
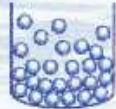

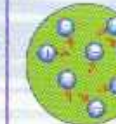


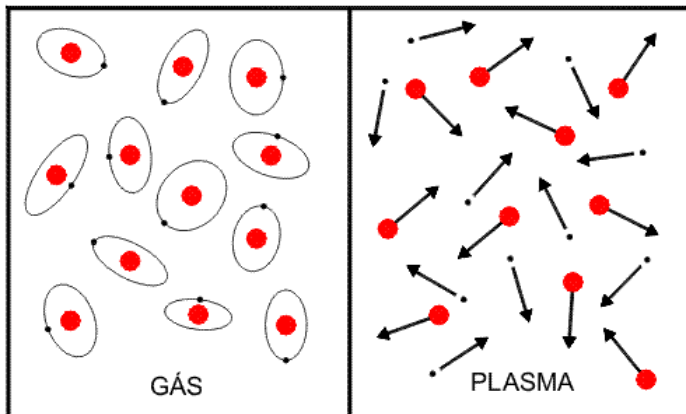
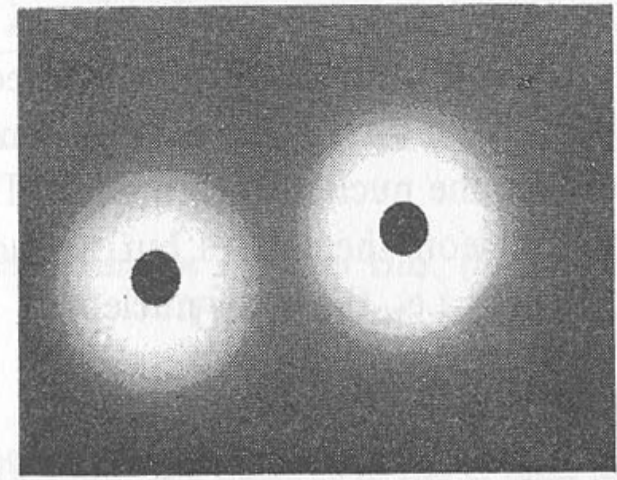



Introduction to Particle In Cell


Giovanni Lapenta

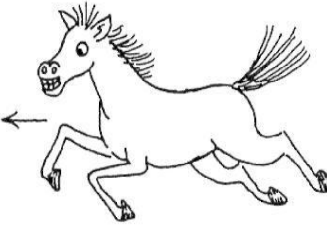
Plasma Physics 0

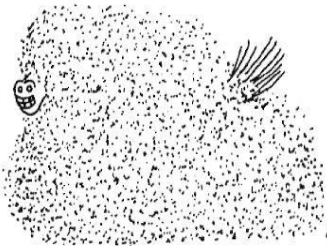
| Solid | Liquid | Gas | Plasma |
|---|---|---|---|
| Example Ice H_2O | Example Water H_2O | Example Steam H_2O | Example Ionized Gas $H_2 \rightarrow H^+ + H^+ + 2e^-$ |
| Cold $T < 0^\circ C$ | Warm $0 < T < 100^\circ C$ | Hot $T > 100^\circ C$ | Hotter $T > 100,000^\circ C$ (> 10 electron Volts) |
|  |  |  |  |
| Molecules Fixed in Lattice | Molecules Free to Move | Molecules Free to Move, Large Spacing | Ions and Electrons Move Independently, Large Spacing |




real particle


quasi particle

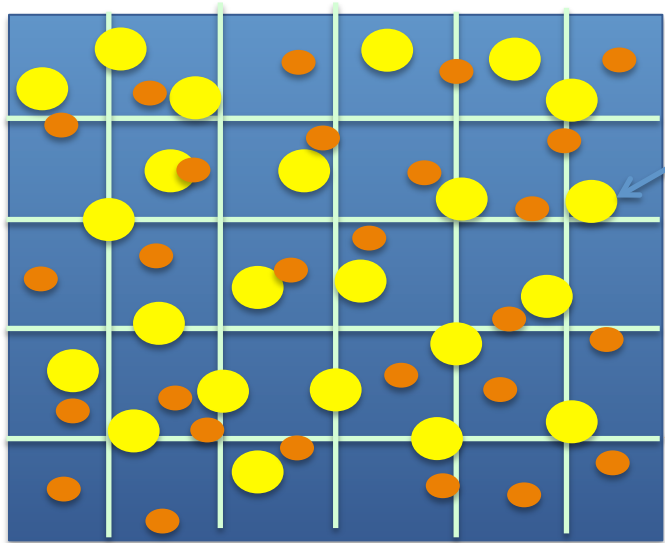

real horse


quasi horse

Quasi Particle Concept

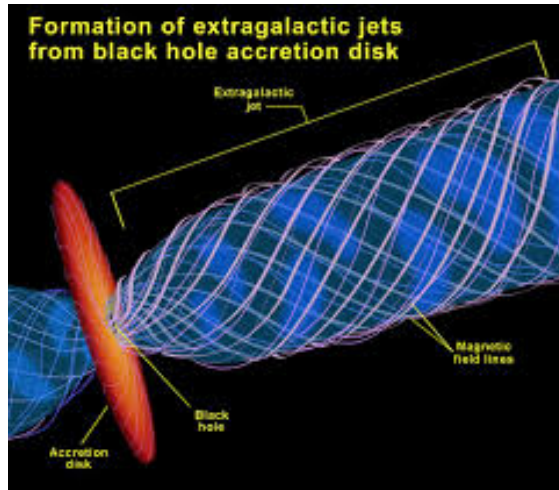
RDM

Challenge of modelling multiple physical levels

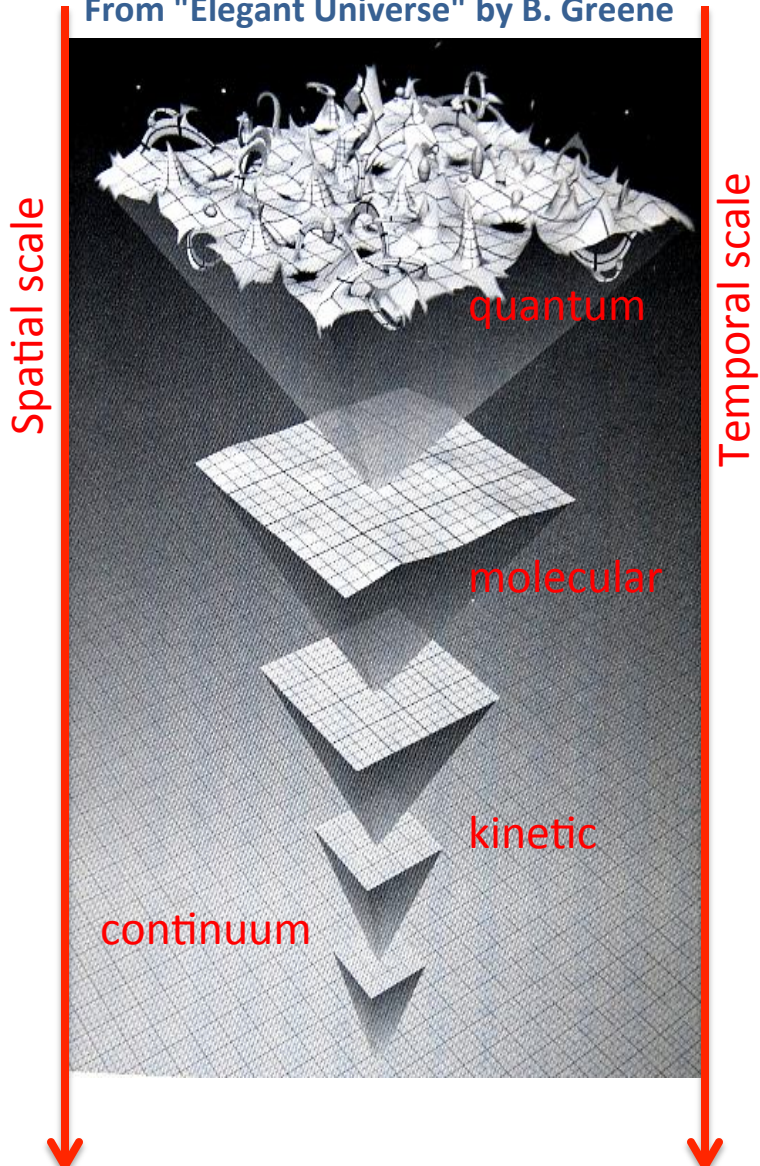


X_p
 P_p

E, B
 ρ, J



From "Elegant Universe" by B. Greene



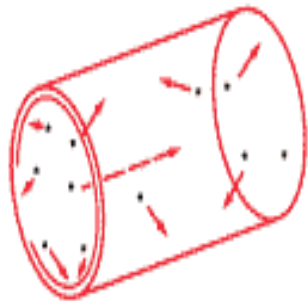
A plasma and its models



GAS



PLASMA



Motion of charged particles
without magnetic field.



Motion of charged particles
with magnetic field.

Single particle level: tracing every single particle and their interactions:

$$\{x_p, v_p\}$$

Kinetic approach: study the distribution function (probability of finding a particle with a given velocity in a given point at a given time):

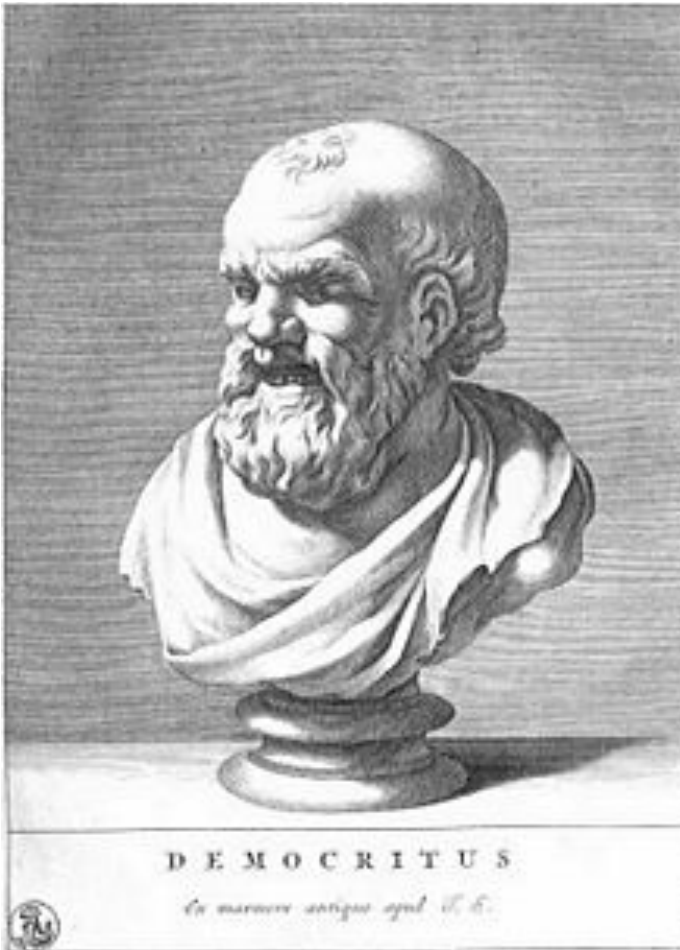
$$f(\mathbf{x}, \mathbf{v}, t)$$

Fluid approach: study local averages (density, average speed, temperature,.....)

$$n(\mathbf{x}, t), \mathbf{U}(\mathbf{x}, t), T(\mathbf{x}, t)$$

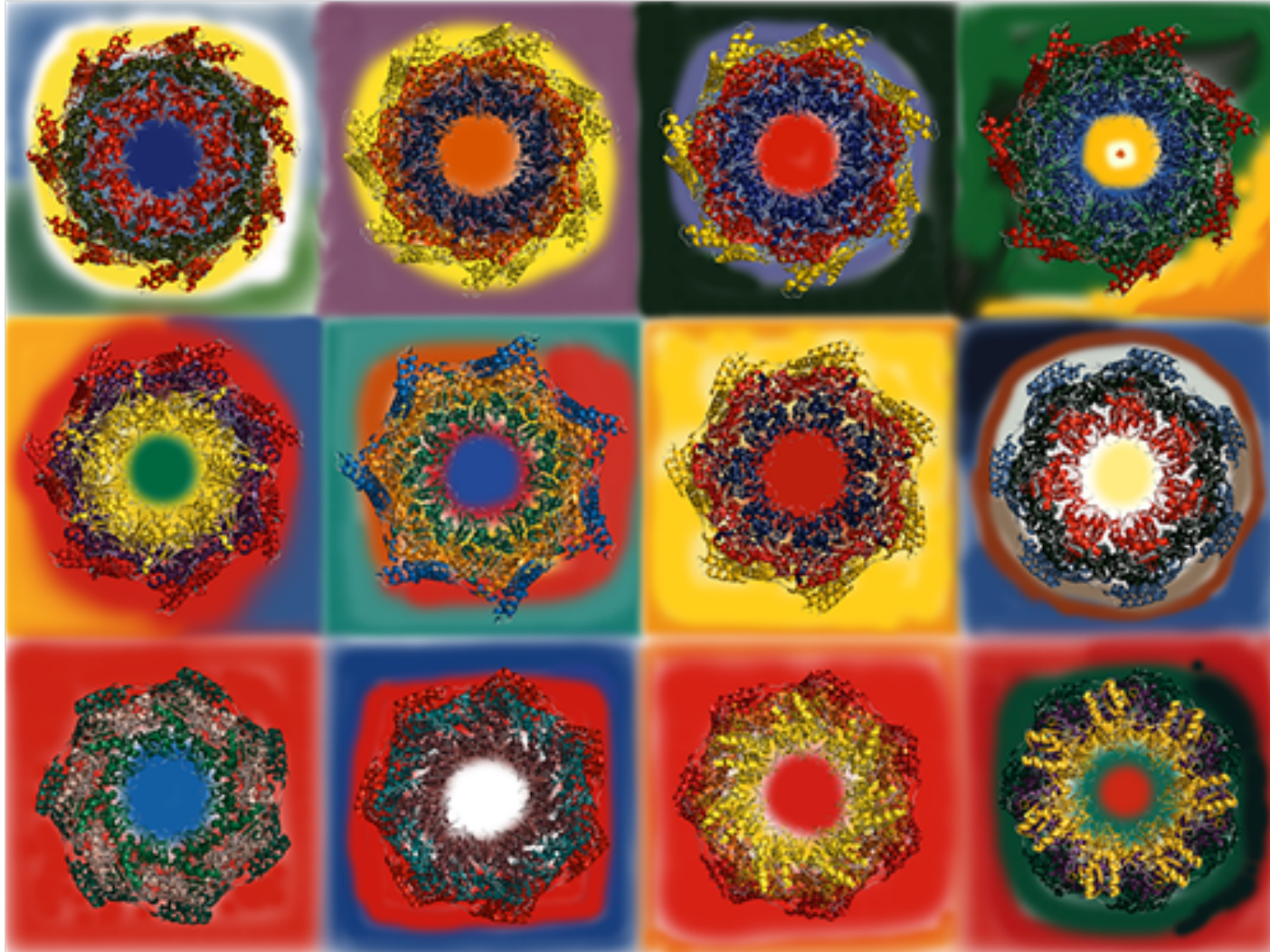


Wassily Kandinsky: Squares and Concentric Rings

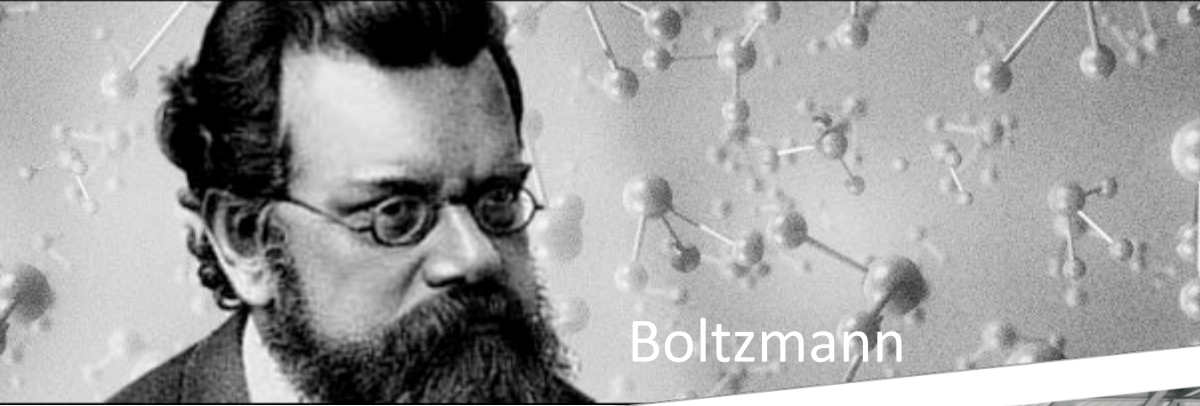


Sweet exists by convention,
Bitter by convention,
Colour by convention;
Atoms and void alone exist in
reality.

Micro: Kinetic Theory



Zheng Yang, Peter Májek, and Ivet Bahar (2009) Allosteric Transitions of Supramolecular Systems Explored by Network Models: Application to Chaperonin GroEL. [PLoS Comput Biol 5\(4\): e1000360](https://doi.org/10.1371/journal.pcbi.1000360).

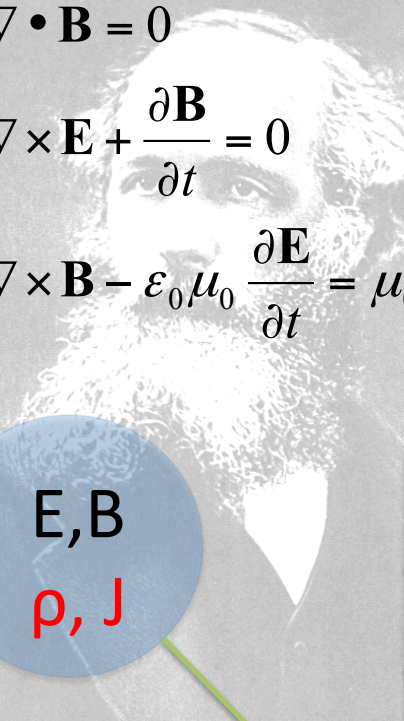


Boltzmann

The two approaches



Fluid Approach



$$\nabla \cdot \mathbf{E} = \rho$$

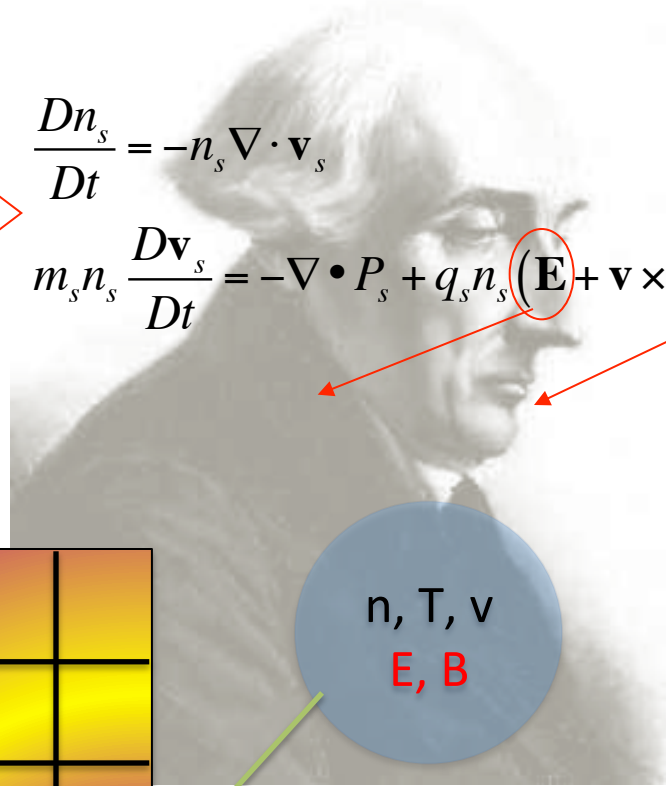
$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0$$

$$\nabla \times \mathbf{B} - \epsilon_0 \mu_0 \frac{\partial \mathbf{E}}{\partial t} = \mu_0 \mathbf{J}$$

E, B
 ρ, \mathbf{J}

Maxwell

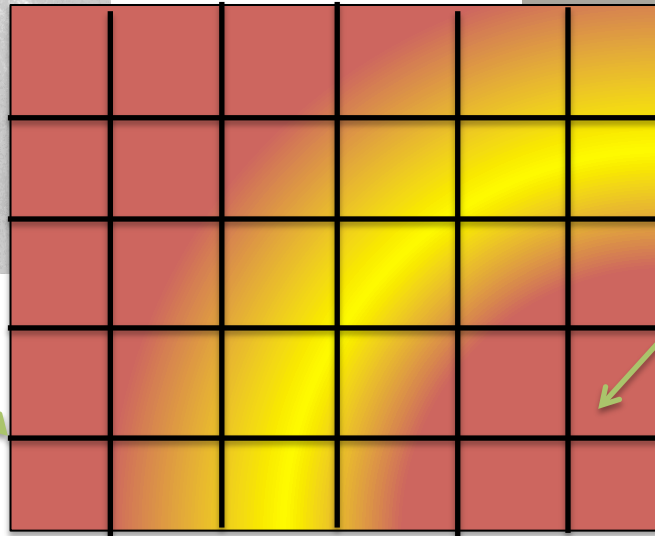



$$\frac{Dn_s}{Dt} = -n_s \nabla \cdot \mathbf{v}_s$$

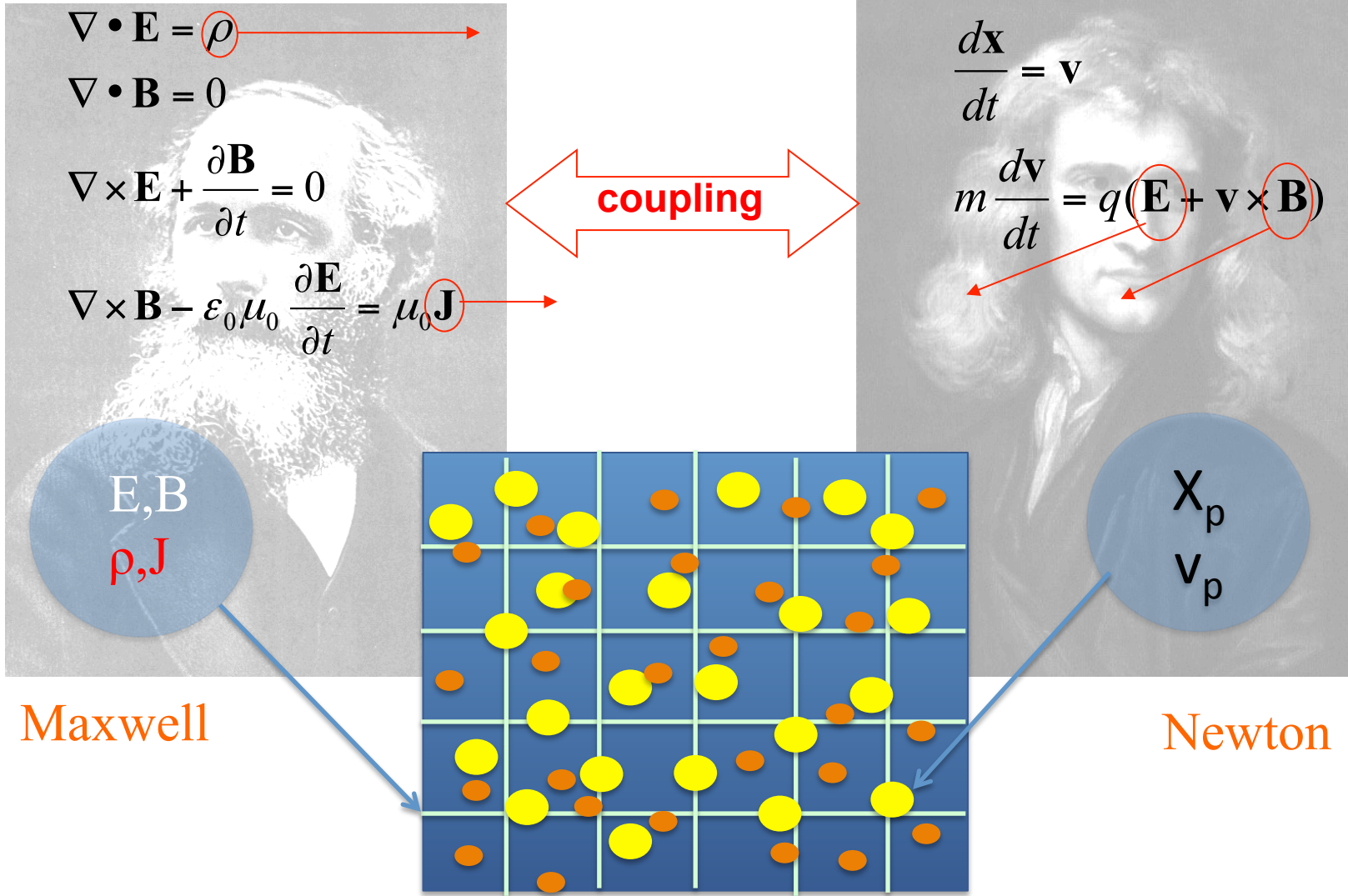
$$m_s n_s \frac{D\mathbf{v}_s}{Dt} = -\nabla \cdot P_s + q_s n_s (\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

n, T, v
E, B

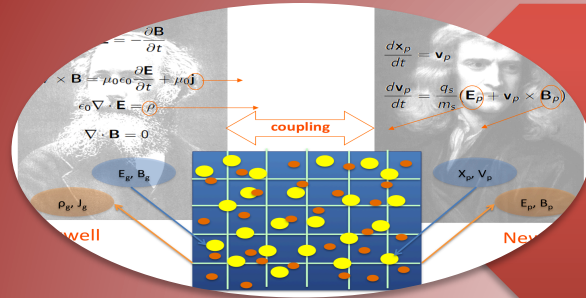
Fluid Equations



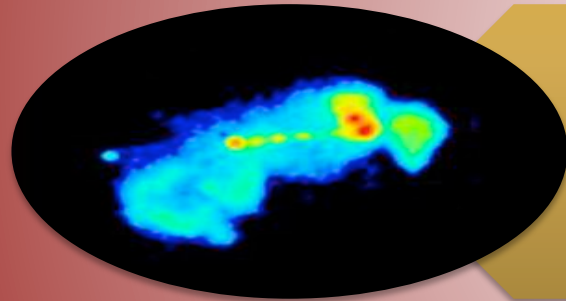
PIC Kinetic model



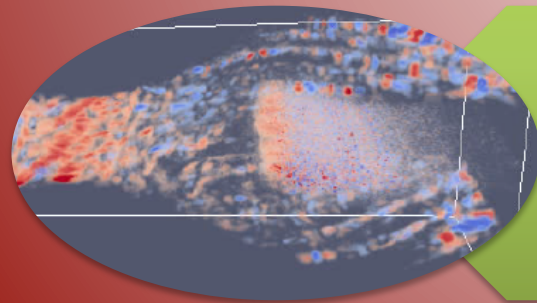
Lecture Structure



Derivation of the basic PIC algorithm

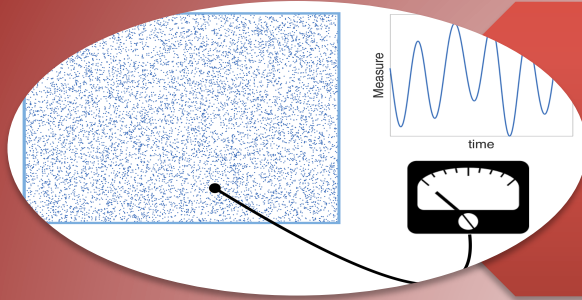


PIC in astrophysical problems



Stability and Multiple scale problems

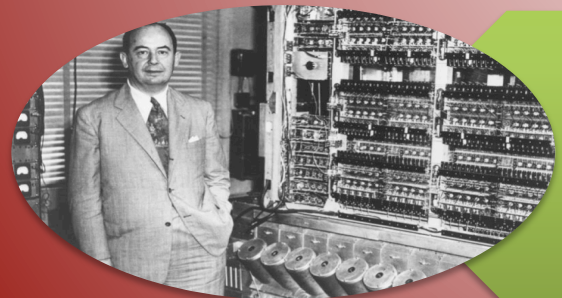
Summary of Lecture 1: Derivation of the basic PIC algorithm



Physical Heuristic
derivation

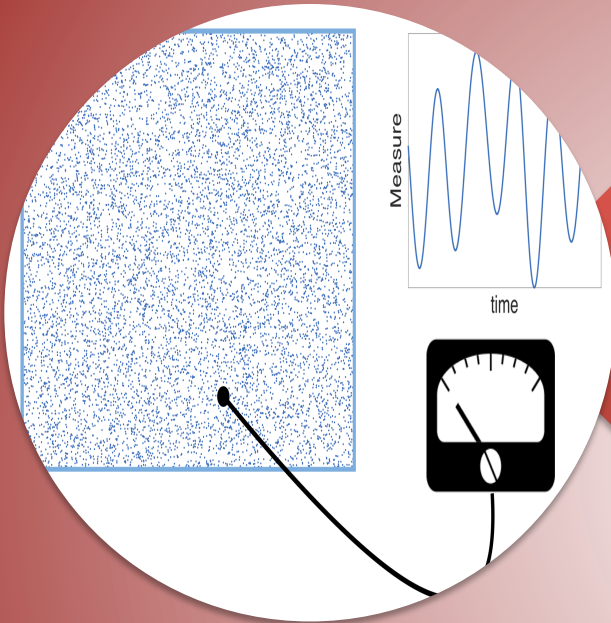


Mathematical
Derivation



Code Skeleton
(Python)

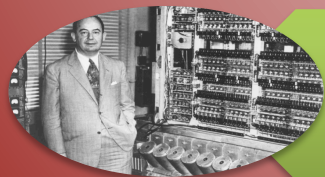
Summary of Lecture 1: Derivation of the basic PIC algorithm



Physical Heuristic derivation

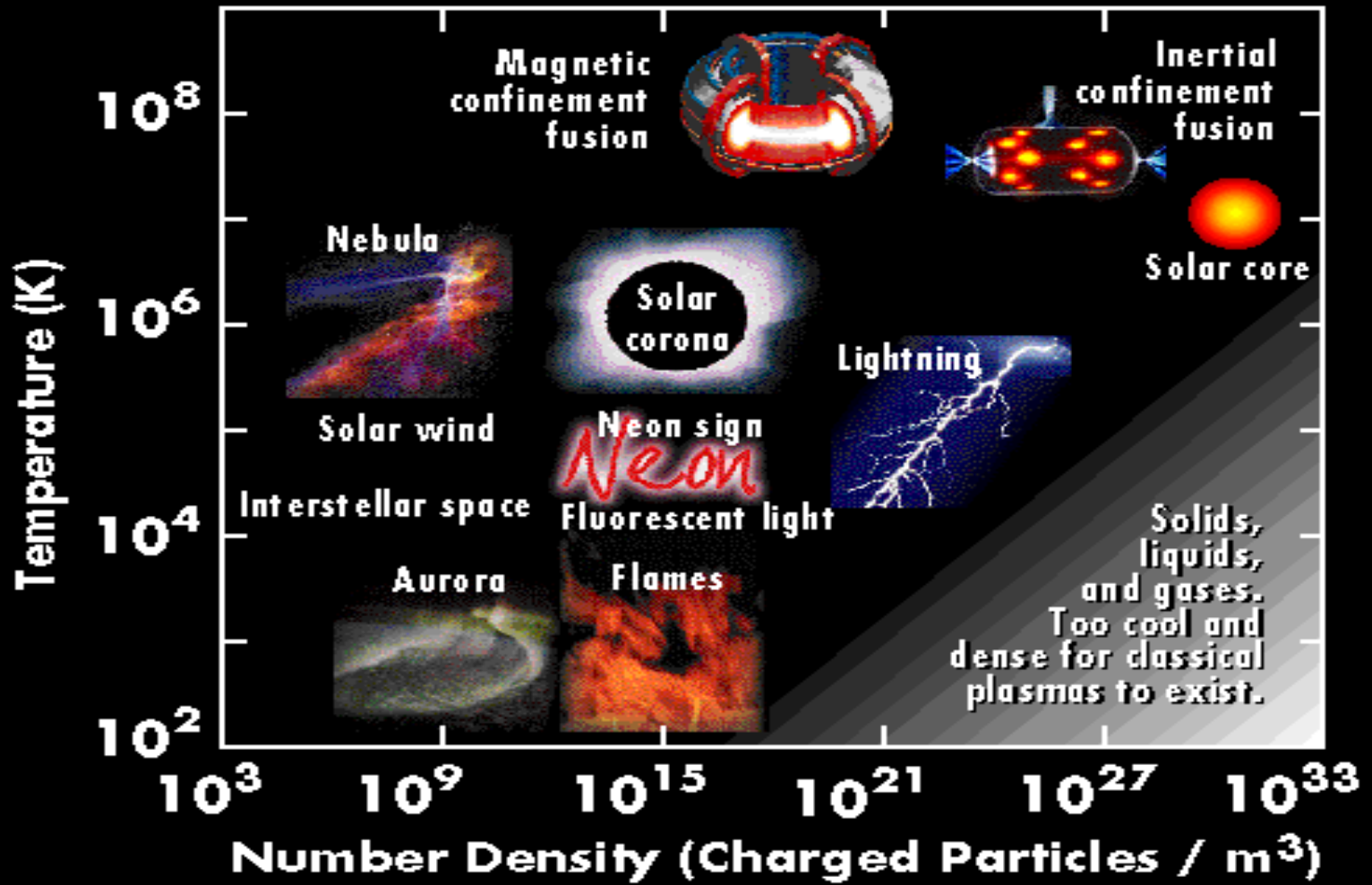


Mathematical Derivation



Code Skeleton (Python)

Typical Plasmas



Copyright 1996 Contemporary Physics Education Project.
Images courtesy of DOE fusion labs, NASA, and Steve Albers.

Examples in nature

APPROXIMATE MAGNITUDES IN SOME TYPICAL PLASMAS

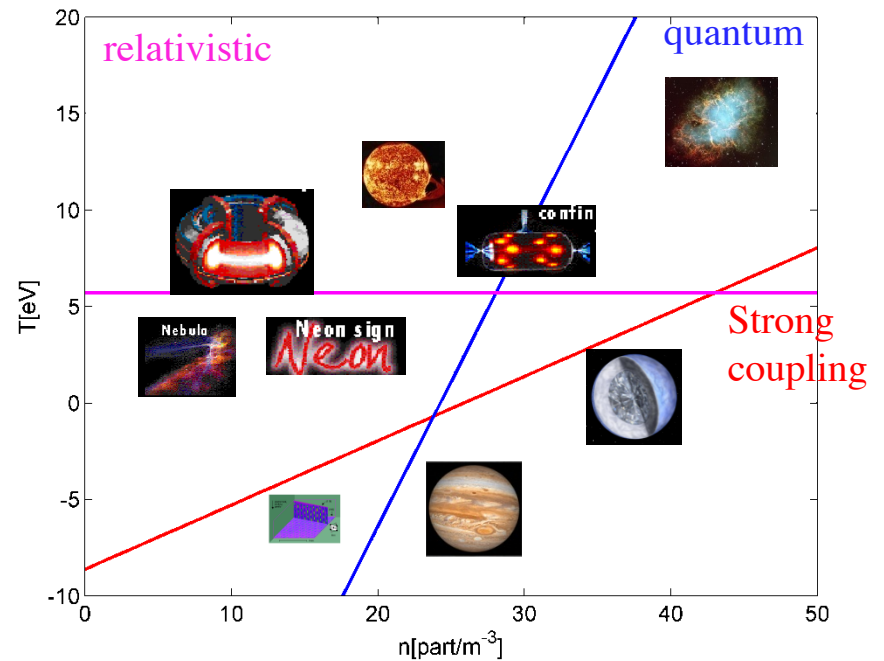
| Plasma Type | $n \text{ cm}^{-3}$ | $T \text{ eV}$ | $\omega_{pe} \text{ sec}^{-1}$ | $\lambda_D \text{ cm}$ | $n\lambda_D^3$ | $\nu_{ei} \text{ sec}^{-1}$ |
|------------------------------------|---------------------|----------------|--------------------------------|------------------------|-----------------|-----------------------------|
| Interstellar gas | 1 | 1 | 6×10^4 | 7×10^2 | 4×10^8 | 7×10^{-5} |
| Gaseous nebula | 10^3 | 1 | 2×10^6 | 20 | 8×10^6 | 6×10^{-2} |
| Solar Corona | 10^9 | 10^2 | 2×10^9 | 2×10^{-1} | 8×10^6 | 60 |
| Diffuse hot plasma | 10^{12} | 10^2 | 6×10^{10} | 7×10^{-3} | 4×10^5 | 40 |
| Solar atmosphere, gas discharge | 10^{14} | 1 | 6×10^{11} | 7×10^{-5} | 40 | 2×10^9 |
| Warm plasma | 10^{14} | 10 | 6×10^{11} | 2×10^{-4} | 8×10^2 | 10^7 |
| Hot plasma | 10^{14} | 10^2 | 6×10^{11} | 7×10^{-4} | 4×10^4 | 4×10^6 |
| Thermonuclear plasma | 10^{15} | 10^4 | 2×10^{12} | 2×10^{-3} | 8×10^6 | 5×10^4 |
| Theta pinch | 10^{16} | 10^2 | 6×10^{12} | 7×10^{-5} | 4×10^3 | 3×10^8 |
| Dense hot plasma | 10^{18} | 10^2 | 6×10^{13} | 7×10^{-6} | 4×10^2 | 2×10^{10} |
| Laser Plasma | 10^{20} | 10^2 | 6×10^{14} | 7×10^{-7} | 40 | 2×10^{12} |

Coupling parameter Γ

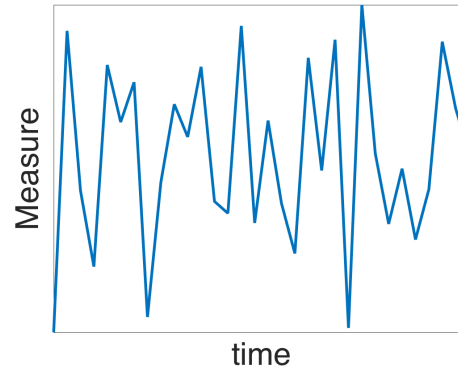
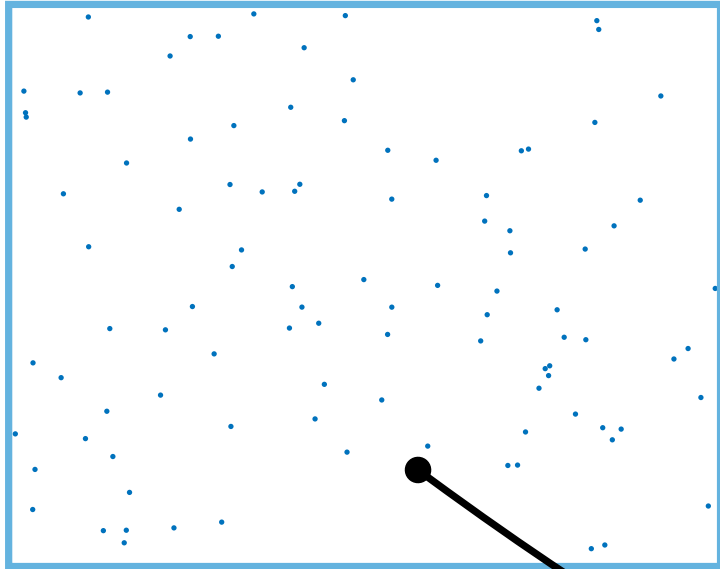
- Plasma Coupling parameter

$$\Gamma \propto n \lambda_D^3$$

- Quantum effects
- Relativistic effects



Strongly Coupled Systems: a thought experiment



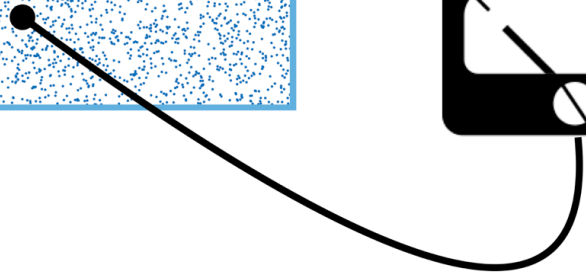
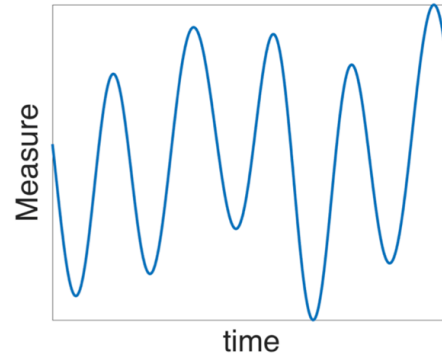
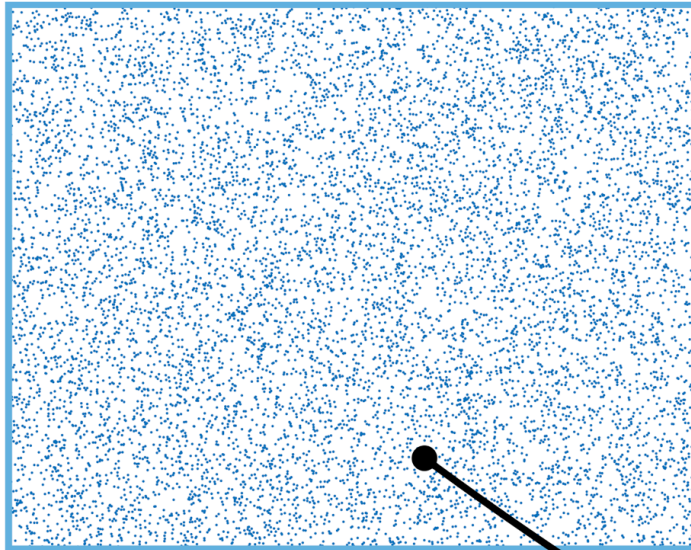
Motion of a particle determined by the near neighbor interactions

Few particle interactions

Mean field approximation not valid

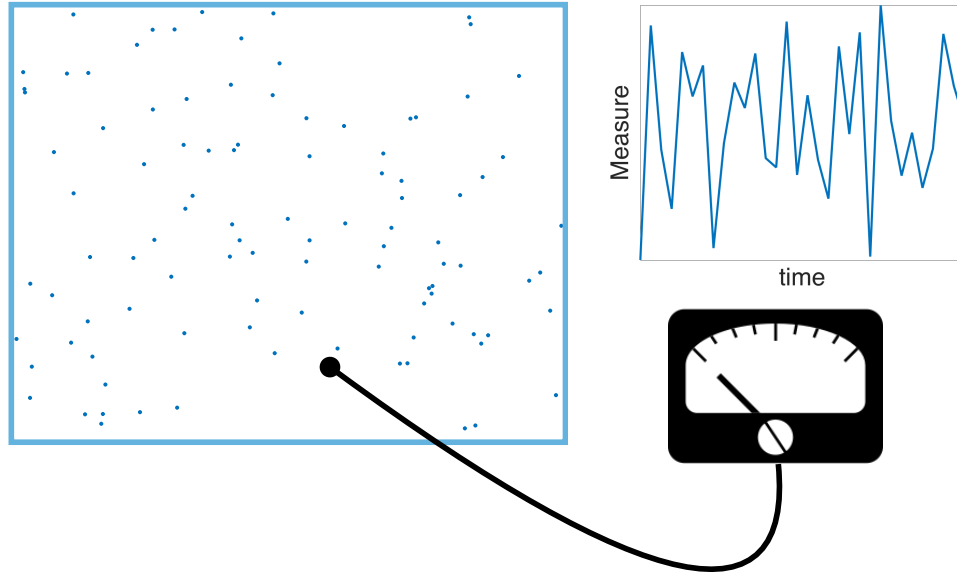
Boltzmann and Klimontovich equation

Weakly Coupled Systems: a thought experiment

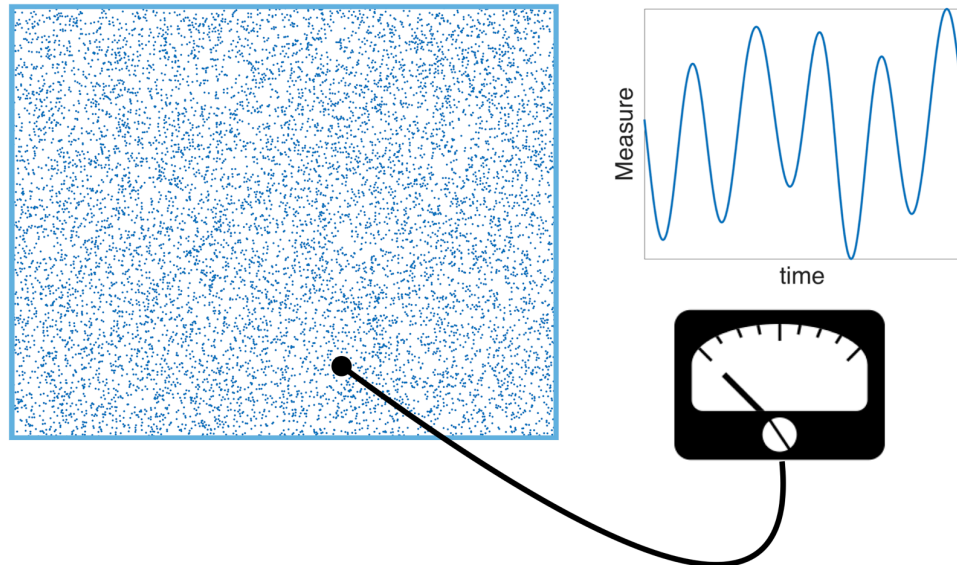


- Motion of a particle determined by the collective behavior of many others
- Many particle interactions
- Mean field approximation
- Vlasov equation

Kinetic simulation models

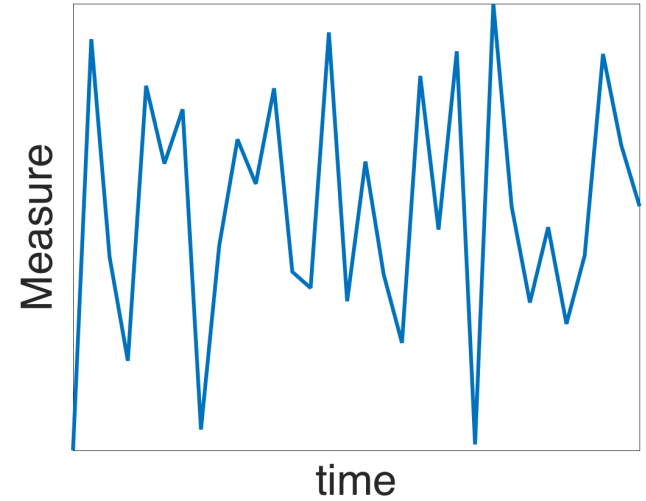
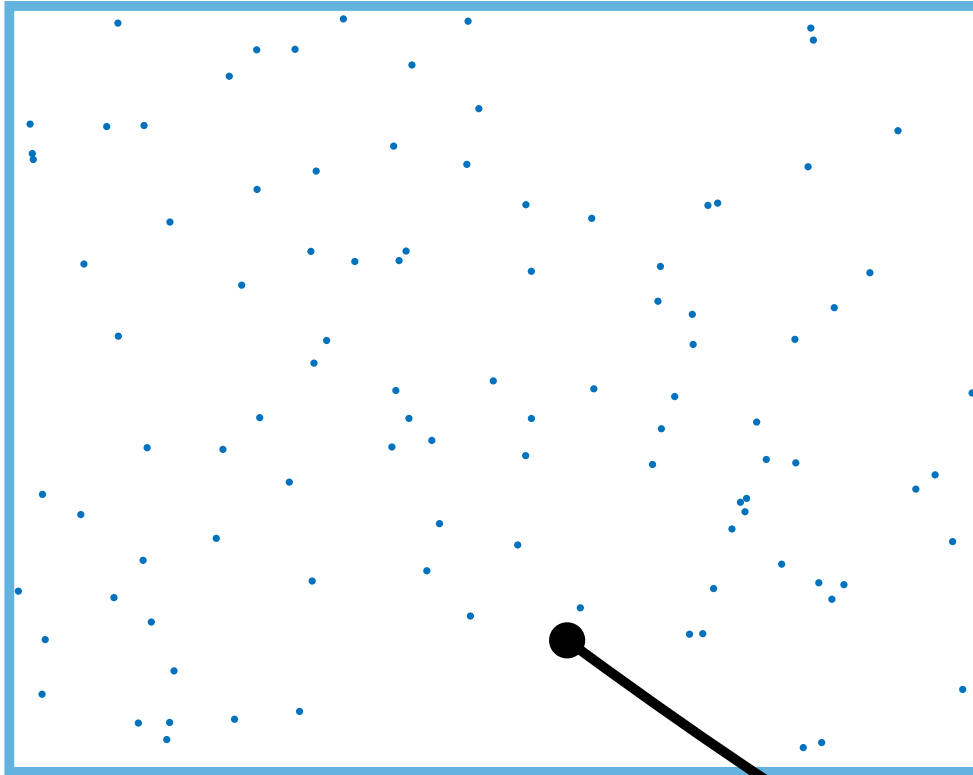


Direct simulation:
Particle-Particle



Indirect simulation:
Particle-Mesh

Particle – Particle (PP) Molecular Dynamics (MD)



Numerical modeling: Particle-Particle or N-body approach

- System of N particles, each interacting with each other. To study the evolution, discretize the time into small steps with interval δt
- Advance particles

$$\begin{aligned}\mathbf{x}_p^{new} &= \mathbf{x}_p^{old} + \delta t \mathbf{v}_p^{old}, \\ \mathbf{v}_p^{new} &= \mathbf{v}_p^{old} + \delta t \mathbf{F}/m,\end{aligned}$$

where \mathbf{F} is summed over all particles

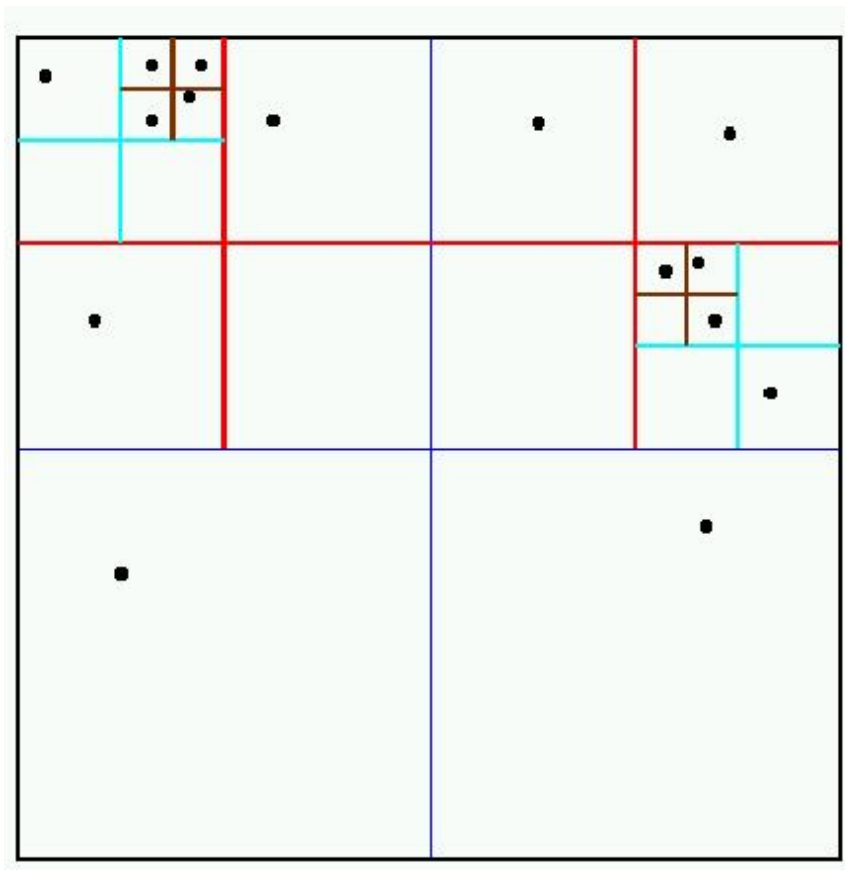
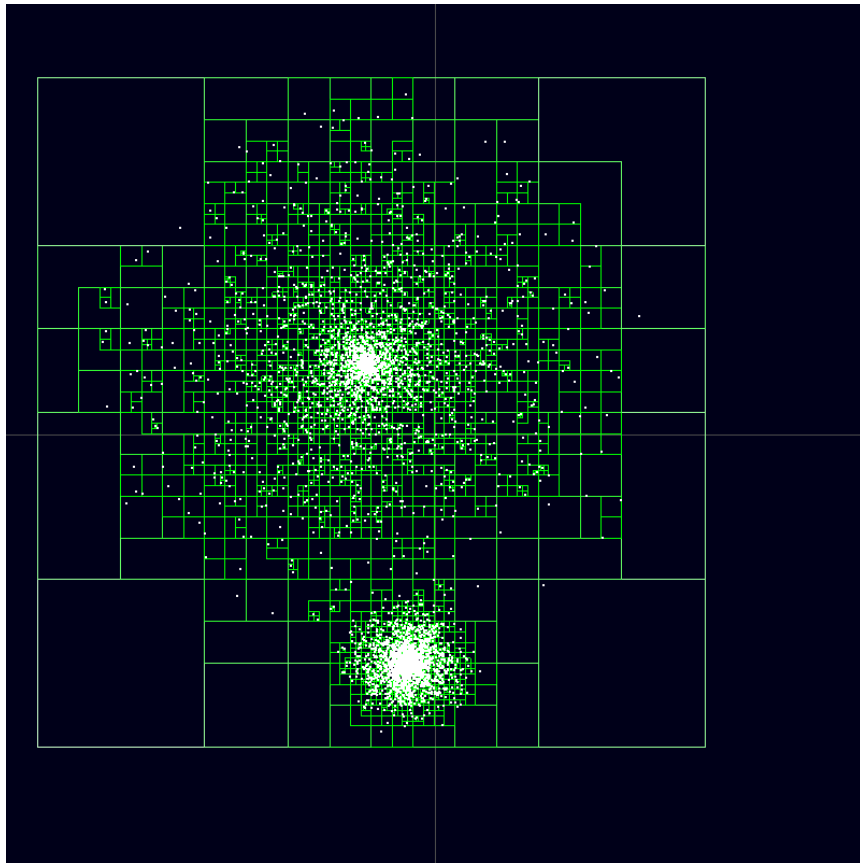
$$\mathbf{F}_p = \sum_{p'} \mathbf{F}_{pp'}.$$

- Gravitational electrostatic, etc., forces. For example, electrostatic:

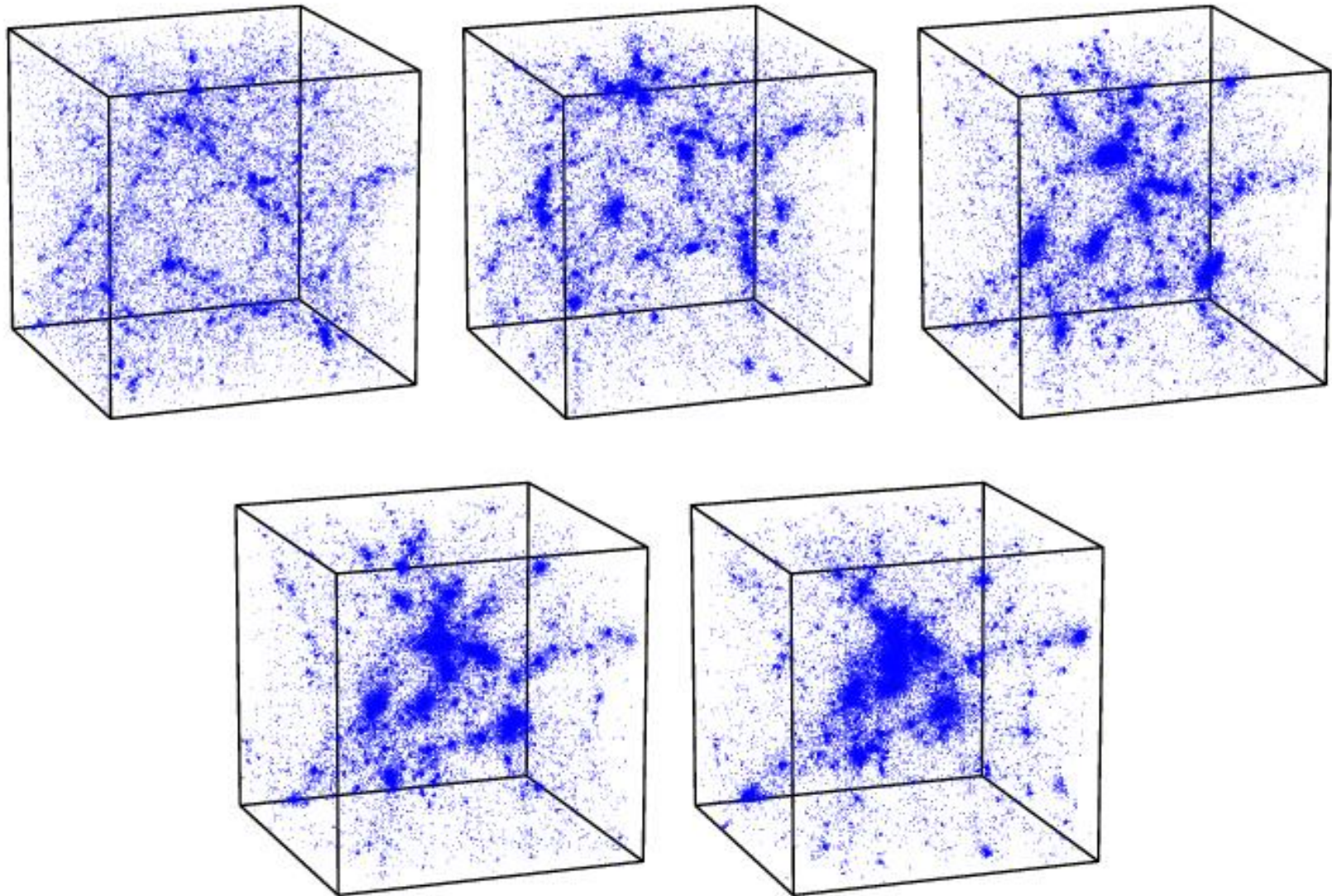
$$\mathbf{F}_{pp'} = \frac{q_p q_{p'}}{4\pi\epsilon_0 |\mathbf{x}_p - \mathbf{x}_{p'}|^2} \cdot \frac{\mathbf{x}_p - \mathbf{x}_{p'}}{|\mathbf{x}_p - \mathbf{x}_{p'}|}.$$

- This cycle repeats until needed.
- The number of force computations at each step is $N(N - 1)/2$.
- Treecode algorithm can reduce to $O(N \log N)$.
- Feasible where plasma approximation is not valid: galaxy mergers, evolution of early Universe, molecular dynamics, etc..

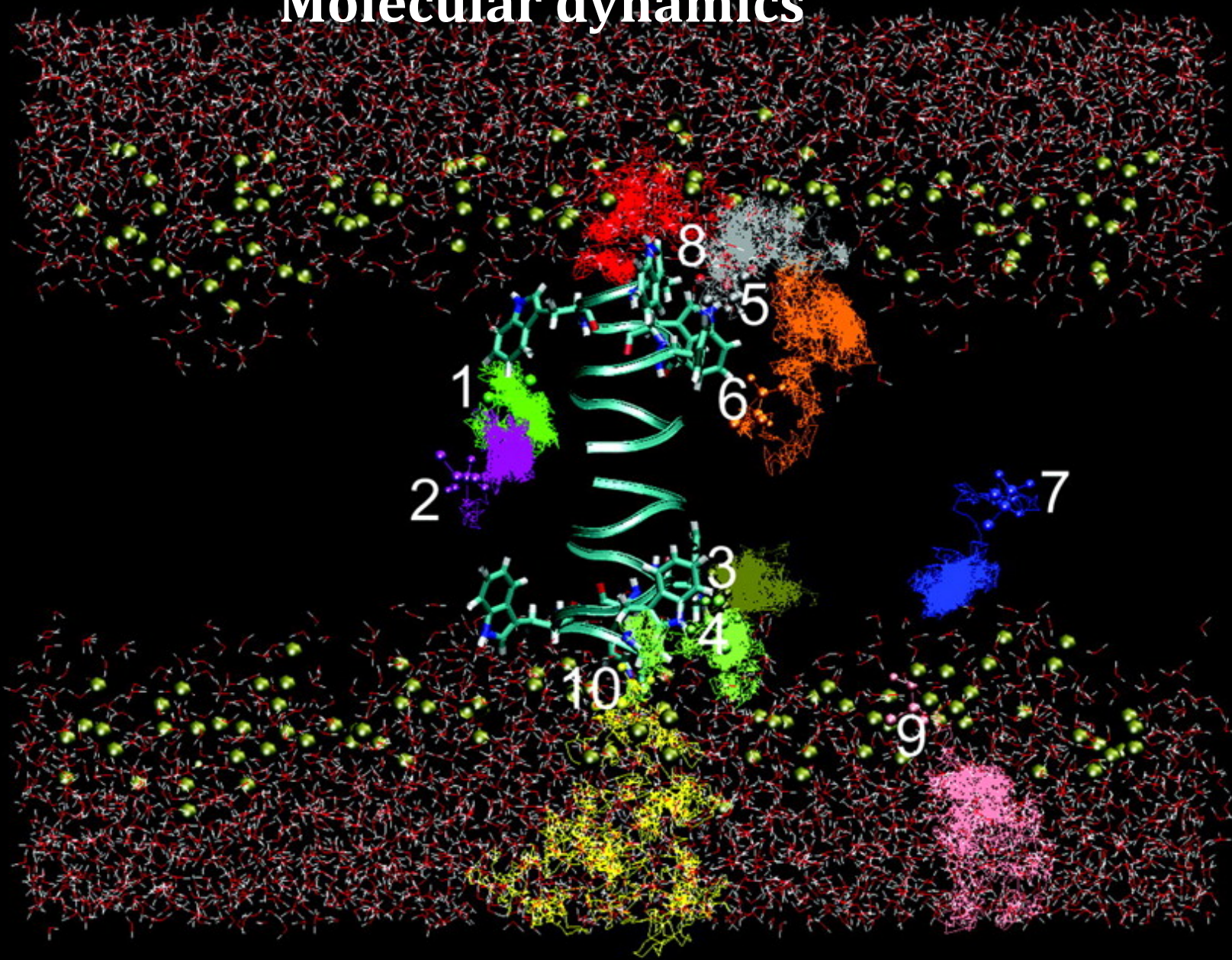
Barnes-Hut Hierarchical Force-calculation Algorithm



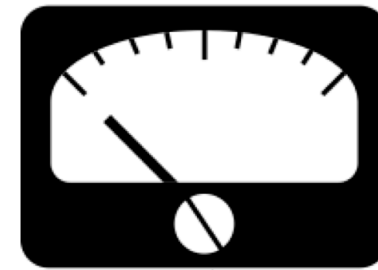
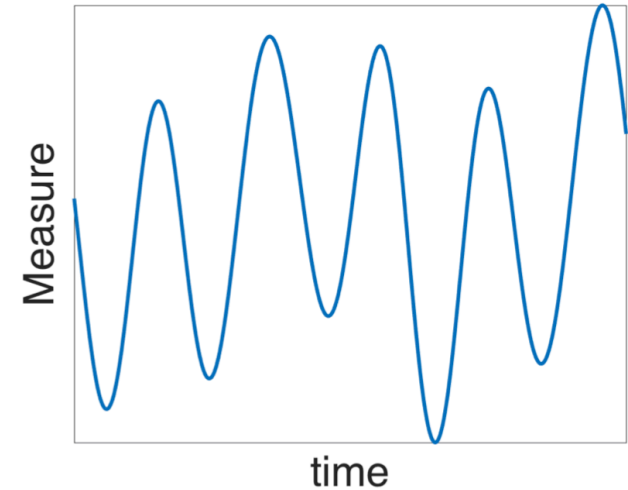
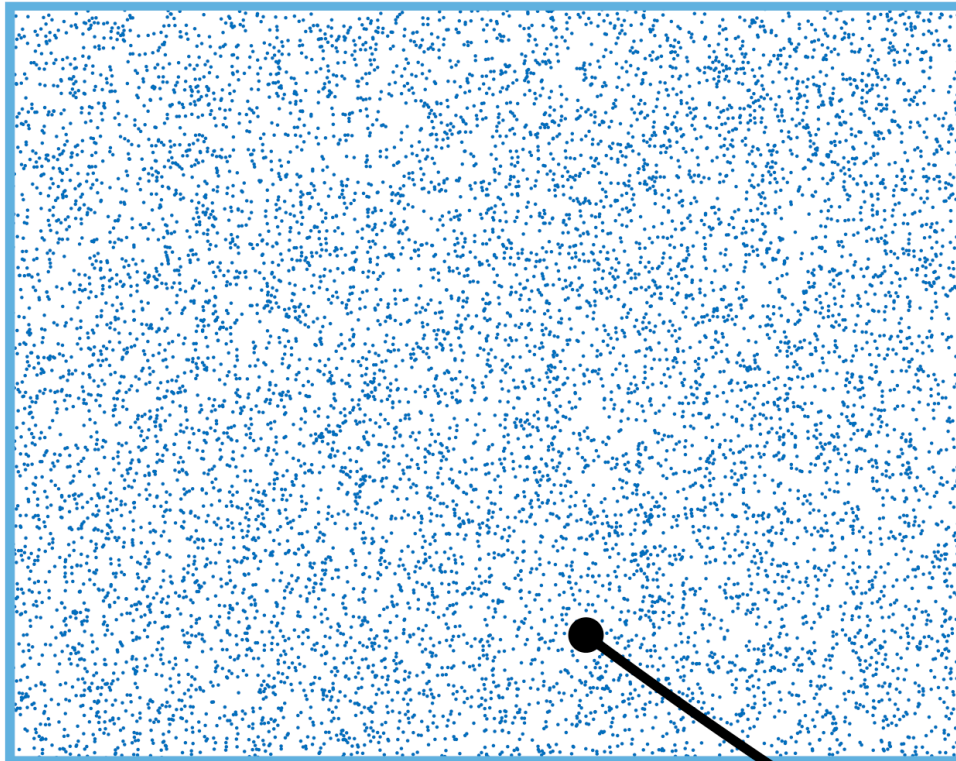
Collapsing void from N-body simulation



Molecular dynamics



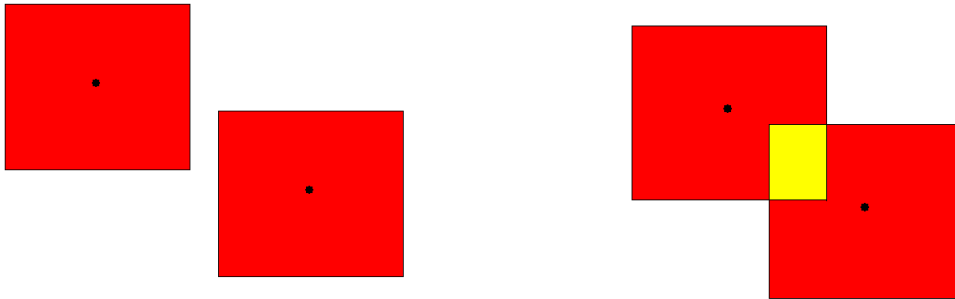
Particle – Mesh (PM) Particle In Cell (PIC)



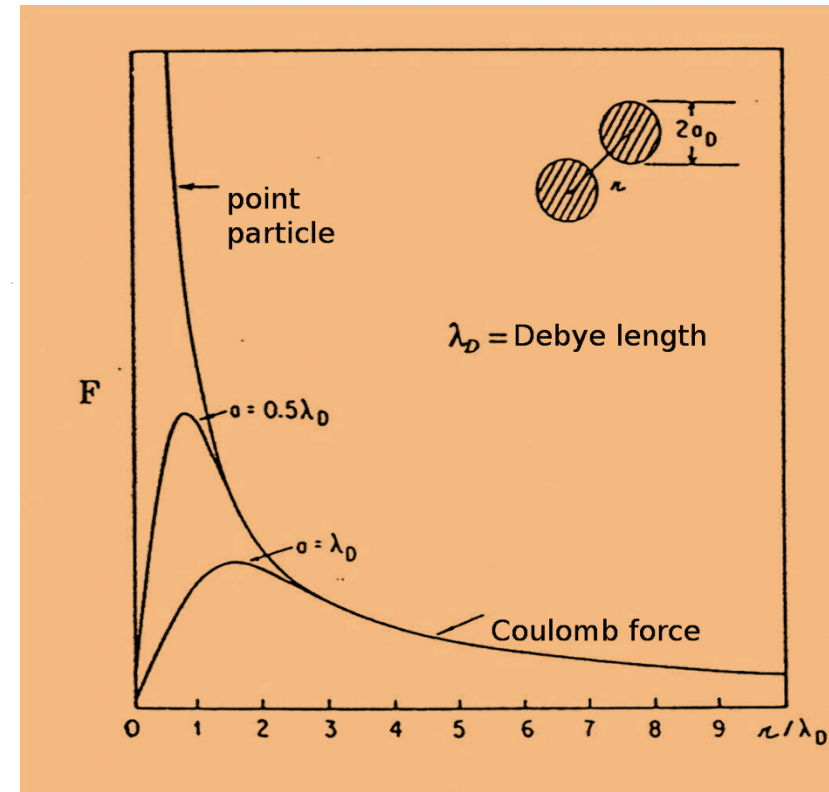
How can this behaviour be represented with a manageable number of computational agents?

Idea of the PIC method: Finite-sized particles

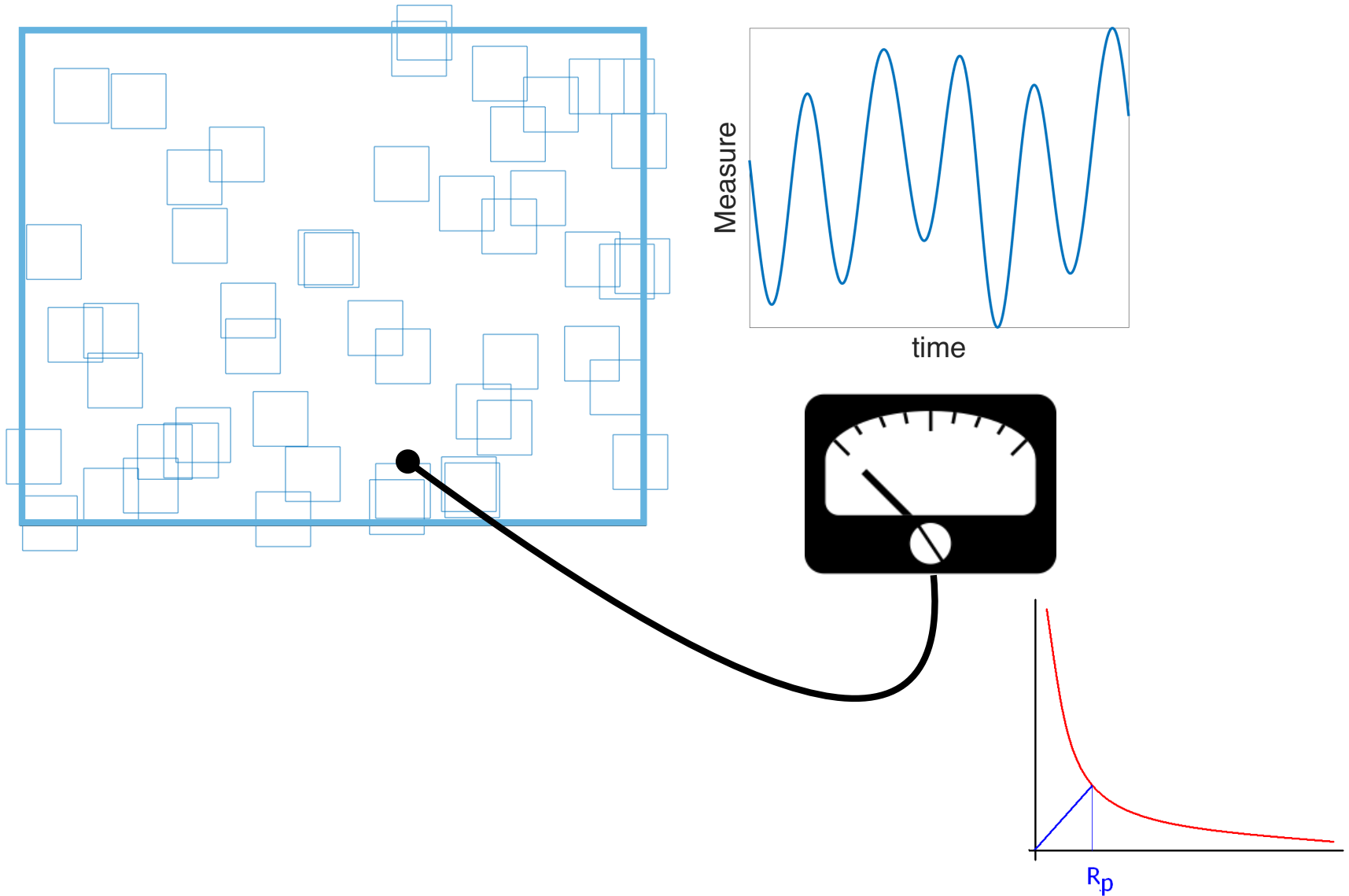
Two finite-sized particles moving closer



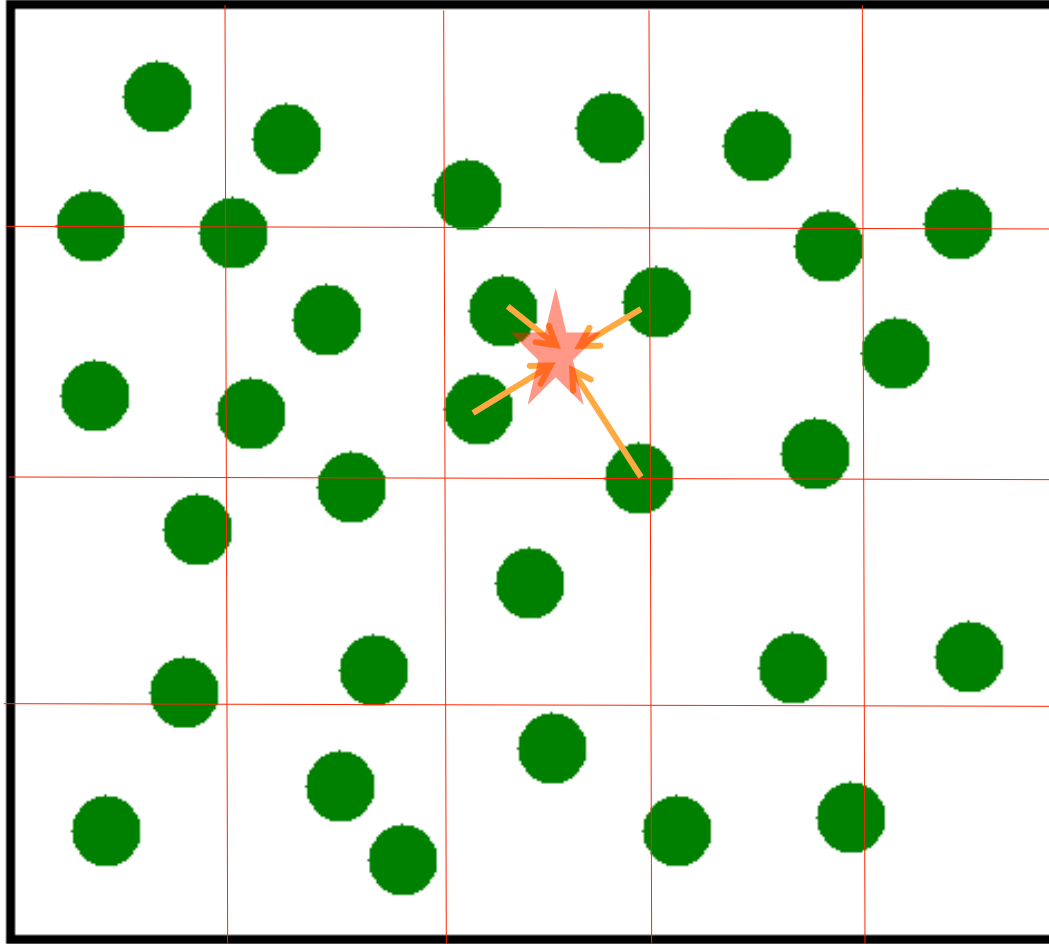
- Finite-size particles
- Far: behave like point particles
- Close (overlap): the area overlapped does not contribute to the force
- At short range the interaction is much weaker



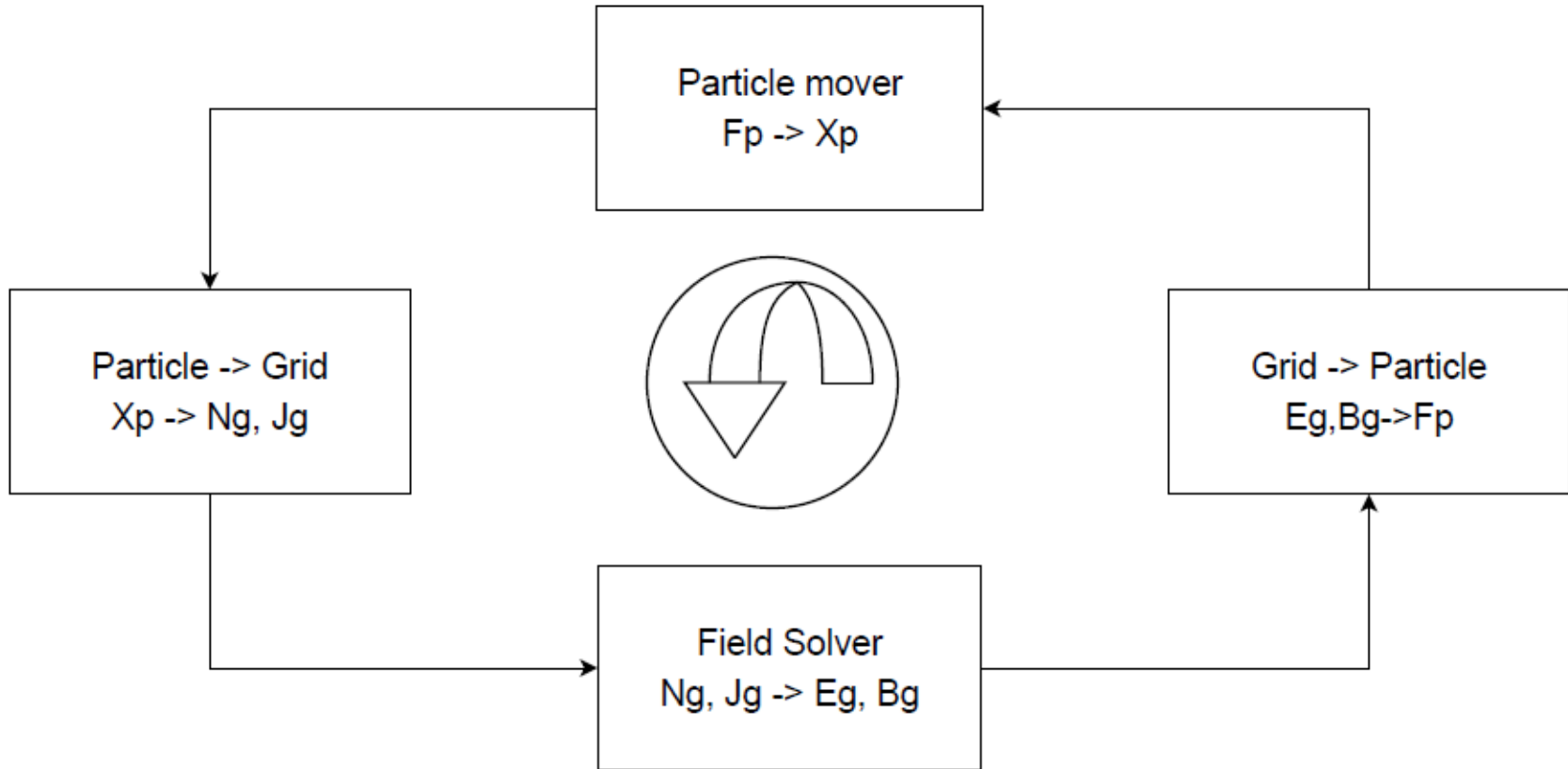
A ensemble of finite-sized particles



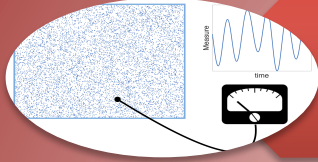
Particle In Cell approach



Using a mesh



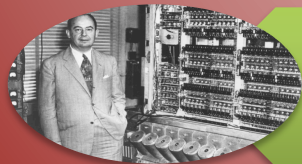
Summary of Lecture 1: Derivation of the basic PIC algorithm



Physical Heuristic derivation



Mathematical
Derivation



Code Skeleton (Python)

Derivation of the Particle-in-Cell (PIC) method

Example of 1D electrostatic

Vlasov equation

$$\frac{\partial f_s}{\partial t} + v \frac{\partial f_s}{\partial x} + \frac{q_s E}{m_s} \frac{\partial f_s}{\partial v} = 0$$

Electric field is given by the Poisson's equation

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

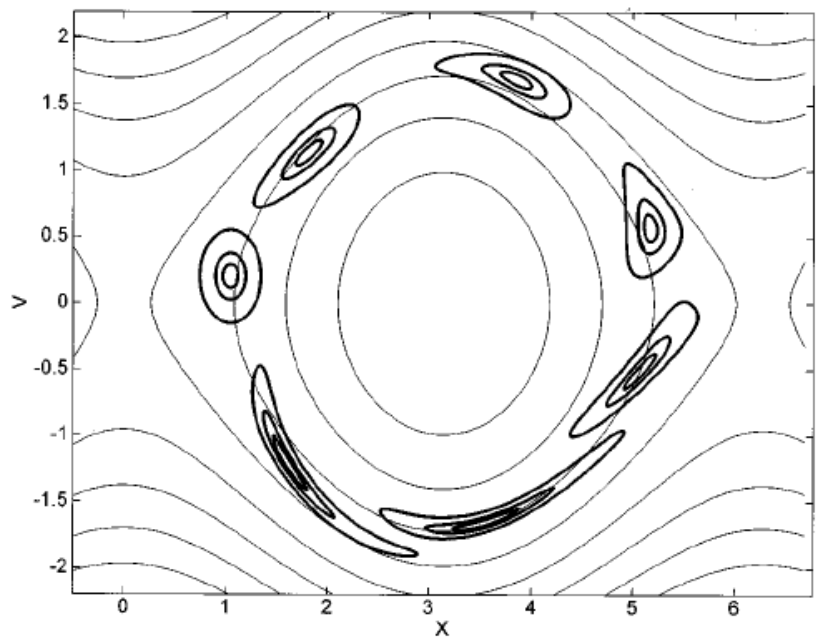
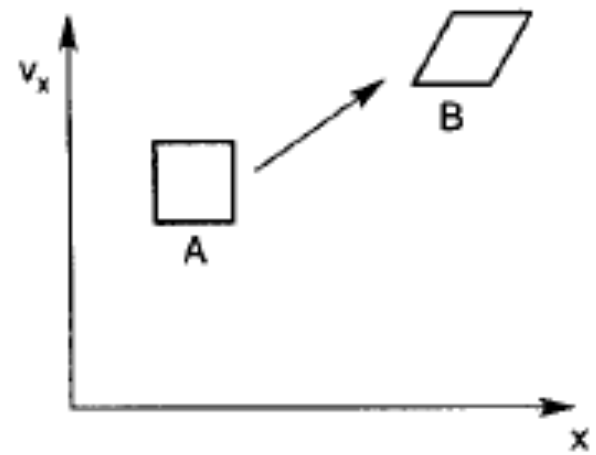
The net charge density is computed from the distribution function

$$\rho(x, t) = \sum_s q_s \int f_s(x, v, t) dv$$

Phase space



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



Derivation of PIC: numerical representation

- The PIC method can be regarded as a **finite element approach but with finite elements that are themselves moving and overlapping**. Assume that the distribution function of each species is given by the superposition of several elements (called computational particles or superparticles):

$$f_s(x, v, t) = \sum_p f_p(x, v, t).$$

- Assign to each computational particle a specific functional form for its distribution, with a number of free parameters whose time evolution will determine the numerical solution of the Vlasov equation. Usually it is the tensor product of the shape in each direction of the phase space:

$$f_p(x, v, t) = N_p S_x(x - x_p(t)) S_v(v - v(t)),$$

where S_x and S_v are the **shape functions**.

- By definition,

$$\int_{-\infty}^{\infty} S_{\xi}(\xi - \xi_p) d\xi = 1.$$

Properties of the Shape functions

A number of properties of the shape functions come from their definition:

1. The support of the shape functions is compact, to describe a small portion of phase space, (i.e. it is zero outside a small range).
2. Their integral over the domain is unitary:

$$\int_{V_\xi} S_\xi(\xi - \xi_p) d\xi = 1 \quad (1.13)$$

where ξ stands for any coordinate or any velocity direction.

3. While not strictly necessary, Occam's razor suggests to choose symmetric shapes:

$$S_\xi(\xi - \xi_p) = S_\xi(\xi_p - \xi) \quad (1.14)$$

Derivation of PIC: shape function choice

- In the velocity direction: Dirac's delta

$$S_v(v - v_p) = \delta(v - v_p).$$

This way, particles stay close together in phase space for longer time.

- In space: b-splines (basic splines). First b-spline is a top-flat function:

$$b_0(x) = \begin{cases} 1 & \text{if } |x| < 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The subsequent are given by the recursive formula

$$b_{n+1} = \int_{-\infty}^{\infty} dx' b_0(x - x') b_n(x').$$

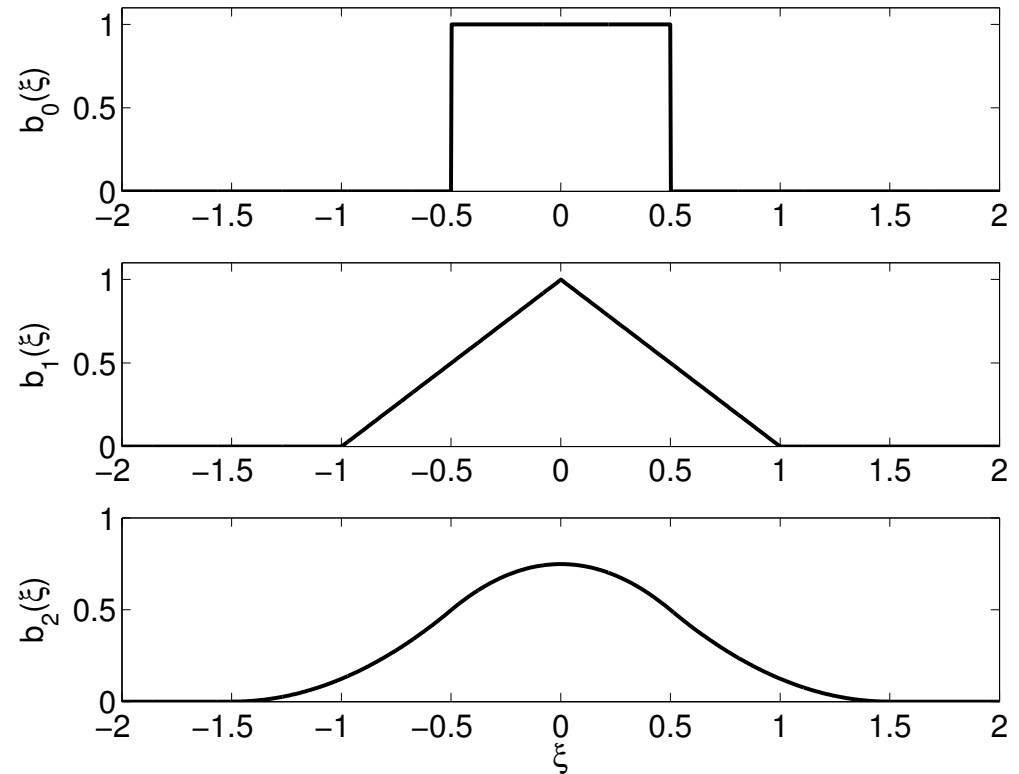
- The spatial shape function for PIC:

$$S_x(x - x_p) = \frac{1}{\Delta_p} b_n\left(\frac{x - x_p}{\Delta_p}\right)$$

B-Spline shape functions

$$b_0(\xi) = \begin{cases} 1 & \text{if } |\xi| < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

$$b_\ell(\xi) = \int_{-\infty}^{\infty} d\xi' b_0(\xi - \xi') b_{\ell-1}(\xi')$$



Crucial properties of the b-splines

- (a) when a b-spline of any order is evaluated on a uniform grid of step 1, the sum over all points of evaluation is unitary, regardless of the central point ξ of the b-spline:

$$\sum_i b_\ell(\xi + i) = 1, \quad (1.18)$$

a property of great convenience in particle interpolation;

- (b) The integral of b-splines of any order is unitary:

$$\int_{-\infty}^{\infty} b_\ell(\xi) d\xi = 1. \quad (1.19)$$

- (c) The Dirac's delta $\delta(\xi)$ can be regarded as $b_{-1}(\xi)$.

PIC: equations of motion 1

$$f_p(x, v, t) = N_p S_x(x - x_p(t)) S_v(v - v_p(t))$$

- We are interested in evolution equations for x_p and v_p .
- By definition, $f_p(x, v, t)$ satisfies Vlasov equation

$$\frac{\partial f_p}{\partial t} + v \frac{\partial f_p}{\partial x} + \frac{q_s E}{m_s} \frac{\partial f_p}{\partial v} = 0$$

It is non-linear because of electric field which depends on $f_s = \sum f_p$.

- The arbitrary functional form chosen for the superparticles (elements) does not satisfy exactly the Vlasov equation. The usual procedure of the finite element method is to require that the moments of the equations be satisfied.

PIC: equations of motion 2

- Use notation $\langle \dots \rangle \equiv \int dx \int dv$
- Moment 0:

$$\frac{\partial \langle f_p \rangle}{\partial t} + \left\langle v \frac{\partial f_p}{\partial x} \right\rangle + \left\langle \frac{q_s E}{m_s} \frac{\partial f_p}{\partial v} \right\rangle = 0$$
$$\frac{dN_p}{dt} = 0$$

- Moment 1_x:

$$\frac{\partial \langle f_p x \rangle}{\partial t} + \left\langle vx \frac{\partial f_p}{\partial x} \right\rangle + \left\langle x \frac{q_s E}{m_s} \frac{\partial f_p}{\partial v} \right\rangle = 0$$
$$\frac{dx_p}{dt} = v_p$$

- Moment 1_v:

$$\frac{\partial \langle f_p v \rangle}{\partial t} + \left\langle v^2 \frac{\partial f_p}{\partial x} \right\rangle + \left\langle v \frac{q_s E}{m_s} \frac{\partial f_p}{\partial v} \right\rangle = 0$$
$$\frac{dv_p}{dt} = \frac{q_s}{m_s} E_p$$

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

PIC: equations of motion 3

- It is a crucial advantage of the PIC method that its evolution equations resemble the same Newton equation as followed by the regular physical particles.

$$\frac{dN_p}{dt} = 0$$

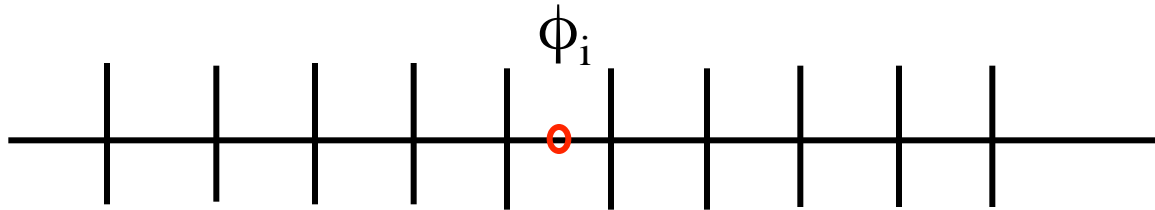
$$\frac{dx_p}{dt} = v_p$$

$$\frac{dv_p}{dt} = \frac{q_s}{m_s} E_p$$

The key difference is that the field is computed as the average over the particles based on the definition of E_p .

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

Field Equations



$$\epsilon_0 \frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta x^2} = -\rho_i \quad (2.24)$$

where the densities ρ_i are similarly defined as average over the cells:

$$\rho_i = \frac{1}{x_{i+1/2} - x_{i-1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} \rho(x) dx \quad (2.25)$$

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \rho(x) dx = \int_{-\infty}^{\infty} b_0 \left(\frac{x - x_i}{\Delta x} \right) \rho(x) dx$$

PIC: field equations 1 (solving Poisson's)

- Discretize electromagnetic field on the rectilinear grid with cell size Δx .
- Simplest finite-difference for Poisson's equation:

$$\epsilon_0 \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} = -\rho_i,$$
$$\rho_i = \frac{1}{x_{i+1/2} - x_{i-1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} \rho(x) dx.$$

- Formulate the density averaged over a cell with the 0-order b-spline

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \rho(x) dx = \int_{-\infty}^{\infty} b_0\left(\frac{x - x_i}{\Delta x}\right) \rho(x) dx$$

- Now, the **interpolation function** is the convolution of the shape function $S_x(x - x_p)$ with the top hat function spanning the cell of the computational grid.

$$W(x_i - x_p) = \int S_x(x - x_p) b_0\left(\frac{x_i - x_i}{\Delta x}\right)$$

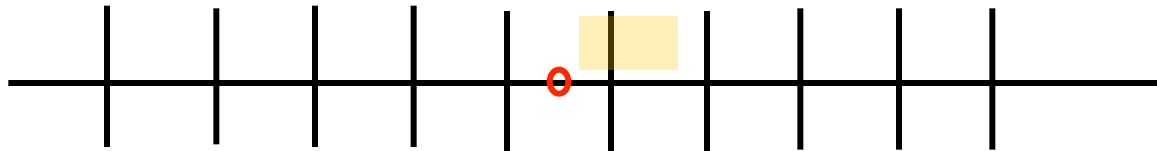
- Finally, average cell charge density is

$$\rho_i = \sum_p \frac{q_p}{\Delta x} W(x_i - x_p), \quad q_p = q_s N_p$$

Interpolation function

$$W(x_i - x_p) = \int S_x(x - x_p) b_0 \left(\frac{x - x_i}{\Delta x} \right) \quad (2.28)$$

$$S_x(x - x_p) = \frac{1}{\Delta_p} b_l \left(\frac{x - x_p}{\Delta_p} \right) \quad (2.11)$$



$$W(x_i - x_p) = b_{l+1} \left(\frac{x_i - x_p}{\Delta_p} \right) \quad (2.30)$$

$$\rho_i = \sum_p \frac{q_p}{\Delta x} W(x_i - x_p) \quad (2.29)$$

PIC: field equations 2 (computing electric field)

- Solution of Poisson's equation gave us electric potential on the grid cells ϕ_i .
- Central difference for the electric field gives:

$$E_i = -\frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x}.$$

- The continuum electric field is reconstructed using the assumption that the field is constant in each cell and equal to its cell-averaged value

$$E(x) = \sum_i E_i b_0 \left(\frac{x - x_i}{\Delta x} \right).$$

- At a particle:

$$E_p = \sum_i E_i \int b_0 \left(\frac{x - x_i}{\Delta x} \right) S_x(x - x_p).$$

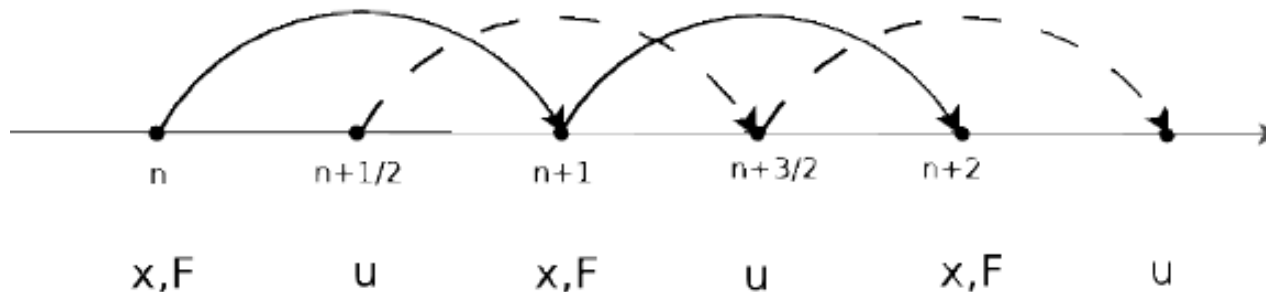
- Using interpolation function, finally,

$$E_p = \sum_i E_i W(x_i - x_p).$$

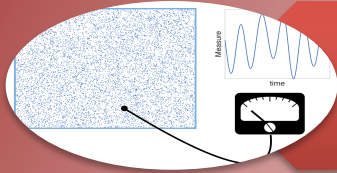
PIC: moving particles

- The equations of motion of superparticles are simple ordinary differential equations with the same form as the regular Newton equations.
- Given the very large number of particles used (billions are now common in published works), the use of complex schemes may result in prohibitively long simulations.
- A simple and widely used is the leap-frog algorithm based on staggering velocity and coordinate computation by a half time step:

$$\begin{aligned}x_p^{n+1} &= x_p^n + \Delta t v_p^{n+1/2} \\v_p^{n+3/2} &= v_p^{n+1/2} + \Delta t \frac{q_s}{m_s} E_p(x_p^{n+1}) \\v_p^{1/2} &= v_p^0 + \Delta t \frac{q_s}{m_s} E_p(x_p^0)\end{aligned}$$



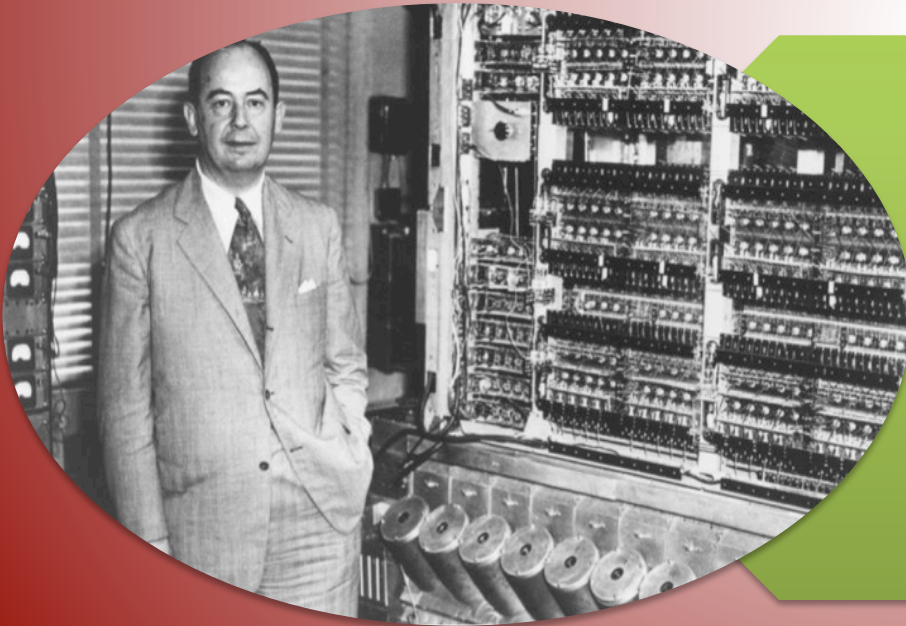
Summary of Lecture 1: Derivation of the basic PIC algorithm



Physical Heuristic derivation

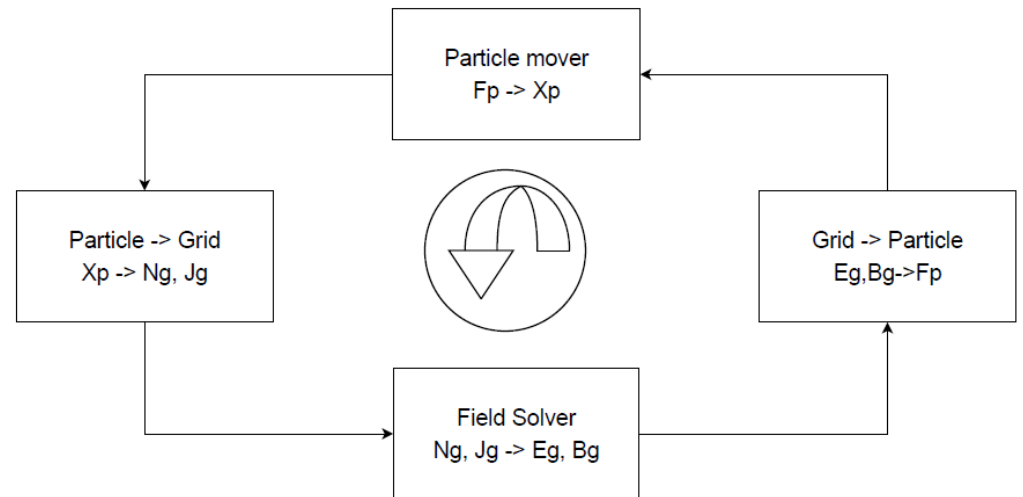
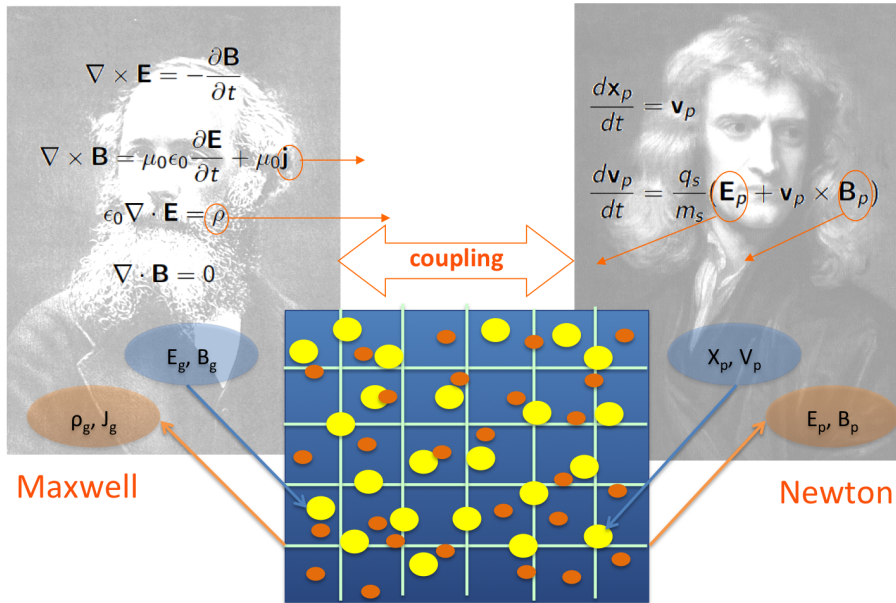


Mathematical Derivation

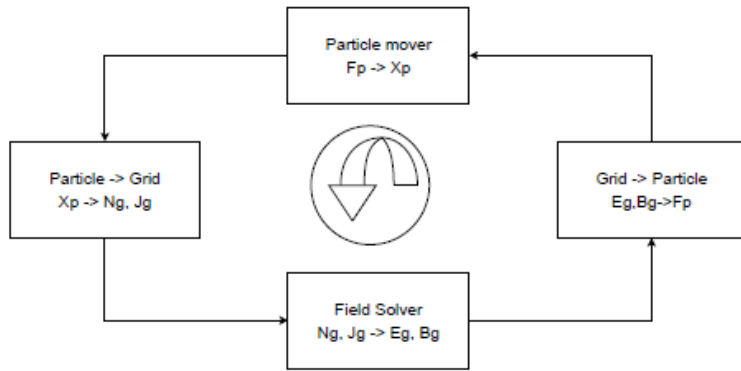


Code Skeleton
(Python)

Developing a simple PIC code



Summary of the PIC method



Algorithm of the PIC method, electrostatic case in 1D

- (i) The plasma is described by a number of computational particles having position x_p , velocity v_p and each representing a fixed number N_p of physical particles.

- (ii) The equations of motion for the particles are advanced by one time step using,

$$x_p^{n+1} = x_p^n + \Delta t v_p^{n+1/2}$$

$$v_p^{n+3/2} = v_p^{n+1/2} + \Delta t \frac{q_s}{m_s} E_p^{n+1}$$

using the particle electric field from the previous time step.

- (iii) The charge densities are computed in each cell using:

$$\rho_i = \sum_p \frac{q_p}{\Delta x} W(x_i - x_p)$$

- (iv) The Poisson equation is solved:

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

and the electric field E_i in each cell is computed:

$$E_i = -\frac{\varphi_{i+1} - \varphi_{i-1}}{2\Delta x}$$

- (v) From the field known in the cells, the field acting on the particles is computed as

$$E_p^{n+1} = \sum_i E_i W(x_i - x_p^{n+1})$$

which is used in the next cycle

- (vi) The cycle restarts.

PIC Code in Python

```
# Move particles
xp += vp * DT
# Enforce periodicity
xp[np.where(xp < 0)] += L
xp[np.where(xp >= L)] -= L

# Project particles->grid
g1 = np.floor(xp/dx - 0.5)
g = np.concatenate((g1, g1+1))
fraz1 = 1 - np.abs(xp/dx - g1 - 0.5)
fraz = np.concatenate((fraz1, 1-fraz1))
g[np.where(g < 0)] += NG
g[np.where(g > NG-1)] -= NG

mat = sparse.csc_matrix((fraz, (p, g)), shape=(N, NG))
rho = Q / dx * mat.toarray().sum(axis=0) + rho_back

# Compute electric field potential
Phi = linalg.spsolve(Poisson, -dx**2 * rho[0:NG-1])
Phi = np.concatenate((Phi, [0]))
# Electric field on the grid
Eg = (np.roll(Phi, 1) - np.roll(Phi, -1)) / (2*dx)

# Project q->p
pp += mat*QM*Eg*DT
gamma = (1. + pp**2)**0.5
# update velocities
vp = pp / gamma
```

Loading the right tools (Anaconda is a good distribution)

As a first step, some python initialisations are needed.

```
import numpy as np
import pylab as plt
from scipy import sparse
from scipy.sparse import linalg
```

We then need to initialise the definition of the simulation, in terms of domain size and of the grid used to discretise it. Next, the time step is set with the number of cycles to be run. The next step is to define the plasma density. We use normalised units where the plasma frequency is set to be unitary. But physical units can be used.

The ion uniform background is set to exactly balance the charge of the particles (that are all electrons in this simple model).

```
# Simulation parameters
L = 20*np.pi #20*np.pi # Domain size

NG = 80 # Number of grid cells
N = NG * 200 # Number of particles (200 per cell for example)
dx = L / NG # Cell size

DT = 0.005 # Time step
NT = 50000 # Number of time steps

WP = 1. # Plasma frequency
QM = -1. # Charge/mass ratio
Q = WP**2 / (QM*N/L) # rho0*L/N: charge carried by a single particle
rho_back = -Q*N/L # Background charge density
```


Particle initial loading

We then initialise the electrons. We choose a classic problem of two-stream instability [24] where the initial electrons are subdivided into two equal beams of opposite mean velocity but equal density and thermal speed.

```
#Particle initial properties
V0 = 0.9 # Stream velocity
VT = 0.0000001 # Thermal speed

# perturbation
XP1 = 1.0
mode = 1

# particles (electrons)
xp = np.linspace(0, L-L/N, N).T # Particle positions
vp = VT * np.random.randn(N) # Particle momentum, initially Maxwellian
pm = np.arange(N)
pm = 1 - 2 * np.mod(pm+1, 2) # Even and odd particles have opposite speed
vp += pm * V0 # Momentum + stream velocity
np.random.shuffle(v) # We reshuffle the indices to avoid any bias

# Add electron perturbation to excite the desired mode
xp += XP1 * (L/N) * np.sin(2 * np.pi * xp / L * mode)
xp[np.where(xp < 0)] += L
xp[np.where(xp >= L)] -= L
```

Example: Two Stream Instability

- Two counter-streaming beams of particles:

$$f(v) = \frac{n_0}{2} \delta(v - v_0) + \frac{n_0}{2} \delta(v + v_0).$$

- The dispersion relation

$$1 - \frac{\omega_p^2}{2} \left(\frac{1}{(\omega - kv_0)^2} + \frac{1}{(\omega + kv_0)^2} \right) = 0.$$

- In relativistic case

$$1 - \frac{\omega_p^2}{2\gamma_0} \left(\frac{1}{(\gamma_0\omega - kv_0\gamma_0)^2} + \frac{1}{(\gamma_0\omega + kv_0\gamma_0)^2} \right) = 0,$$

$$\gamma_0 = \sqrt{1 - \frac{v_0^2}{c^2}}.$$

In the 1D electrostatic limit, the Maxwell equations reduce to just the Poisson equation:

$$-\epsilon_0 \nabla^2 \phi = en_i - en_e \quad (1.67)$$

where n_i is the uniform background ion density and n_e is the electron density projected to the grid from the particles. The ∇^2 operator is discretised as in the simplest textbook finite difference method [53]:

$$-\epsilon_0(\phi_{i+1} + \phi_{i-1} - 2\phi_i) = (en_i - en_e)\Delta x^2 \quad (1.68)$$

The potential is computed in the cell centres. The continuum is subdivided in cells indexed by i with $i \in [0, \text{NG} - 1]$ (note than python as C and C++ counts the indices in vectors from 0, in MATLAB the same index range would be 1 to NG).

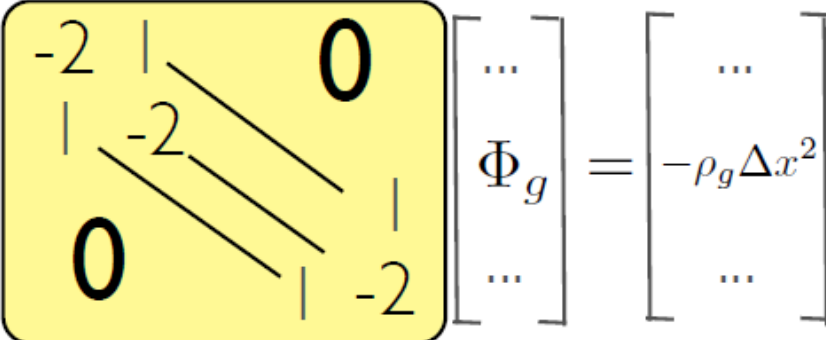
```
# Auxiliary vectors
p = np.concatenate([np.arange(N), np.arange(N)]) # Particle indices up to N
Poisson = sparse.spdiags(([1, -2, 1] * np.ones((1, NG-1), dtype=int)).T, \
                        [-1, 0, 1], NG-1, NG-1)
Poisson = Poisson.tocsr()
```

Poisson solver

The Poisson equation can be discretized in 1D using the classic three-points formula:

$$\frac{\Phi_{g+1} - 2\Phi_g + \Phi_{g-1}}{\Delta x^2} = -\rho_g$$

When casted to matrix form:
(tridiagonal matrix)


$$\begin{bmatrix} -2 & 1 & & 0 \\ & 1 & -2 & \\ & & 1 & -2 \\ 0 & & & 1 & -2 \end{bmatrix} \begin{bmatrix} \dots \\ \Phi_g \\ \dots \end{bmatrix} = \begin{bmatrix} \dots \\ -\rho_g \Delta x^2 \\ \dots \end{bmatrix}$$

Once the Potential is known, the electric field(- gradient of Phi) can be calculated by central difference:

$$E_g = -\frac{\Phi_{g+1} - \Phi_{g-1}}{2\Delta x}$$

MATRIX Poisson:

$$Poisson(g1, g2) = \begin{pmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & 1 & -2 \end{pmatrix}$$

$$\epsilon_0 \frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta x^2} = \rho_i \quad (2.24)$$

Main Cycle

```
# Main cycle
for it in xrange(NT+1):
    # update particle position xp
    xp += vp * DT
    # Periodic boundary condition
    xp[np.where(xp < 0)] += L
    xp[np.where(xp >= L)] -= L

    # Project particles->grid
    csi = xp/dx
    g1 = np.floor(csi - 0.5) # Distance from the centre of the cell
    g = np.concatenate((g1, g1+1))
    fraz1 = 1 - np.abs(xp/dx - g1 - 0.5)
    fraz = np.concatenate((fraz1, 1-fraz1))
    g[np.where(g < 0)] += NG
    g[np.where(g > NG-1)] -= NG
    mat = sparse.csc_matrix((fraz, (p, g)), shape=(N, NG))
    rho = Q / dx * mat.toarray().sum(axis=0) + rho_back

    # Compute electric field potential
    Phi = linalg.spsolve(Poisson, -dx**2 * rho[0:NG-1])
    Phi = np.concatenate((Phi, [0]))

    # Electric field on the grid
    Eg = (np.roll(Phi, 1) - np.roll(Phi, -1)) / (2*dx)

    # interpolation grid->particle and velocity update
    vp += mat * QM * Eg * DT
```

Details of implementation - Particle mover

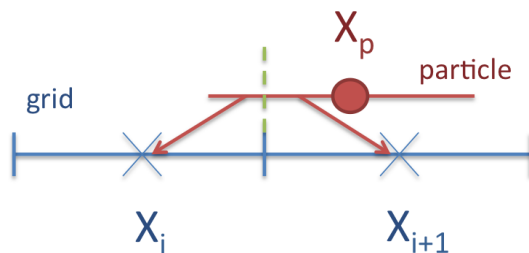
$$x_p^{n+1} = x_p^n + \Delta t v_p^{n+1/2}$$

$$v_p^{n+3/2} = v_p^{n+1/2} + \Delta t \frac{q_s}{m_s} E_p(x_p^n)$$

```
for it=1:NT
    xp=xp+vp*DT;
    ...
    * interpolate particle
    informations to grid to
    calculate charge density
    * calculate the field Eg
    ...
    vp=vp+mat*QM*Eg*DT;
end
```

Particle projection: We introduce a *logical coordinate* defined as $\xi = x/\Delta x$. The particles and the cells have logical coordinates, where the interpolation function is more easily defined as $W_{ip} = b_\ell(\xi_p - \xi_i)$. Cell centres have coordinates $\xi_i = (i + 1/2)$, $i \in [0, \text{NG} - 1]$ and particles have $\xi_p = x/\Delta x$. We consider the case of particle shapes given by b-splines of order 0 and interpolation functions, consequently, of order 1:

$$W_{ip} = b_1(\xi_p - \xi_i) \equiv \begin{cases} 1 - |\xi_p - \xi_i|, & \text{if } |\xi_p - \xi_i| < 1 \\ 0, & \text{otherwise} \end{cases} \quad (1.69)$$



The code in the example uses a matrix notation to project the particle avoiding loops but a simple loop over the particles would work as well. The matrix "mat" is equal to the interpolation function

$$W_{ip} = W(x_p - x_i)$$

that indeed has two indexes.

The charge is then the matrix W_{ip} applied to the vector q_p :

$$q_i = W \mathbf{q} \equiv \sum_p W_{ip} q_p$$

Interpolation Functions

$$W(x_i - x_p) = \int S_x(x - x_p) b_0 \left(\frac{x - x_i}{\Delta x} \right) \quad (2.28)$$

$$W(x_i - x_p) = b_{l+1} \left(\frac{x_i - x_p}{\Delta_p} \right) \quad (2.30)$$

$$\mathit{mat}(p, g) = \begin{pmatrix} \mathit{fraz1} & 1 - \mathit{fraz1} & 0 & 0 \\ 0 & \mathit{fraz1} & 1 - \mathit{fraz1} & 0 \\ \mathit{fraz1} & 1 - \mathit{fraz1} & 0 & 0 \\ 1 - \mathit{fraz1} & 0 & 0 & \mathit{fraz1} \end{pmatrix}$$

Example: Two Stream Instability

- Two counter-streaming beams of particles:

$$f(v) = \frac{n_0}{2} \delta(v - v_0) + \frac{n_0}{2} \delta(v + v_0).$$

- The dispersion relation

$$1 - \frac{\omega_p^2}{2} \left(\frac{1}{(\omega - kv_0)^2} + \frac{1}{(\omega + kv_0)^2} \right) = 0.$$

- In relativistic case

$$1 - \frac{\omega_p^2}{2\gamma_0} \left(\frac{1}{(\gamma_0\omega - kv_0\gamma_0)^2} + \frac{1}{(\gamma_0\omega + kv_0\gamma_0)^2} \right) = 0,$$

$$\gamma_0 = \sqrt{1 - \frac{v_0^2}{c^2}}.$$

3.3 Diagnostics

The great advantage of the PIC method is that it provides quantities very similar to those provided in a actual plasma experiments: the user has both information on the distribution of the plasma particles and of the fields. Below we describe some of the most used diagnostics.

Typical *particle diagnostics* are:

- phase space plots (x_p vs v_p)
- more complex phase space density plots where the phase space is discretized in a grid of cells and the number of particles in each cell is counted to give the distribution function
- Velocity distribution functions (e.g. histogram of particle counts in energy bins)

Typical *field diagnostics* are:

- plots of the fields (e.g. E , φ , ρ , ...) versus space and/or time
- FFT modes of the field quantities to measure growth rates of specific modes.

Furthermore, information can be derived from integral quantities. Examples are: the *total kinetic energy*:

$$E_k = \frac{1}{2} \sum_p m_p v_p^2 \quad (3.4)$$

the *total momentum*:

$$P = \sum_p m_p v_p \quad (3.5)$$

the *total energy of the field*:

$$E_E = \Delta x \sum_c \frac{\epsilon_0 E^2}{2} \equiv \Delta x \sum_c \frac{\rho \varphi}{2} \quad (3.6)$$

- Lapenta, Notes provided:
<https://github.com/CmPA/iPic3D/wiki/School-in-Les-Houches>



- Hockney & Eastwood, *Computer simulation using particles*
- Birdsall, *Plasma physics via computer simulation*