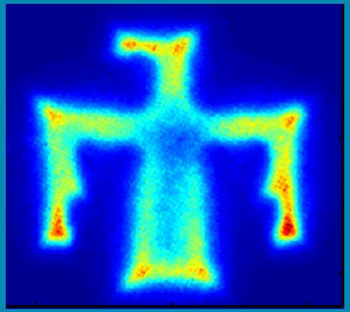
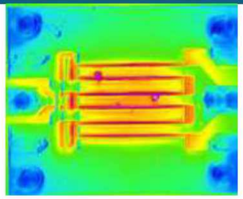
SAND2020-10816C

Particle Methods for Revealing Kinetic Plasma Behavior





PRESENTED BY

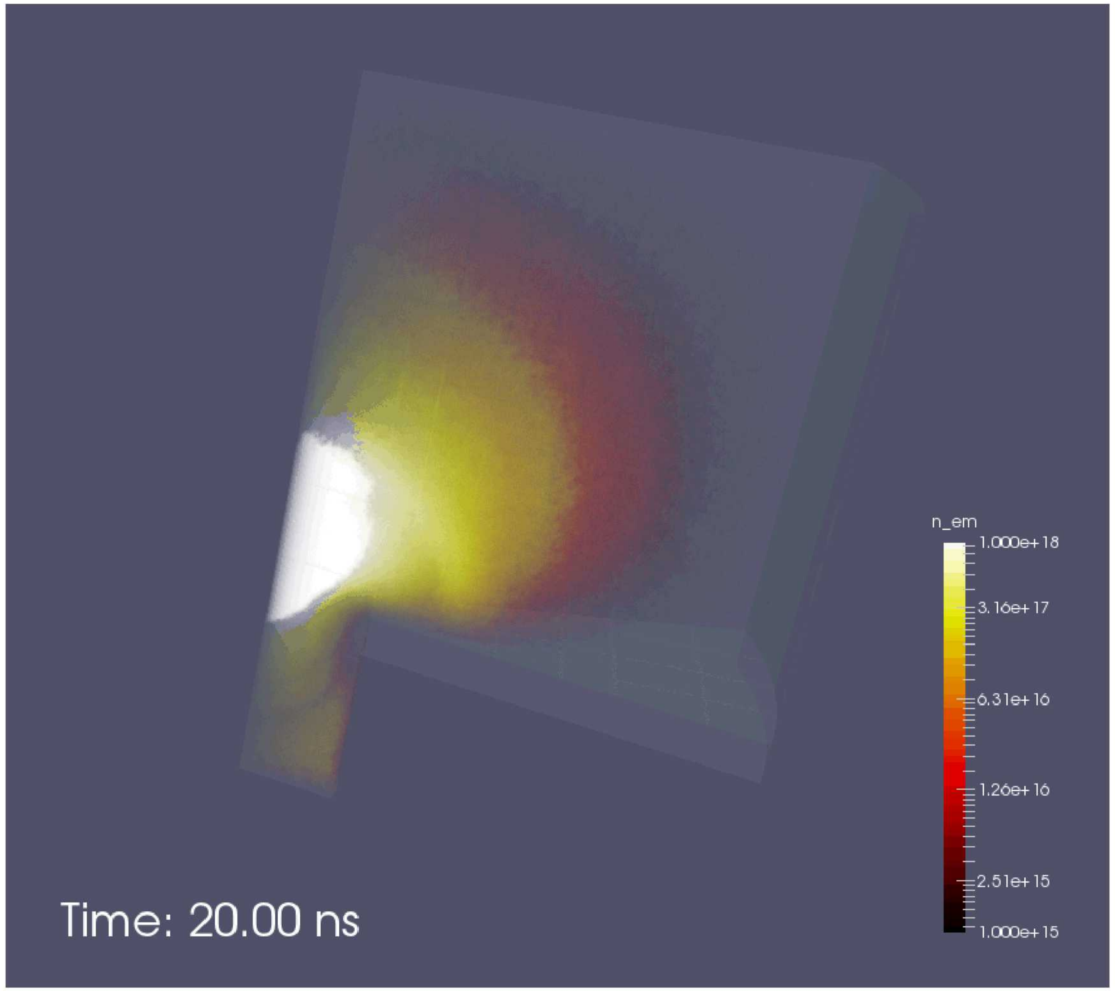
B.T. Yee, M.M. Hopkins

Outline

- Motivation
- Physical Models
- Numerical Methods

  Break

- A Simple PIC code
- Application: Streaming Instabilities

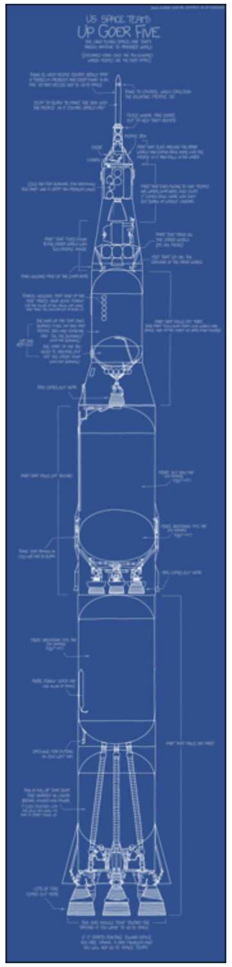


Background and Motivation

What use is computer modeling?

(an incomplete list)

- To test physical assumptions
- Explore complex behavior unamenable to theory
- In the design and engineering of systems
- Identifying failure mechanisms
- Learn physical intuition



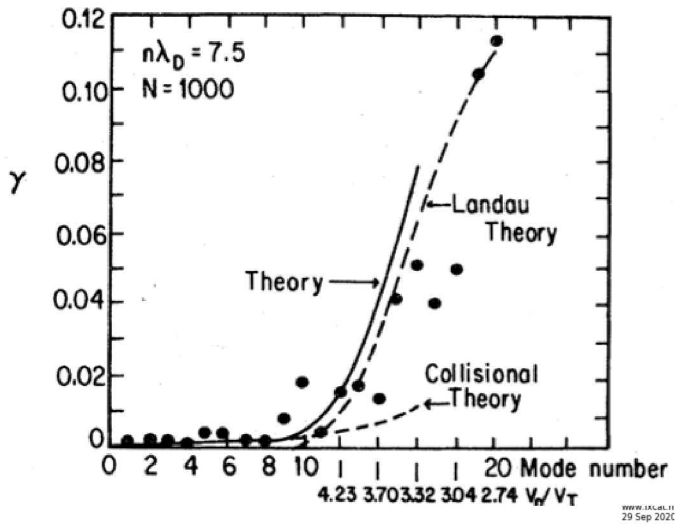
"Spot the cow.gif" by Nepluno is licensed under CC BY-SA 4.0

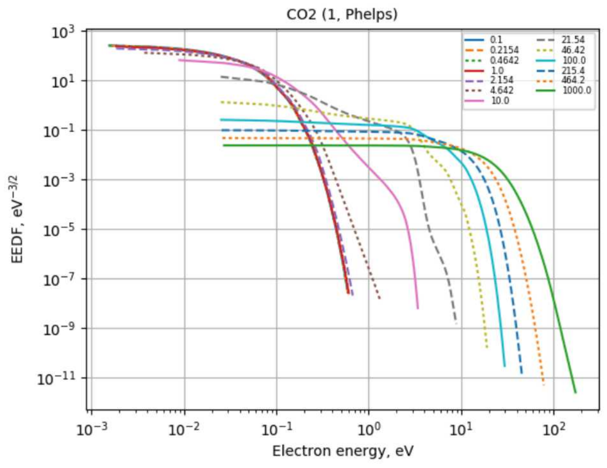
"xkcd: Up Goer Five" by Randall Munroe is licensed under CC BY-NC 2.5

"The Great Wave off Kanagawa" by Katsushika Hokusai, public domain

Why use a kinetic description?

- Capture behavior not represented by fluids
  - Wave-particle interactions
  - Runaway electrons
  - Nonequilibrium excitation
- Explore regimes of model validity
- Investigate microscale physics
- Not because it's fast





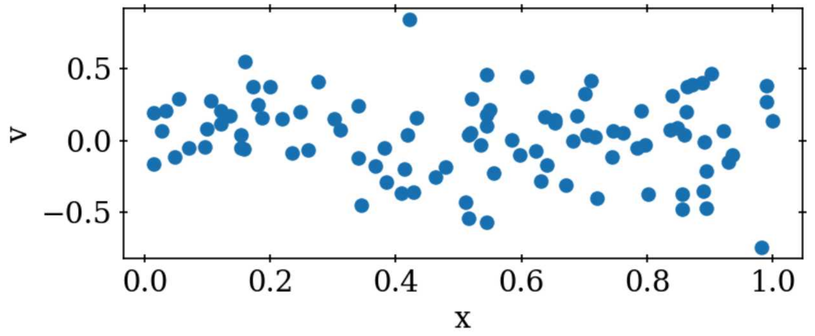
Dawson, Rev. Mod. Phys. **55**, 403-447 (1983).

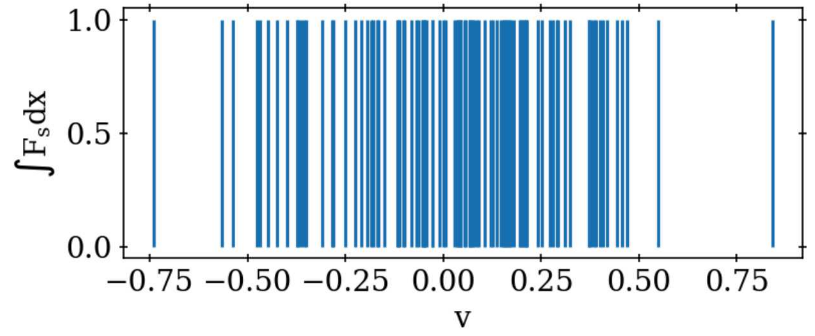
Physical Models

The Klimontovich Equation

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

- Each particle's velocity and position as a □ -function in phase-space
- F_s : summation over all N particles in a system
- Given initial conditions, system can be solved for all time
- Implies a "spikey" distribution function



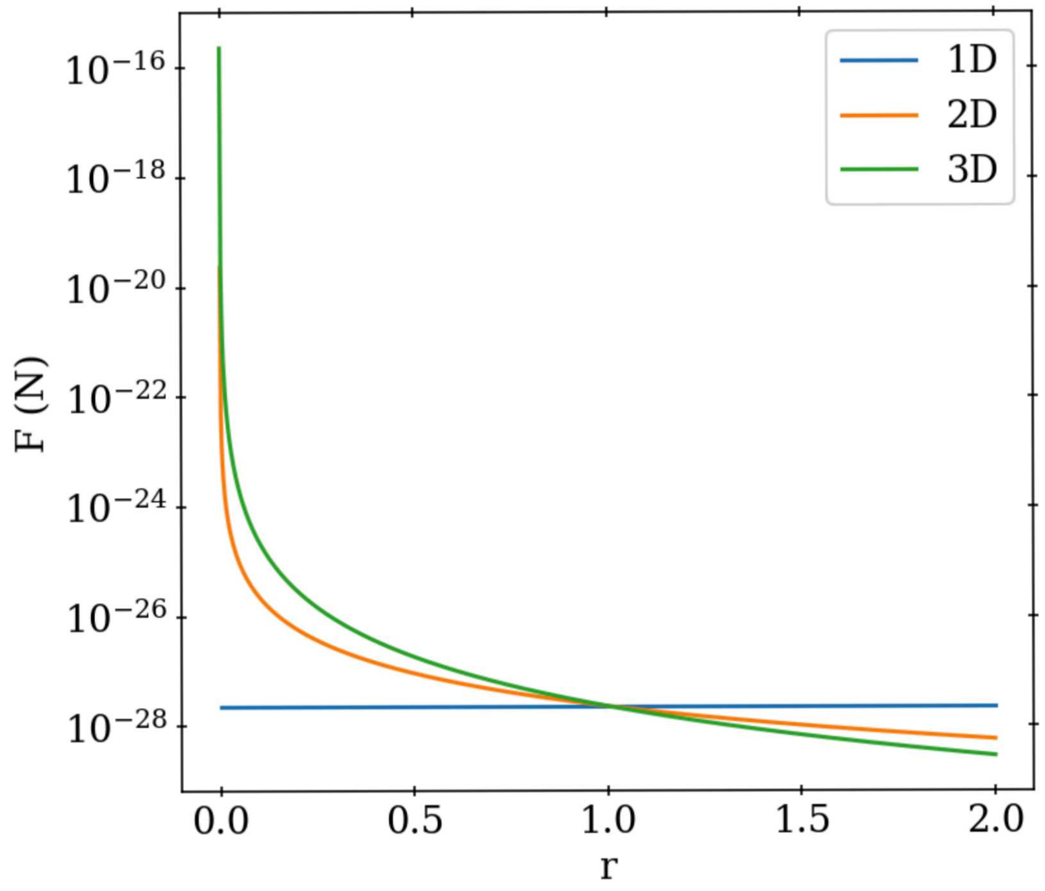


See D.R. Nicholson's "Introduction to Plasma Theory" for more details

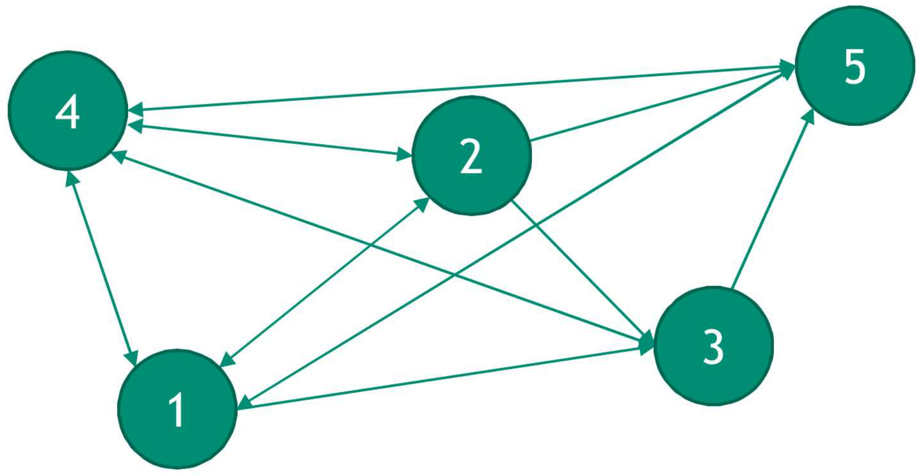
The Poisson Equation

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}, \quad \nabla\mathbf{E} = -\phi \quad \rightarrow \quad \nabla^2\phi = -\frac{\rho}{\epsilon_0}$$

- **B** is assumed negligible or fixed $\mathbf{B}_{\mathbf{ext}}$
- Simple use of Poisson's equation for fields
- Dimensionality important in determining force between particles
- In 1-D (used here) the force does not vary with distance



That’s it, right?



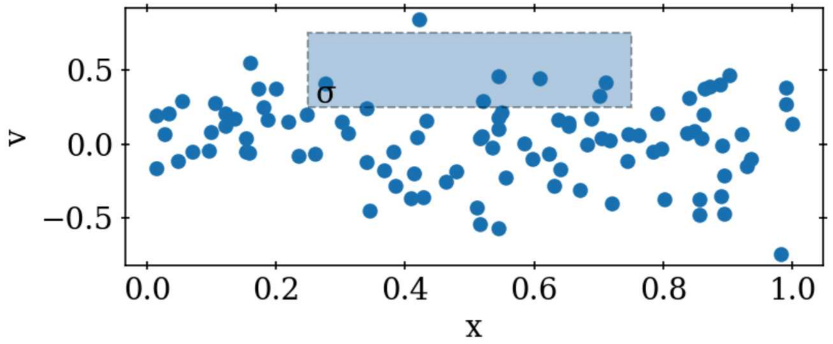
- Can we just solve the Klimontovich equation directly?
  - Set initial particle positions and velocities
  - Calculate forces
  - Move forward in time
  - Repeat!
- This can work…sometimes, mostly impractical
- Number of force calculations is $O(N^2)$

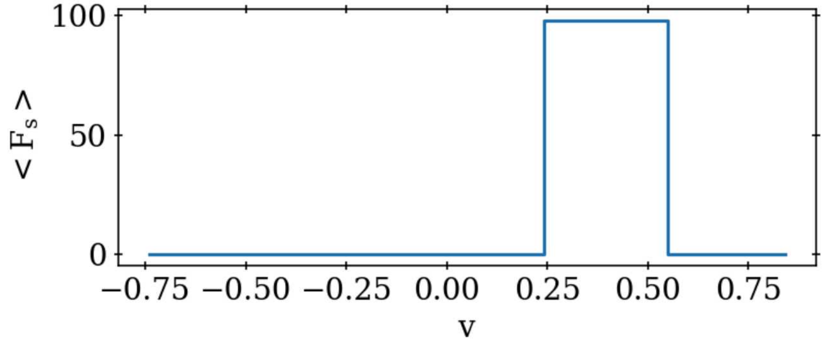
- Consider a typical low temperature plasma
  - Volume ~ 10 cm^3, density 10^{10} cm^{-3}
  - 10^{11} particles in the system
- ~ 10^{22} calculations *per timestep*

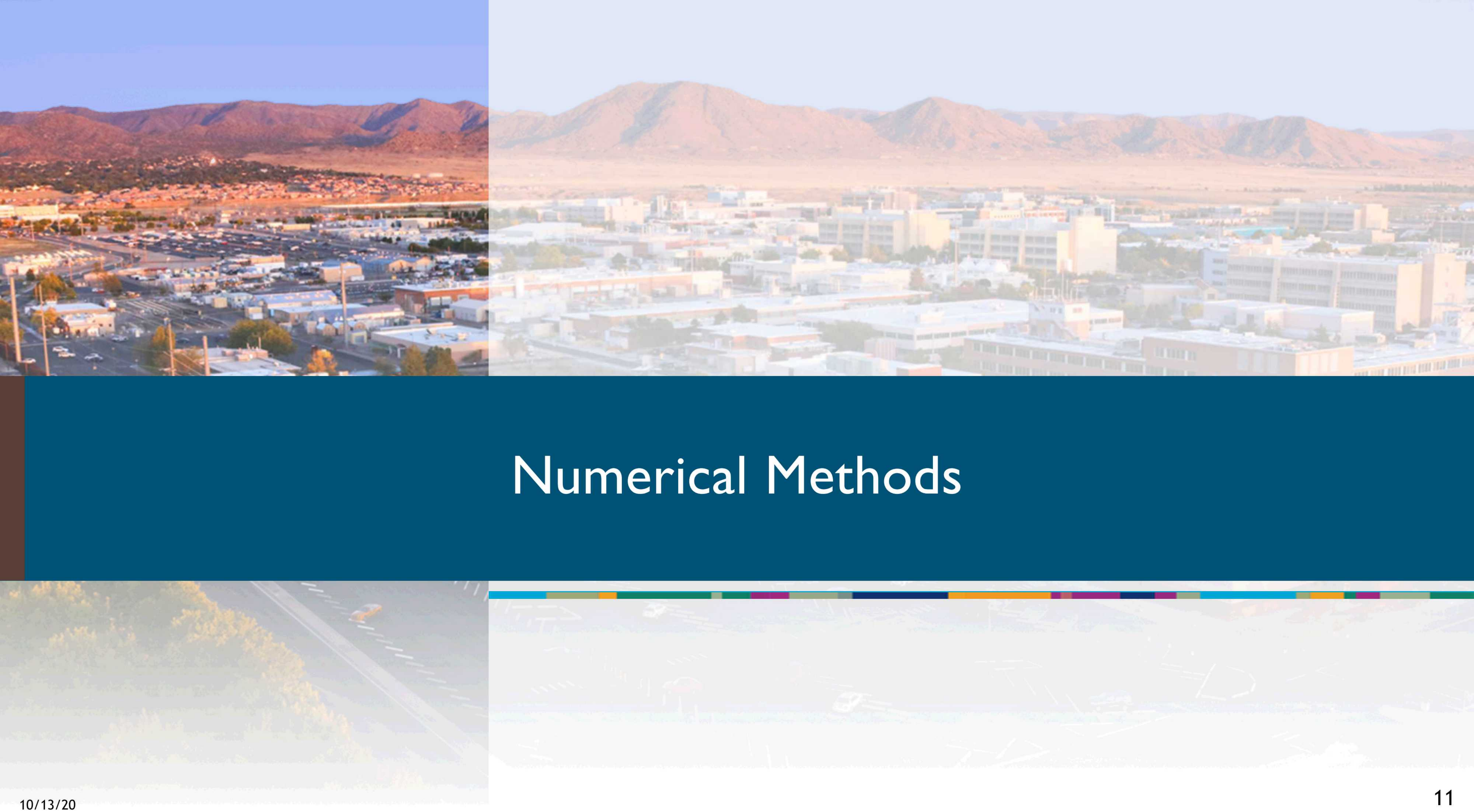
From reduced distribution functions to macroparticles

$$\langle \ldots \rangle \equiv \frac{1}{(nV)^N} \prod_{l=1}^{N_s} \int_\sigma d\mathbf{x}_l d\mathbf{v}_l f_l(\mathbf{v}_l)(\ldots)$$

- In kinetic theory use ensemble averaging to reduce system
- Similarly reduce complexity for modeling
- Segments of phase space → single "macroparticle"
- Spatial scale set by range of electrostatic interactions (λ_D)
- After all that, just *F*=*ma*. Really.



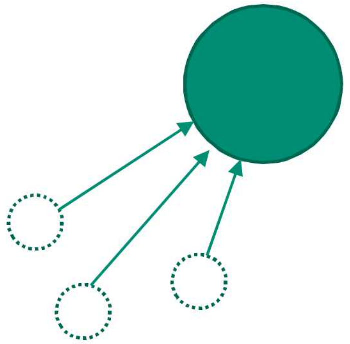


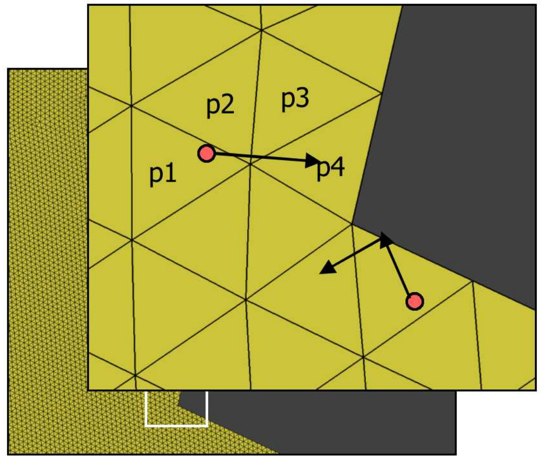


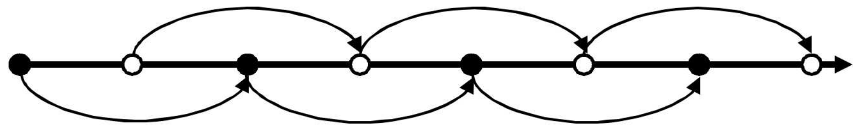
Numerical Methods

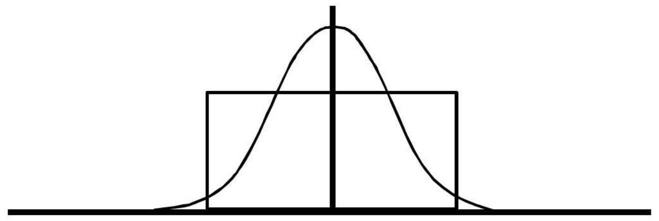
Particle-in-Cell Basics

- **Macroparticles** – discrete computational particles representing ensembles
- **Particle Mapping** - interpolating particle charges to the mesh
- **Field Solve** - solving for electric fields on the mesh and interpolating to particles
- **Particle Updates** – advancing particle velocity and positions in time



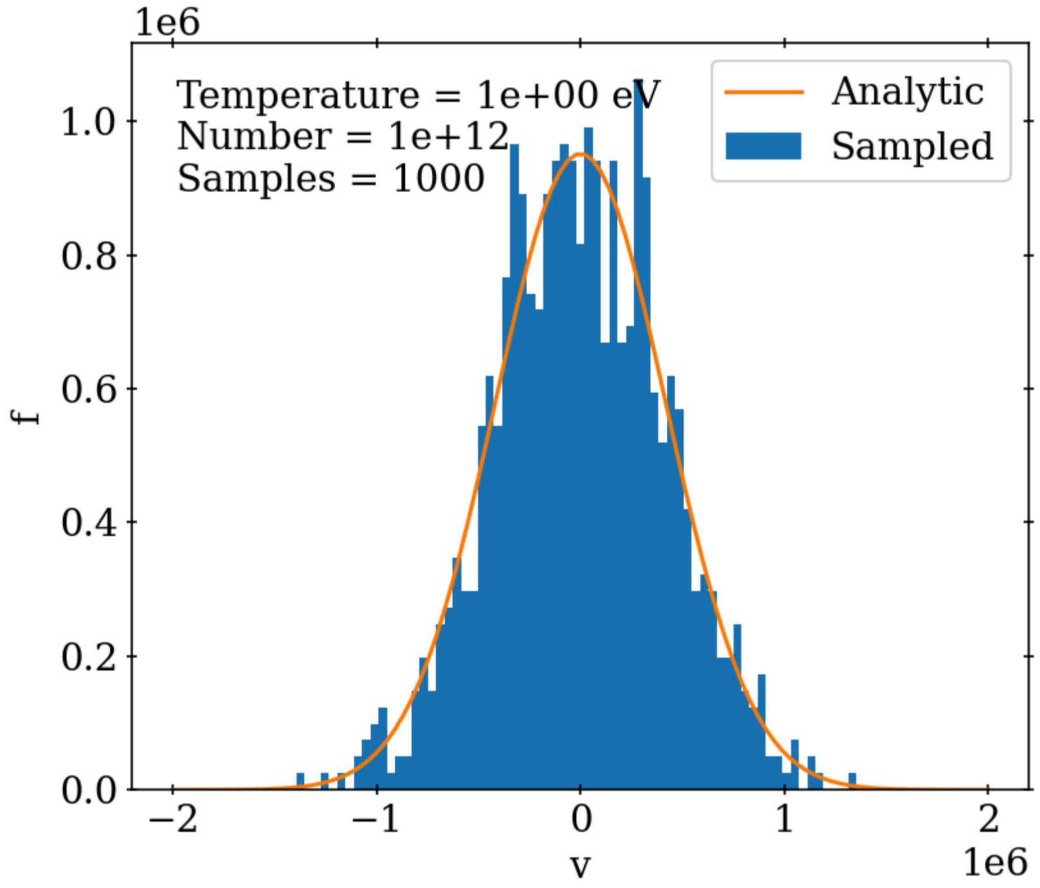






Macroparticles

- AKA, super or notional or computational or representative particles
- Can be thought of as representing some number of physical particles[1]
- Correspondence between physical and computational particle given by "weight"
- Each macroparticle possesses its own
  - Position, velocity, mass, charge, …
- Initialize particles by sampling from proscribed distributions
- Number needed set by resolution of f

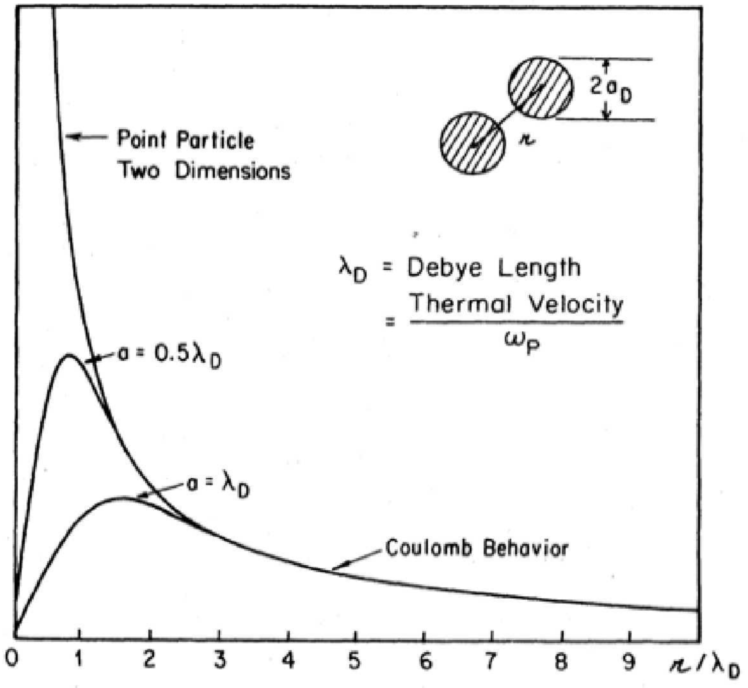


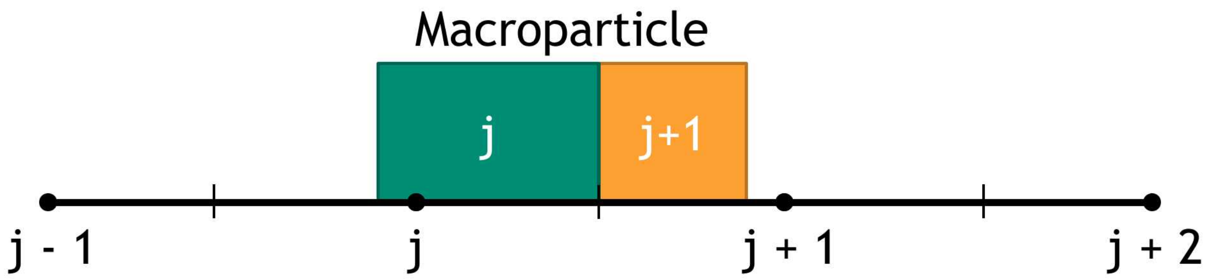
[1]Not strictly the case, they typically have altered interaction potentials and a finite (as opposed to singular) spatial extent. Morse and Nielson (1969) suggest that instead of considering them "very large real particles" to treat them as "random Lagrangian mesh points imbedded in a collisionless phase fluid," see previous discussion of ensemble average.

Particle Mapping

- Macroparticles convert 10^{11} particles to 10^5 still O(10^{10}) calculations per timestep
- Use particle-grid mapping to get $O(N^2) \rightarrow O(N)$
- Implies finite particle size (recall reduced distribution)
- Particle "shapes" determine how charge is distributed to grid
- Influences particle-particle collisions and noise

$$\rho_j = \sum_{|x_i - X_j| < \Delta X} w_i q_i \left(1 - \frac{x_i - X_j}{\Delta X}\right)$$



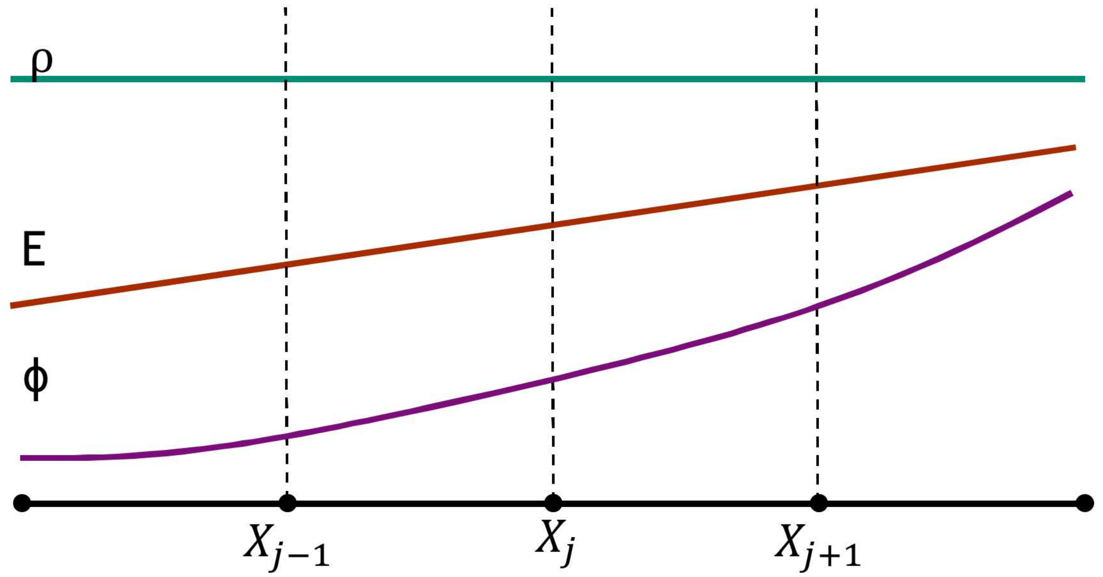


Dawson, Rev. Mod. Phys. **55**, 403-447 (1983).

Electric Field Solution

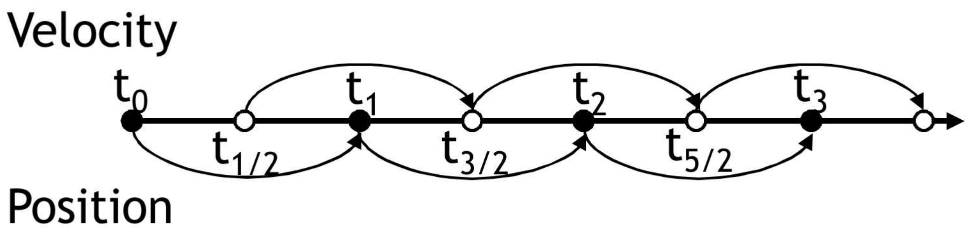
$$\frac{d^2}{dx^2}\phi(x) = -\frac{\rho}{\epsilon_0} \approx \frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{\delta x^2} \rightarrow \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & 0 \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{N-2} \\ \phi_{N-1} \end{bmatrix} = -\frac{\Delta x^2}{\epsilon_0} \begin{bmatrix} \rho_1 - V_0 \\ \rho_2 \\ \rho_3 \\ \vdots \\ \rho_{N-2} \\ \rho_{N-1} - V_L \end{bmatrix}$$

- Use Poisson's equation to solve for $\mathbf{E}$[1]
- Calculate ∇^2 with a three-point stencil
- Form tridiagonal matrix for entire grid ($A \cdot x = b$)
- Get the inverse of A however you like (Gaussian elimination, Cholesky decomposition, …)
- Multiply by charge vector to find ϕ



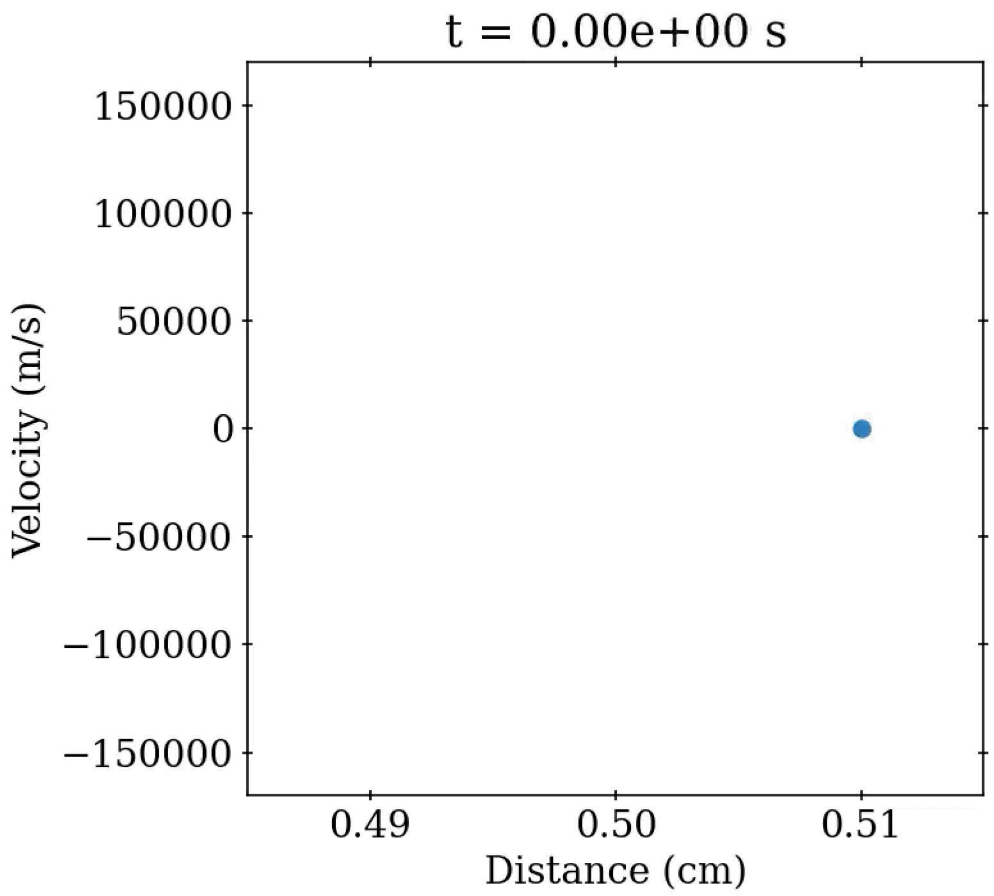
[1] Other approaches are both possible and common. A particularly fast method is by solving the problem in Fourier space by means of FFTs. It is also possible to simply integrate the charge density to obtain E. Meanwhile, electromagnetic codes must approach the problem from a different angle entirely.

Particle Updates



- Advance particles with leapfrog algorithm
- Calculated for each particle independently
- Fast with many desirable properties:

1. $v_{n+1/2} = v_n + \Delta t/2\ (E_n q/m)$
2. $x_{n+1} = x_n + v_{n+1/2} \Delta t$
3. Field update (see previous)
4. $v_{n+1} = v_{n+1/2} + \Delta t/2\ (E_{n+1} q/m)$



Solution Constraints

For a valid solution, must meet three basic constraints

1. **Resolve Debye Length ($\Delta x < \lambda_D$)**
   - Properly resolve shielding effects
   - Avoid grid heating and instability
2. **Resolve Plasma Frequency ($\Delta t < 1/f_p$)**
   - Capture plasma oscillations and electron dynamics
3. **Meet CFL condition ($\Delta x/\Delta t > v_{max}$)[1]**
   - Necessary for convergence
   - 'Speed of light' for an explicit finite difference scheme

These are necessary but not sufficient.

[1] It's perhaps instructive to note that the first two requirements are equivalent to the CFL condition if v_{max} is taken to be the electron thermal velocity. The electron thermal velocity is often an appropriate choice for the fastest characteristic velocity in a system.

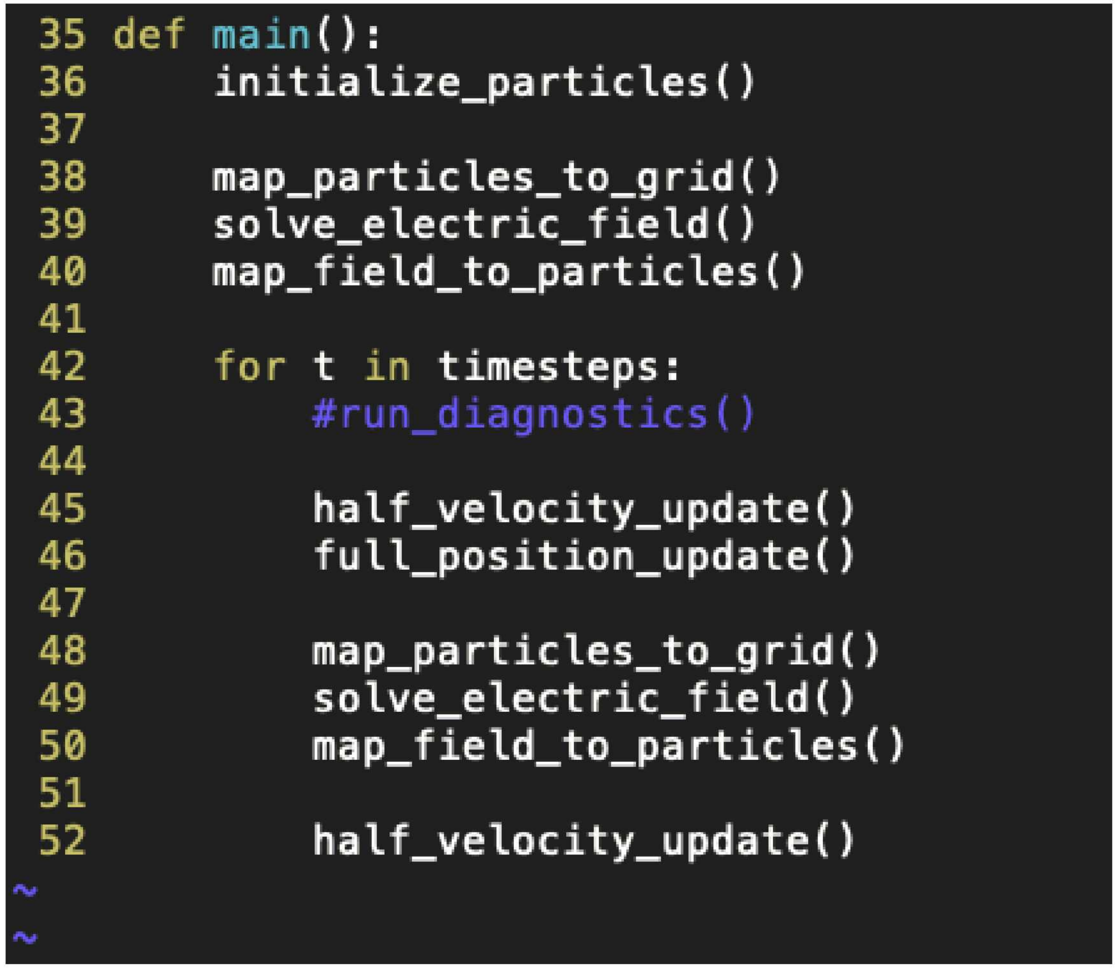
Intermission! (Take 5)

A Simple PIC Code

Anatomy of a PIC code

Most PIC codes have this structure:

1. Initialize particles
2. Solve fields
3. Update velocity by dt/2
4. Update position
5. Solve fields
6. Update velocity by dt/2
7. Goto (3)



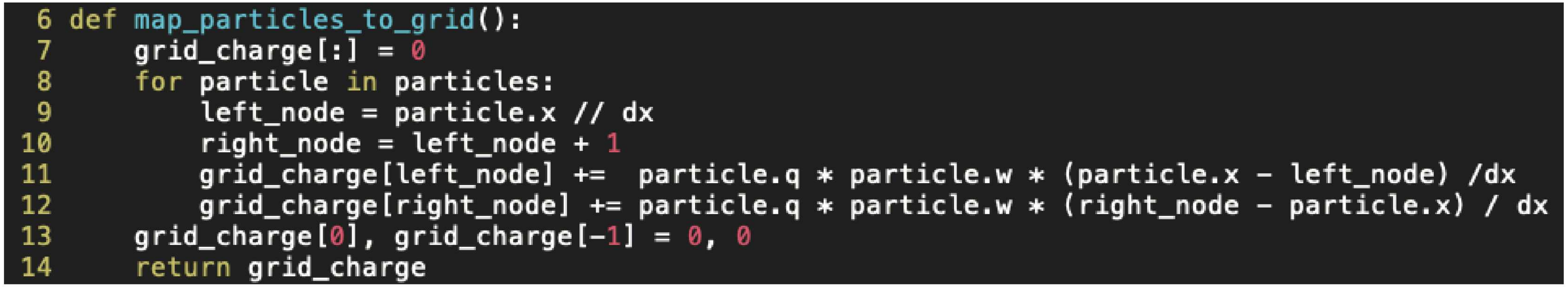
```
def main():
    initialize_particles()

    map_particles_to_grid()
    solve_electric_field()
    map_field_to_particles()

    for t in timesteps:
        #run_diagnostics()

        half_velocity_update()
        full_position_update()

        map_particles_to_grid()
        solve_electric_field()
        map_field_to_particles()

        half_velocity_update()
~
~
```

Particle Mapping

For each particle:

1. Find its member cell
2. Calculate fraction to assign to left and right nodes
3. Adjust for particle charge, weight, and cell size
4. Repeat for all particles

$$\rho_j = \sum_{|x_i - X_j| < \Delta X} w_i q_i \left(1 - \frac{x_i - X_j}{\Delta X}\right)$$



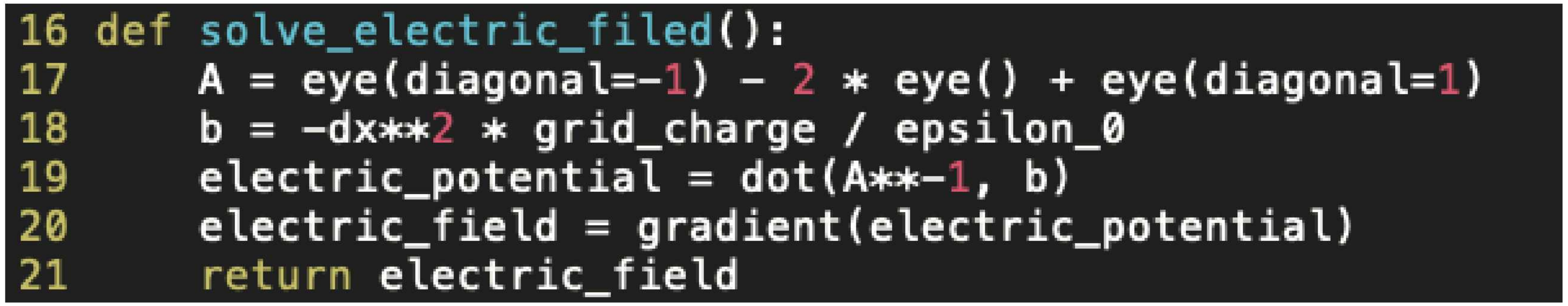
```python
def map_particles_to_grid():
    grid_charge[:] = 0
    for particle in particles:
        left_node = particle.x // dx
        right_node = left_node + 1
        grid_charge[left_node] +=  particle.q * particle.w * (particle.x - left_node) /dx
        grid_charge[right_node] += particle.q * particle.w * (right_node - particle.x) / dx
    grid_charge[0], grid_charge[-1] = 0, 0
    return grid_charge
```

Field Solutions

To update the electric field:

1. Form the stencil and take its inverse[1]
2. Calculate the grid charge (see previous)
3. Dot the two together to get the potential
4. Take the gradient to obtain the field

$$\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & 0 \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{N-2} \\ \phi_{N-1} \end{bmatrix} = -\frac{\Delta x^2}{\epsilon_0} \begin{bmatrix} \rho_1 - V_0 \\ \rho_2 \\ \rho_3 \\ \vdots \\ \rho_{N-2} \\ \rho_{N-1} - V_L \end{bmatrix}$$


```
def solve_electric_filed():
    A = eye(diagonal=-1) - 2 * eye() + eye(diagonal=1)
    b = -dx**2 * grid_charge / epsilon_0
    electric_potential = dot(A**-1, b)
    electric_field = gradient(electric_potential)
    return electric_field
```

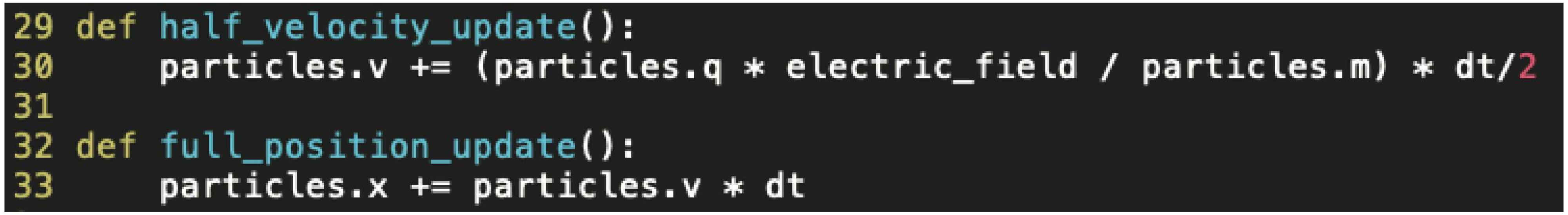
[1] For our purposes, the inverse can be calculated once before the time iteration and used for the entire simulation. This is often not possible for large systems.

Position and Velocity Updates

- Particle velocity and position updates staggered
- Velocity update based on electric field interpolated to particle
- Interpolation should match particle shape function
- When writing output, should ensure velocity and position are aligned

$$v_{n+1/2,i} = v_{n,i} + \frac{\Delta t}{2}\left(\frac{E_{n,i} q_i}{m_i}\right)$$

$$x_{n+1,i} = x_{n,i} + v_{n+1/2,i}\Delta t$$



```python
def half_velocity_update():
    particles.v += (particles.q * electric_field / particles.m) * dt/2

def full_position_update():
    particles.x += particles.v * dt
```

SIMPIC: An Introduction

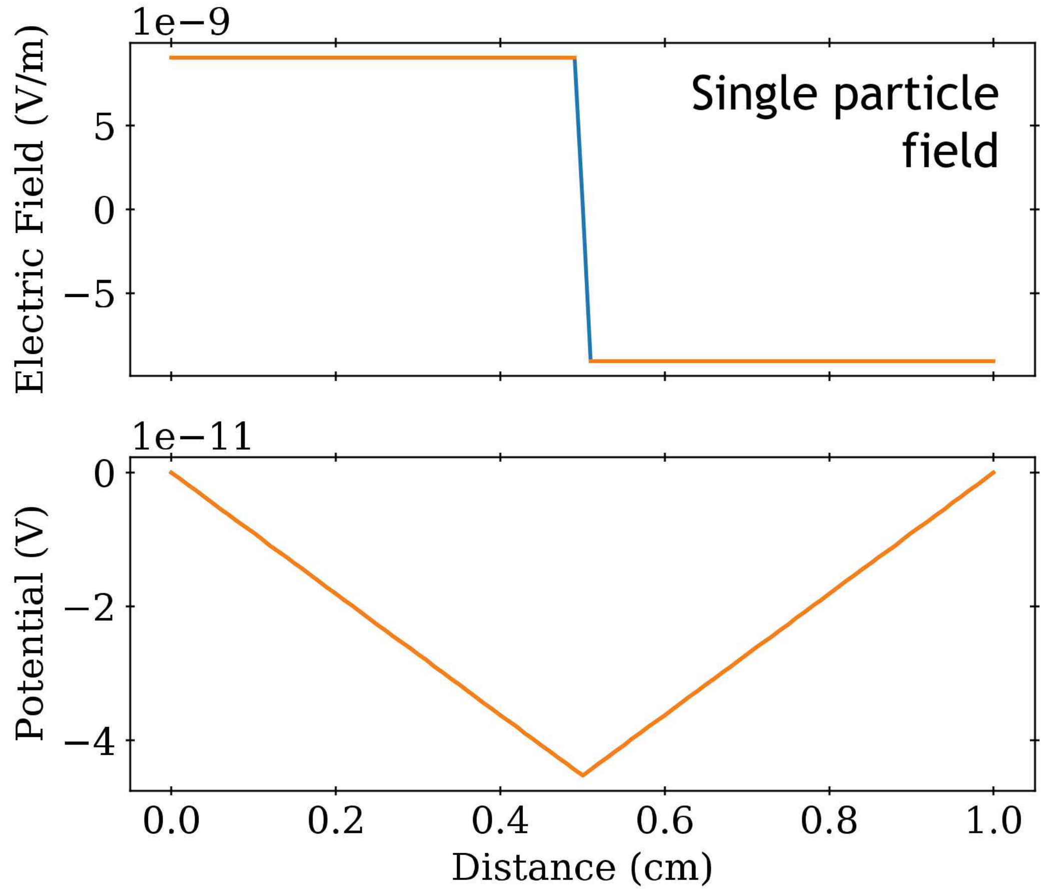
https://gitlab.com/btyee/simpic/

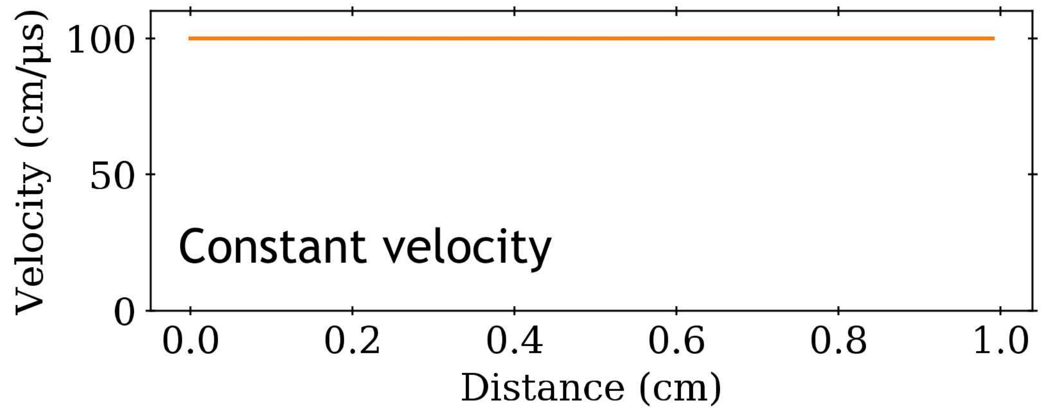
A **sim**ple **PIC** code written specifically for this tutorial

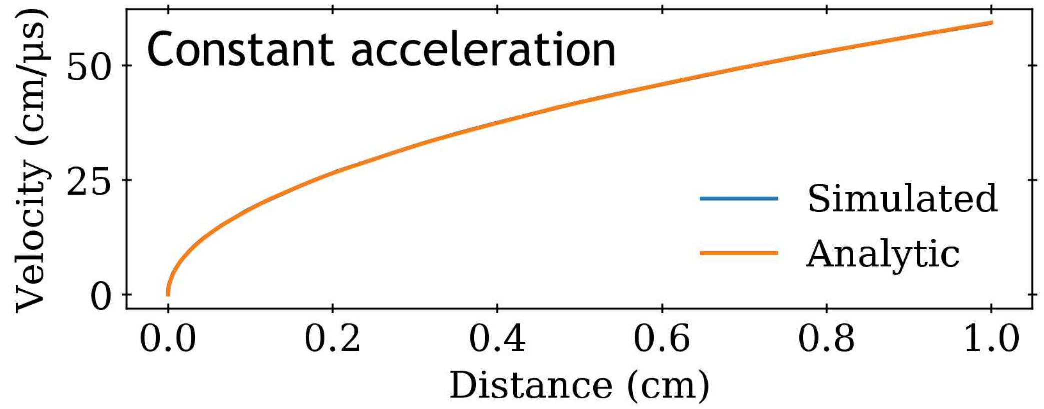
- 1D, bounded
- Electrostatic
- Non-relativistic
- Leap-frog update
- Particle resampling on exit
- Particle heating
- Variety of spatial/velocity sampling methods
- Single core

- Fast enough
- Concise, readable, heavily commented
- Portable to many systems with few dependencies (Python 3+, numpy 1.17+)
- Simple input deck, easily extendable
- Freely available

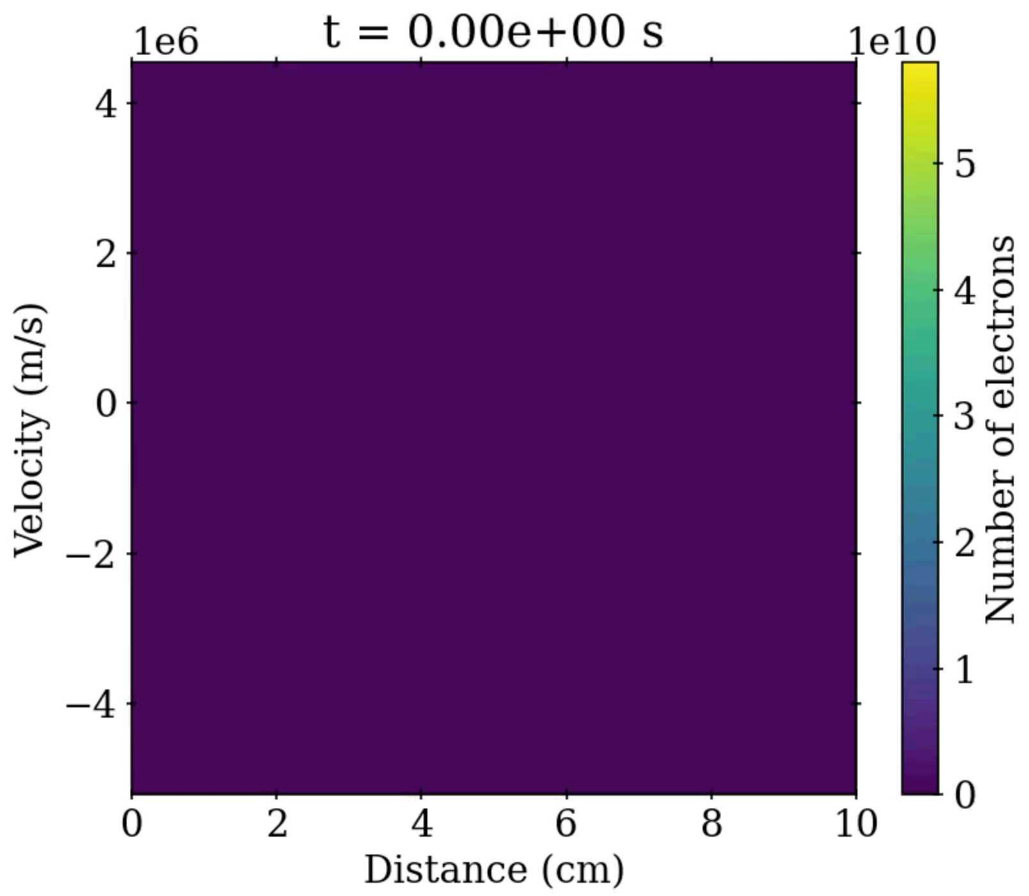
Basic verification and validation

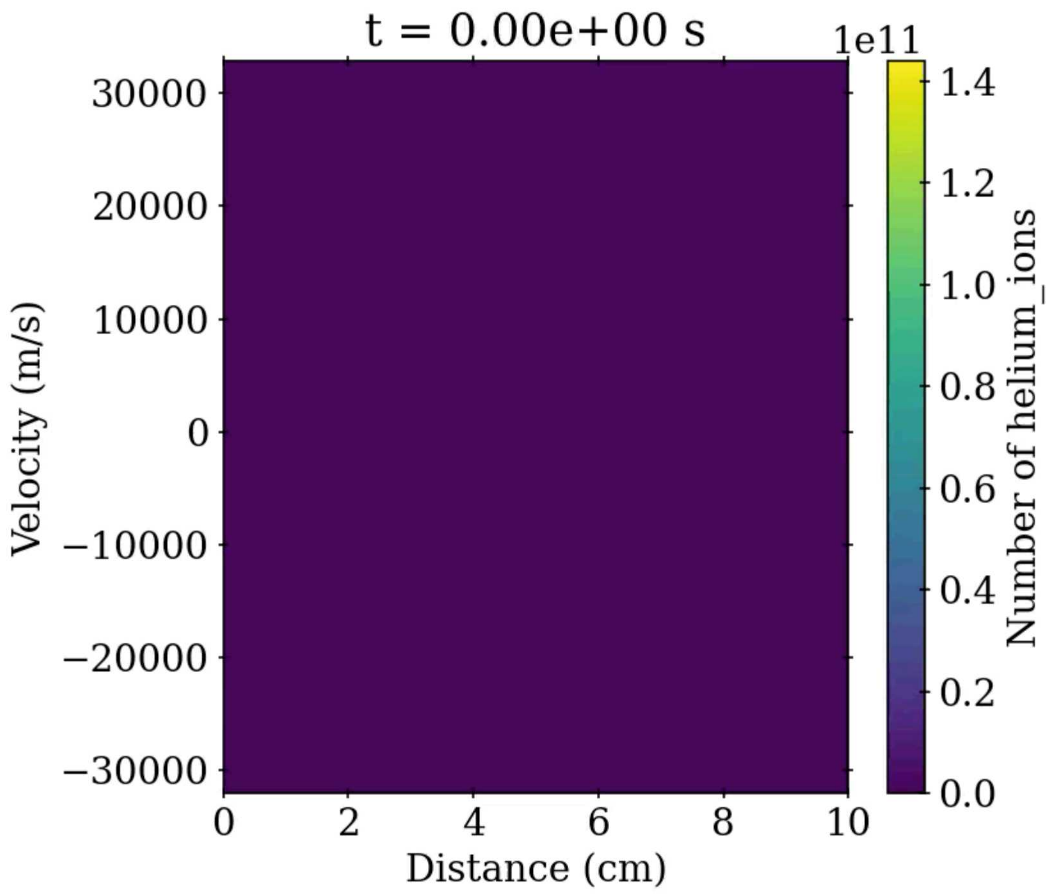






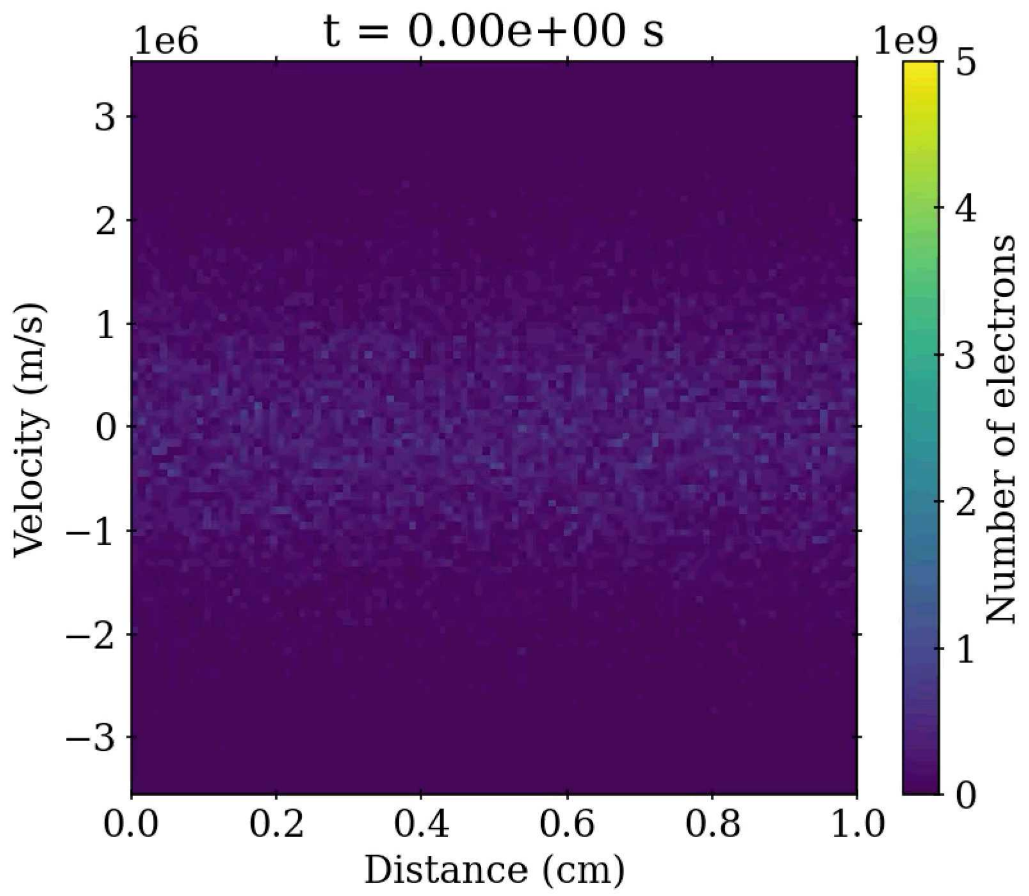
Matching Fluid Results: Stable Sheath

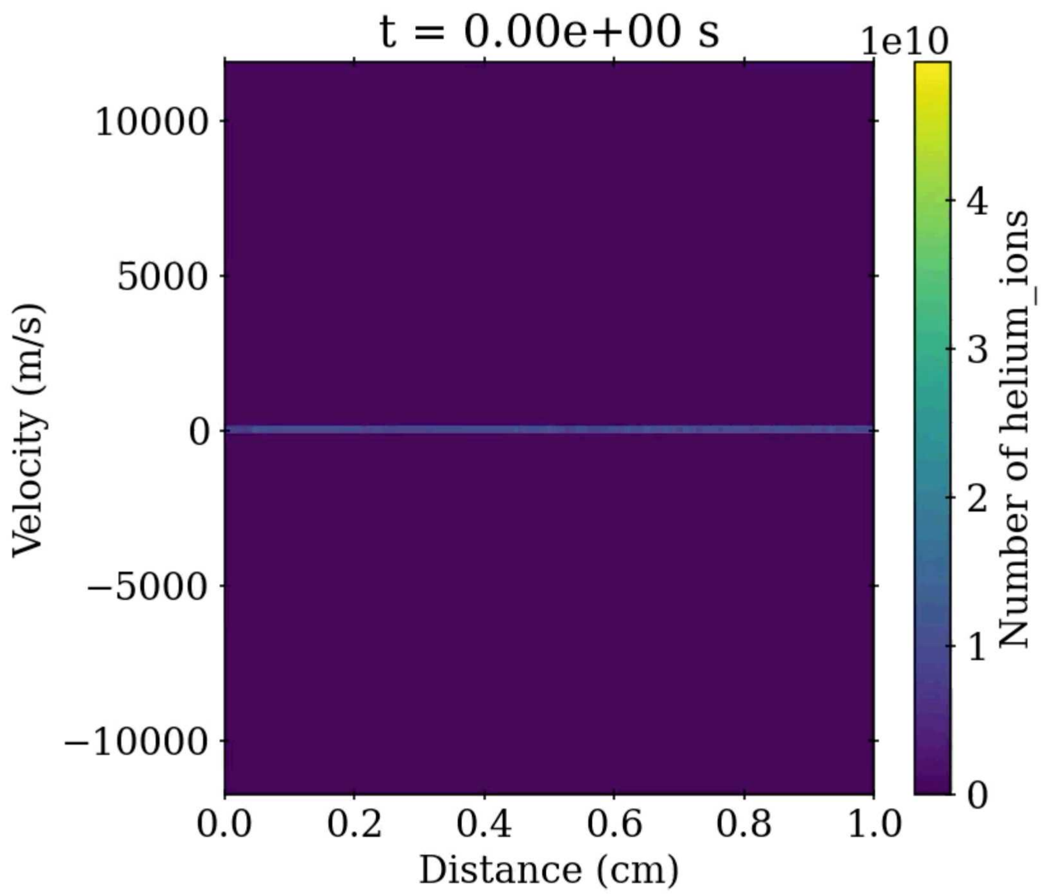




This solution is only marginally stable for the given electron-ion temperature ratio. See: Baalrud, Plasma Sources Sci. Technol. **25**, 025008 (2016).

Uniquely Kinetic Behavior: Tonks-Langmuir Plasma



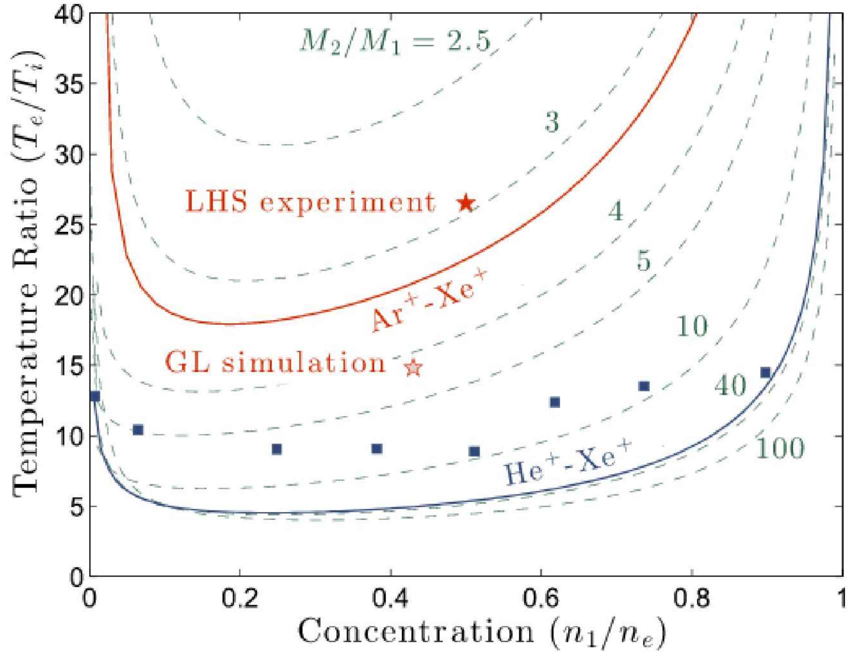


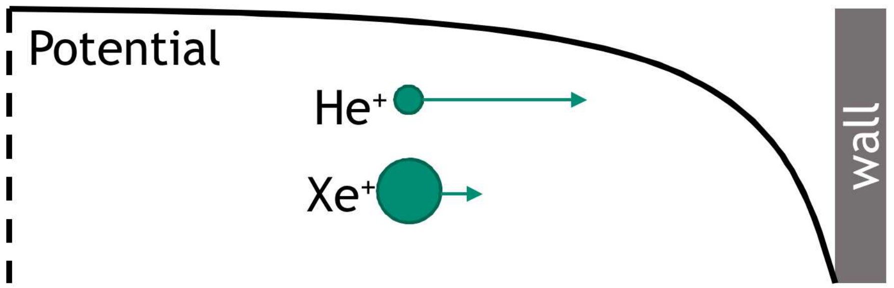
Harrison, Thompson, Proc. Phys. Soc. **74**, 145-152 (1959).

Application: Streaming Instabilities

Problem Description

- Plasma bound between two equipotential electrodes
- Composed of electrons, helium ions, and xenon ions
- Bohm criterion states $c_s = \sqrt{kT_e/m_i}$
- Ions are accelerated in the same field, thus differential flow
- Can lead to instabilities under right conditions





Baalrud et al. Plasma Sources Sci. Technol. **24**, 015034 (2015)

Choosing model parameters: part one

Nominal parameters

- Plasma density: 10^{15} m^{-3} (a “real” density)
- Electron temperature: 4.5 eV (by theory)

$\rightarrow \lambda_D = 500$ um, $f_p = 10$ GHz

System size

- Lower limits:
  - Much larger than the sheath (many λ_D)
  - Long enough presheath for wave growth
- Upper limits:
  - Solution time (field calculations and particle number)

$\rightarrow$ L = 10 cm

Spatial discretization

- Lower limits:
  - Solution time (more cells, longer field solve)
  - CFL condition (balance with timestep)
- Upper limits:
  - Gradients, prevent grid heating ($\sim \lambda_D$)

$\rightarrow \Delta$x = 50 μm

- Diagnostic output
  - Often enough to resolve wave behavior

Choosing model parameters: part two

- Temporal discretization
  - Resolve plasma frequency
  - Satisfy CFL

  → Δt = 40 ps

- Total simulation time
  - Sufficient for steady state solution (several ion transit times)

  → T = 100 μs

- Repopulation rate
  - Determined by steady state flux out of system, approximate with Bohm flux (iterate)

  → R_{iz} = 10^{19} s^{-1} (iterate)

- Particle Temperatures
  - Initial conditions chosen so that T_e/T_i is in the unstable regime
  - Distributions evolve self-consistently, limited control

  → T_e = 4.5 eV, T_i = 0.3 eV

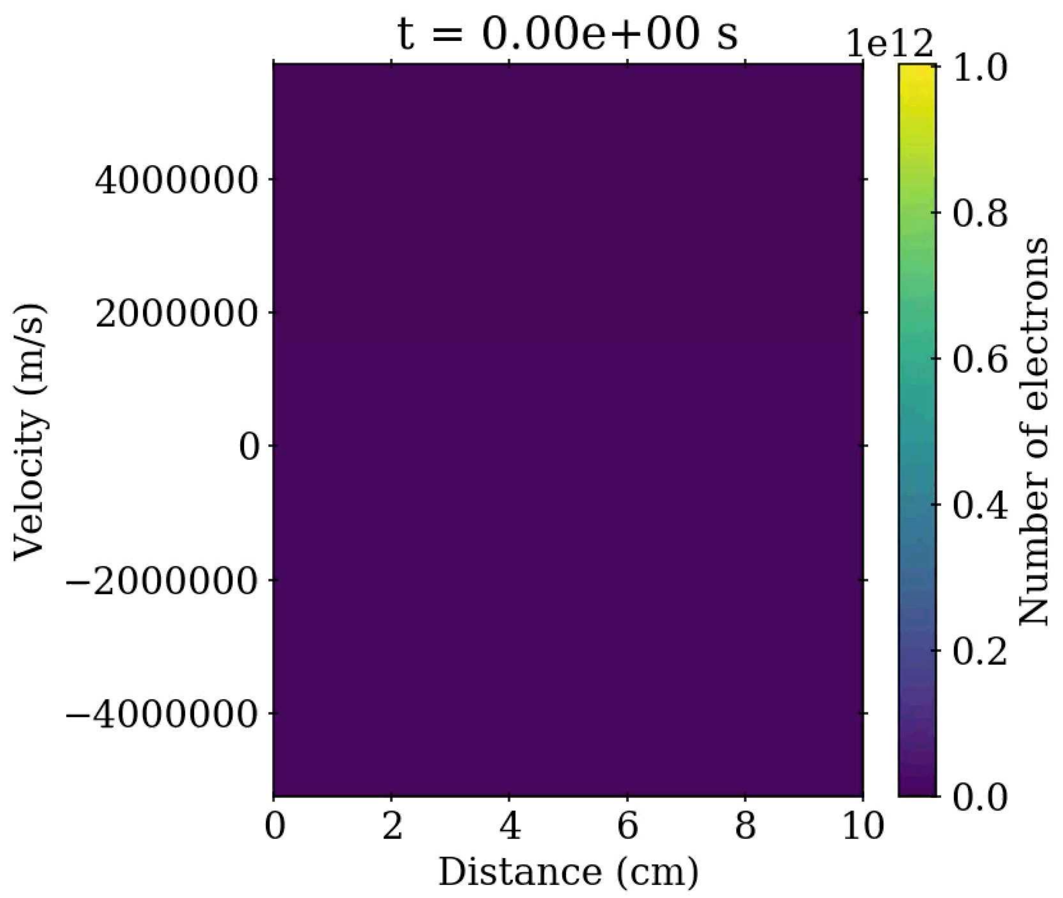
- Heating Rate
  - Keeps electrons from cooling by collection at walls
  - Add through random kicks in velocity space

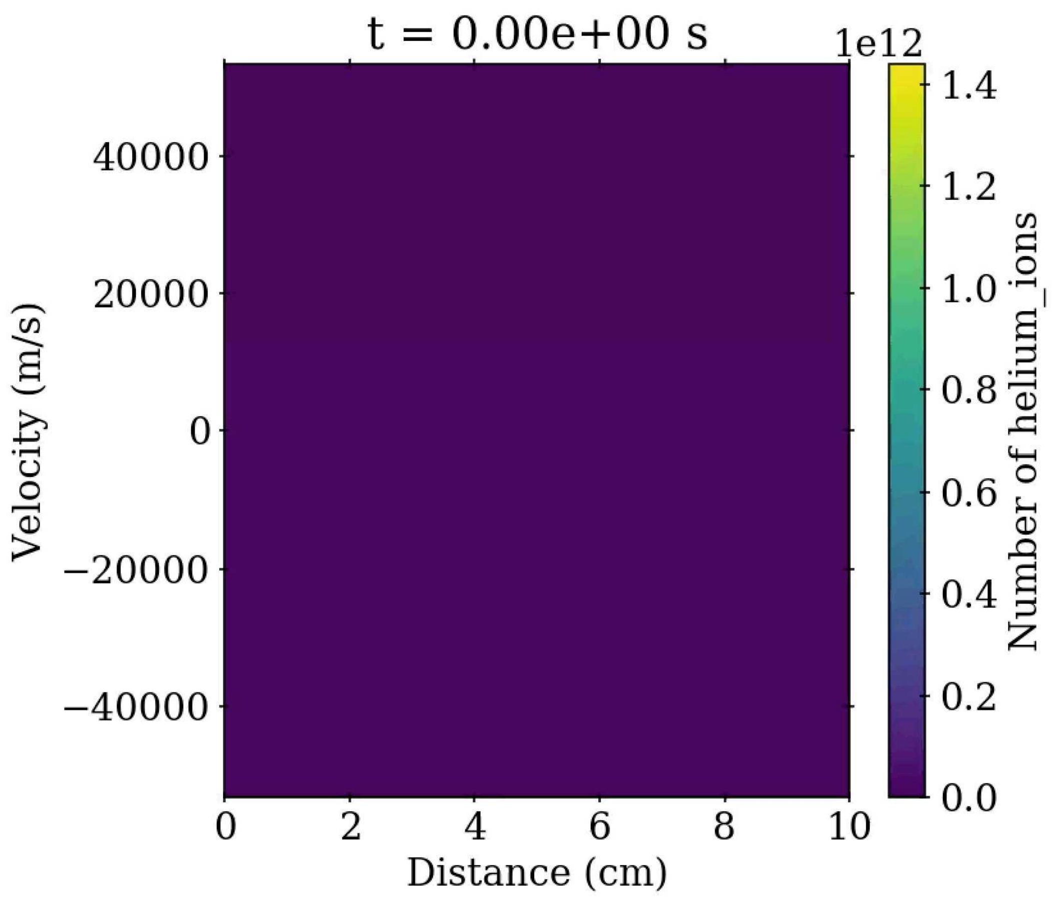
  → ΔP = $5\text{x}10^{-19}$ W (iterate)

- Runtime
  - As much as needed

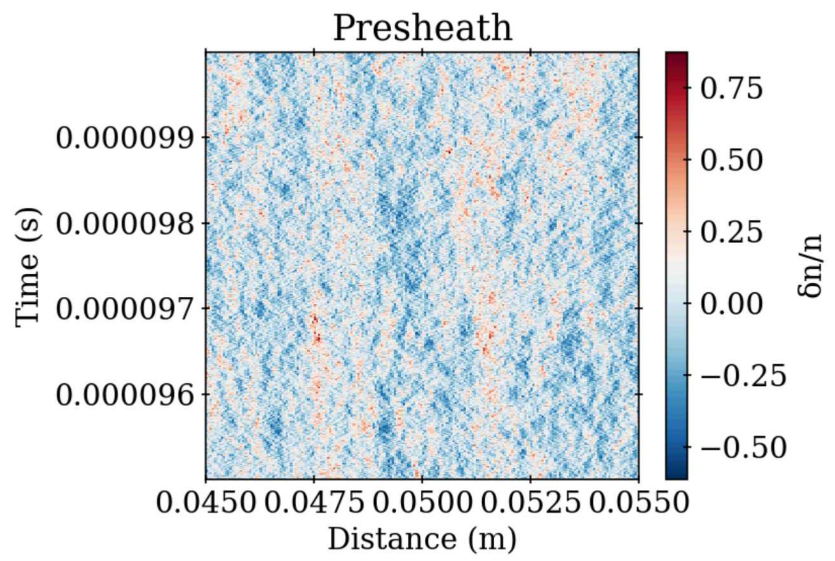
  → ~ 24 hours

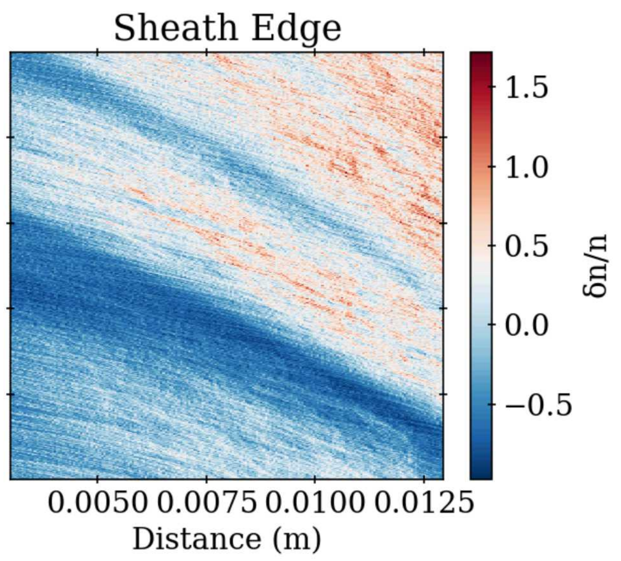
Ion-Ion Streaming Instability: Phase Space Results

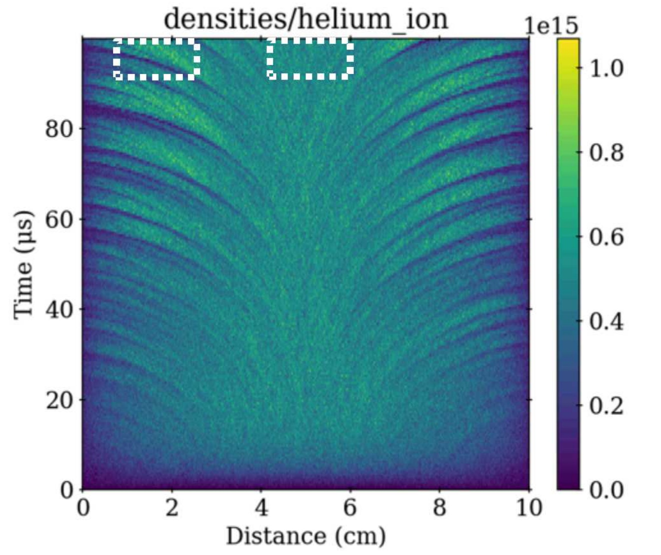




Ion-Ion Instability: Time Domain Analysis

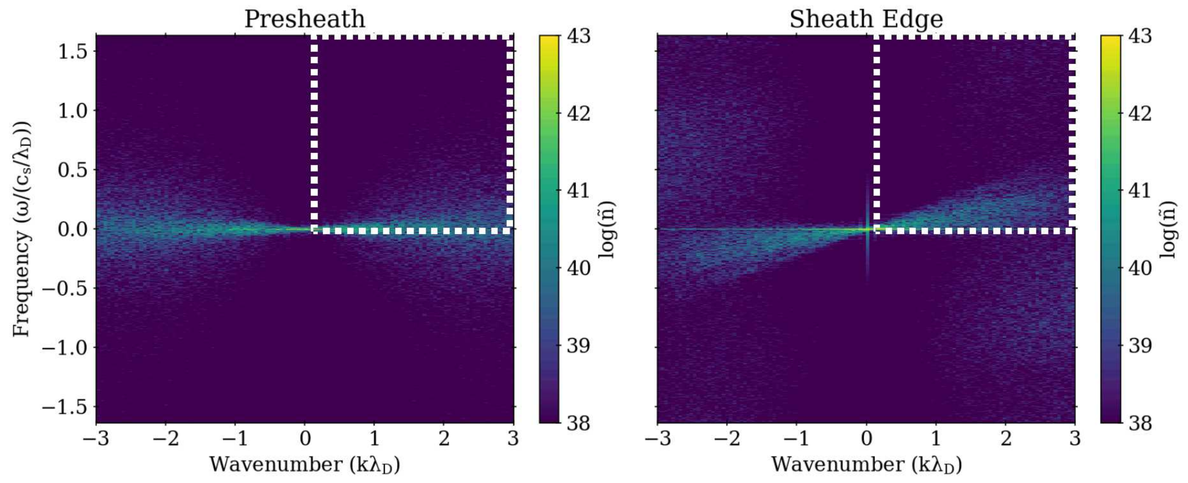


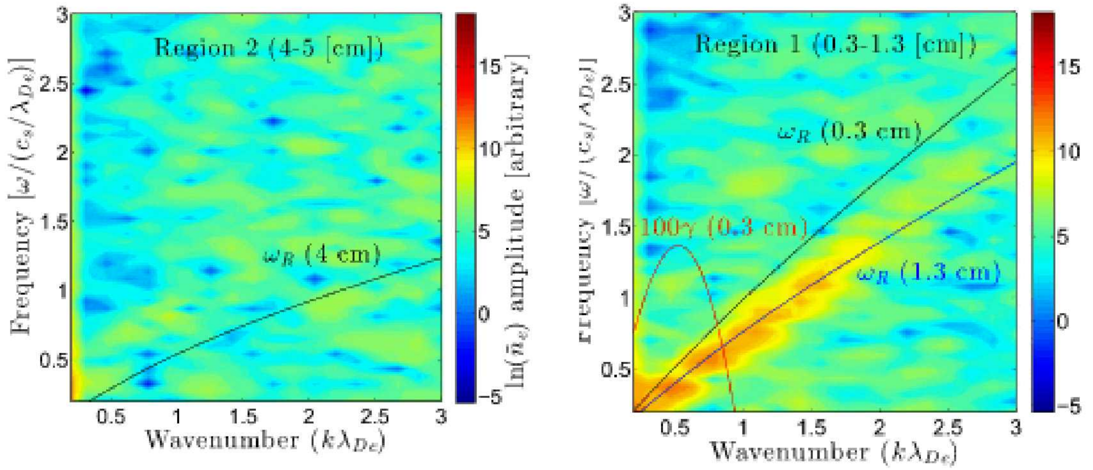




- Presheath sampled at center of domain, minimal flow
- Sheath edge sampled just prior to departure from quasineutrality
- Presheath quiescent relatively little structure
- Strong waves originating in presheath, propagating into boundary
- Appears in all species (strongest in helium, weakest in electrons)

Ion-Ion Instability: Frequency Domain Analysis





- Take 2D Fourier transform of helium ion density for comparison to dispersion relation
- Both presheath and sheath edge have strong low frequency content
- Likely due to simulation noise (add more particles)
- Sheath edge shows strong signal in region where ion-ion instability is expected
- Can reproduce cutting edge sheath physics results with simple 1D PIC code (and enough time)

Baalrud et al. Plasma Sources Sci. Technol. **24**, 015034 (2015)

Summary

- Particle simulations can accurately reproduce detailed kinetic behavior of plasmas
- While they are derived from kinetic theory, writing them can be easy as $F=ma$ and $\nabla^2\phi=-\rho/\varepsilon_0$
- Particle-in-cell simulations should
  - Resolve the Debye length
  - Resolve the electron plasma frequency
  - Satisfy the CFL condition
- Finding a solution is an iterative process

Acknowledgements

- Scott Baalrud
- Greg Severn
- Steve Shannon
- Venkattraman Ayyaswamy

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences under contract number DE-NA0003525. This research used resources of the Low Temperature Plasma Research Facility at Sandia National Laboratories, which is a collaborative research facility supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences.

Questions?

Contact: btyee@sandia.gov, mmhopki@sandia.gov

Advanced Topics & Resources

Software

- *Many* PIC codes available with a variety of licenses and features
- Models and methods often chosen for particular application (wakefield accelerators, beams, gas dynamics, etc.)
- Compatible computer architectures and ease of use also widely vary

- Aleph
- CHICAGO
- EMPIRE
- MAGIC
- OSIRIS
- Vsim
- VPIC
- Warp(X)
- XPDP1, XOOPIC, …

Research Areas

- Heterogeneous architectures
- Implicit methods
- Hybrid models
- Uncertainty quantification
- Strongly-coupled plasmas
- Adaptive methods
- Photon transport
- Reweighting and merging algorithms
- Kinetic chemistry models
- Solid state transitions
- Multiphysics coupling
- Surface interactions
- Finite element methods

Literature

- Birdsall and Langdon – "Plasma Physics via Computer Simulation"
- Hockney and Eastwood – "Computer Simulation Using Particles"
- Bird – "Molecular Gas Dynamics and the Direct Simulation of Gas Flows"

(Optional) Homework Problems

- Include neutral particle collision effects through Monte-Carlo collisions. What new solution constraints should be introduced?
- Implement secondary electron emission from the electrodes. How could a highly emissive surface restructure a plasma sheath?
- Incorporate the ability to solve for a periodic domain using an FFT for the field solutions. What are some advantages to this approach, how could it be used for finite systems?
- Explore the use of higher-order particle shapes, how do these impact solution time, simulation noise, and large scale behavior?
- Change the coordinate system to cylindrical or spherical to explore unequal electrode area properties. Why could a particle approaching $r = 0$ be a problem, how might you deal with it?
- Develop a merge algorithm to dynamically limit the number of computational particles. How do you preserve the characteristics of the distribution function when removing particles?
- Explore the equilibration of non-Maxwellian distribution functions, how fast should this occur based on kinetic theory? Can PIC reproduce the appropriate equilibration rate?
- Add a model assuming a Boltzmann relation for the electrons, only treating the ions kinetically. Why is this sometimes desirable? What limits does this place on the types of phenomena that can be modeled?