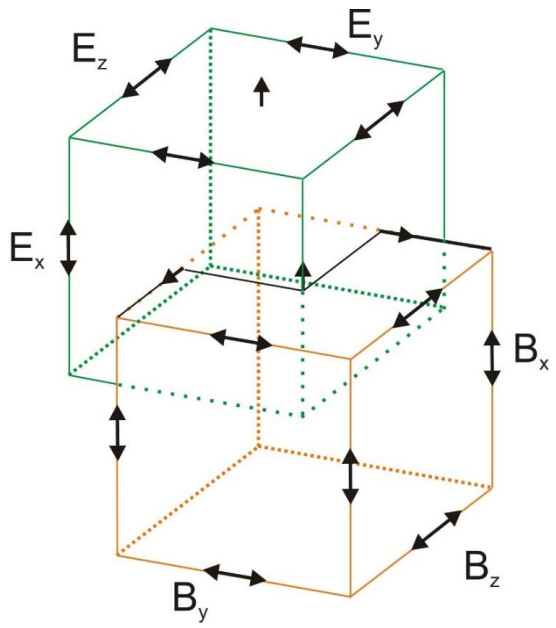
$$\left(\nabla^2 + \mu_0\epsilon_0\omega^2\right)\mathbf{E}(\mathbf{x},t) = i\omega\mathbf{j}(\mathbf{x},t)$$

Here **j** is externally induced currents and from plasma

$$\mathbf{B}(\mathbf{x},t) = \frac{i}{\omega}\nabla\times\mathbf{E}(\mathbf{x},t)$$

PIC : ElectroMagnetic field solution FDTD

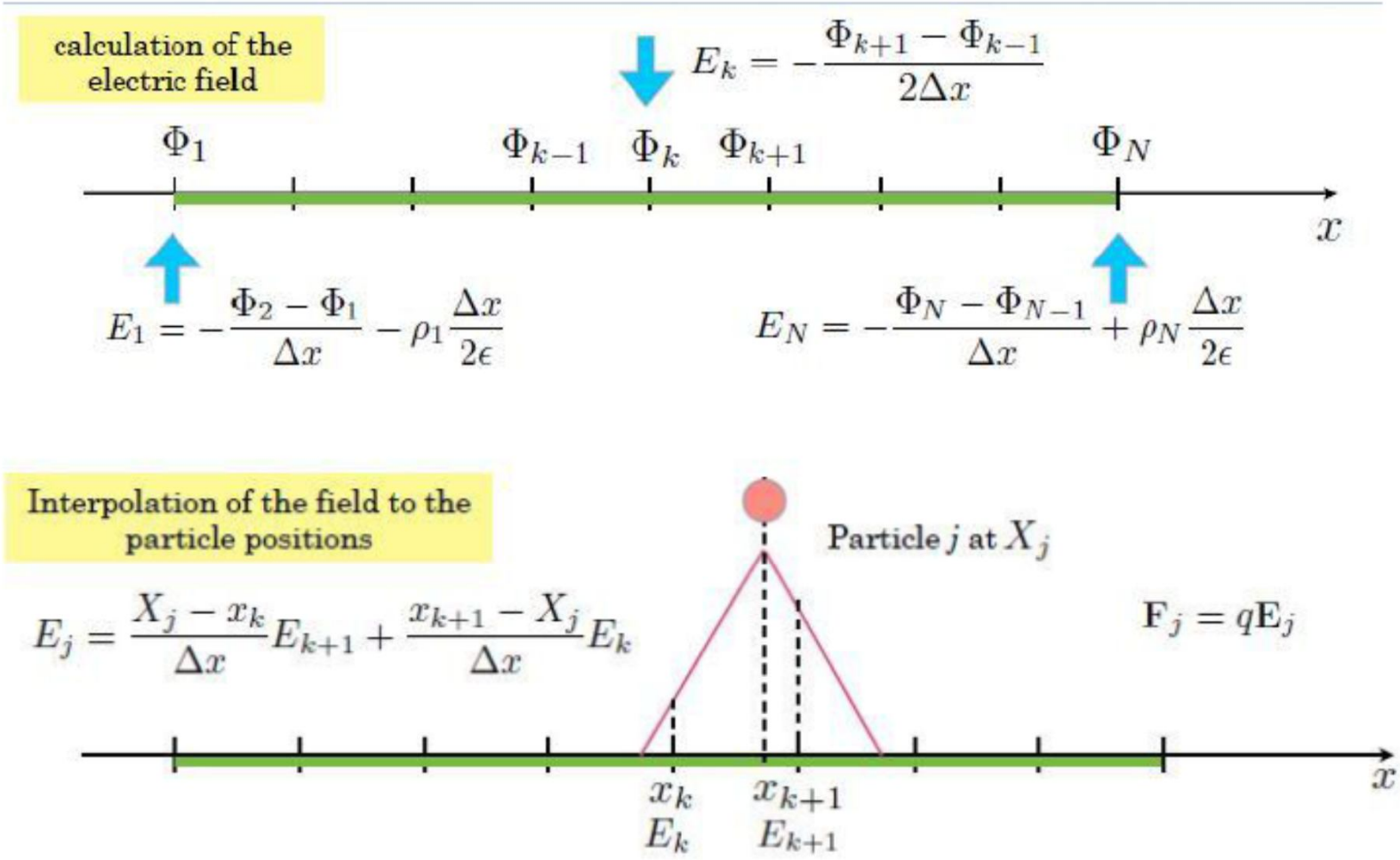
- In Finite difference Time domain staggered space and time stencil call Yee grid is used to solve Maxwell‘s equation for **E** and **B**
- Current deposion using continuity equation in which case Gauss law is preserved
- In direct current deposit method Boris correction is used additionally which is modified form of Gauss‘ laws



$$\frac{B_{y_{k+1/2}}^{n+1/2} - B_{y_{k+1/2}}^{n-1/2}}{\Delta t} = -\frac{E_{x_{k+1}}^{n} - E_{x_k}^{n}}{\Delta z}$$

$$\frac{E_{x_k}^{n+1} - E_{x_k}^{n}}{\Delta t} = -c^2 \left(\frac{B_{y_{k+1/2}}^{n+1/2} - B_{y_{k-1/2}}^{n+1/2}}{\Delta z} \right) - \mu_0 c^2 J_{x_k}^{n+1/2}$$

Electric field at Particle Positions



Poisson Equation

The most general for the *variable-coefficient* Poisson equation

$$\nabla \cdot [u(\mathbf{r})\nabla\phi(\mathbf{r})] = f(\mathbf{r})$$

Where u , ϕ and f are scalar functions of position vector **r**

Special case $u = 1$ Classical form of Poisson Equ.

$$\nabla^2\phi(\mathbf{r}) = f(\mathbf{r})$$

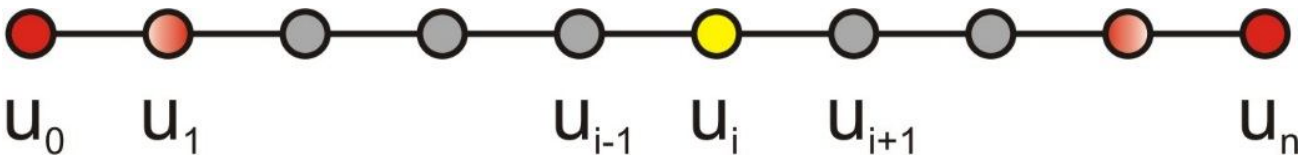
Forcing term f =0 , gives Laplace Equation

$$\nabla^2\phi(\mathbf{r}) = 0$$

Discretization of Poisson Equation

Finite Domain Method (FDM) : Continuous function represented on finite intervals, the variables are calculated at finite number of locations called *grid-points*.

1D stencil : n –grid points



Discretized form of Poisson‘ equation in 1D

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} = f_i$$

Discretization of Poisson Equation ...

$$\frac{u_2 - 2u_1 + u_0}{\Delta x^2} = f_1$$
$$\frac{u_3 - 2u_2 + u_1}{\Delta x^2} = f_2$$
$$\vdots$$
$$\frac{u_n - 2u_{n-1} + u_{n-2}}{\Delta x^2} = f_{n-2}$$

u_o and u_n are necessary known boundary values

The Matrix Ax = b

$$\frac{u_2 - 2u_1 + u_0}{\Delta x^2} = f_1$$
$$\frac{u_3 - 2u_2 + u_1}{\Delta x^2} = f_2$$
$$\vdots$$
$$\frac{u_n - 2u_{n-1} + u_{n-2}}{\Delta x^2} = f_{n-2}$$

u_o and u_n are necessary known boundary values

Matrix Equation of form

$$A\mathbf{x} = b$$

The Matrix Ax = b

$$A\mathbf{x} = b$$

$$\begin{bmatrix} -2 & 1 & \cdot & \cdot & & & & 0 \\ \cdot & 1 & -2 & 1 & \cdot & \cdot & & \\ \cdot & & 1 & -2 & 1 & \cdot & \cdot & \\ & & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & 1 & -2 & 1 \\ 0 & & & & \cdot & \cdot & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdot \\ \cdot \\ u_{nx-2} \\ u_{nx-1} \end{bmatrix} = \begin{bmatrix} f_1 + left\ BC \\ f_2 \\ f_3 \\ \cdot \\ \cdot \\ f_{nx-2} \\ f_{nx-1} + right\ BC \end{bmatrix}$$

Boundary conditions

For an ordinary differential equation

$$y'' + y = 0$$

Dirichlet Boundary Condition

The condition on interval $[a, b]$ is given as

$$y(a) = \alpha, \qquad y(b) = \beta$$

where α and β are constant numbers

Boundary conditions

For an ordinary differential equation

$$y'' + y = 0$$

Neumann Boundary Condition

The condition on interval $[a, b]$ is given as

$$y'(a) = \alpha, \qquad y'(b) = \beta$$

where α and β are constant numbers

Solving Ax = b

- Direct method
  - Fast inversion of Sparse matrx
  - Computes precise solution to problem
  - Example: Gaussian Elimination
- Iterative methods
  - Most preferred method
  - Starting from Initial guess, in successive approximations solution is achieved
  - Example: Jacobi, Gauss-Seidel, Successive Over Relaxation, Multigrid
- Spectral method
  - Fast Fourier Transform
  - Periodic boundary conditions can be easily applied

Direct Method

These methods are based on triangularization of matrix. In these methods the unknowns are eliminated in a systematic way, so that one ends up with a triangular system, which can then be easily solved.

Gaussian Elimination
a sequence of elementary row operations to modify the matrix until the lower left-hand corner of the matrix is filled with zeros, as much as possible.

LU decomposition
This method is based on a decomposition of the matrix A into a lower- and upper-triangular matrix: $A = LU$.
The system $Ax = b$ is then equivalent to $LUx = b$, which decomposes into two triangular systems $Ly = b$ and $Ux = y$.

Iterative Methods : Mesh relaxation

$$u_i = -f_i \frac{\Delta x^2}{2} + \frac{1}{2}\left(u_{i+1} + u_{i-1}\right)$$

Jacobi / Point-Jacobi Method

$$\begin{aligned}
&\text{do } \{ \\
&\quad \mathtt{u_{temp}} = \mathtt{u} \\
&\quad \mathtt{u_{i,j}} = \mathtt{u^{+}_{temp}} \\
&\quad \mathtt{du} = \left|\mathtt{u_{temp}} - \mathtt{u}\right| \\
&\quad \} \text{ while } (\ \mathtt{du} \geq \mathtt{tol}\ )
\end{aligned}$$

- Easy to implement
- Slow

Iterative Methods : Mesh relaxation

$$u_i = -f_i \frac{\Delta x^2}{2} + \frac{1}{2}\left(u_{i+1} + u_{i-1}\right)$$

Gauß – Seidel Method

$$\begin{aligned}
&\text{do } \{ \\
&\quad \mathtt{u_{temp}} = \mathtt{u} \\
&\quad \mathtt{u_{i,j}} = \mathtt{u^{+}_{i,j}} \\
&\quad \mathtt{du} = \left|\mathtt{u_{temp}} - \mathtt{u}\right| \\
&\} \text{ while } (\ \mathtt{du} \geq \mathtt{tol}\ )
\end{aligned}$$

- an improvement on the Jacobi algorithm. We no longer need to allocate two arrays. Instead, we can make just a single array and carry out all the updates in situ.
- But the Gauß Seidel implementation introduces an additional complication that the order in which the updates are applied those will affect the values.

Iterative Methods : Mesh relaxation

$$u_i = -f_i \frac{\Delta x^2}{2} + \frac{1}{2}\left(u_{i+1} + u_{i-1}\right)$$

Successive Over Relaxation

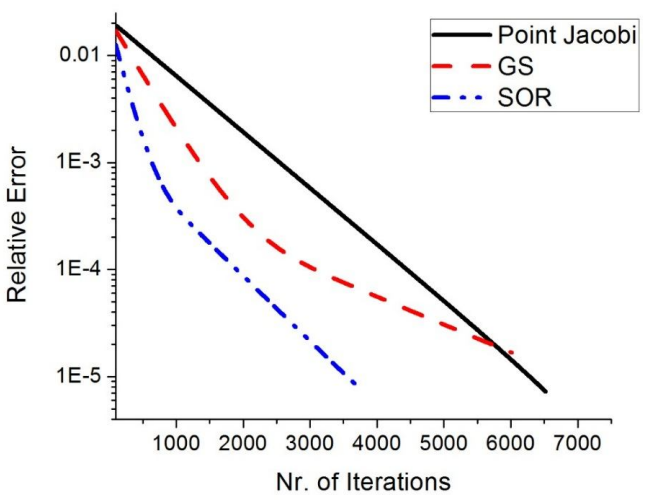
$$\begin{aligned}
&\text{do } \{ \\
&\quad u_{temp} = u \\
&\quad u_{new} = u^{+}_{i,j} \\
&\quad u = u + \omega \left(u_{new} - u \right) \\
&\quad du = \left| u_{temp} - u \right| \\
&\} \text{ while } \left(du \geq tol \right)
\end{aligned}$$

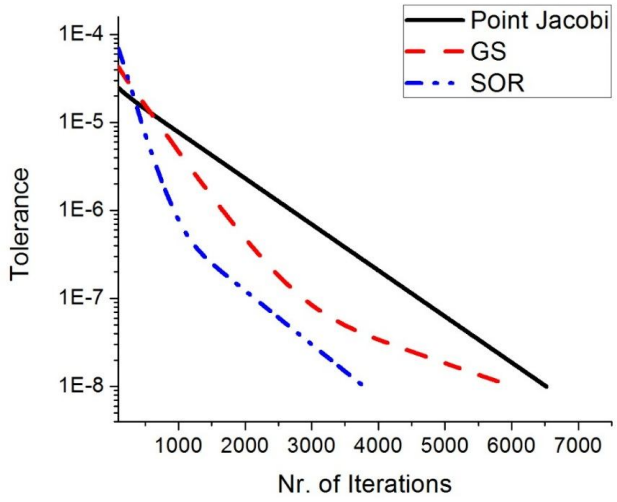
- Refinement for Gauß-Seidel Method
- Displacement vector Δu is calculated and solution is moved with factor ω

Iterative Methods

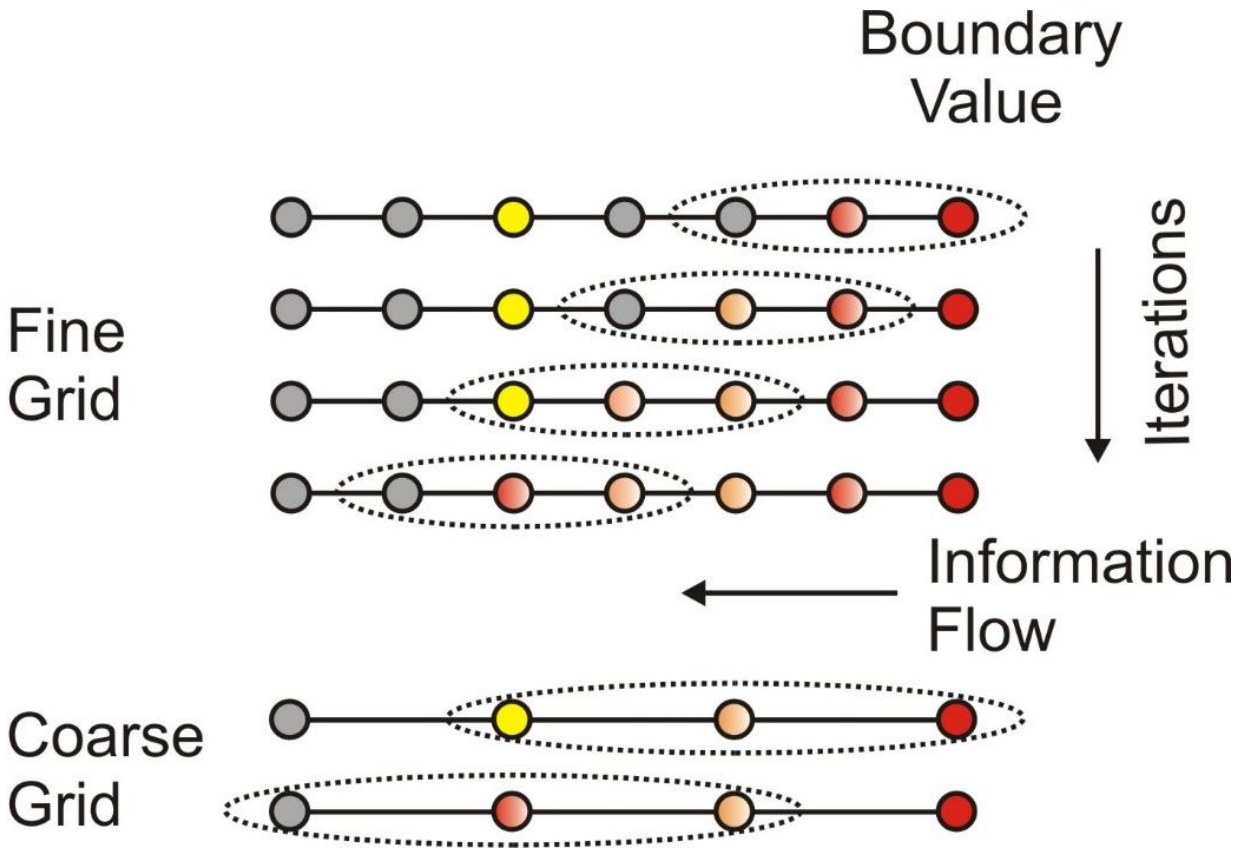
- Jacobi (Point-Jacobi),
- Gauss-Seidel
- Succesive Over Relation(SOR)

In each iterative step $e = ||\mathbf{A}x - b||$ is minimized
In practical example we use tolerance or the change from last step





Multi-grid solver



Multi-grid solver

- **Smoothing** - Reducing high frequency errors, for example using a few iterations of the Gauss-Seidel or SOR method.
- **Residual Computation** - Computing residual error after the smoothing operation/s.
- **Restriction** - Downsampling the residual error to a coarser grid.
- **Prolongation** - Interpolation of a correction computed on a coarser grid into a finer grid.
- **Correction** - Adding prolongated coarser grid solution onto the finer grid.

Algorithm

We start by few iterations on fine grid

$$A(\phi_f^n)^* = b_f$$

Compute Residue and check for termination

$$R_f^n = A(\phi_f^n)^* - b_f$$

Multi-grid solver ...

Algorithm ...

Interpolation of residue on coarser grid : *Restriction operation*

$$R_f \rightarrow R_c$$

Since we have residue we solve for error vector on finer grid

$$A\epsilon_c = R_c$$

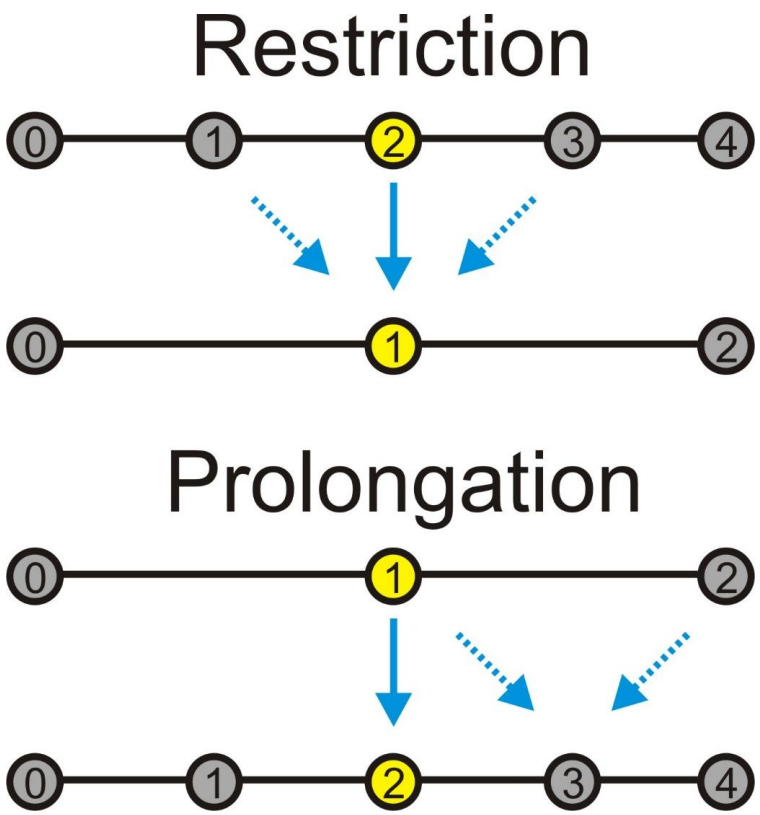
Interpolate error vector on finer grid : *Prolongation operation*

$$\epsilon_c \rightarrow \epsilon_f$$

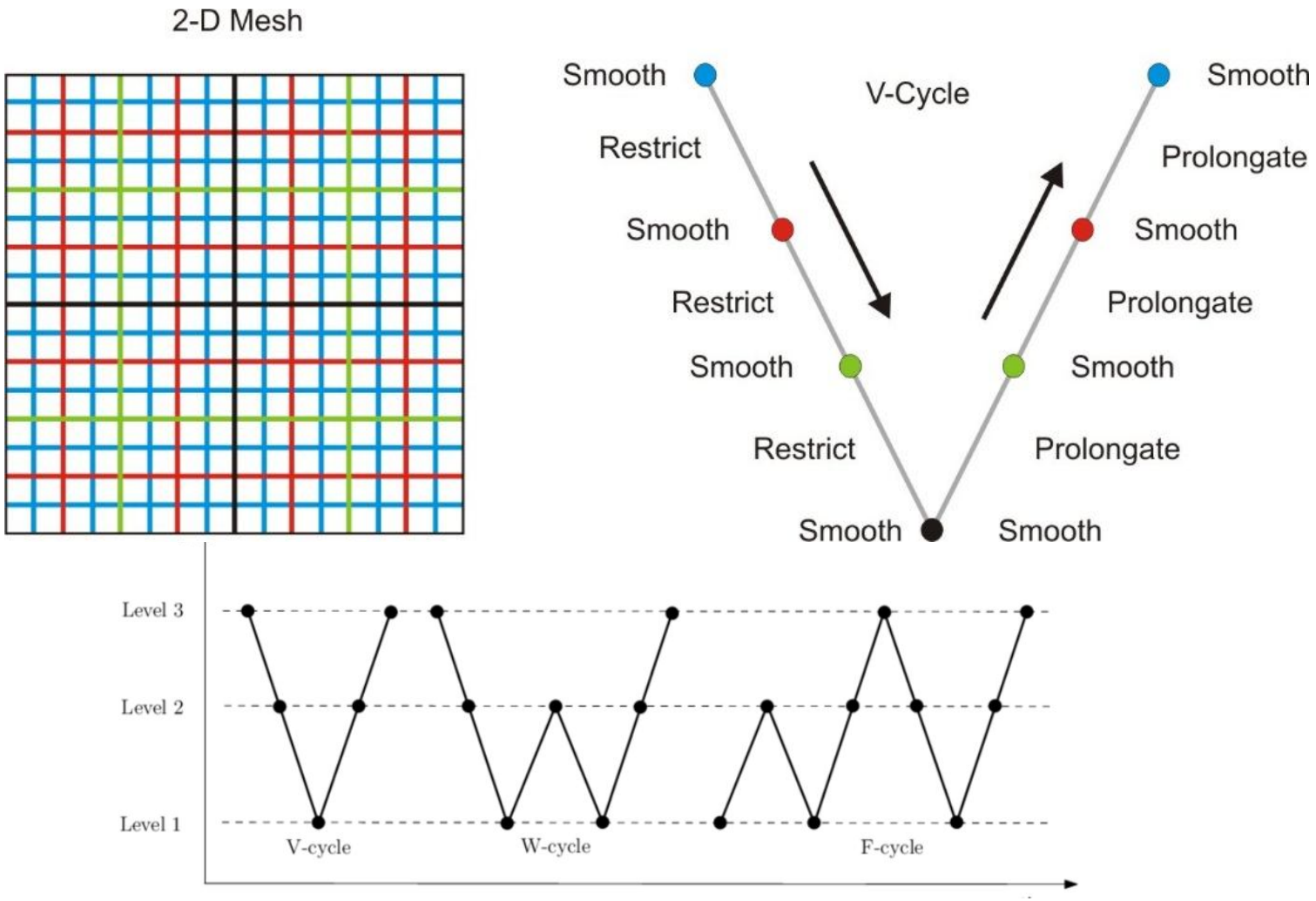
Finally correction on finer grid

$$\phi_f^n = \left(\phi_f^n\right)^* - \epsilon_f$$

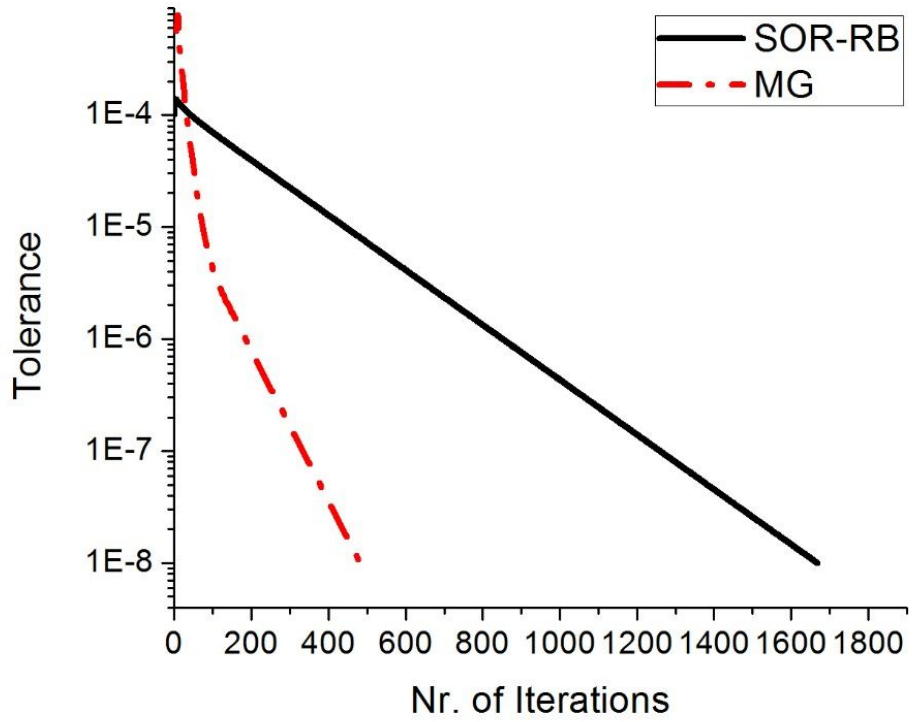
Restriction - Prolongation



The V-cycle


2-D Mesh
Smooth
V-Cycle
Smooth
Restrict
Prolongate
Smooth
Smooth
Restrict
Prolongate
Smooth
Smooth
Restrict
Prolongate
Smooth
Smooth
Level 3
Level 2
Level 1
V-cycle
W-cycle
F-cycle

Multigrid : Comparison with SOR



Spectral Method

$$\frac{d^2\phi}{dx^2} = \rho(x)$$

Express f and ρ in terms of their Fourier transforms

$$f(x) = \frac{1}{\sqrt{2\pi}} \int g(k) e^{ikx} dk \qquad \rho(x) = \frac{1}{\sqrt{2\pi}} \int \sigma(k) e^{ikx} dk$$

The equation is diagonalized in k-space

$$g(k) = -\frac{\sigma(k)}{k^2} \qquad -k^2 g(k) = \sigma(k)$$

The solution is given by the inverse transform

$$f(x) = -\frac{1}{\sqrt{2\pi}} \int \frac{\sigma(k)}{k^2} e^{ikx} dk$$

Particle in Cell : Constraints

Two important conditions to be satisfied

$$\omega_0 \Delta t < 2$$
$$\Delta x \leq \lambda_D$$

Third condition $N_D >> 1$ holds true for plasmas except strongly coupled plasmas

According to Tskhakaya et al, and Hockeney and Eastwood modified to

$$\omega_{pe} \Delta t \leq 0.2$$
$$\Delta x < 3.4 \lambda_D$$

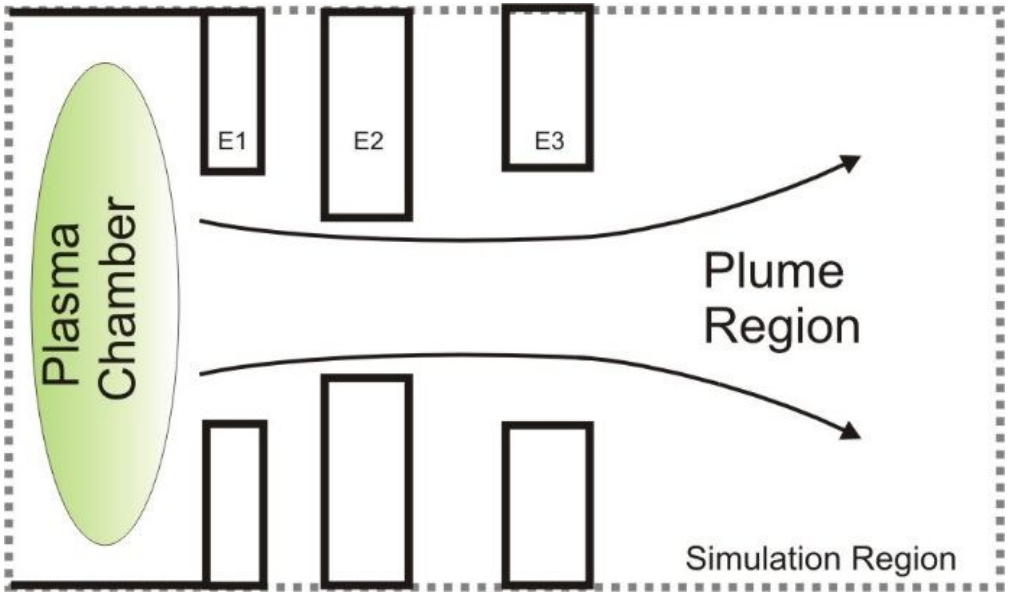
Example: Consider plasma with density 10^{16} m^{-3} and temperature 10 eV

This gives Debye length $\lambda_D = 2.35 \times 10^{-4}$ m and plasma frequency $= 5.64 \times 10^{9}$ s^{-1}

Hence we have

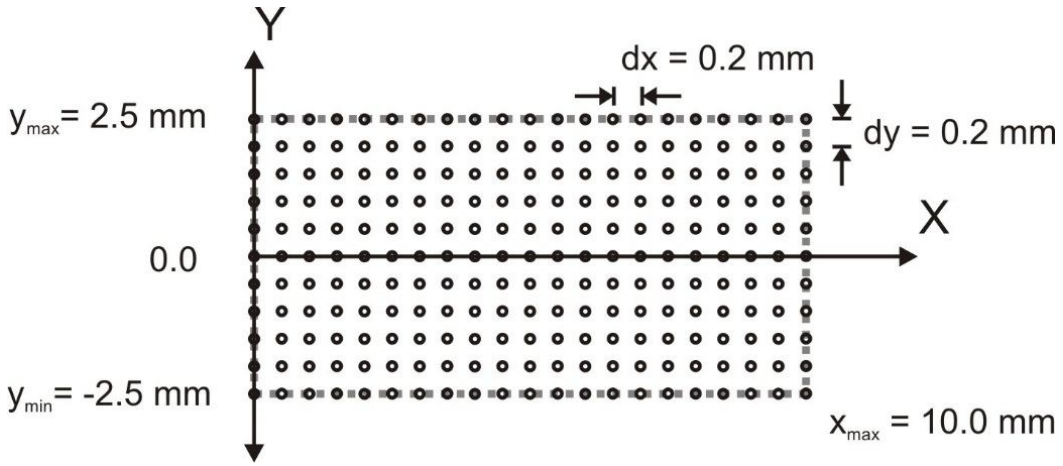
$\Delta t \leq 3.55 \times 10^{-11}$ s and $\Delta x < 8 \times 10^{-4}$ m

Scope of the simulation project



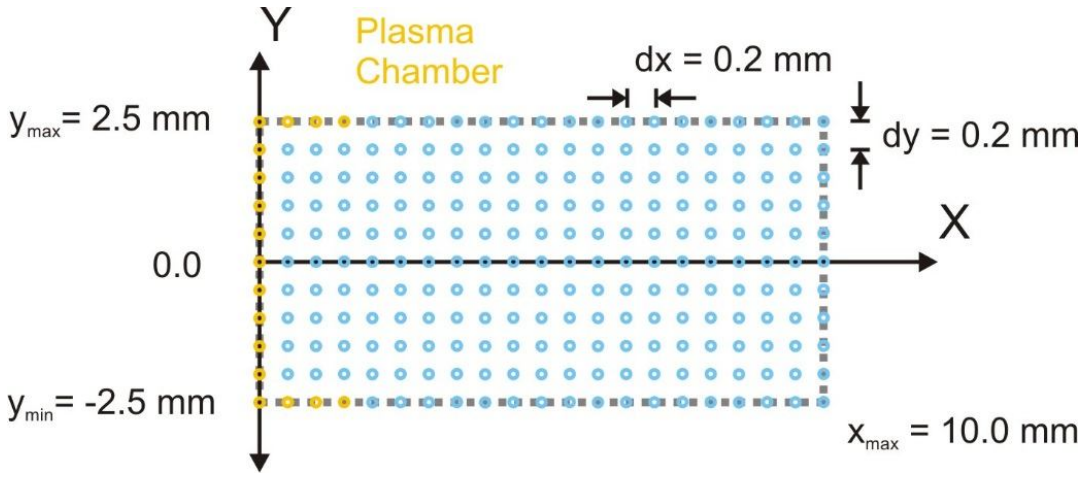
Domain : 2D
Plasma inside chamber
Extraction of ions
Formation of exhaust plume

Domain

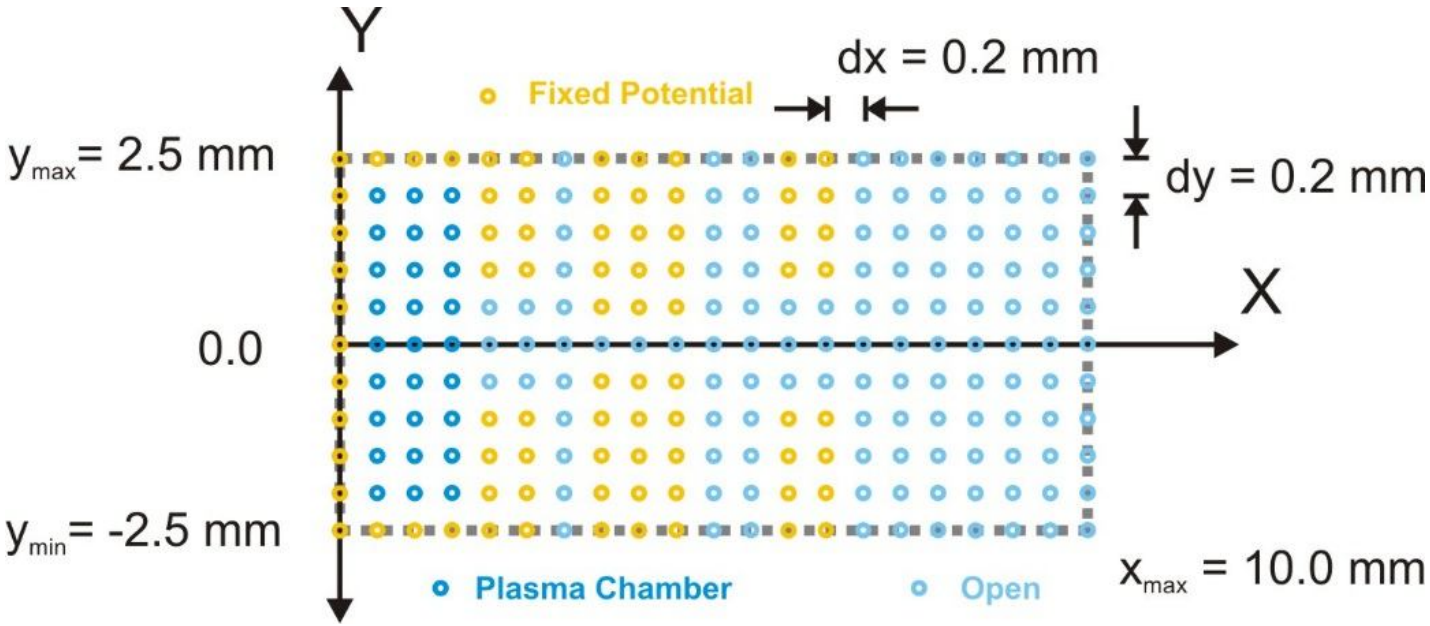


Domain : 2D
Ni = 51
Nj = 26

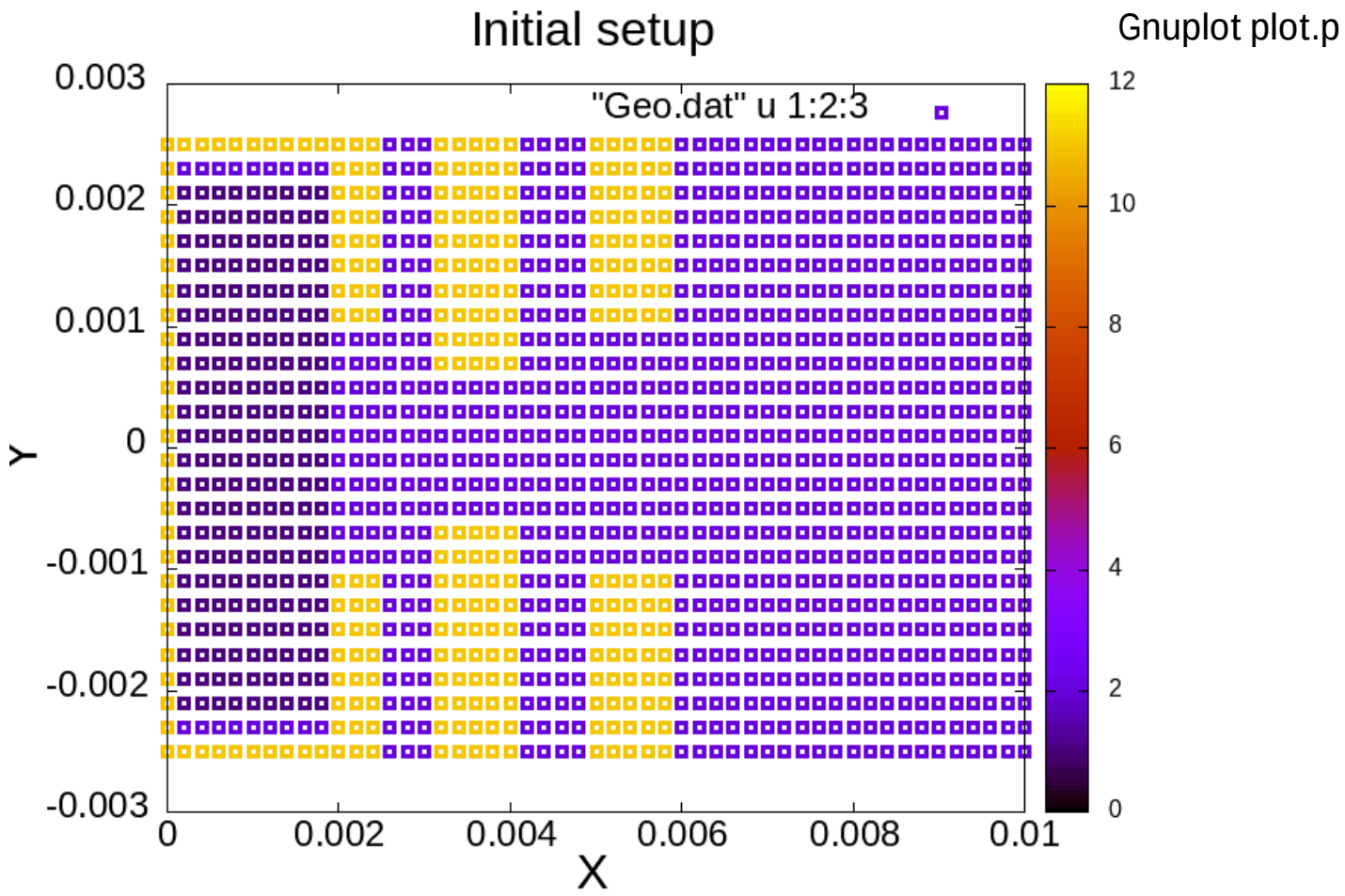
Plasma
Chamber len = 2 mm
Chamber r =2.2 mm



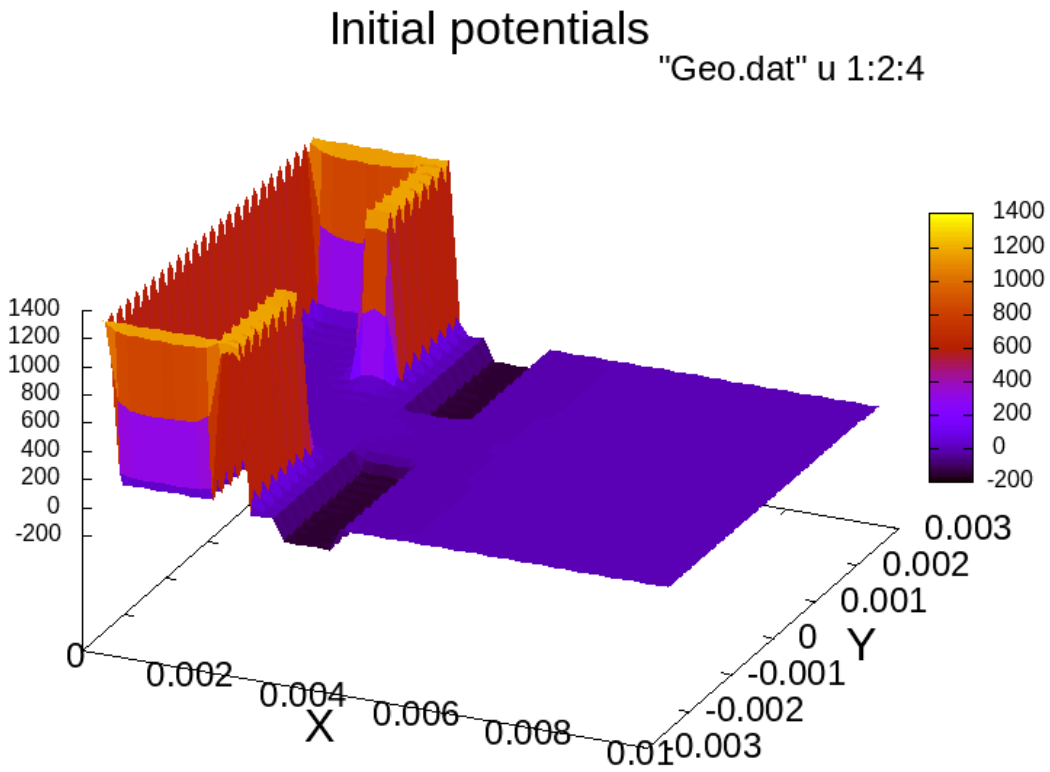
Domain ...


Y
dx = 0.2 mm
Fixed Potential
y_{max}= 2.5 mm
dy = 0.2 mm
X
0.0
y_{min}= -2.5 mm
x_{max} = 10.0 mm
Plasma Chamber
Open

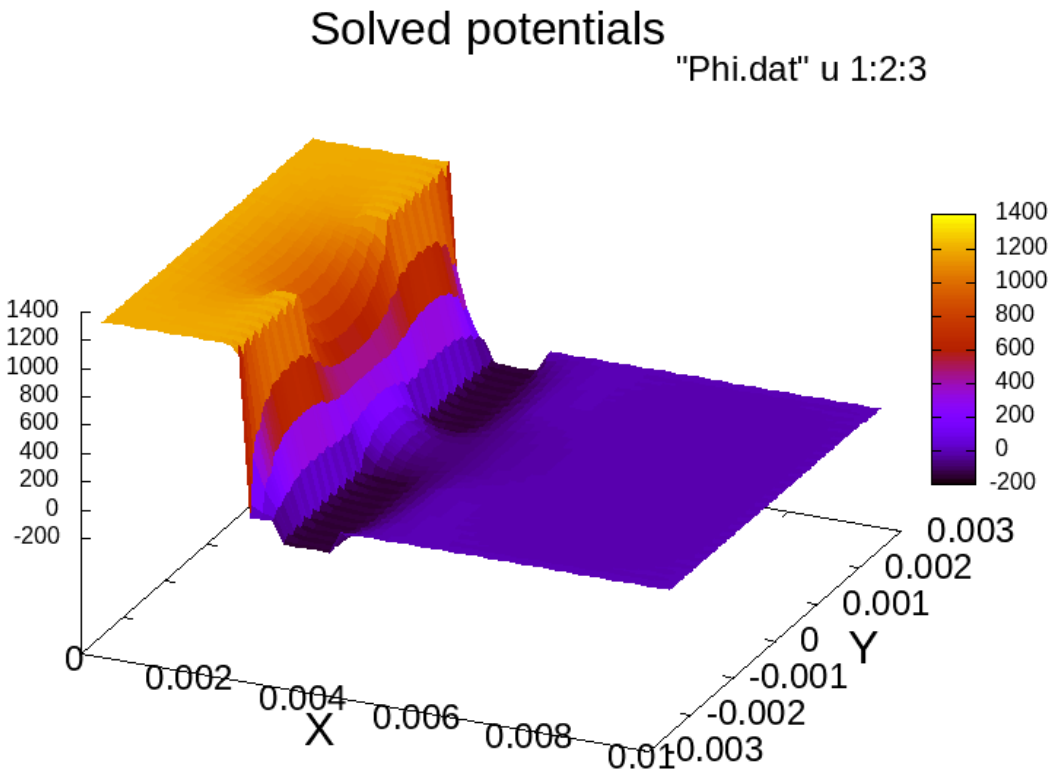
Domain ...


Initial setup
"Geo.dat" u 1:2:3
Gnuplot plot.p
0.003
0.002
0.001
0
-0.001
-0.002
-0.003
Y
0
0.002
0.004
0.006
0.008
0.01
X
12
10
8
6
4
2
0

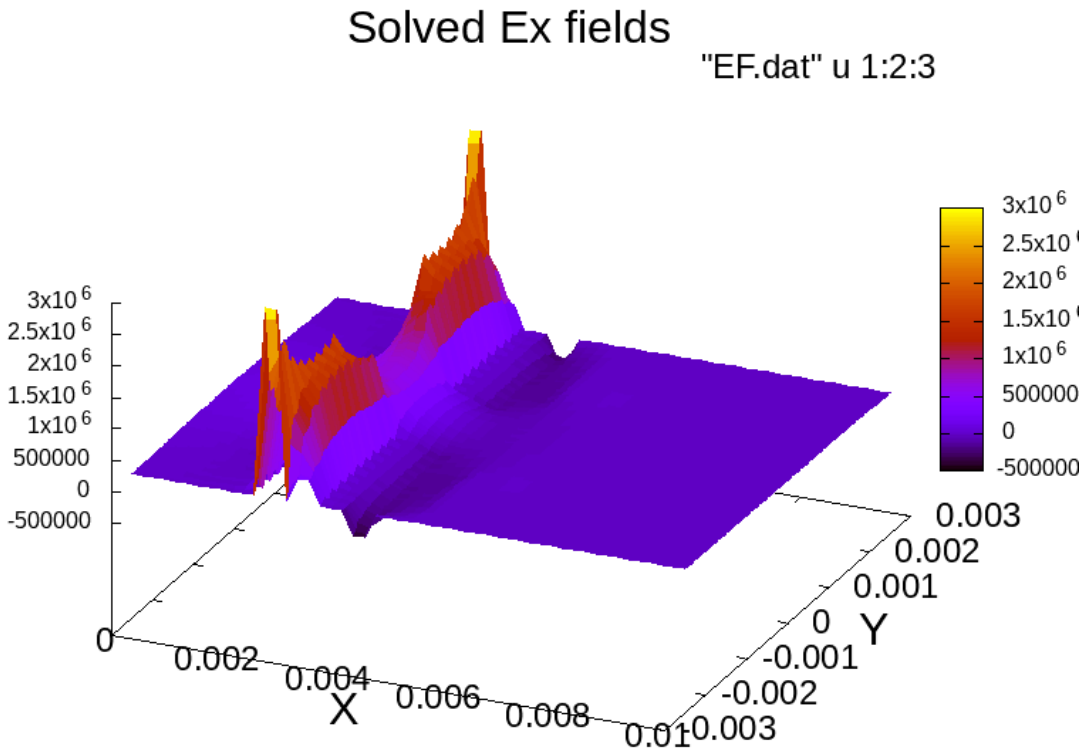
Domain ...


Initial potentials
"Geo.dat" u 1:2:4
1400
1200
1000
800
600
400
200
0
-200
0
0.002
0.004
0.006
0.008
0.01
X
0.003
0.002
0.001
0
-0.001
-0.002
-0.003
Y

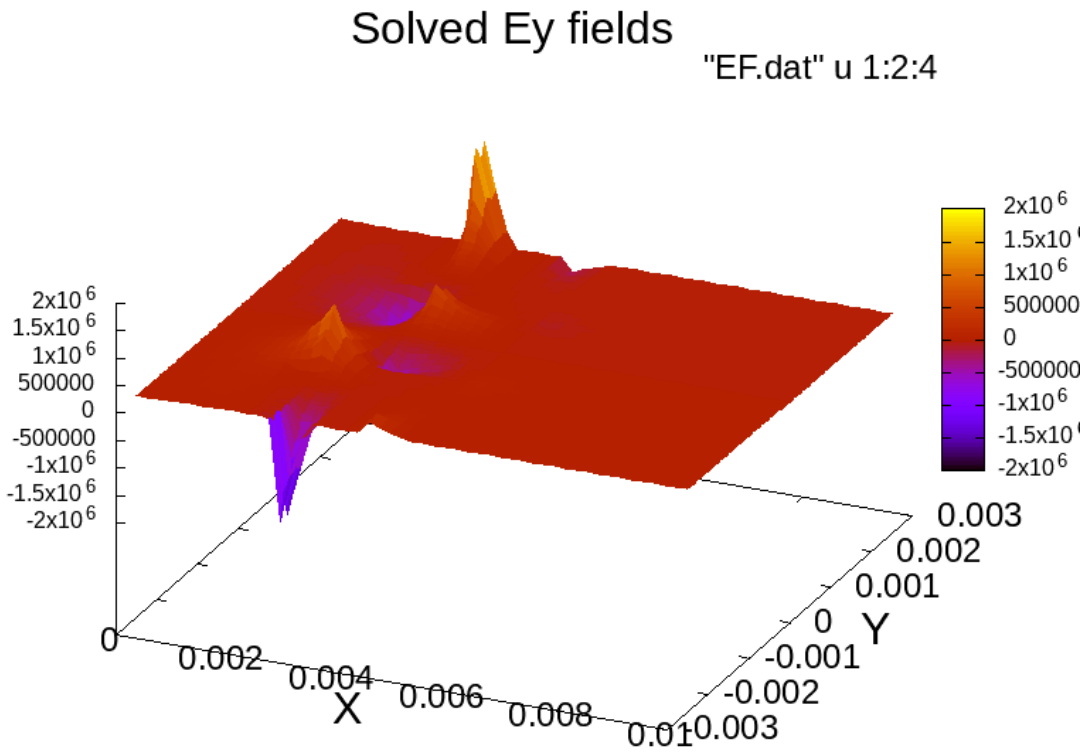
Domain ... Solving Poisson Equation



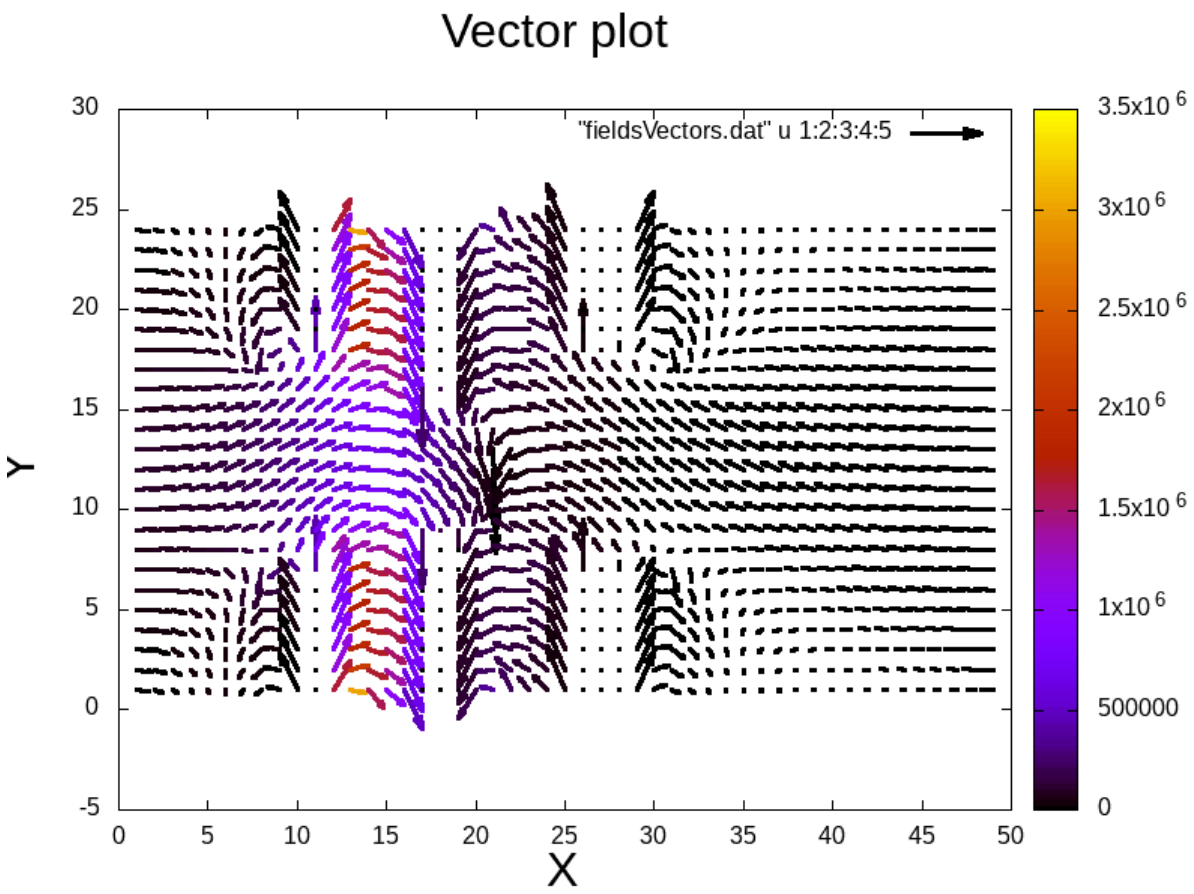
Domain ... Plotting Electric fields



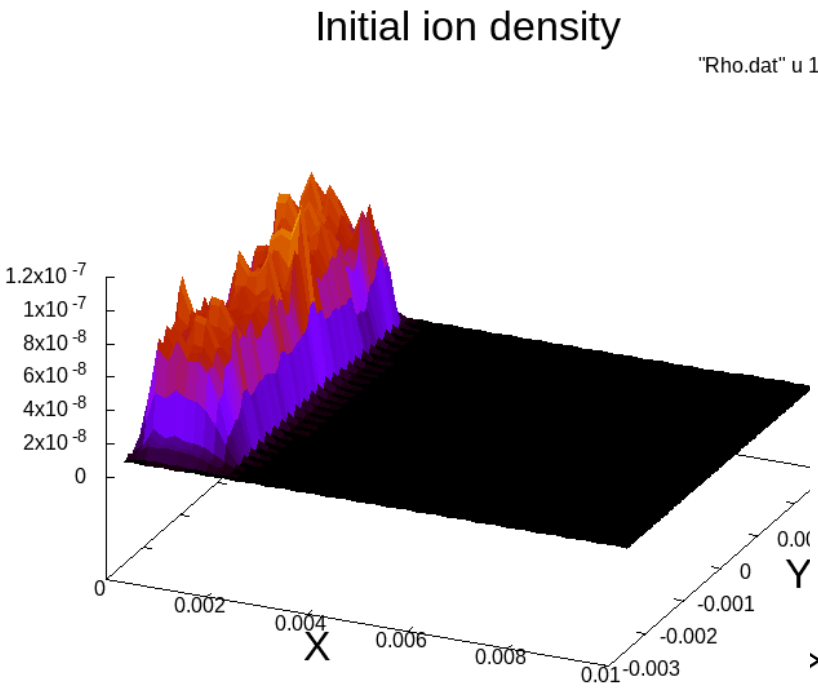
Domain ... Plotting Electric fields



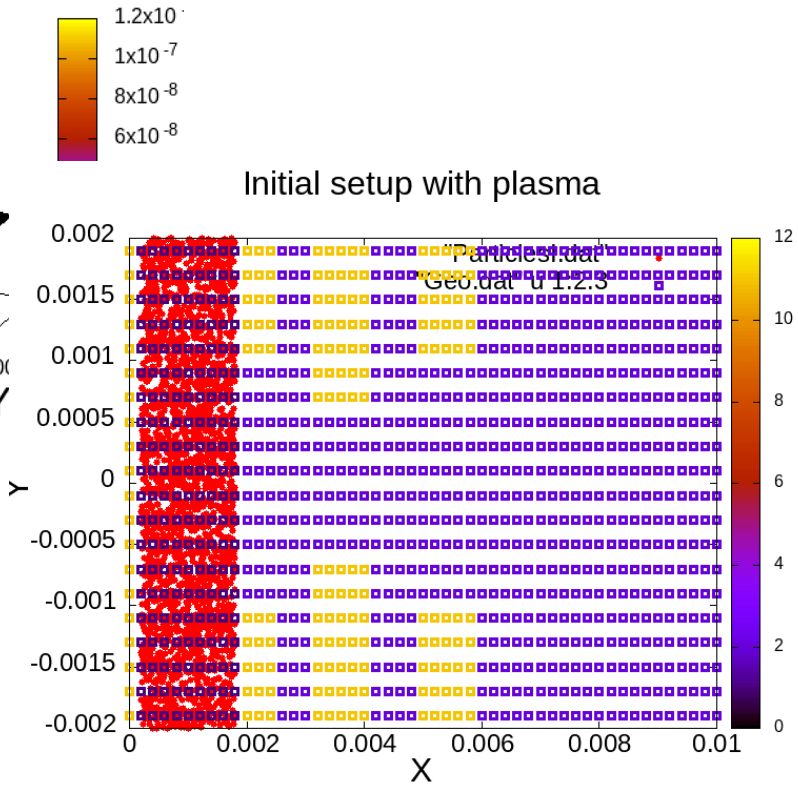
Domain ... Plotting Electric fields



Species ... Distribution

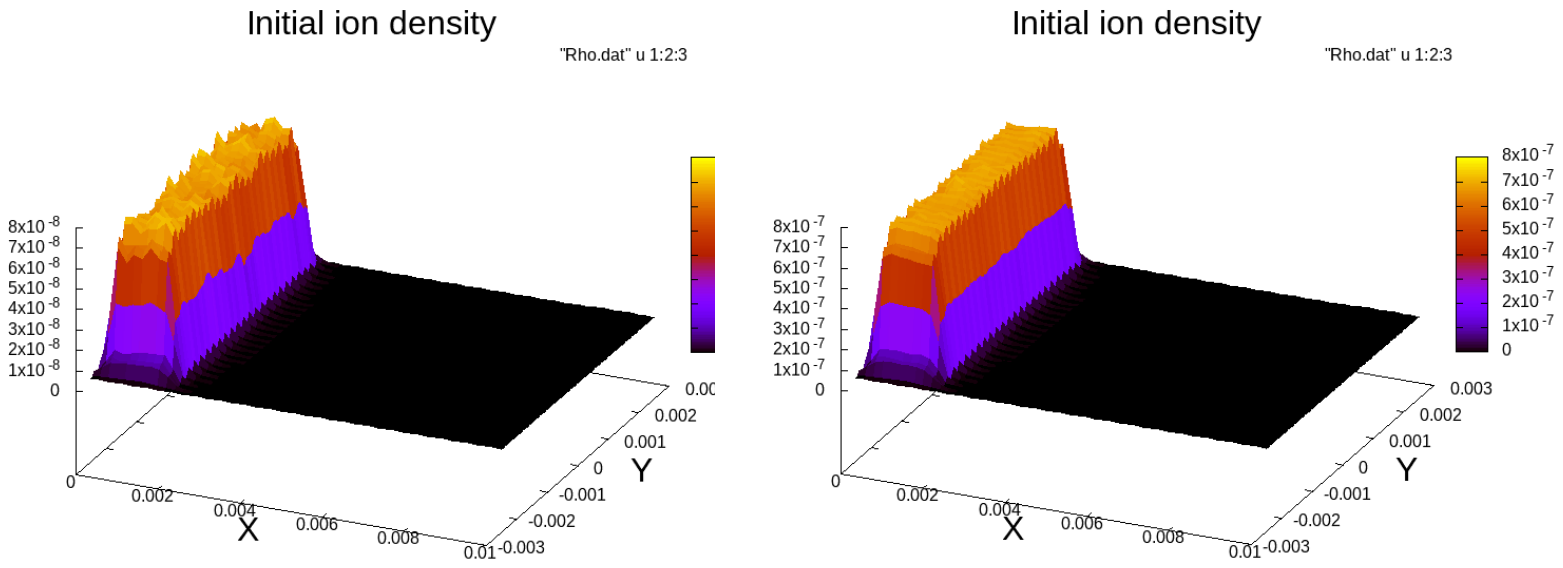


Step5.cpp



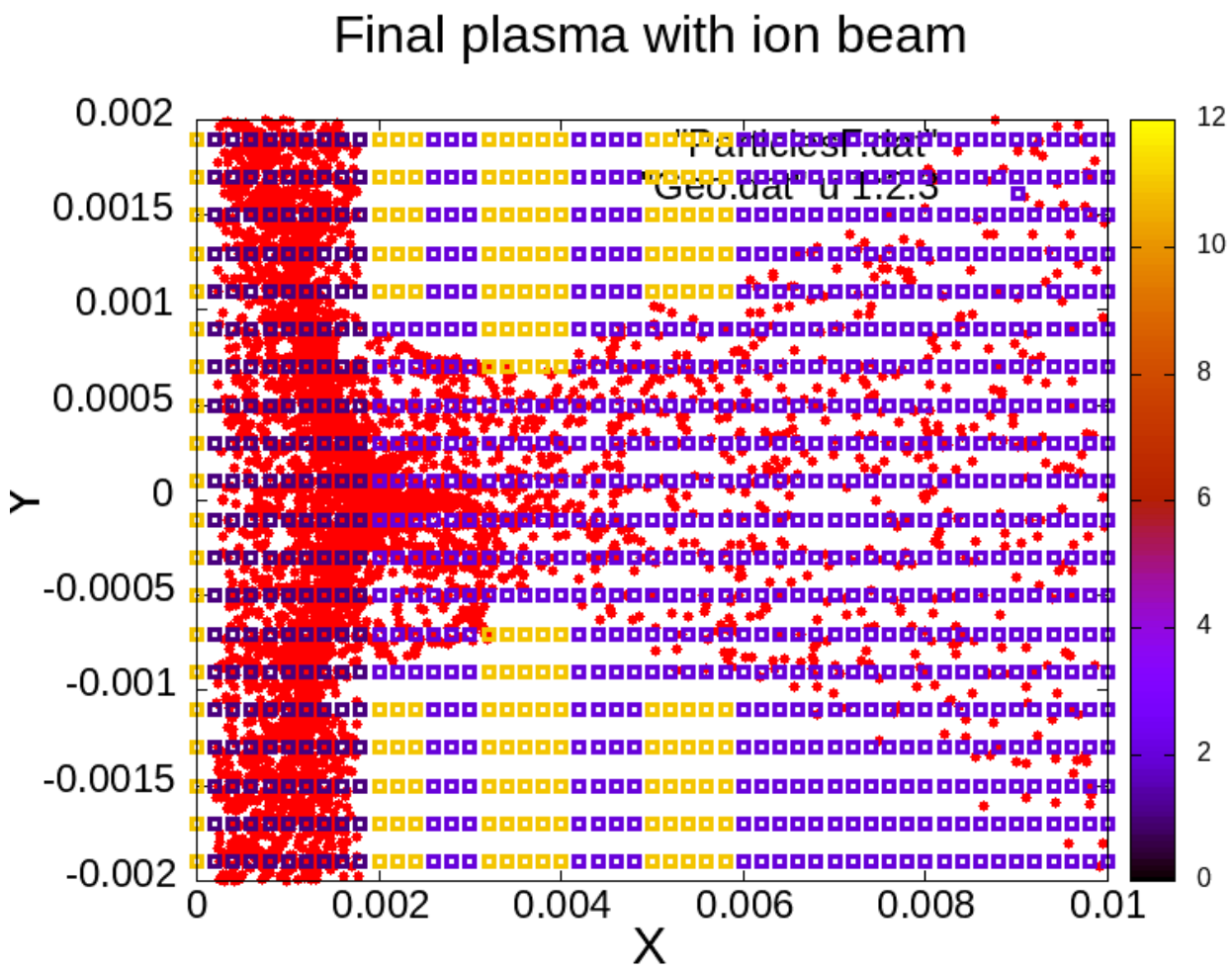
Species ... density

Step5.cpp

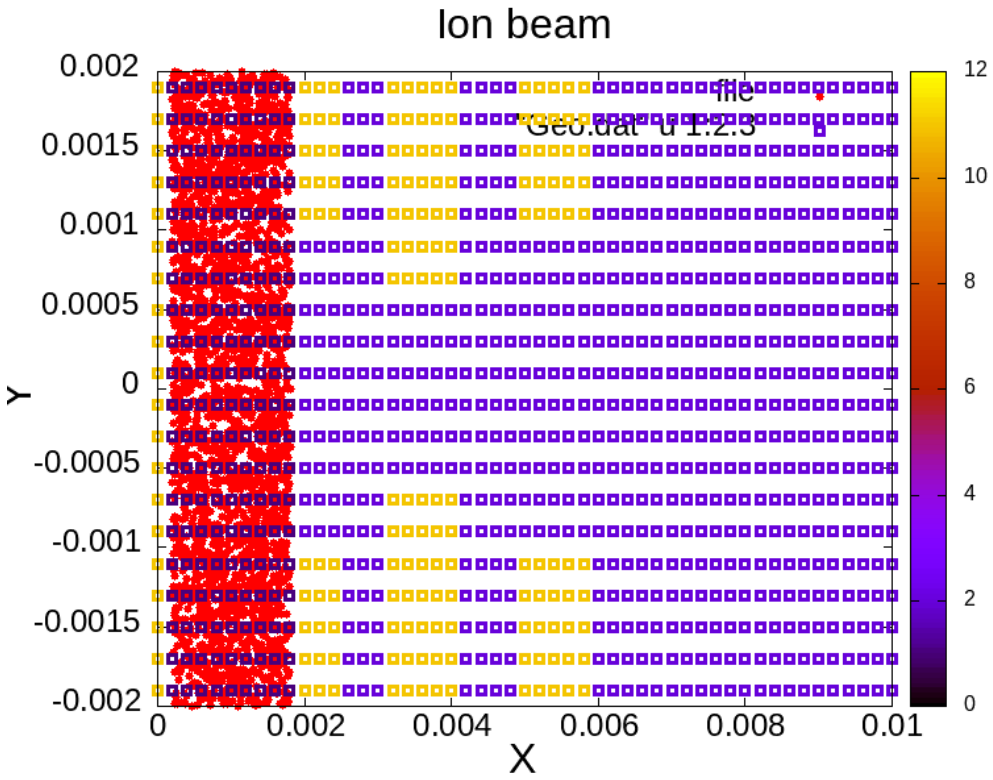


Flatter distribution with increasing number density

Species ... Beam formation



Species ... Beam formation



Create folder named results
Run Step7.cpp
Copy plotLoop.p, Geo.dat
Then run
gnuplot plotLoop.p
Animate using convert command under Linux