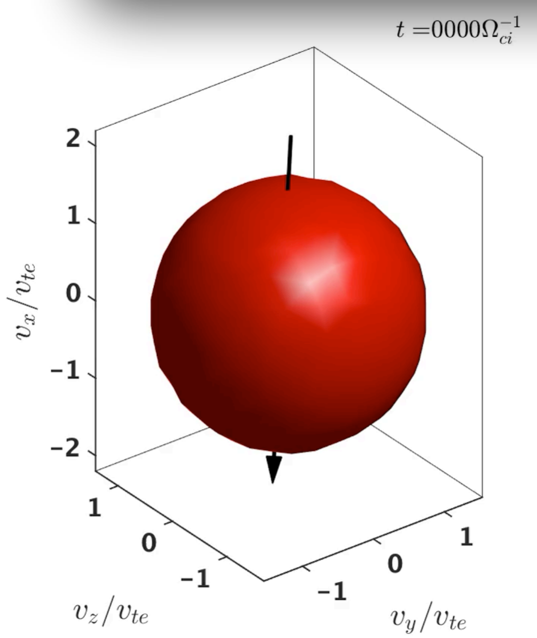
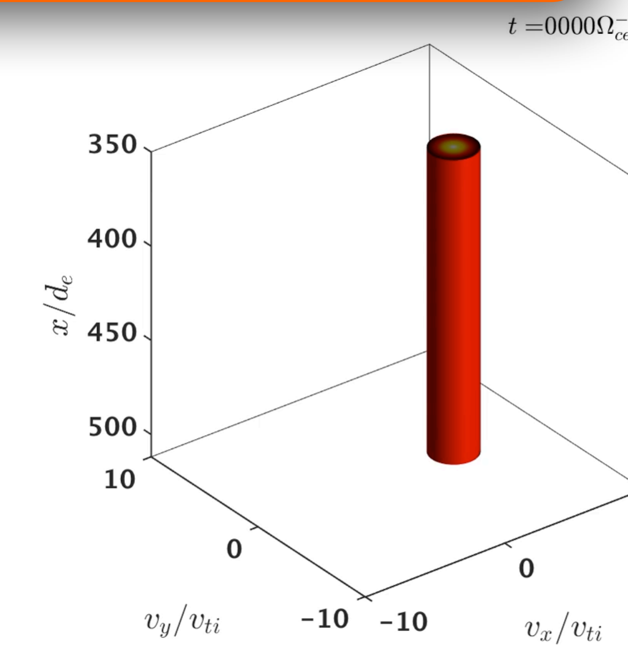
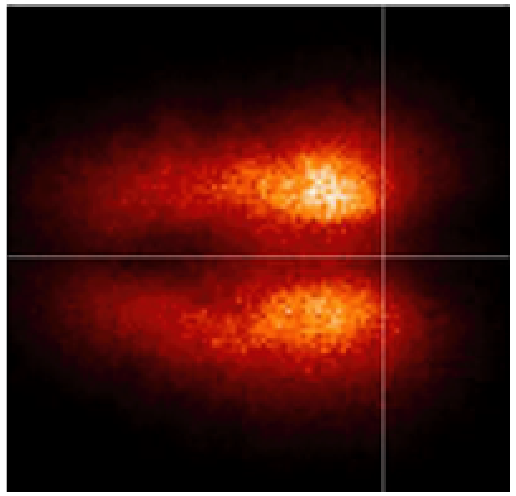
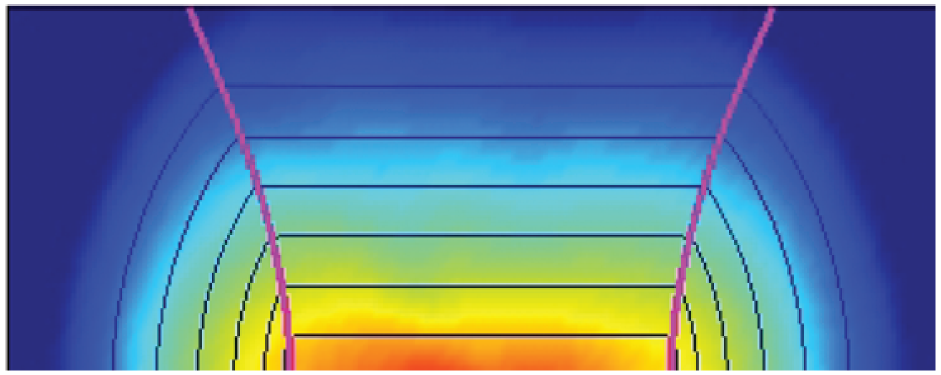
Numerical Methods for Kinetic Plasmas

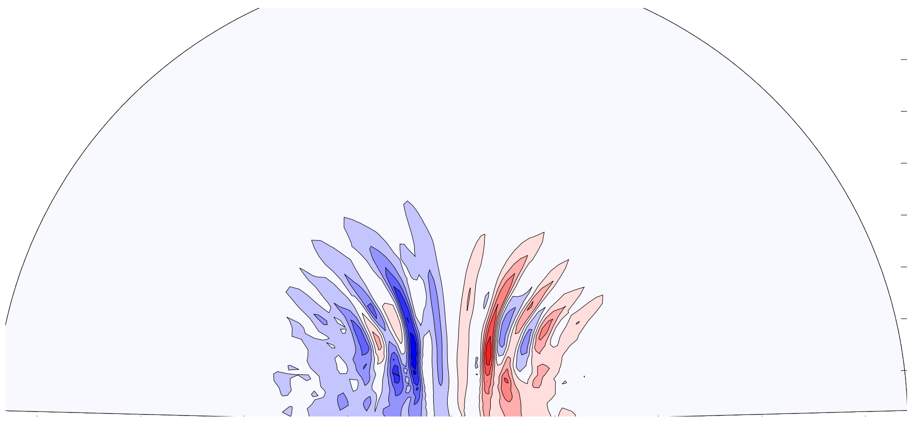


Jason M. TenBarge

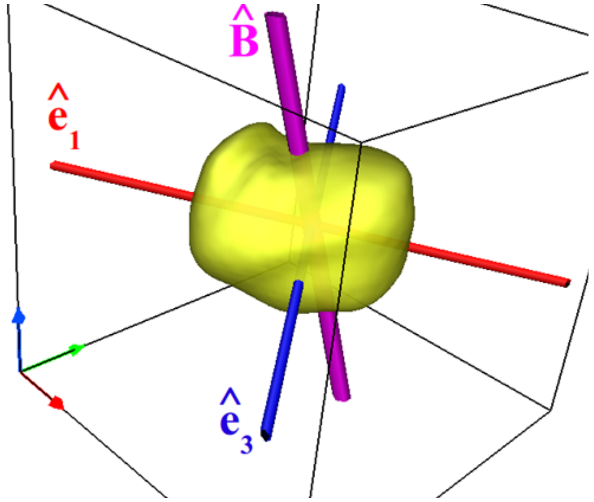




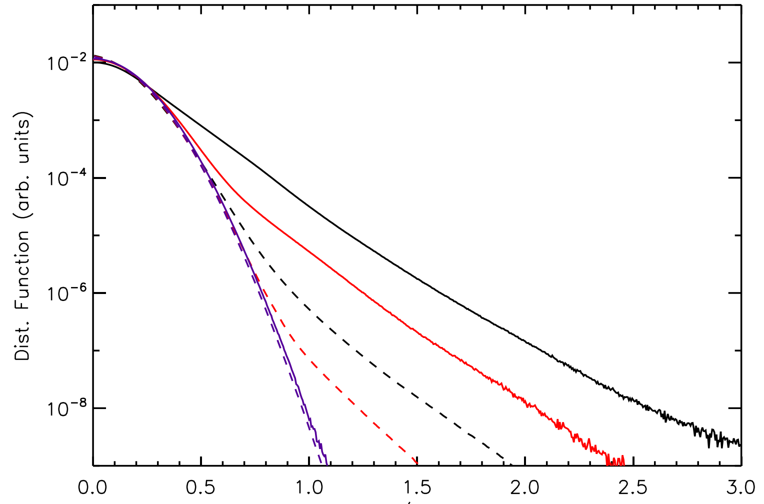




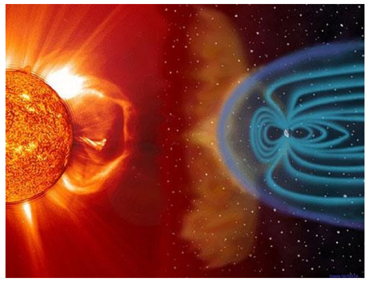
June 2019



PRINCETON UNIVERSITY



Range of scales

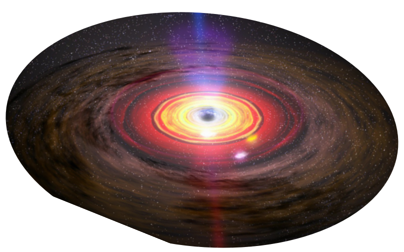


ℓ

ρ_e ρ_i λ_mfp

λ_D d_e d_i L

10^{-10} 10^{-6} 10^{-2} 1 au

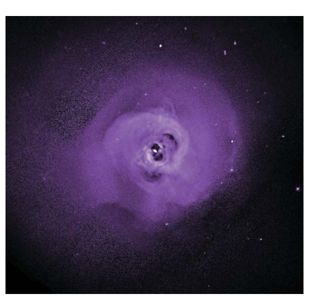


ℓ

ρ_e ρ_i λ_mfp

λ_D d_e d_i L

10^{-14} 10^{-11} 0.1 pc



ℓ

ρ_e ρ_i λ_mfp

λ_D d_e d_i L

10^{-18} 10^{-14} 1 10^2 kpc

Hierarchy of simulation methods

1) Individual particles: Klimontovich equation

2) Kinetic
- -Eulerian
- -Lagrangian: Particle-in-cell (PIC)

everything \ {  }

- -Gyrokinetics

$$\omega/\Omega \ll 1 \quad k_{\|} L \sim k_{\perp} \rho \sim 1$$

and small fluctuations

- -Kinetic MHD

$$k\rho \sim \omega/\Omega \ll 1, \;\; \mathrm{Ma} \sim 1$$

- -Drift kinetics

$$k\rho \sim \omega/\Omega \ll 1, \;\; \mathrm{Ma} \ll 1$$

3) Hybrid
- -One species kinetic, the other fluid

$$\lambda_{\mathrm{D}}, \;\; m_{\mathrm{e}}/m_{\mathrm{i}} \to 0$$

fluid electrons, kinetic ions

Hierarchy of simulation methods

1) Individual particles: Klimontovich equation
2) Kinetic
   -Eulerian
   -Lagrangian: Particle-in-cell (PIC)
   -Gyrokinetics
   -Kinetic MHD
   -Drift kinetics
3) Hybrid
   -One species kinetic, the other fluid

4) Fluid
   -Landau fluid — fluid equations + closure mimicking collisionless damping

   -Braginskii — fluid equations + anisotropic transport due to magnetization

   -Extended MHD — MHD + vestiges of kinetic effects
     * Two fluid, Hall MHD, CGL

   -MHD

   -Incompressible MHD, reduced MHD — additional assumptions added to further simplify MHD

First, a brief review of where Vlasov comes from…

(due to Klimontovich)

$$F_\alpha(\boldsymbol{r},\boldsymbol{v},t) = \sum_{i=1}^{N_\alpha} \delta(\boldsymbol{r} - \boldsymbol{R}_{\alpha i}(t))\delta(\boldsymbol{v} - \boldsymbol{V}_{\alpha i}(t))$$

positions of particles of species α

velocities of particles of species α

$$\lim_{\mathrm{d}\boldsymbol{r}\mathrm{d}\boldsymbol{v}\to 0} \int \mathrm{d}\boldsymbol{r}\mathrm{d}\boldsymbol{v}\, F_\alpha(\boldsymbol{r},\boldsymbol{v},t)$$ is either 1 or 0

if you know $\boldsymbol{R}_{\alpha i}(0)$ and $\boldsymbol{V}_{\alpha i}(0)$, and can solve

$$\frac{\mathrm{d}\boldsymbol{R}_{\alpha i}}{\mathrm{d}t} = \boldsymbol{V}_{\alpha i} \qquad \frac{\mathrm{d}\boldsymbol{V}_{\alpha i}}{\mathrm{d}t} = \frac{q_\alpha}{m_\alpha}\left[\boldsymbol{E}_\mathrm{m}(\boldsymbol{R}_{\alpha i},t) + \frac{1}{c}\boldsymbol{V}_{\alpha i}\times\boldsymbol{B}_\mathrm{m}(\boldsymbol{R}_{\alpha i},t)\right]$$

then you know everything. Done.

“Microphysical” fields computed from Maxwell’s equations

$$\boldsymbol{\nabla} \cdot \boldsymbol{B}_{\mathrm{m}} = 0$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{E}_{\mathrm{m}} = 4\pi \sum_{\alpha} q_{\alpha} \int \mathrm{d}\boldsymbol{v}\, F_{\alpha}(\boldsymbol{r}, \boldsymbol{v}, t)$$

$$\boldsymbol{\nabla} \times \boldsymbol{B}_{\mathrm{m}} = \frac{1}{c} \frac{\partial \boldsymbol{E}_{\mathrm{m}}}{\partial t} + \frac{4\pi}{c} \sum_{\alpha} q_{\alpha} \int \mathrm{d}\boldsymbol{v}\, \boldsymbol{v} F_{\alpha}(\boldsymbol{r}, \boldsymbol{v}, t)$$

$$\boldsymbol{\nabla} \times \boldsymbol{E}_{\mathrm{m}} = -\frac{1}{c} \frac{\partial \boldsymbol{B}_{\mathrm{m}}}{\partial t}$$

Rather than evolve $\boldsymbol{R}_{\alpha i}$ and $\boldsymbol{V}_{\alpha i}$, solve

$$\partial F_{\alpha}(\boldsymbol{r}, \boldsymbol{v}, t)/\partial t = \frac{\partial}{\partial t} \sum_{i=1}^{N_{\alpha}} \delta(\boldsymbol{r} - \boldsymbol{R}_{\alpha i}(t))\delta(\boldsymbol{v} - \boldsymbol{V}_{\alpha i}(t))$$

“Microphysical” fields computed from Maxwell’s equations

$$\nabla \cdot \boldsymbol{B}_{\mathrm{m}} = 0$$

$$\nabla \cdot \boldsymbol{E}_{\mathrm{m}} = 4\pi \sum_{\alpha} q_{\alpha} \int \mathrm{d}\boldsymbol{v}\, F_{\alpha}(\boldsymbol{r}, \boldsymbol{v}, t)$$

$$\nabla \times \boldsymbol{B}_{\mathrm{m}} = \frac{1}{c}\frac{\partial \boldsymbol{E}_{\mathrm{m}}}{\partial t} + \frac{4\pi}{c} \sum_{\alpha} q_{\alpha} \int \mathrm{d}\boldsymbol{v}\, \boldsymbol{v} F_{\alpha}(\boldsymbol{r}, \boldsymbol{v}, t)$$

$$\nabla \times \boldsymbol{E}_{\mathrm{m}} = -\frac{1}{c}\frac{\partial \boldsymbol{B}_{\mathrm{m}}}{\partial t}$$

Rather than evolve $\boldsymbol{R}_{\alpha i}$ and $\boldsymbol{V}_{\alpha i}$, solve

$$\left[\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla + \frac{q_{\alpha}}{m_{\alpha}} \left(\boldsymbol{E}_{\mathrm{m}} + \frac{1}{c}\boldsymbol{v} \times \boldsymbol{B}_{\mathrm{m}}\right) \cdot \frac{\partial}{\partial \boldsymbol{v}}\right] F_{\alpha}(\boldsymbol{r}, \boldsymbol{v}, t) = \frac{DF_{\alpha}}{Dt} = 0$$

“Klimontovich equation”

An important note

The Klimontovich equation is equivalent to phase-space conservation, but it is NOT a statistical equation.
It *looks* like the Vlasov equation, but it is completely different!

With proper initial conditions,
it is *deterministic*, not *probabilistic*.

This makes it cumbersome… but it *does* form the basis of particle-in-cell (PIC) methods and statistical plasma kinetics.

Let's see the latter…

Averaging

Ensemble averaging over all microscopic realizations of the macroscopic plasma (which is equivalent to a coarse-graining procedure by ergodicity),

$$\left[\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla} + \frac{q_\alpha}{m_\alpha}\left(\boldsymbol{E} + \frac{1}{c}\boldsymbol{v}\times\boldsymbol{B}\right)\cdot\frac{\partial}{\partial \boldsymbol{v}}\right] f_\alpha(\boldsymbol{r},\boldsymbol{v},t) =$$

$$-\frac{q_\alpha}{m_\alpha}\left\langle\left(\delta\boldsymbol{E} + \frac{1}{c}\boldsymbol{v}\times\delta\boldsymbol{B}\right)\cdot\frac{\partial F_\alpha}{\partial \boldsymbol{v}}\right\rangle$$

LHS = Vlasov equation

RHS = collisions due to discrete nature of particles

$\sim \Lambda^{-1} \doteq (n\lambda_{\mathrm{D}}^3)^{-1} \ll 1$ the LHS

this is probabilistic (even more so once the RHS is simplified)

Eulerian (Continuum) vs Lagrangian (PIC)

solve

$$\left[\frac{\partial}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla} + \frac{q_\alpha}{m_\alpha}\left(\boldsymbol{E} + \frac{1}{c}\boldsymbol{v}\times\boldsymbol{B}\right)\cdot\frac{\partial}{\partial \boldsymbol{v}}\right] f_\alpha(\boldsymbol{r},\boldsymbol{v},t) = \left(\frac{\partial f_\alpha}{\partial t}\right)_{\mathrm{coll}}$$

in 6D phase space (“Eulerian”)

or

solve

$$\frac{\mathrm{d}\boldsymbol{R}_{\alpha i}}{\mathrm{d}t} = \boldsymbol{V}_{\alpha i} \qquad \frac{\mathrm{d}\boldsymbol{V}_{\alpha i}}{\mathrm{d}t} = \frac{q_\alpha}{m_\alpha}\left(\boldsymbol{E}_{\mathrm{m}} + \frac{1}{c}\boldsymbol{V}_{\alpha i}\times\boldsymbol{B}_{\mathrm{m}}\right)$$

for a finite number of (macro)particles (“Lagrangian”)

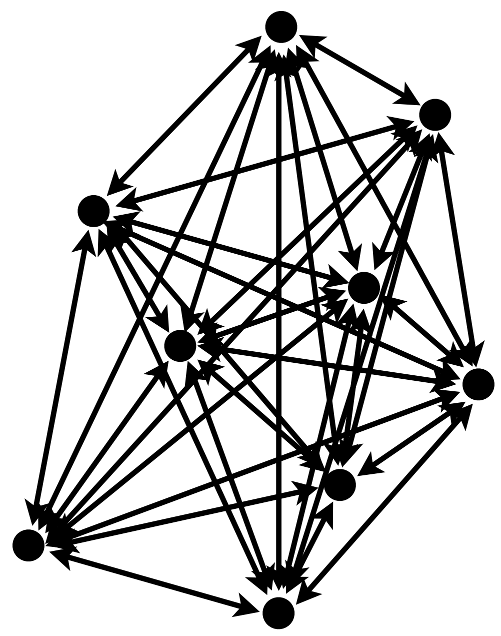
(f = const on these characteristics)

Macroparticles plus a grid

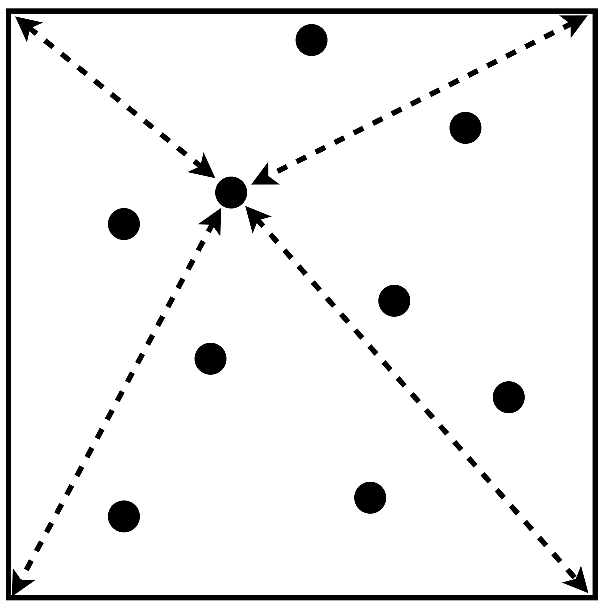
In the Lagrangian case, you really don't want to do particle pairing for ~10^{10} particles per Debye cloud!

concept of (macro)particles communicating with one another electromagnetically via a grid; reduction in # of pairings

particle-particle



particle-mesh (PIC)



number of pairs: $\frac{N(N-1)}{2} \propto N^2$

$\propto N$

Lagrangian (Klimontovich/PIC)

- Only 3D grid needed for real space; Monte-Carlo sampling of velocity space; means that parallelization is easy and usually gives good scaling
- Easy to write
- “Unlimited” dynamical range for particle velocities; no boundary conditions on **v**

- Difficult to include explicit collisions; usually not even implemented
- Limited phase-space density resolution
- Errors from finite-size particles (smoothing)
- Load balancing issues

- √N noise! Need lots of particles to capture phase mixing, collisionless damping, and small-amplitude fluctuations properly
- Things can go unpredictably wrong

Eulerian (Vlasov-Landau)

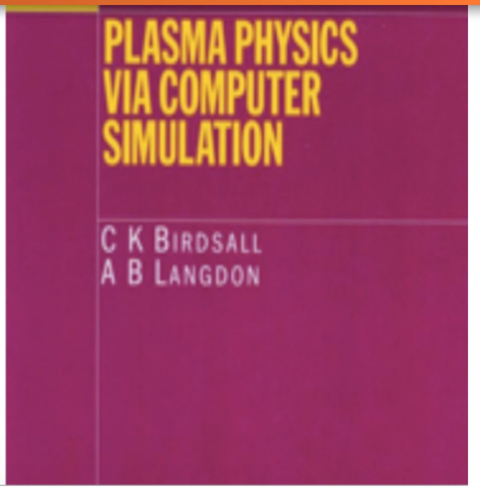
- No noise
- Good control over dissipation; easier to include collisions
- No issues if plasma very inhomogeneous

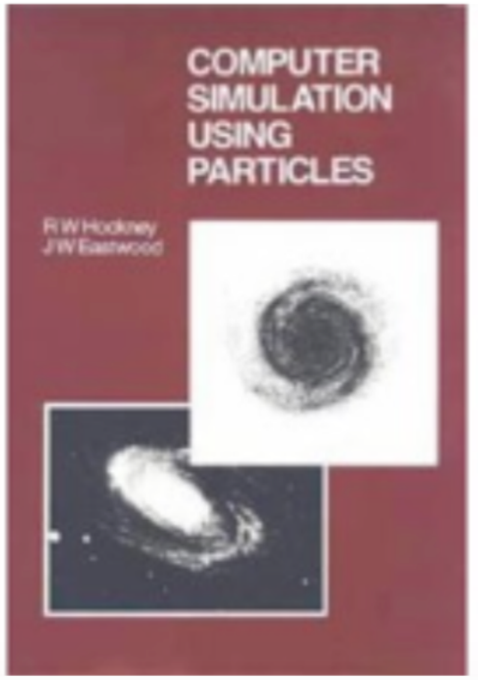
- 6D grid -> extremely expensive; often results in poor velocity-space resolution
- Difficult to parallelize efficiently

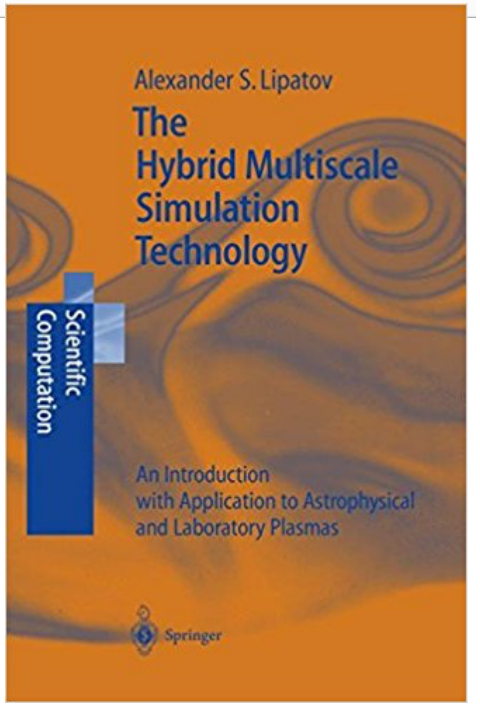
- Velocity space isn't (easily) adaptable, …

PIC simulations: Some history

- Dawson's sheet model (1962): 1000 sheets in 1D; started late 1950s at Princeton, later @ UCLA
- Hockney, Buneman (1965): introduced grids and direct Poisson solve
- Finite-size particles and PIC (Dawson et al. 1968; Birdsall et al. 1968)
- Short-wavelength and high-frequency particle noise minimized via charge sharing and smoothing schemes; noise studied by fluctuation-dissipation theorem (Klimontovich 1967; Langdon 1979; Birdsall & Langdon 1983; Krommes 1993 for GK PIC)
- 1980s-90s 3D electromagnetic PIC booms; "PIC bibles" 1988 and 1990



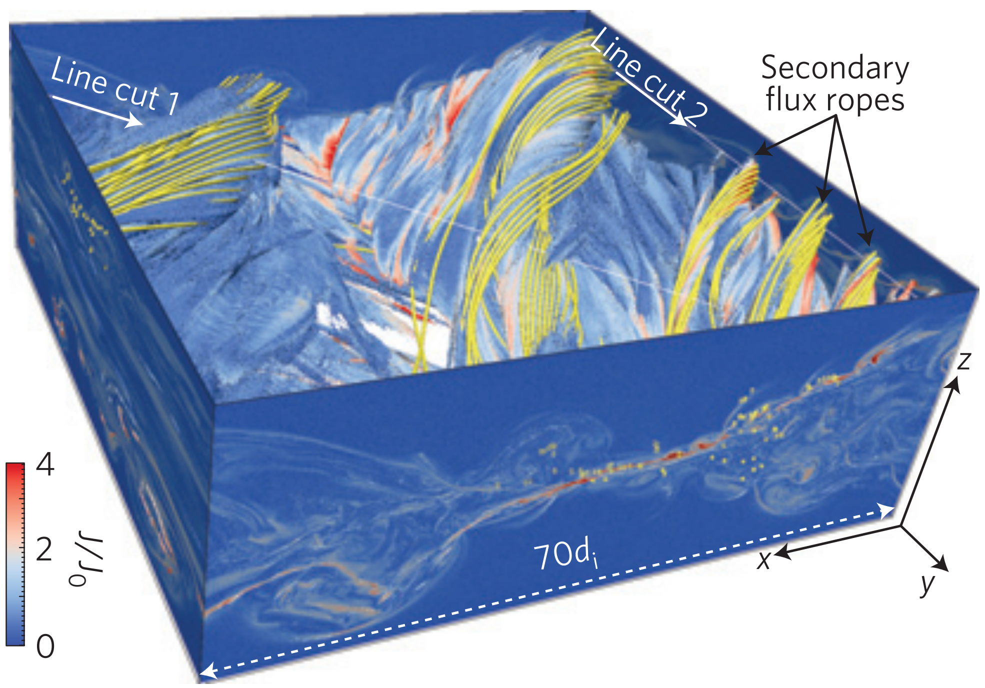




PIC simulation successes

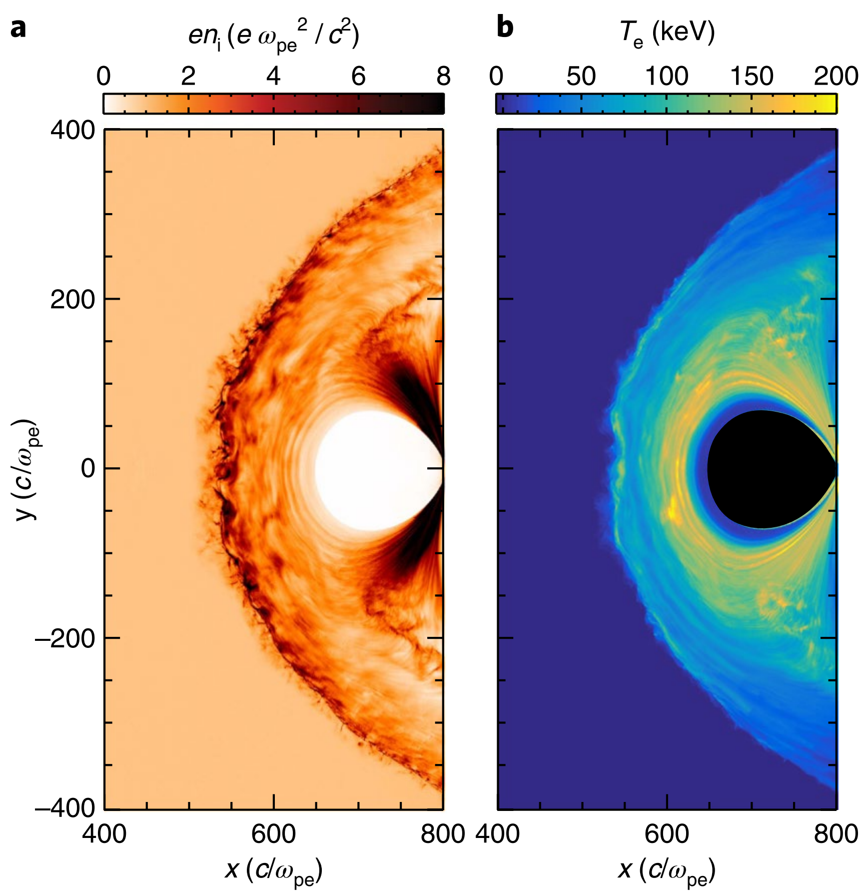
PIC has been enormously successful for modeling large amplitude, kinetic phenomena

Magnetic reconnection (VPIC)



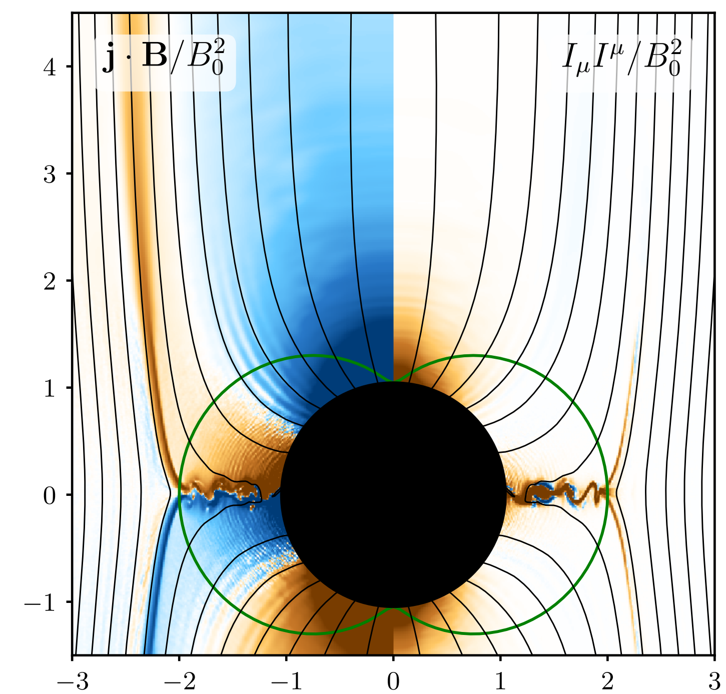
Daughton et al Nature Phys (2011)

Shocks (OSIRIS)



Rigby et al Nature Phys (2018)

Black hole jet formation (ZELTRON)



Parfrey et al PRL (2019)

PIC algorithm [Birdsall and Langdon (1991)]

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

- Solve Vlasov equation along characteristics

$$m \frac{d\mathbf{v}}{dt} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \; \frac{d\mathbf{x}}{dt} = \mathbf{v}$$

- Describes “single” particle evolution in Lagrangian framework

$$\nabla \cdot \mathbf{E} = 4\pi \rho_c$$

$$\frac{1}{c} \frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{B} - \frac{4\pi}{c} \mathbf{J}$$

- Each particle is really a super particle, representing many real particles, although q/m is kept the same

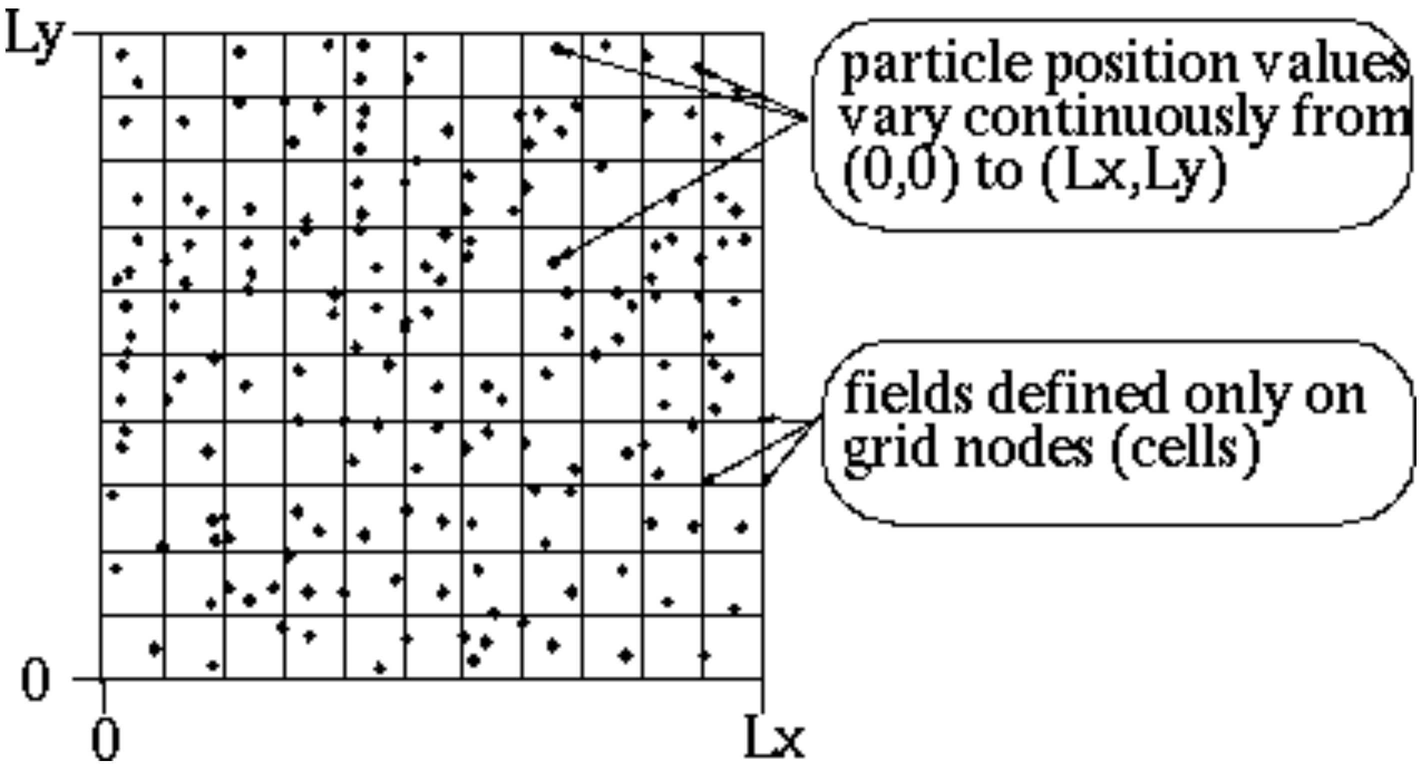
$$c \nabla \times \mathbf{E} = - \frac{\partial \mathbf{B}}{\partial t}$$

- Fields are not between individual particles

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

Solving Maxwell’s equations requires a grid, e.g.,

$$\frac{E_j - E_{j-1}}{\Delta x} = 4\pi \rho_{cj}$$



PIC simulations: General idea

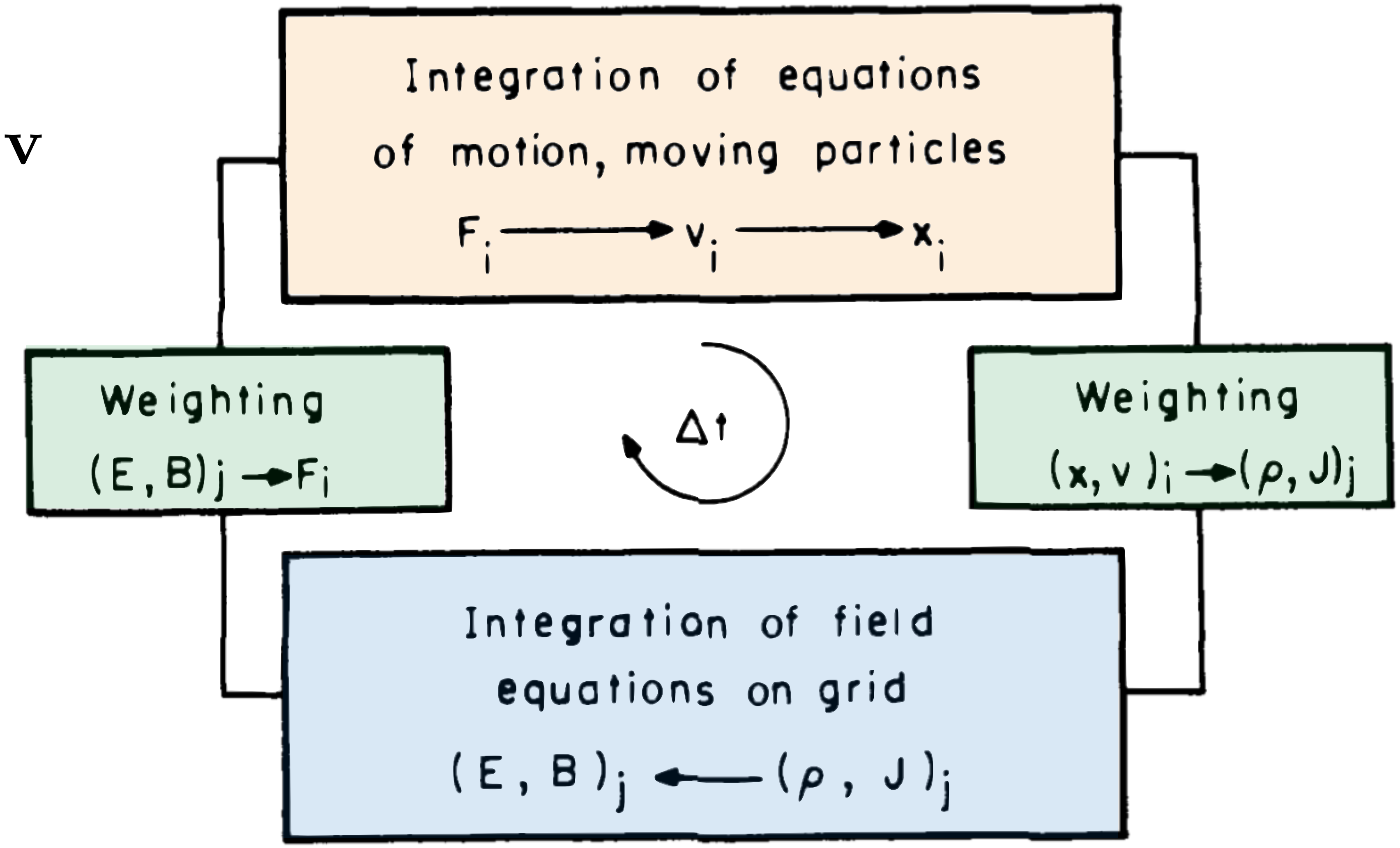
$$m\frac{d\mathbf{v}}{dt} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}),\ \frac{d\mathbf{x}}{dt} = \mathbf{v}$$

$$\nabla \cdot \mathbf{E} = 4\pi\rho_c$$

$$\frac{1}{c}\frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{B} - \frac{4\pi}{c}\mathbf{J}$$

$$c\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

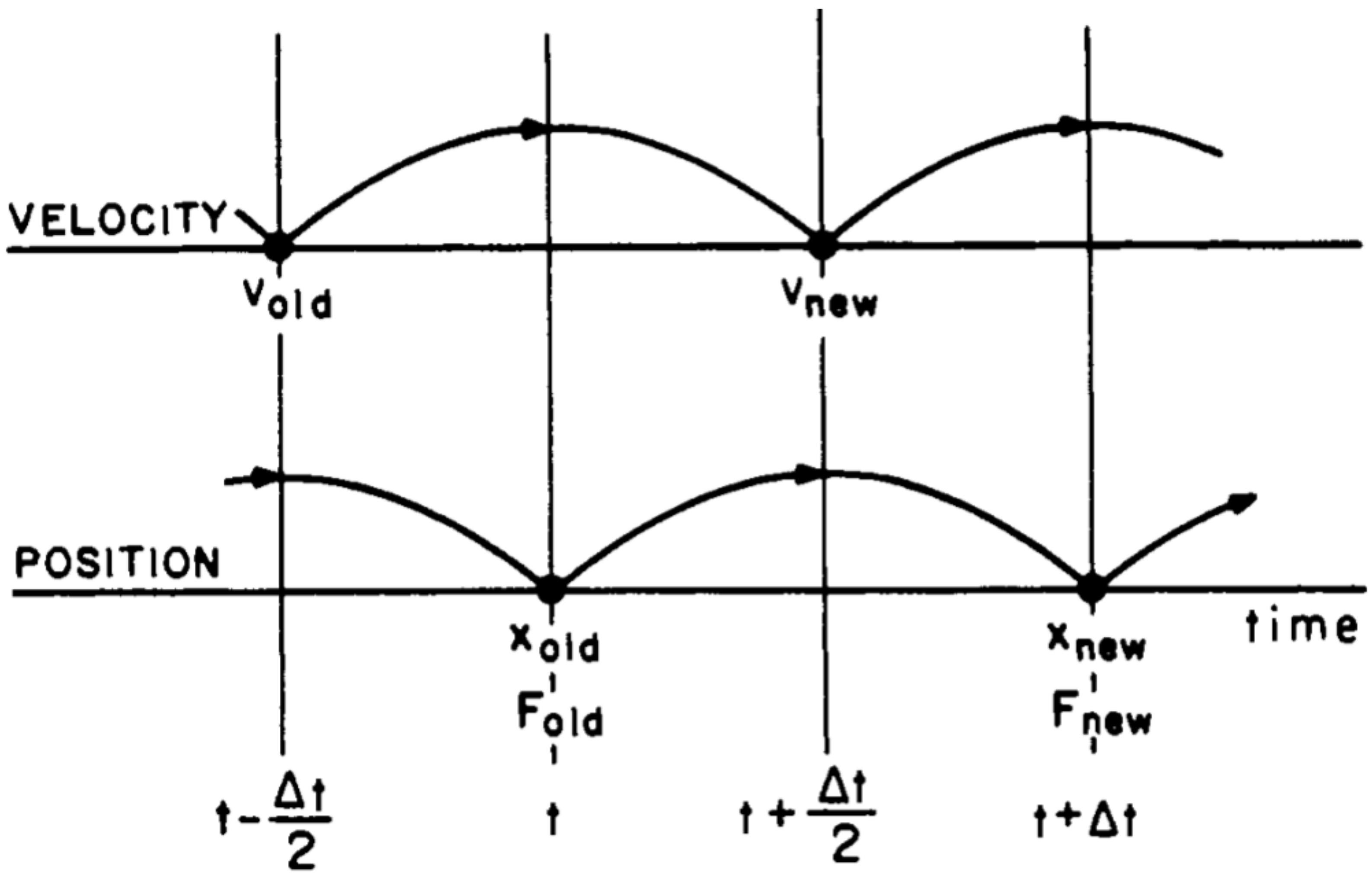
$$\nabla \cdot \mathbf{B} = 0$$



note: sometimes fields are subcycled to reduce cost, but great care must be taken to avoid instability

Step 1: Push Particles

leapfrog algorithm



2nd-order accurate

time-reversible

symplectic

?

$$\frac{\boldsymbol{V}_i^{n+1/2} - \boldsymbol{V}_i^{n-1/2}}{\Delta t} = \boldsymbol{E}_i^n(\boldsymbol{R}_i^n) + \boldsymbol{V}_i^n \times \boldsymbol{B}^n(\boldsymbol{R}_i^n)$$

$$\frac{\boldsymbol{R}_i^{n+1} - \boldsymbol{R}_i^n}{\Delta t} = \boldsymbol{V}_i^{n+1/2}$$

Step 1: Push Particles

Crank-Nicolson (Buneman 1967): $\boldsymbol{V}_i^n = \dfrac{\boldsymbol{V}_i^{n+1/2} + \boldsymbol{V}_i^{n-1/2}}{2}$

$$\Longrightarrow \frac{\boldsymbol{V}_i^{n+1/2} - \boldsymbol{V}_i^{n-1/2}}{\Delta t} = \boldsymbol{E}_i^n(\boldsymbol{R}_i^n) + \frac{\boldsymbol{V}_i^{n+1/2} + \boldsymbol{V}_i^{n-1/2}}{2} \times \boldsymbol{B}^n(\boldsymbol{R}_i^n)$$

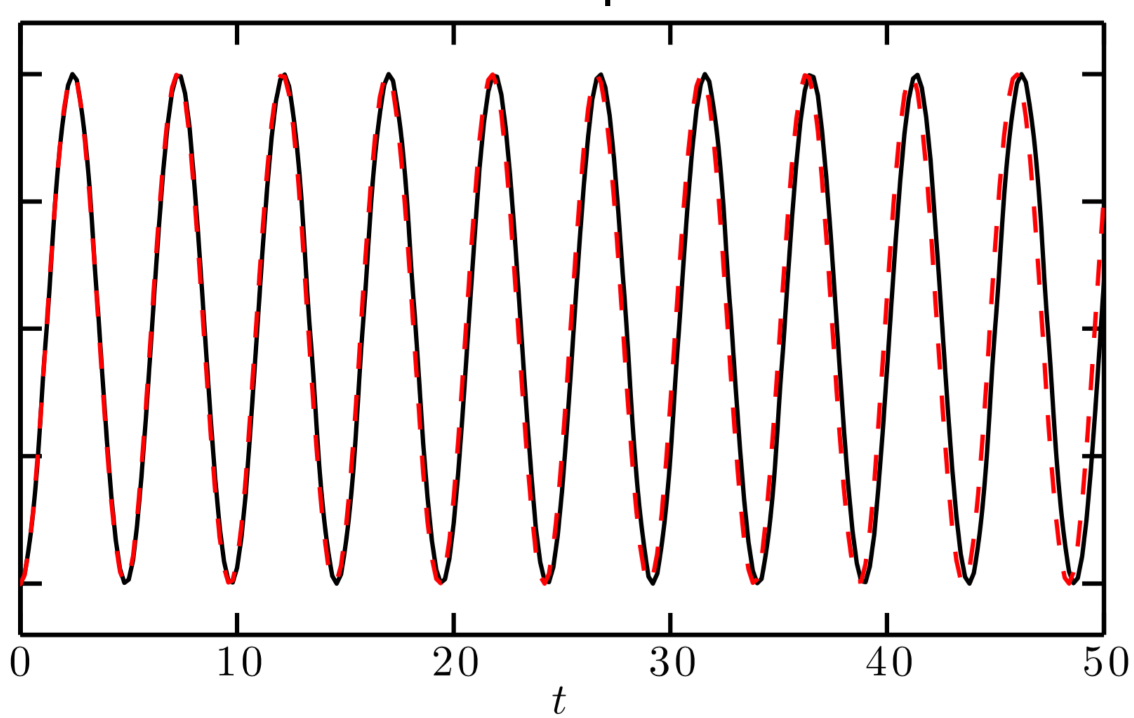
Boris (1970) algorithm (time-reversible, conserves energy and phase space volume):

$$\mathbf{V}_i^{n-1/2} = \mathbf{V}_i^{-} - \mathbf{E}_i^n(\mathbf{R}_i^n)\frac{\Delta t}{2}$$

$$\mathbf{V}_i^{n+1/2} = \mathbf{V}_i^{+} + \mathbf{E}_i^n(\mathbf{R}_i^n)\frac{\Delta t}{2}$$

$$\frac{\mathbf{V}_i^{+} - \mathbf{V}_i^{-}}{\Delta t} = \frac{\mathbf{V}_i^{+} + \mathbf{V}_i^{-}}{2} \times \mathbf{B}^n(\mathbf{R}_i^n)$$

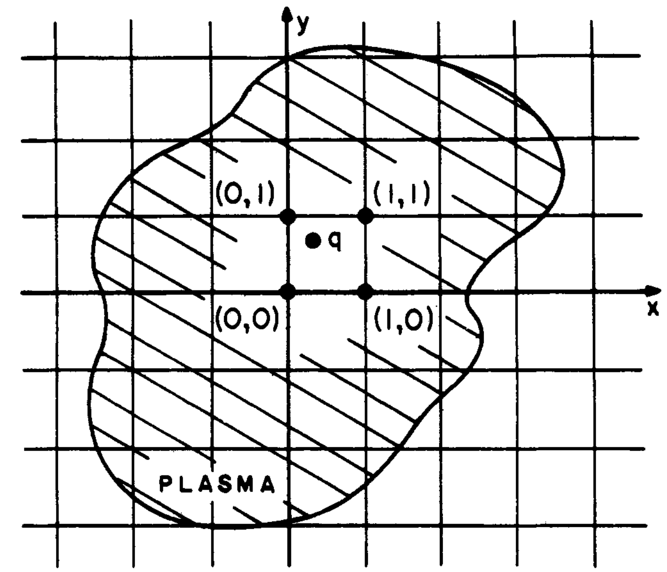
makes small phase error



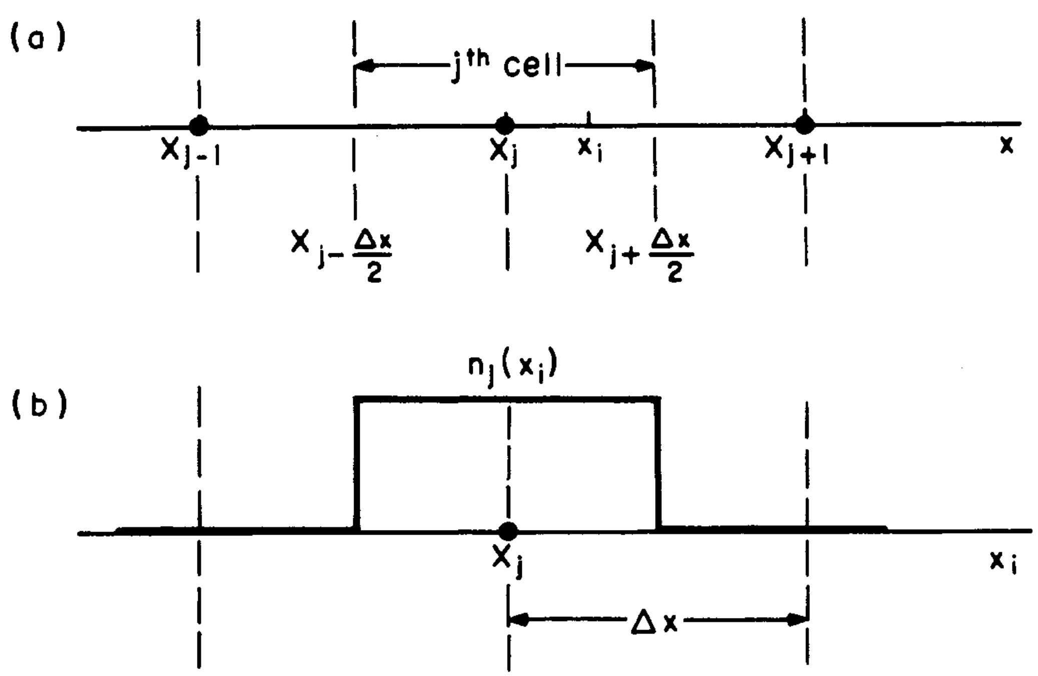
can overstep gyromotion without stability issues (just accuracy issues…)

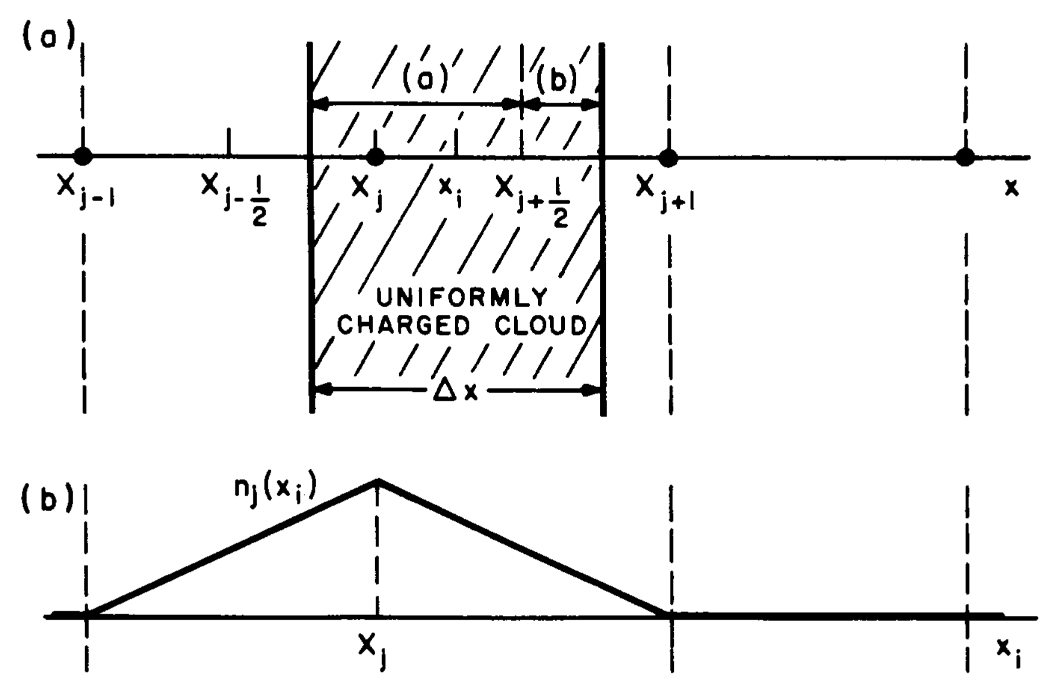
How do we put the particles on the grid?

Consider a cloud of plasma in your simulation domain. How do you represent the charge density on the grid?



Particles must have a shape associated with them. The shape corresponds to the order of interpolation, with higher orders leading to less noise.





Step 2: Deposit particles to grid

Simulation particles are not delta functions in real space; they represent large number of physical particles: "macroparticles" or "Lagrangian markers"

$$n_\alpha(\boldsymbol{r}) = \int \mathrm{d}\boldsymbol{v}\, F_\alpha = \sum_{i=1}^{N_\alpha} \delta(\boldsymbol{r} - \boldsymbol{R}_{\alpha i}) \rightarrow \sum_{i=1}^{N_\alpha} S(\boldsymbol{r} - \boldsymbol{R}_{\alpha i})$$

$$n_\alpha(\boldsymbol{r})\boldsymbol{u}_\alpha(\boldsymbol{r}) = \int \mathrm{d}\boldsymbol{v}\, \boldsymbol{v} F_\alpha = \sum_{i=1}^{N_\alpha} \boldsymbol{V}_{\alpha i}\, \delta(\boldsymbol{r} - \boldsymbol{R}_{\alpha i}) \rightarrow \sum_{i=1}^{N_\alpha} \boldsymbol{V}_{\alpha i}\, S(\boldsymbol{r} - \boldsymbol{R}_{\alpha i})$$

"shape function"

dictates how much phase-space density is assigned to a given grid cell

Coulomb force between finite-size particles

Finite-size particles considerably reduce Coulomb interactions

inter-particle forces inside a cell are underestimated; collisions must be re-introduced for controlled dissipation (rarely done)

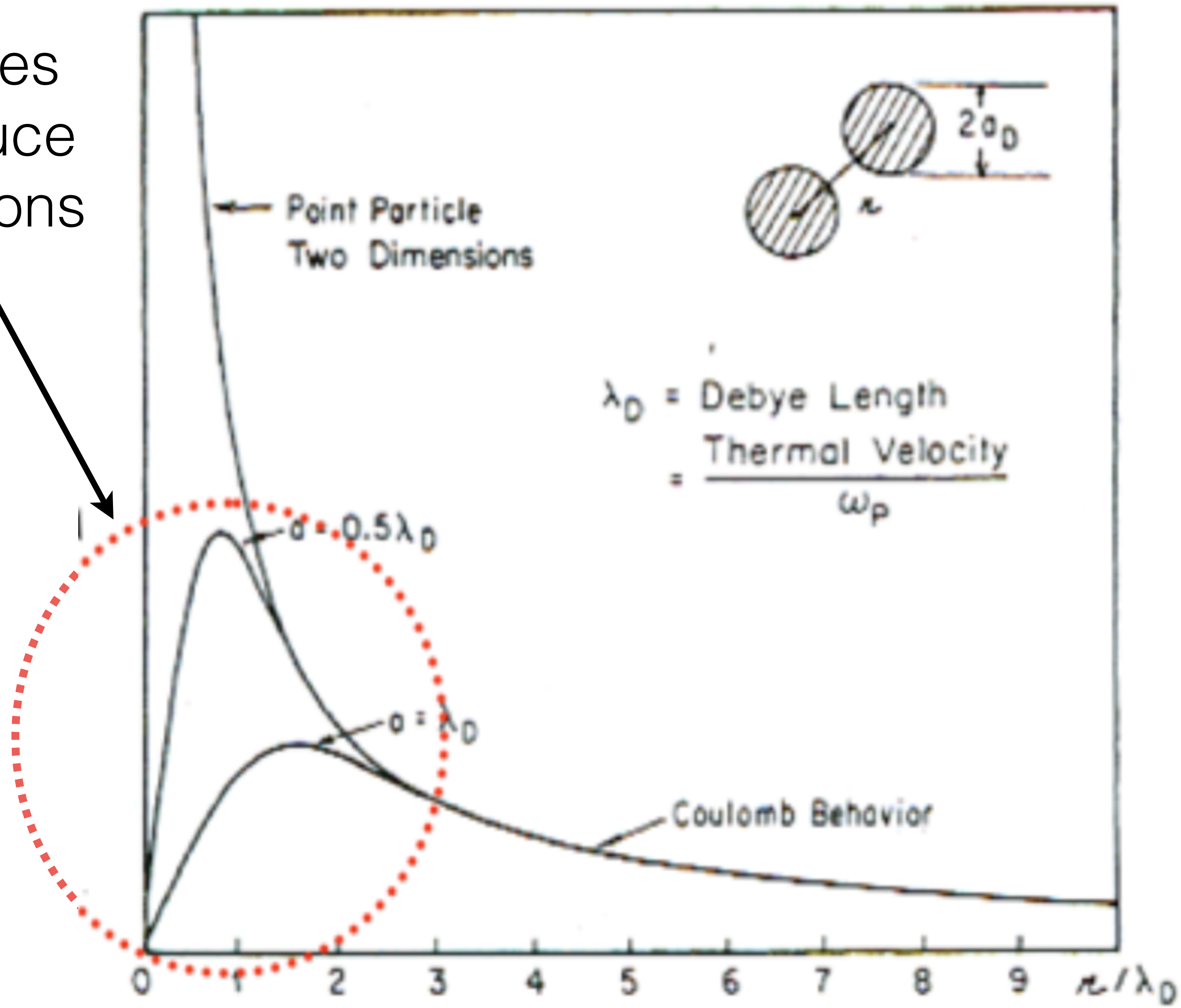
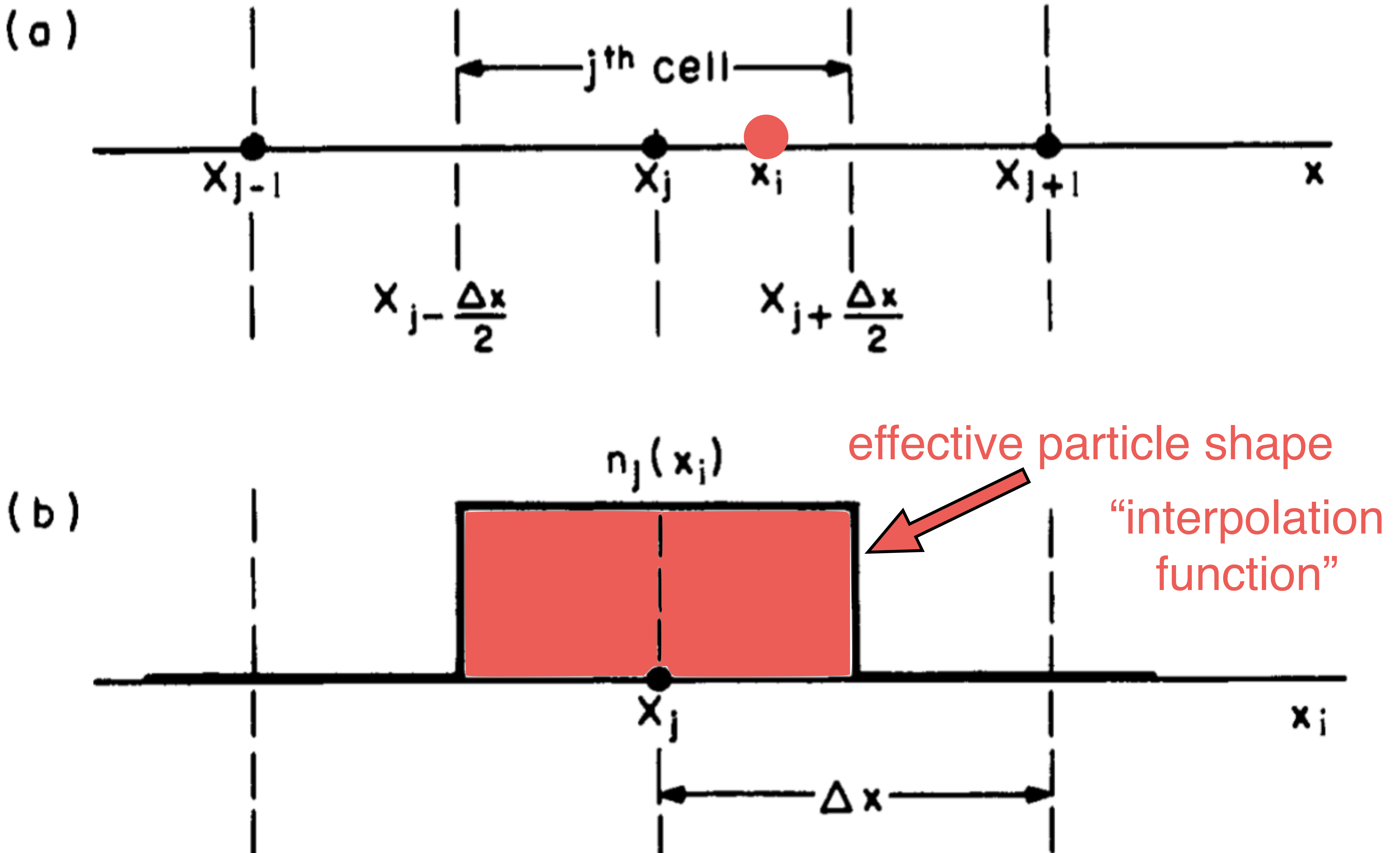


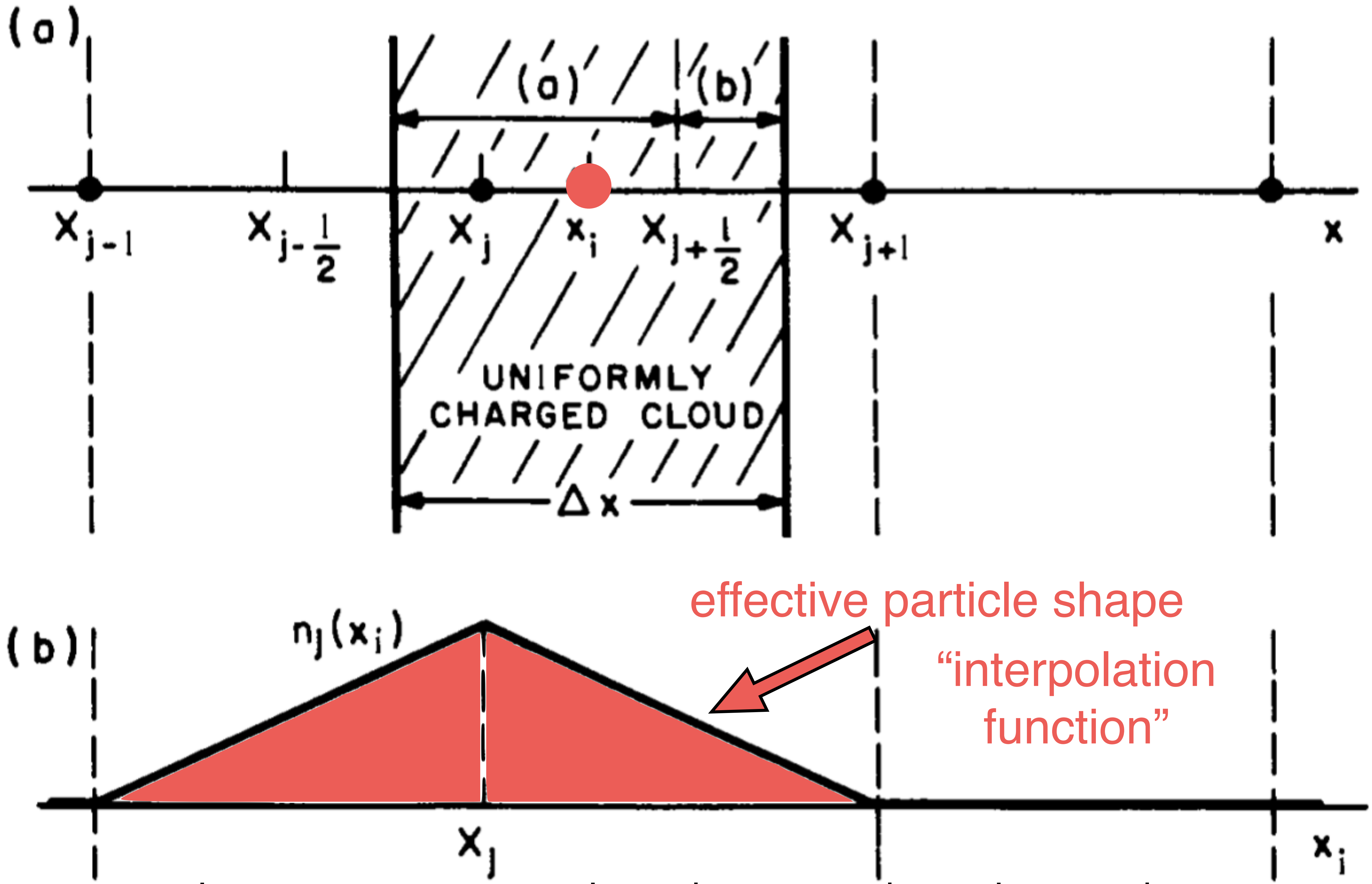
FIG. 2. Force law between finite-size particles in two dimensions for various sized particles. A Gaussian-shaped charge-density profile was used.

0th-order particle weighting (nearest neighbor)



assigned to whatever cell contains particle
(bad: discontinuous forces)

1st-order particle weighting (cloud-in-cell; CIC)



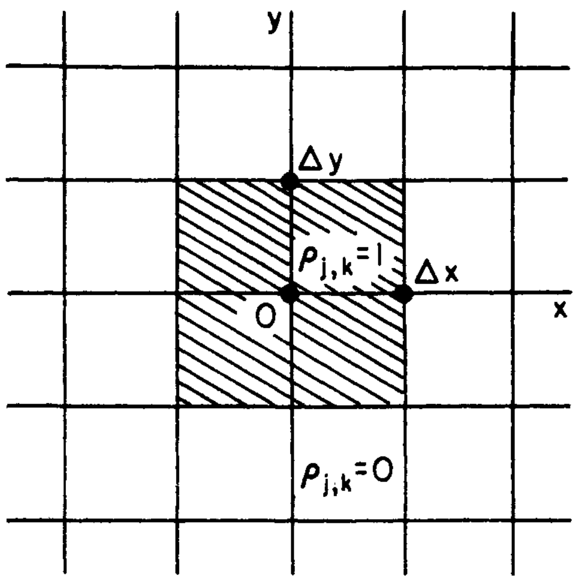
assignment proportional to overlapping volume
(ok: continuous forces, discontinuous derivatives)

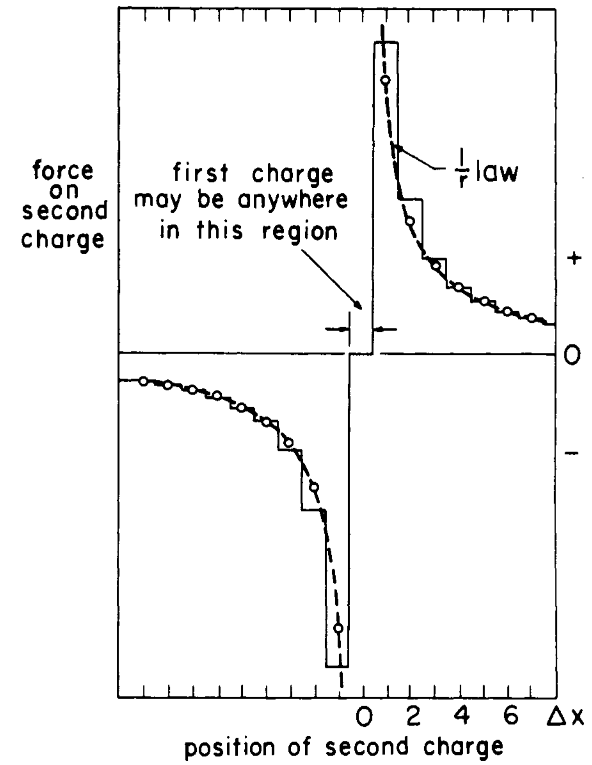
What sort of error does the particle shape cause?

Particle shape

Force on second particle

0th order



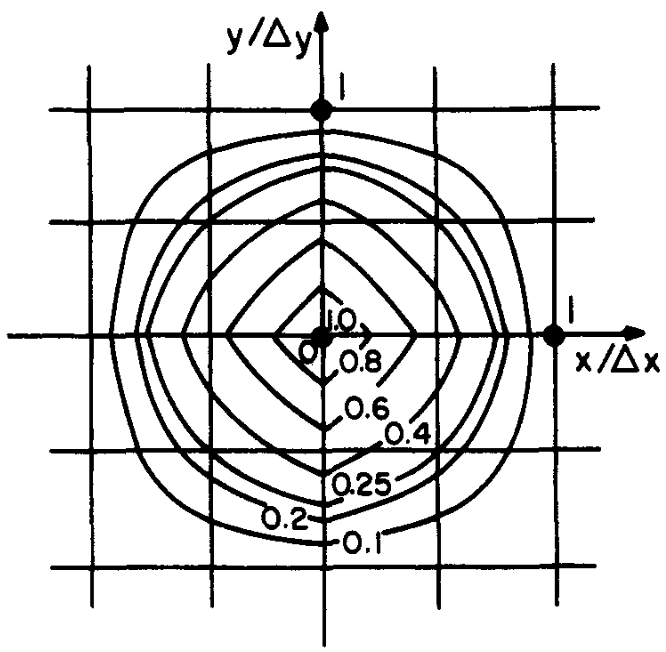


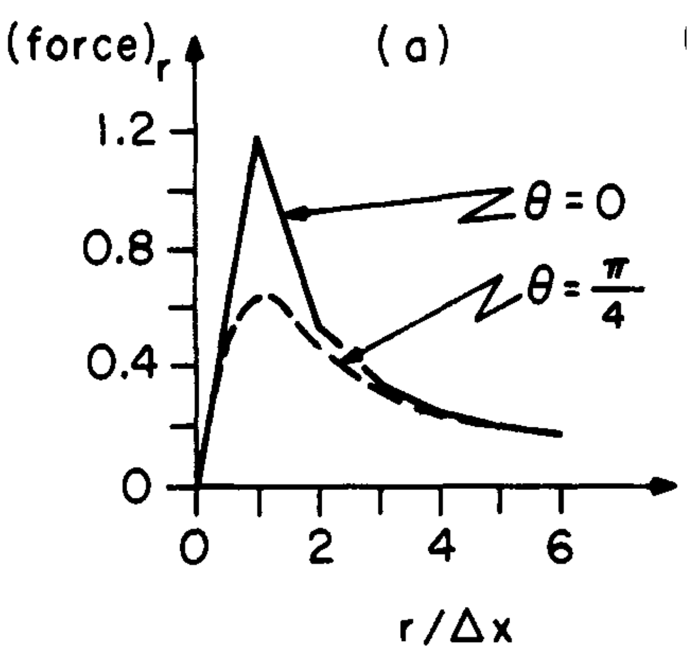
Error in the force

$$\delta F = q(E - E_0)$$

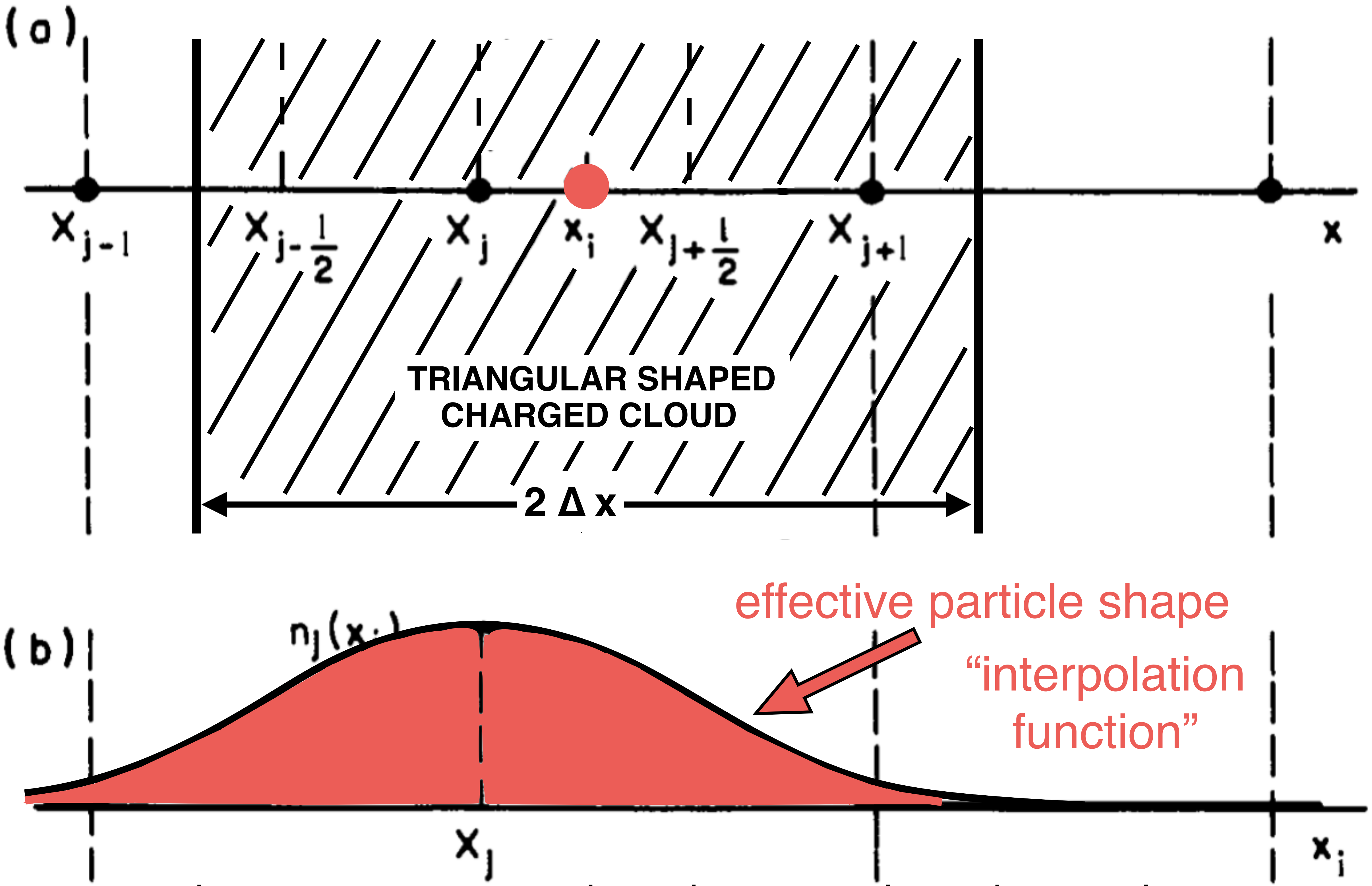
leads to noise and heating

1st order





2nd-order particle weighting (triangular shaped cloud; TSC)



assignment proportional to overlapping volume
(good: continuous forces and first derivatives)

Particle shape in practice

In principle, higher-order shape functions can be used, which result in better spatial filtering of high-frequency components; but these require a larger stencil, which means many more accesses of memory

> 2nd-order deposition rarely used

Instead, spatial filtering performed to smooth moments

spectral code: trivially done in k space

grid code: done by “digital filtering” (Hamming 77)

Step 3: Update fields

evolution equations:

$$\frac{\partial \boldsymbol{B}}{\partial t} = -c\boldsymbol{\nabla} \times \boldsymbol{E}$$

$$\frac{\partial \boldsymbol{E}}{\partial t} = c\boldsymbol{\nabla} \times \boldsymbol{B} - 4\pi \sum_{\alpha} q_{\alpha} \sum_{i=1}^{N_{\alpha}} \boldsymbol{V}_{\alpha i}\, S(\boldsymbol{r} - \boldsymbol{R}_{\alpha i})$$

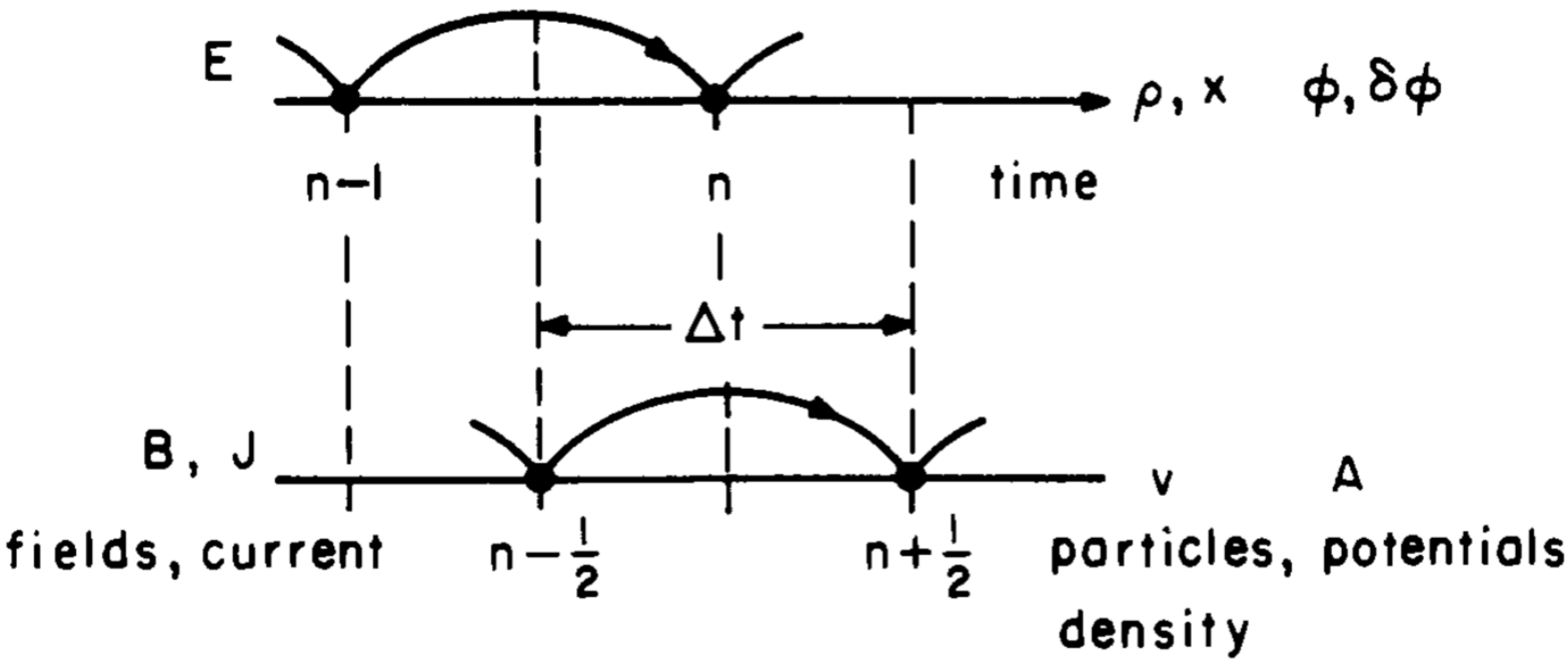
constraints:

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

$$\boldsymbol{\nabla} \cdot \boldsymbol{E} = 4\pi \sum_{\alpha} q_{\alpha} \sum_{i=1}^{N_{\alpha}} S(\boldsymbol{r} - \boldsymbol{R}_{\alpha i})$$

broken by truncation error if you’re not careful!

Symmetry of Maxwell's equations suggests leapfrog:

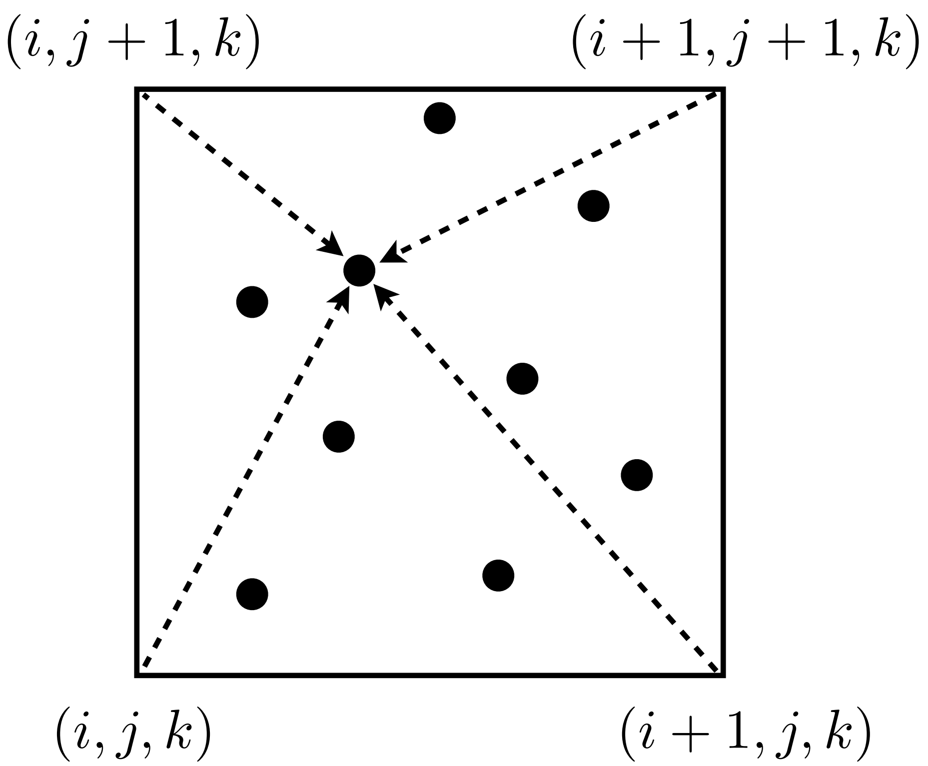


$$\frac{\boldsymbol{E}^{n}-\boldsymbol{E}^{n-1}}{\Delta t}=\boldsymbol{\nabla}\times\boldsymbol{B}^{n-1/2}-\boldsymbol{J}^{n-1/2}$$

$$\frac{\boldsymbol{B}^{n+1/2}-\boldsymbol{B}^{n-1/2}}{\Delta t}=-\boldsymbol{\nabla}\times\boldsymbol{E}^{n}$$

Step 4: Interpolate grid to particles

Interpolation to/from grid must be done in same way,
or else you get self-force



$$\boldsymbol{E}(\boldsymbol{R}_p) = \sum_{i,j,k} \boldsymbol{E}(\boldsymbol{r}_{i,j,k})\, S(\boldsymbol{r}_{i,j,k} - \boldsymbol{R}_p)$$

$$\boldsymbol{B}(\boldsymbol{R}_p) = \sum_{i,j,k} \boldsymbol{B}(\boldsymbol{r}_{i,j,k})\, S(\boldsymbol{r}_{i,j,k} - \boldsymbol{R}_p)$$

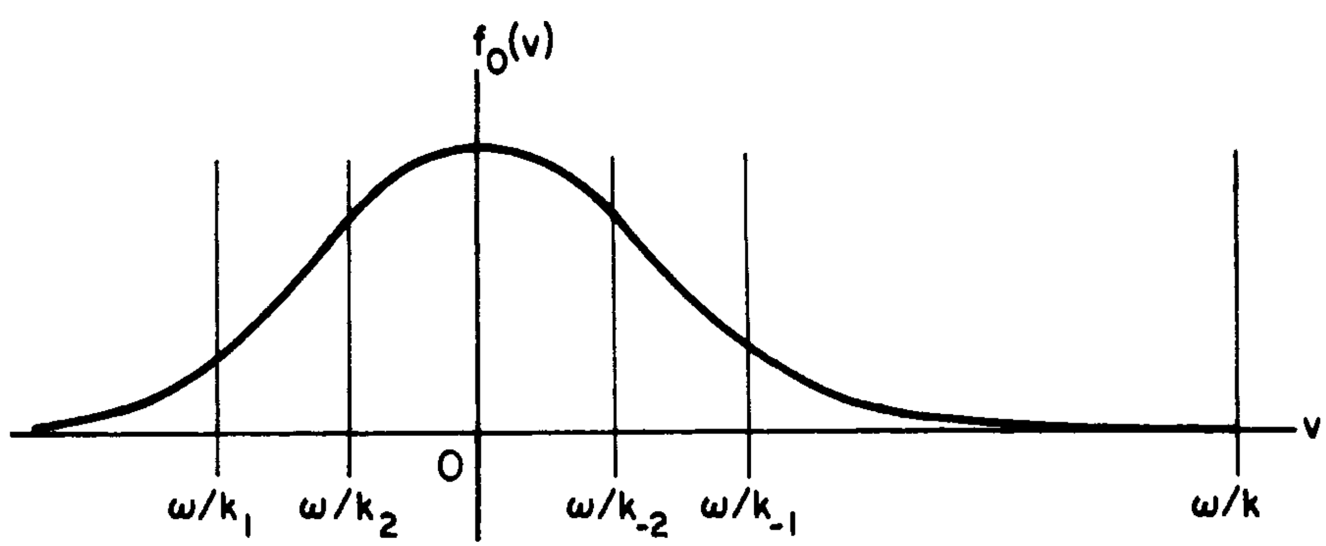
$$\sum_{i,j,k} \boldsymbol{E}(\boldsymbol{r}_{i,j,k}) \cdot \boldsymbol{B}(\boldsymbol{r}_{i,j,k})\, S(\boldsymbol{r}_{i,j,k} - \boldsymbol{R}_p) \stackrel{?}{=} \boldsymbol{E}(\boldsymbol{R}_p) \cdot \boldsymbol{B}(\boldsymbol{R}_p)$$

better be…

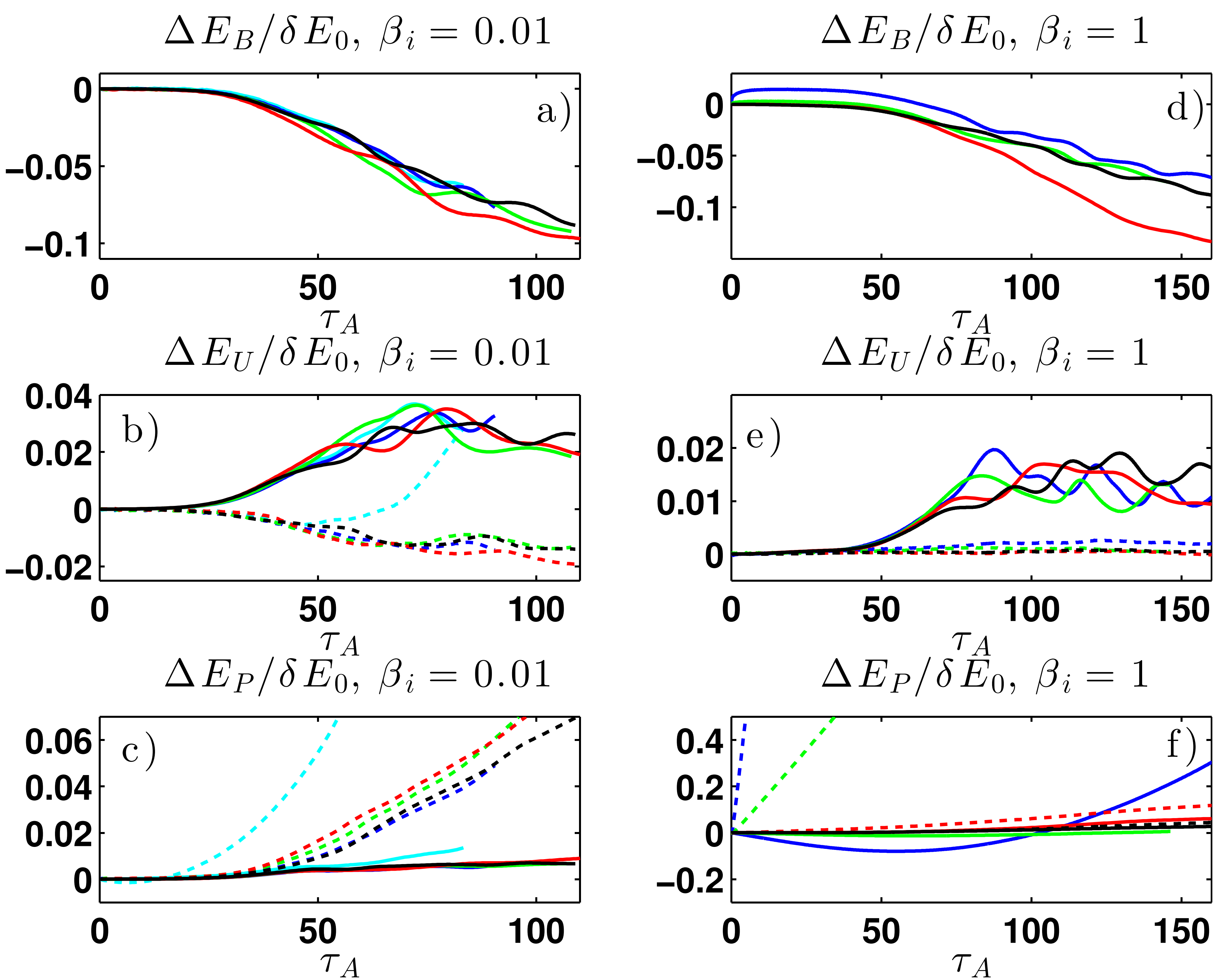
Stability conditions

Time: Courant–Friedrichs–Lewy (CFL) $$\Delta t \leq \frac{1}{c}\left(\frac{1}{(\Delta x)^2}+\frac{1}{(\Delta y)^2}+\frac{1}{(\Delta z)^2}\right)^{-1/2}$$

- Particles live on a continuous domain, which contains the minimum possible wavelength to which the particles can respond, $k_p\lambda_e = 1$, but the grid is discrete and has a lower bound on the wavelength, k_g.
- This means that a particle wave may have phase speed $v_p = \omega/k_p$, but the grid wave will have $v_g = \omega/k_g$.
- For $k_g \gg 1/\lambda_e$, $v_g \gg v_p$, and the particles may be strongly resonant with the wave, leading to self-heating.
- In practice, the plasma heats until the Debye length is resolved.



PIC numerical heating [TenBarge et al, PoP (2014)]



Black = continuum, gyrokinetic result

Colors = PIC with different guide field strength (SNR)

Some popular PIC codes

XOOPIC (2D RPIC, free unix version, Mac and Windows are paid through Tech-X);

VORPAL (1,2,3D RPIC, hybrid, sold by Tech-X)

TRISTAN (public serial version), 3D RPIC (also have 2D), becoming public now

OSIRIS (UCLA) 3D RPIC, mainly used for plasma accelerator research

Apar-T, Zeltron.

PIC-on-GPU — open source

LSP -- commercial PIC and hybrid code, used at national labs

VLPL -- laser-plasma code (Pukhov ~2000)

Reconnection research code (UMD, UDelaware)

Every national lab has PIC codes. (VPIC at Los Alamos)

All are tuned for different problems, and sometimes use different formulations (e.g. vector potential vs fields, etc). Direct comparison is rarely done.

Hands on demonstration using OSIRIS

Please go to: https://jupyter.picksc.org/

bwinjum@ucla.edu

Gkeyll Simulation Framework

What does a Langmuir wave simulation look like in continuum Vlasov?

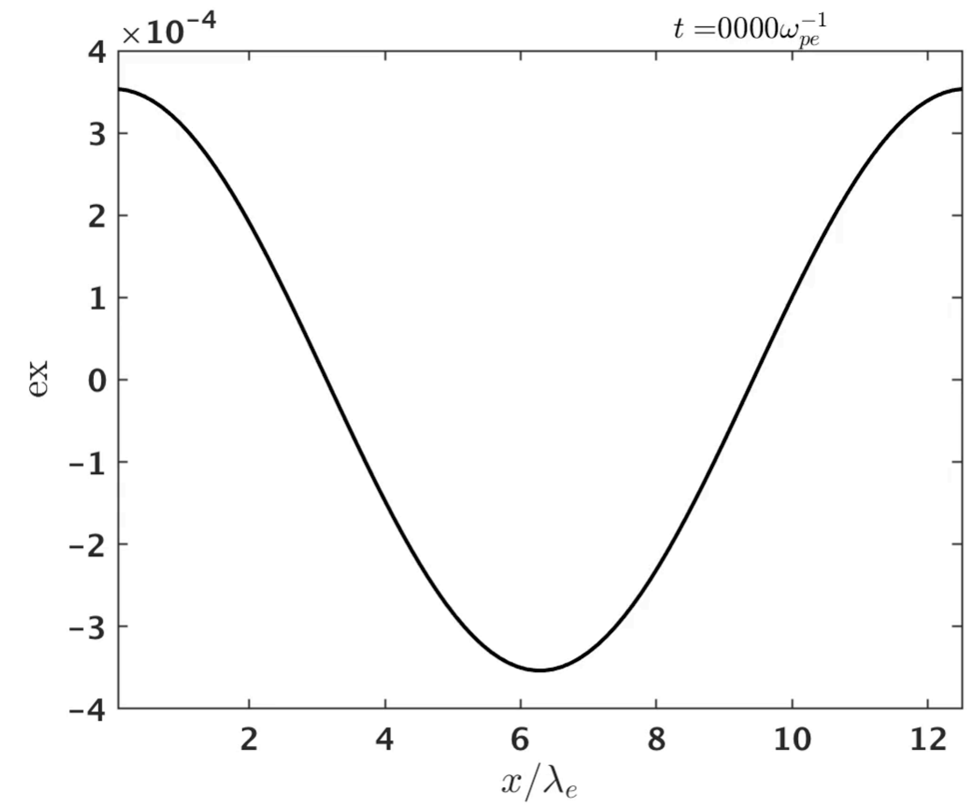
Let's look at a continuum Vlasov simulation of a single Langmuir wave using the Gkeyll (https://gkyl.readthedocs.io/) simulation code

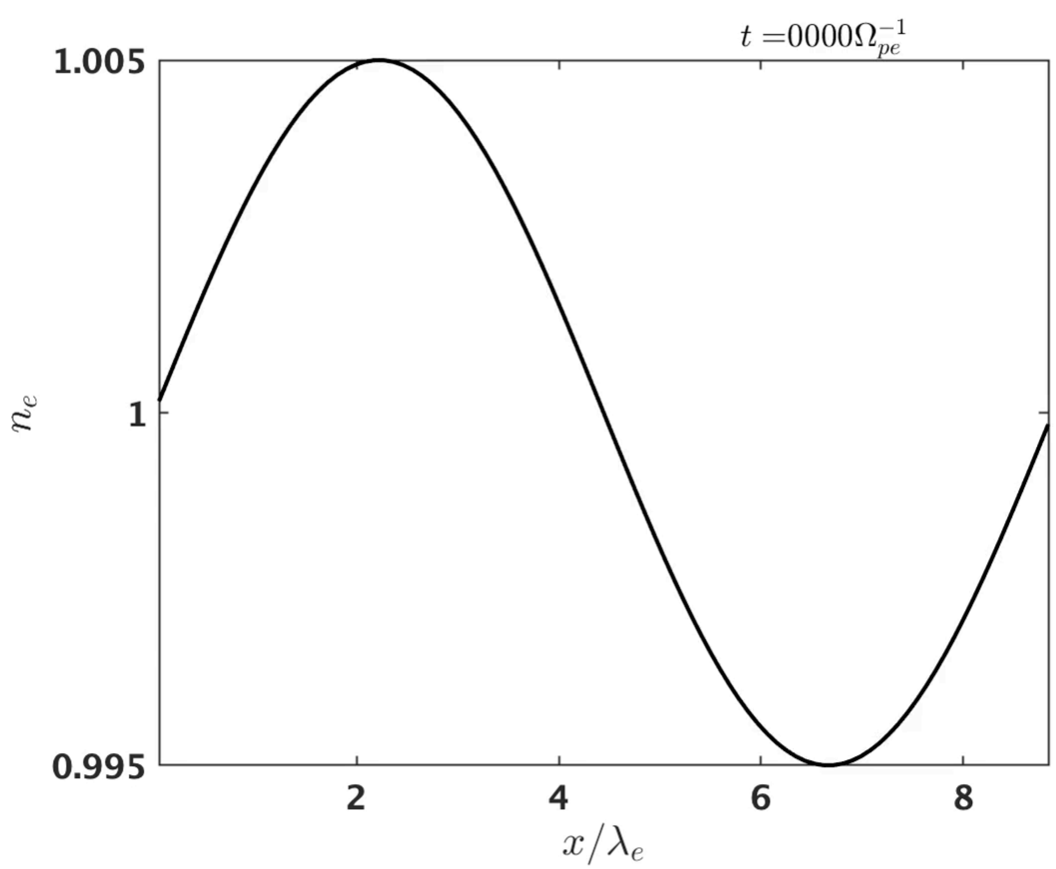
$$\delta n/n_0 = 0.005$$

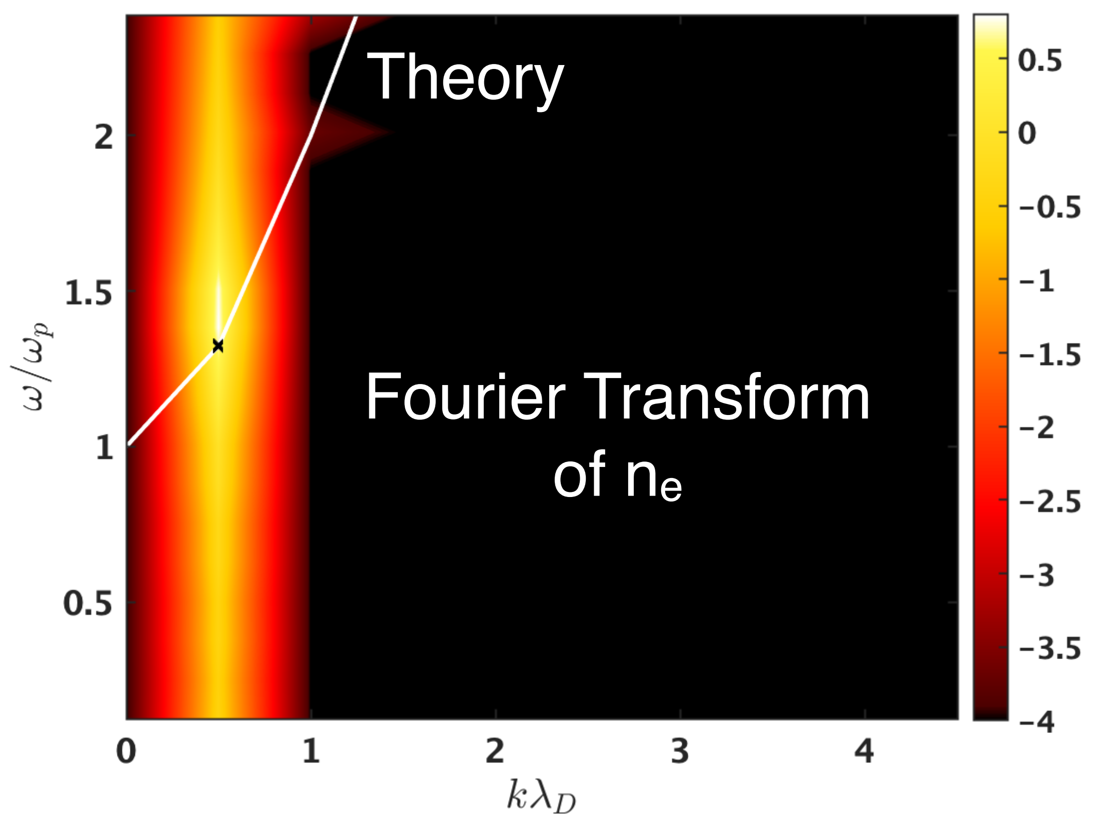
$$(n_x, n_v) = (32, 64), p = 2$$

$$L_x = 4\pi\lambda_D$$

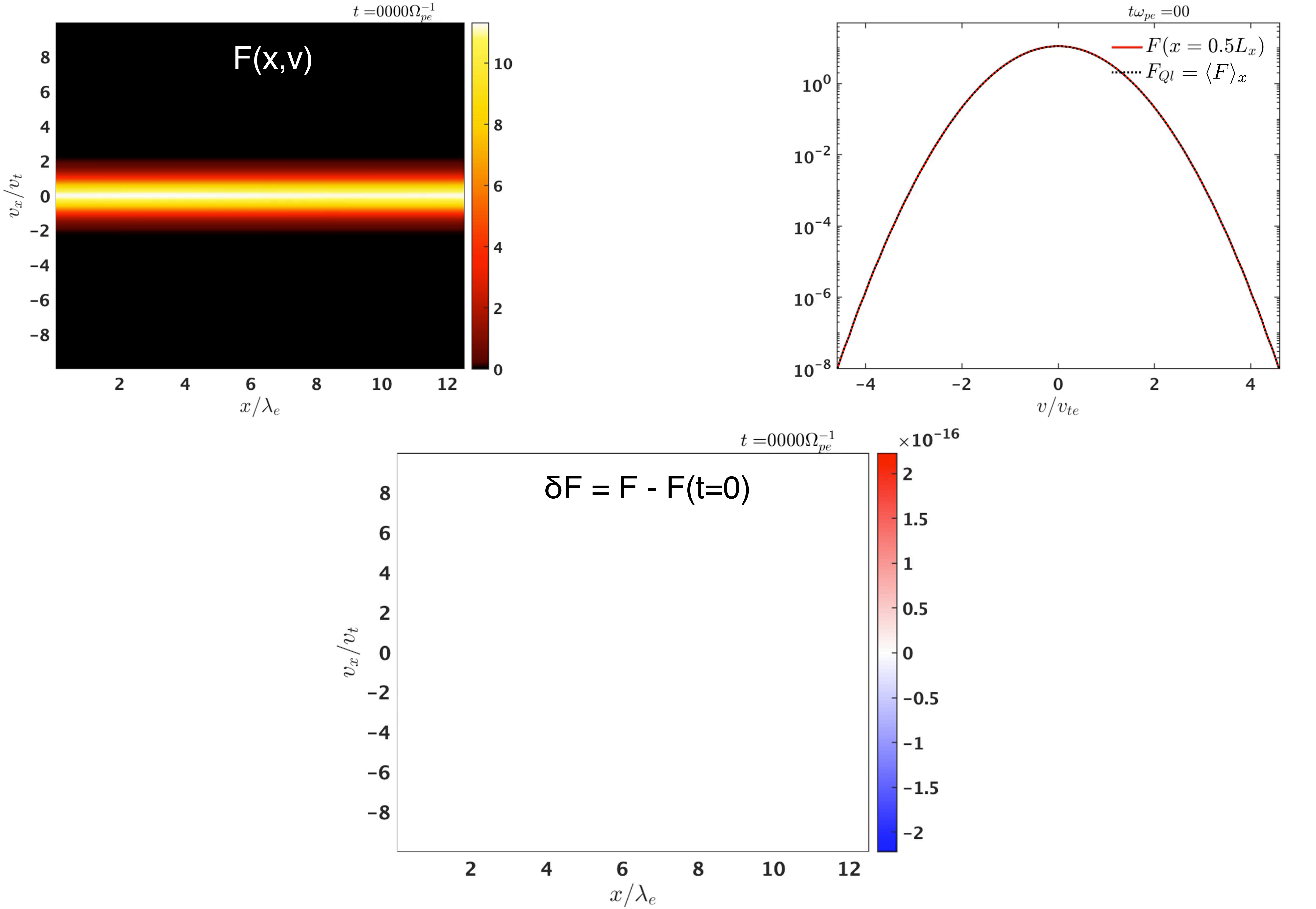
$$-10v_{te} \leq v \leq 10v_{te}$$







How about the distribution function?



Concluding thoughts

- PIC is a powerful and efficient tool for exploring collisionless kinetic physics
- As with any tool, it should be applied with care (black boxes are dangerous!)
- Many improvements continue to be made to PIC, e.g., porting to GPUs/heterogenous architectures, AMR, vectorization, improved sorting and load balancing
- Yet, some problems remain better suited to alternative kinetic approaches, e.g., Eulerian Vlasov or gyrokinetics
- Especially global simulations, where hybrid kinetic or extended MHD models still rule

Thanks for listening and participating!

Special thanks to
Ben Winjum and Matt Kunz