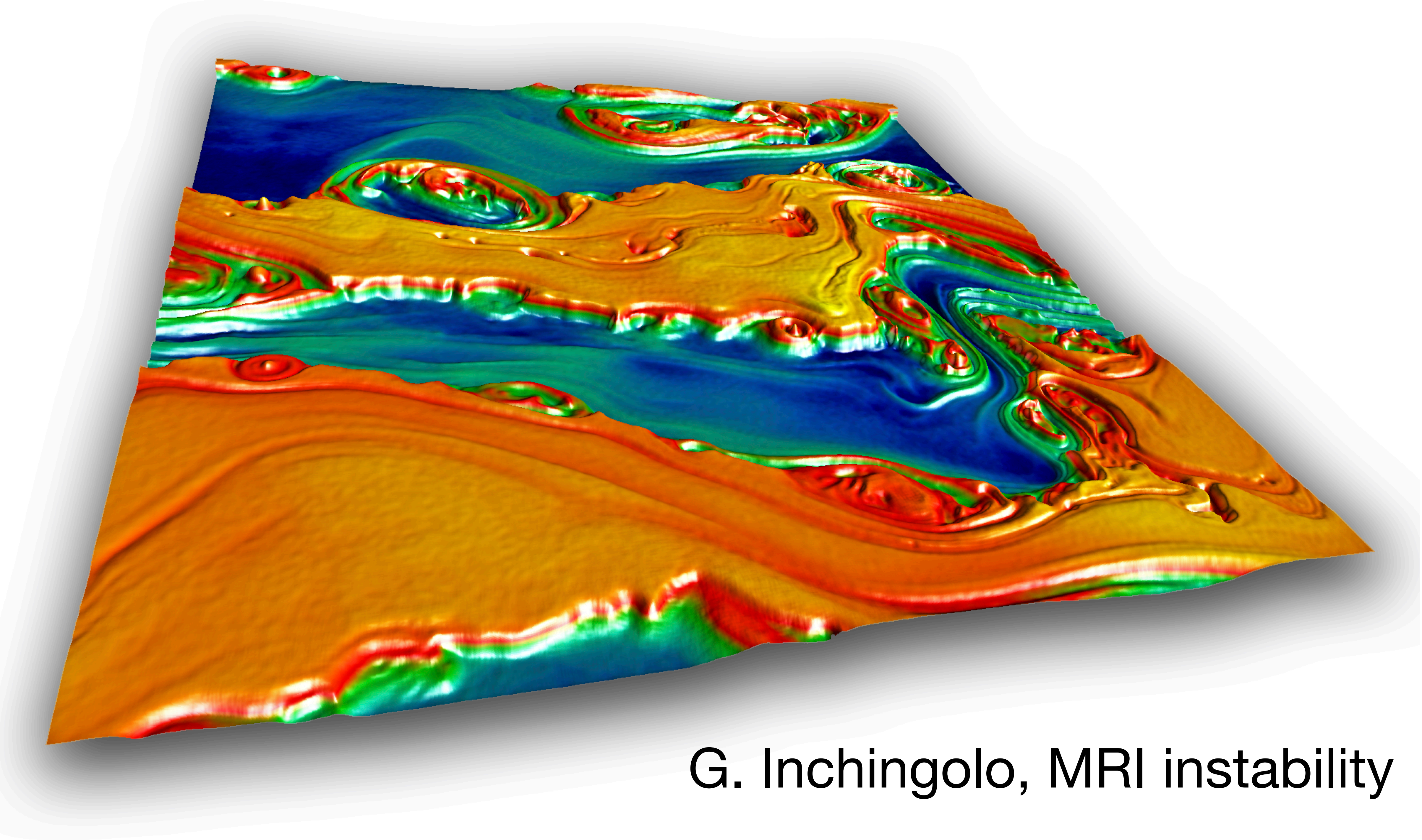
Simulating plasmas

Marija Vranic

GoLP / Instituto de Plasmas e Fusão Nuclear
Instituto Superior Técnico, Lisbon, Portugal

epp.tecnico.ulisboa.pt || golp.tecnico.ulisboa.pt



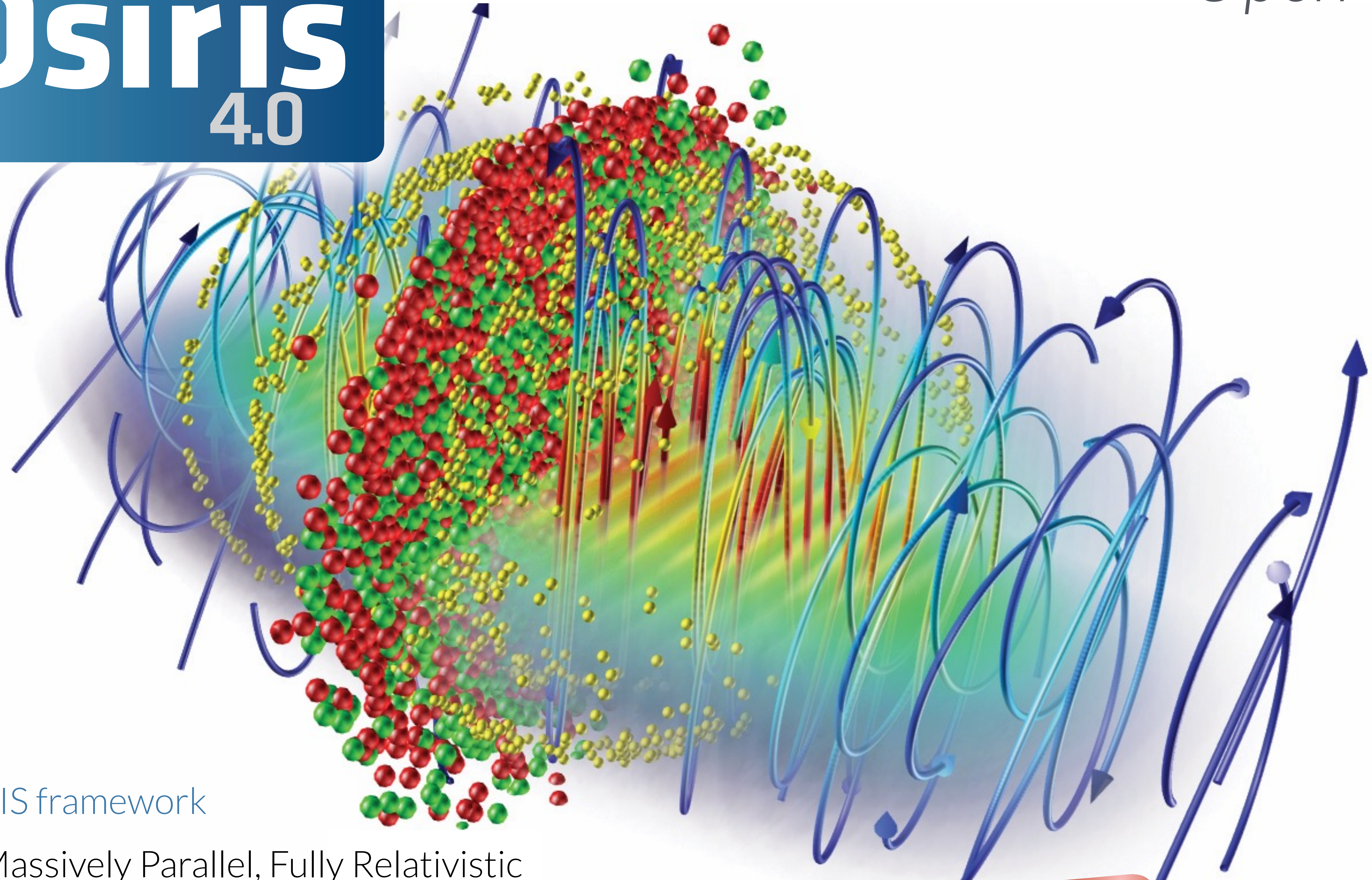
G. Inchingolo, MRI instability

INSTITUTO DE PLASMAS E FUSÃO NUCLEAR

TÉCNICO LISBOA

Osiris 4.0

Open-source version available



Open-access model

- 40+ research groups worldwide are using OSIRIS
- 400+ publications in leading scientific journals
- Large developer and user community
- Detailed documentation and sample inputs files available
- Support for education and training

Using OSIRIS 4.0

- The code can be used freely by research institutions after signing an MoU
- Open-source version at:

https://osiris-code.github.io/

OSIRIS framework

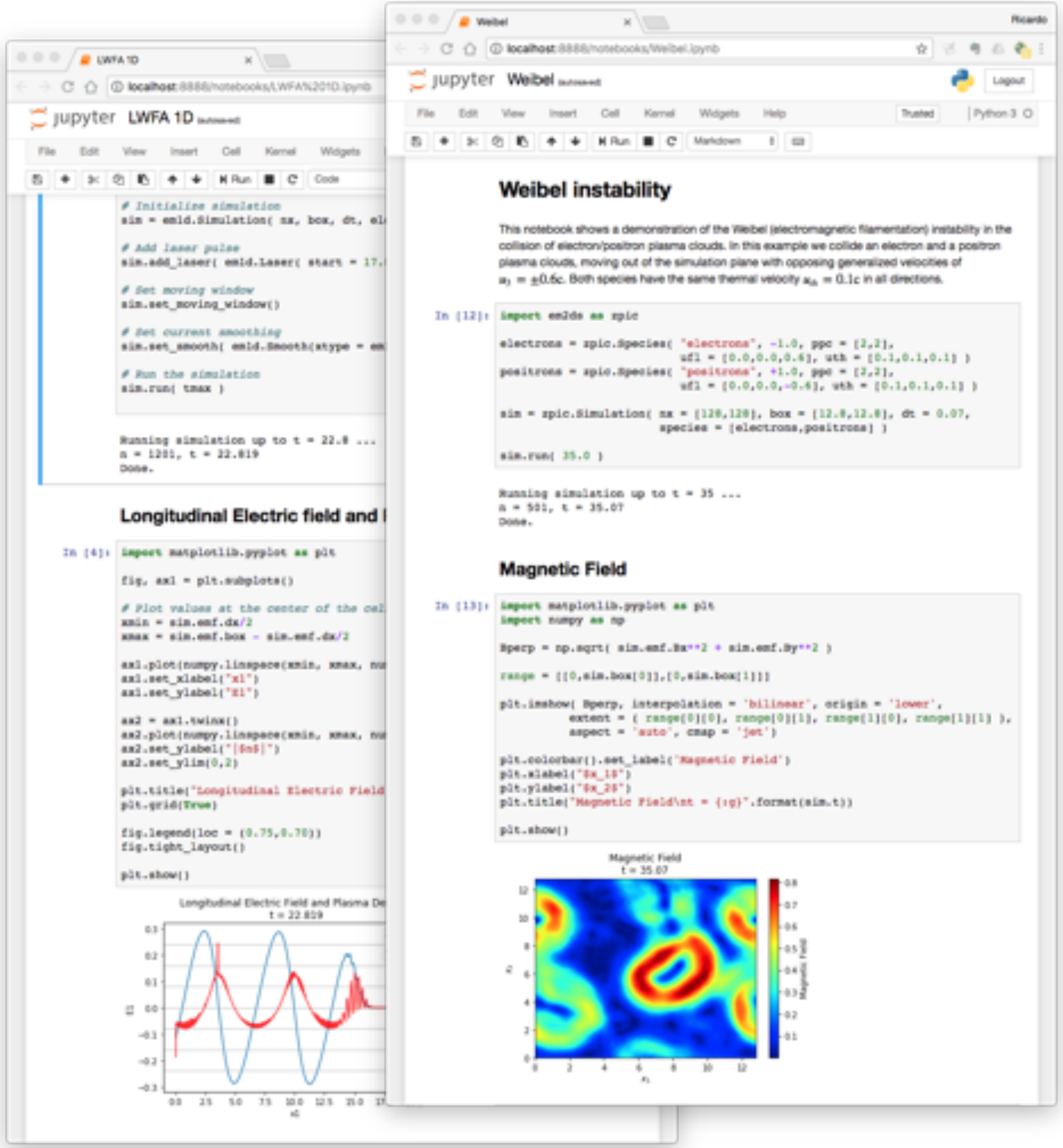
- Massively Parallel, Fully Relativistic Particle-in-Cell Code
- Support for advanced CPU / GPU architectures
- Extended physics/simulation models
- AI/ML surrogate models and data-driven discovery

UNIVERSITY OF MICHIGAN   UCLA   TÉCNICO LISBOA

Ricardo Fonseca: ricardo.fonseca@tecnico.ulisboa.pt

The ZPIC educational code suite

- **ZPIC code suite**
  - Open-source PIC code suit for plasma physics education
  - Fully relativistic 1D and 2D EM-PIC algorithm
  - Eletrostatic 1D/2D PIC algorithm
- **Requirements**
  - No external dependencies, requires only C99 compiler
  - Python interface (optional)
- **Jupyter Notebooks**
  - Includes set of Python notebooks with example problems
  - Detailed explanations of code use and physics
- **Also available through Docker**
  - If you just want to run the notebooks you can use a Docker image available on DockerHub: **zamb/zpic**

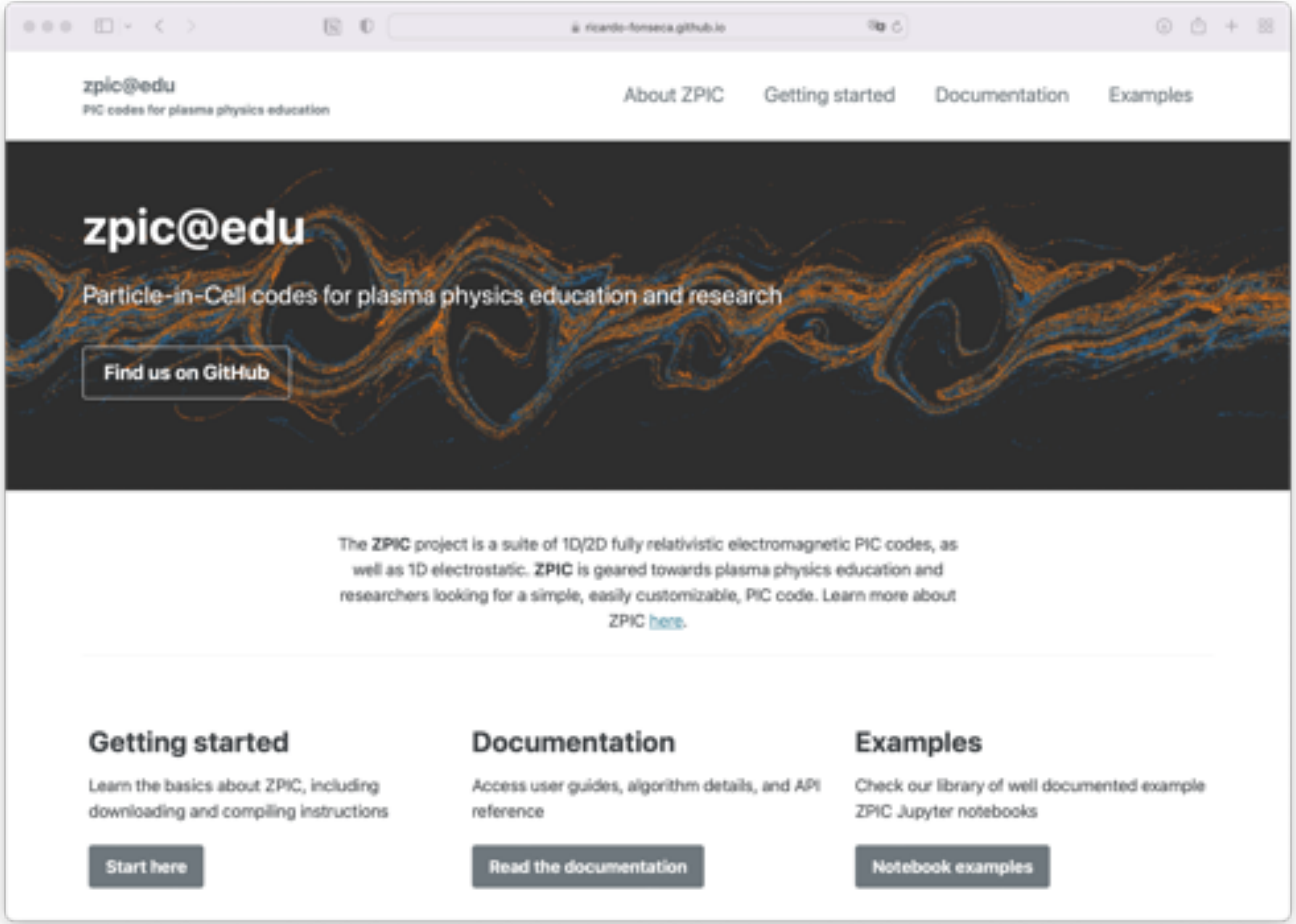


zpic@edu

Come find us on GitHub
github.com/ricardo-fonseca/zpic

ZPIC documentation

https://zpic-plasma.github.io/

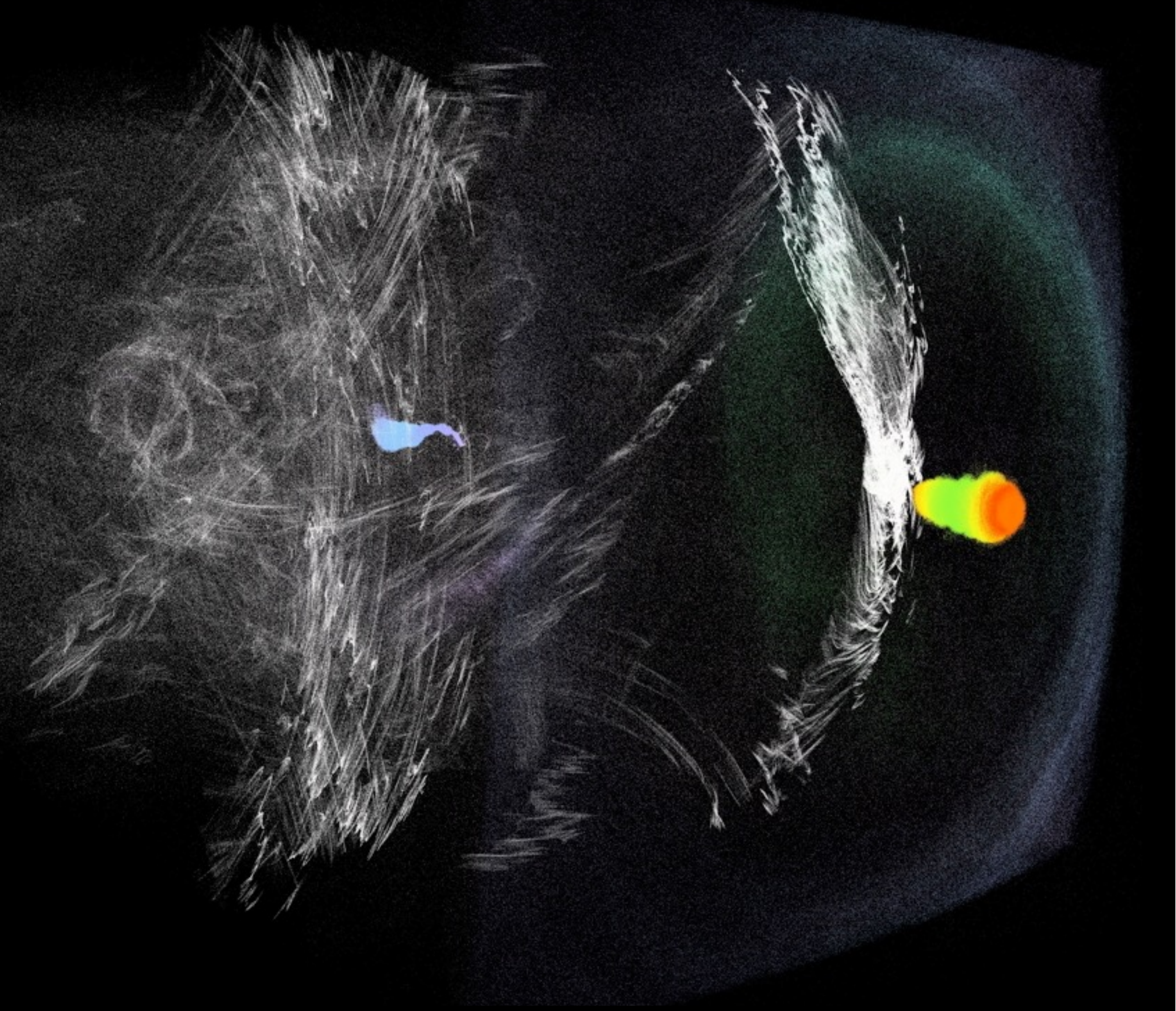


Examples

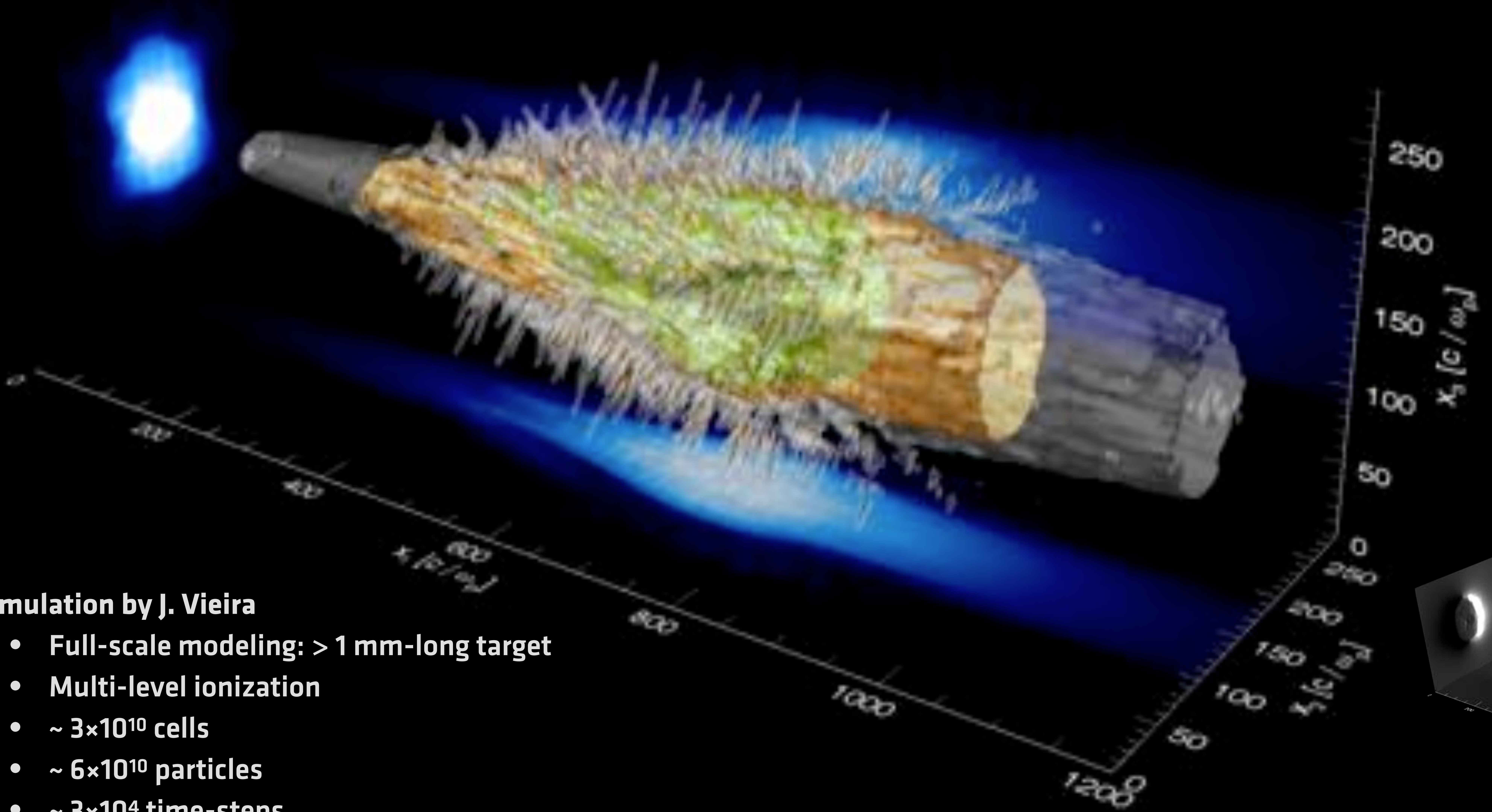
Large Scale Numerical Experiments

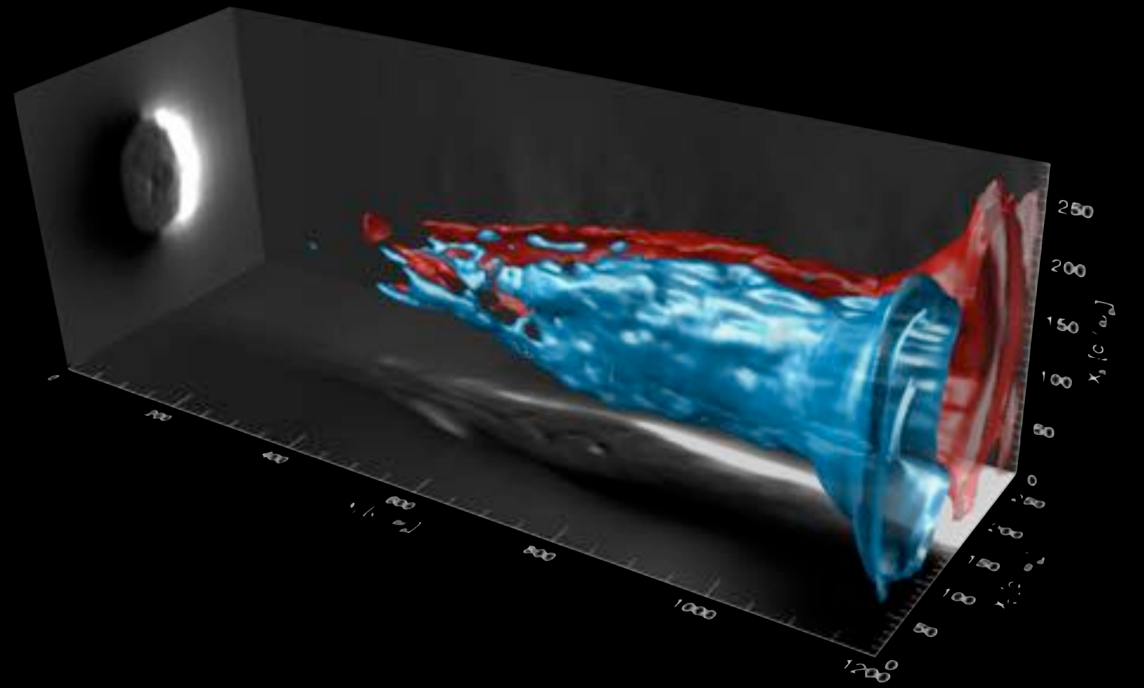
- **Computational Physics needs to deal with increasingly complex systems**
  - Highly nonlinear processes
  - Multi-scale phenomena
  - Purely theoretical descriptions impossible

- **Example: 10 GeV Laser-plasma accelerator**
  - Laser λ_0 ~ 1 µm
  - Propagation length L ~ 0.5 m
  - ~ 10^9 grid cells
  - ~ 10^{10} particles
  - Iterations ~ 10^6 - 10^7
  - Memory ~ 1 - 10 TB
  - Operations ~ 10^{18} - 10^{19}
  - **Run-time on single core ~ 4 years!**



Full scale 3D LWFA modeling

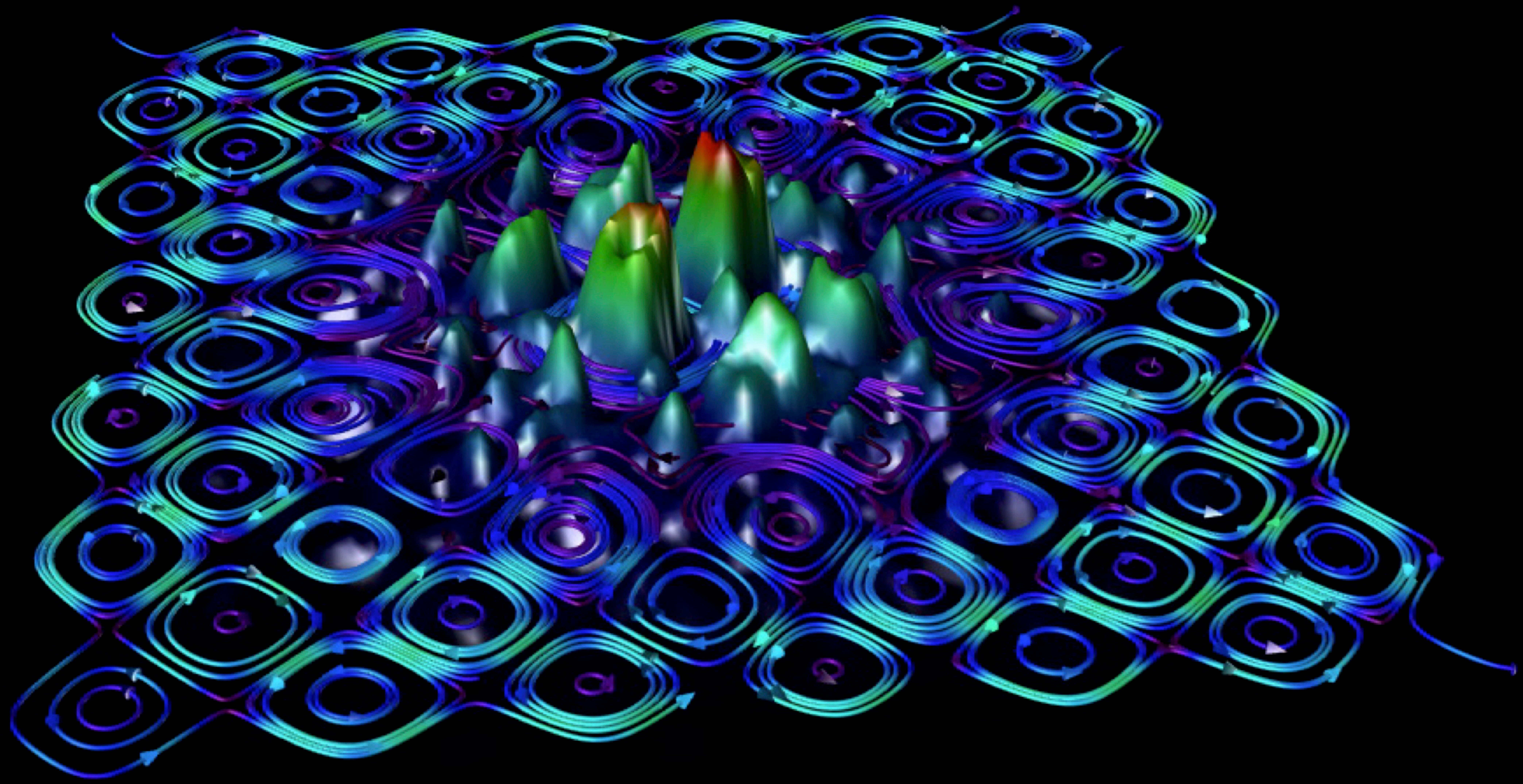




Simulation by J. Vieira

- Full-scale modeling: > 1 mm-long target
- Multi-level ionization
- ~ 3×10^{10} cells
- ~ 6×10^{10} particles
- ~ 3×10^{4} time-steps
- ~ 0.4 million core h (~ 16 k€)

Plasma in extreme conditions - kinetic PIC + Monte Carlo

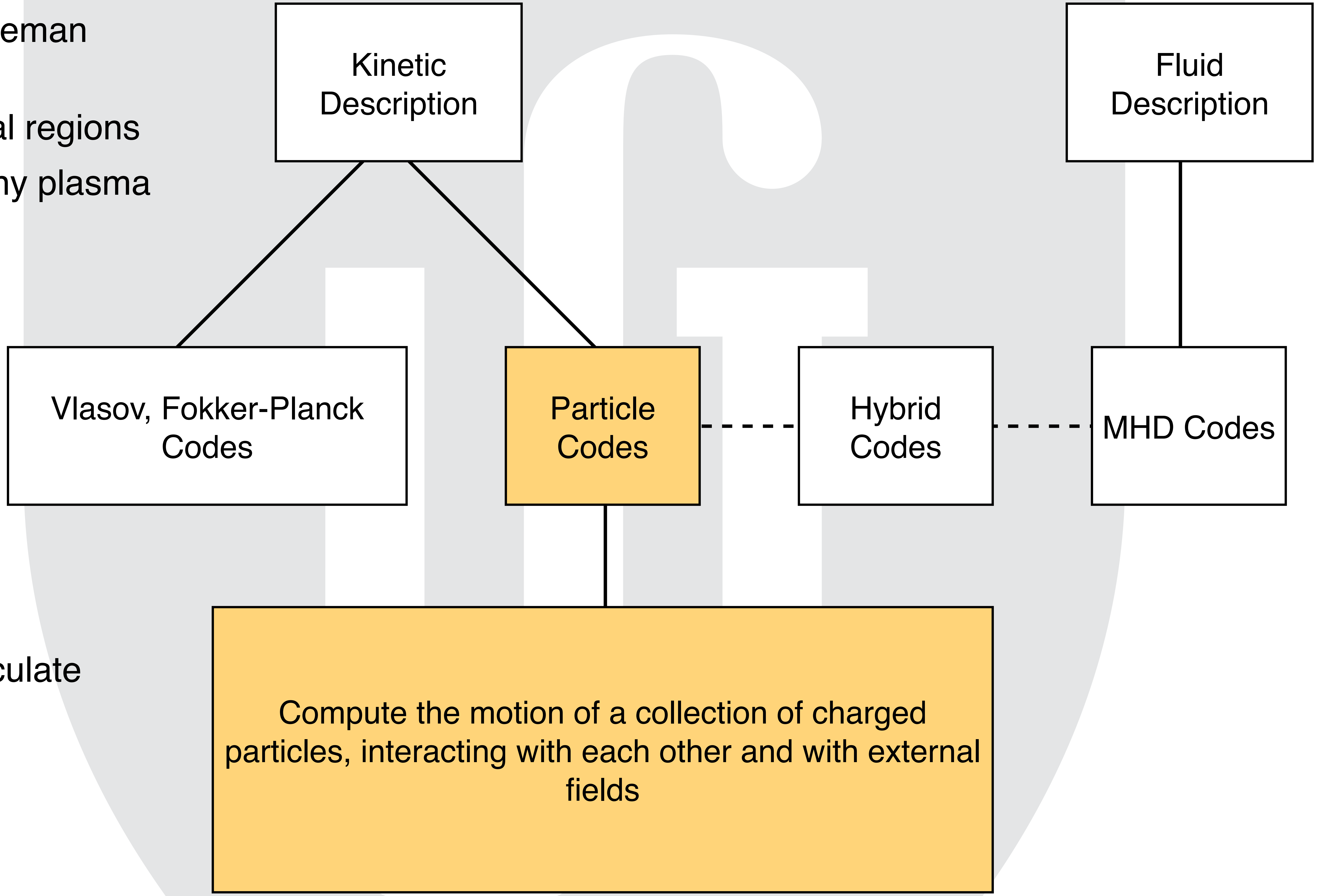


Marija Vranic et al, 4-laser cascade

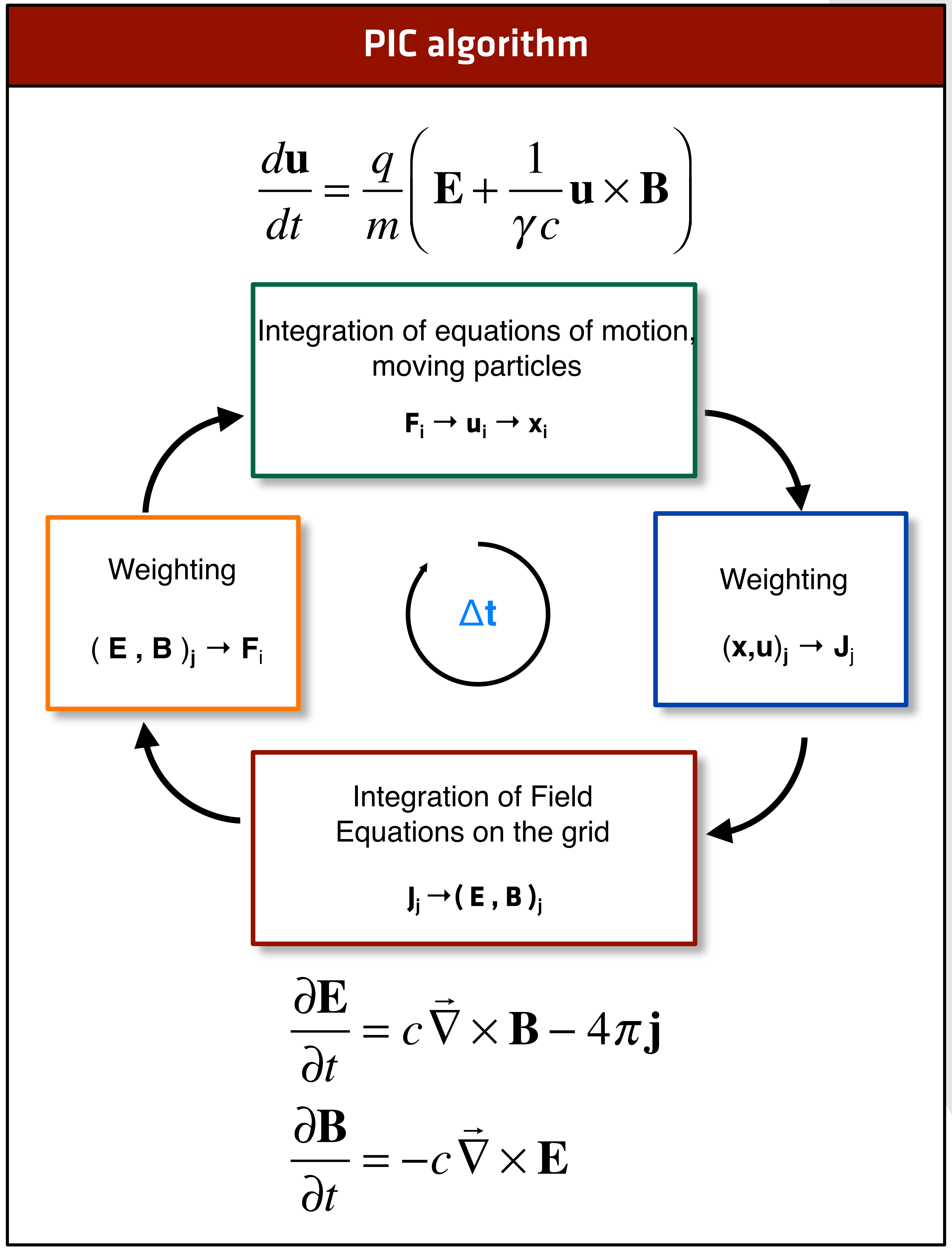
The PIC algorithm

Overview of Plasma Simulation Algorithms

- **Plasma Simulations Using Particles**
  - Pioneered by John Dawson and Oscar Buneman circa 1960
  - Use macro particles to simulate large spatial regions
    - 1 simulation particle corresponds to many plasma particles
  - Particle-Particle simulations
    - Computations go with $O(N_p^2)$
    - Computationally very demanding
- **Particle-In-Cell algorithms**
  - Interact particles through fields
  - Discretize fields on grids
  - Interpolate fields at particle positions to calculate forces
  - Deposit particle charge/current on a grid
  - Particle-Mesh algorithm
    - Computations go with $O(N_p)$
    - Still computationally heavy but much more tractable



The particle-in-cell (PIC) Algorithm



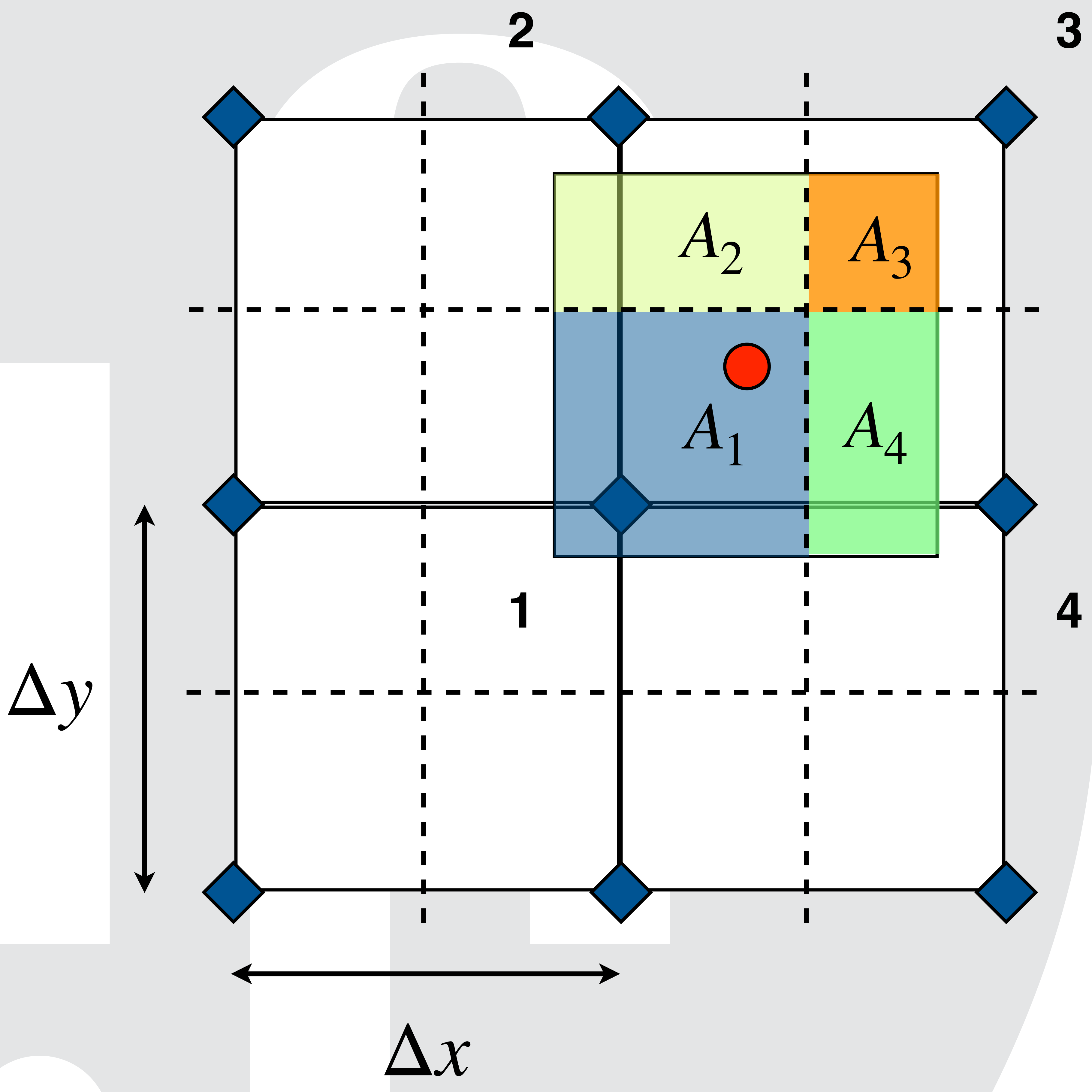
- **Fully Relativistic, Electromagnetic Particle-In-Cell algorithm**
  - Discretize Electric and Magnetic fields on a grid
  - Cell size must resolve shortest relevant lengths in the simulation
    - Typically the laser wavelength or the plasma skin depth
  - Represent plasma particles with simulation macro-particles
    - Free to move in entire *n*D-3V phase-space
    - Each macro-particle represents several plasma particles
  - Must have enough particles per cell to properly resolve velocity distributions
- **Fields and particles don't exist in the same simulation topology**
  - Field quantities are limited to grid points
  - Field interpolation connects fields → particles
  - Current deposition connects particles → fields
- **Four major steps**
  - Field interpolation
  - Particle advance
  - Current deposition
  - Field advance

zpic@edu

Interpolating the fields

- **Particles are free to move to any position**
  - Field are discretized on a grid
  - Field values at particle positions are required to advance particle momenta
- **Interpolate fields at particle positions**
  - ZPIC uses linear interpolation
  - In 2D this can be viewed as area weighting
  - The interpolating scheme must be consistent with charge / current deposition
- **Momentum conserving algorithm**
  - Avoids self-forces
  - $dp/dt = 0$ for single particle

$$A = \Delta x \Delta y$$



$$E_{part} = \frac{A_1 E_1 + A_2 E_2 + A_3 E_3 + A_4 E_4}{A}$$

Pushing the particles

- **Advance generalized velocity and position of individual particles**
  - ZPIC is a fully relativistic code so we work with $\mathbf{u} = \gamma\, \boldsymbol{\beta}$ instead $\mathbf{v}$.
  - We use a leap-frog scheme to integrate particle motion:
    - Positions ($\mathbf{x}$) are defined at integral time-steps t^n
    - Velocities ($\mathbf{u}$) are defined at half time-steps $t^{n+½}$
  - Second-order accuracy in time
- **Velocities are integrated using a relativistic Boris pusher**
  - Separate **E** and **B** contributions
    i. Accelerate with ½ electric impulse
    ii. Full magnetic field rotation
    iii. Add remaining ½ electric impulse
  - Fully relativistic, second order time accurate
  - By construction, no work from **B** field
- **Position advance is straightforward**
  - ZPIC stores cell index and position inside cell

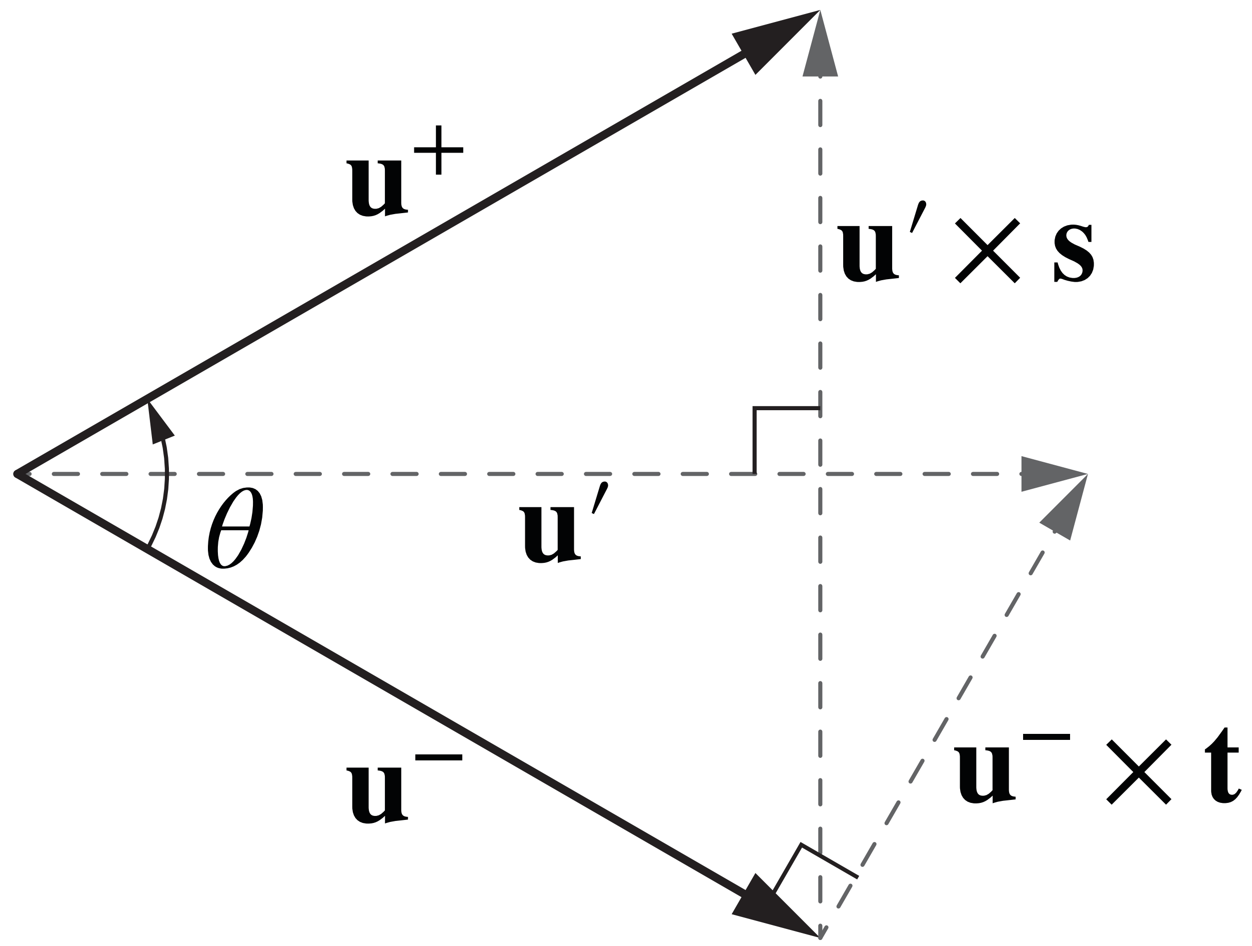
Advance momenta

i. $$\mathbf{u}^- = \mathbf{u}^{n-1/2} + \frac{q\mathbf{E}^n}{m}\frac{\Delta t}{2}$$

ii. $$\mathbf{u}' = \mathbf{u}^- + \mathbf{u}^- \times \mathbf{t}$$

iii. $$\mathbf{u}^+ = \mathbf{u}^- + \mathbf{u}' \times \mathbf{s}$$

iv. $$\mathbf{u}^{n+1/2} = \mathbf{u}^+ + \frac{q\mathbf{E}^n}{m}\frac{\Delta t}{2}$$



$$\mathbf{t} = -\frac{q\mathbf{B}^n}{\gamma^n mc}\frac{\Delta t}{2}$$

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

Advance positions

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \frac{\mathbf{u}^{n+1/2}}{\gamma^{n+1/2}}\Delta t$$

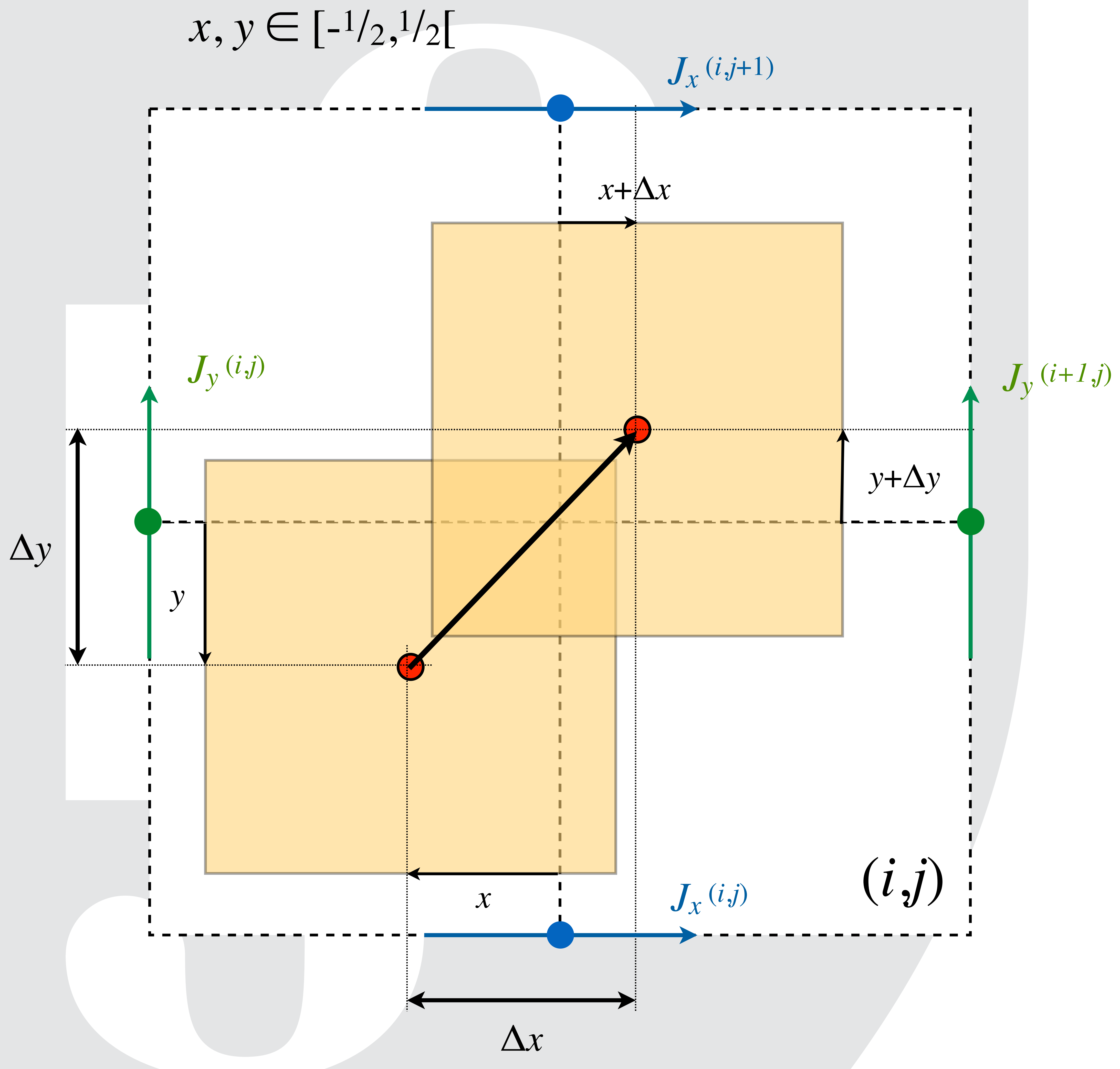
Boris, JP Proc. Fourth Conf. Num. Sim. Plasmas (1970) 3-67

Depositing the current

- **Connects particle motion to field equations**
  - Current deposition must satisfy continuity equation:

$$\frac{\partial \rho}{\partial t} = -\nabla' \cdot \mathbf{j}$$

  - The operator ∇' corresponds to the finite difference approximation
  - Simply depositing $\rho \mathbf{v}$ does not conserve charge
  - Critical to guarantee the solutions to Maxwell's equations are self-consistent
- **Exact charge conserving current deposition scheme**
  - Developed by Villaseñor and Buneman for linear interpolation
  - Looks at particle motion, not velocity
  - Limited to motion inside single cell
    - If particles cross cell boundary, motion is split into segments that don't cross boundaries



Villasenor & Buneman Comp. Phys. Comm. 69 (1992) 306-316

Advancing the EM fields

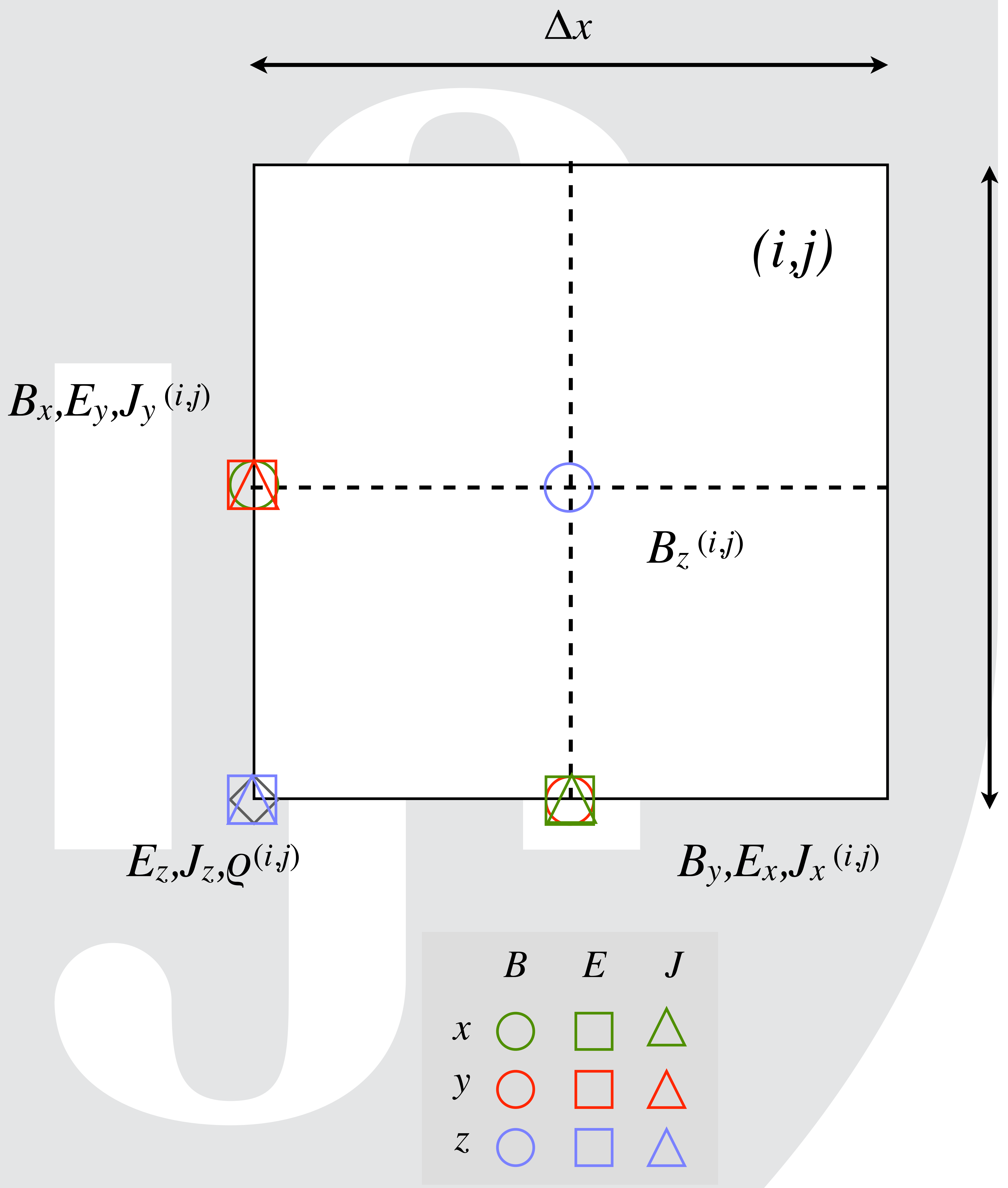
- **EM Fields are advanced in time using Maxwell's equations using the deposited current as source terms**
  - Rearrange Ampére's and Faraday's laws:

$$\frac{\partial \mathbf{E}}{\partial t} = \nabla' \times \mathbf{B} - \mathbf{j}$$

$$\frac{\partial \mathbf{B}}{\partial t} = - \nabla' \times \mathbf{E}$$

  - Discretize temporal and spatial derivatives using finite differences

- **Careful time and spacial centering of quantities leads to 2nd order accuracy**
  - ZPIC uses the Finite Difference Time Domain (FDTD) algorithm
  - Fields are staggered in time for 2nd order accuracy
    - **E** is defined at times t^n
    - **B** and **j** are defined at times $t^{n+½}$
    - **B** is later time centered for use in particle advance
  - And also in space:
    - Spatial derivatives are also 2nd order accurate



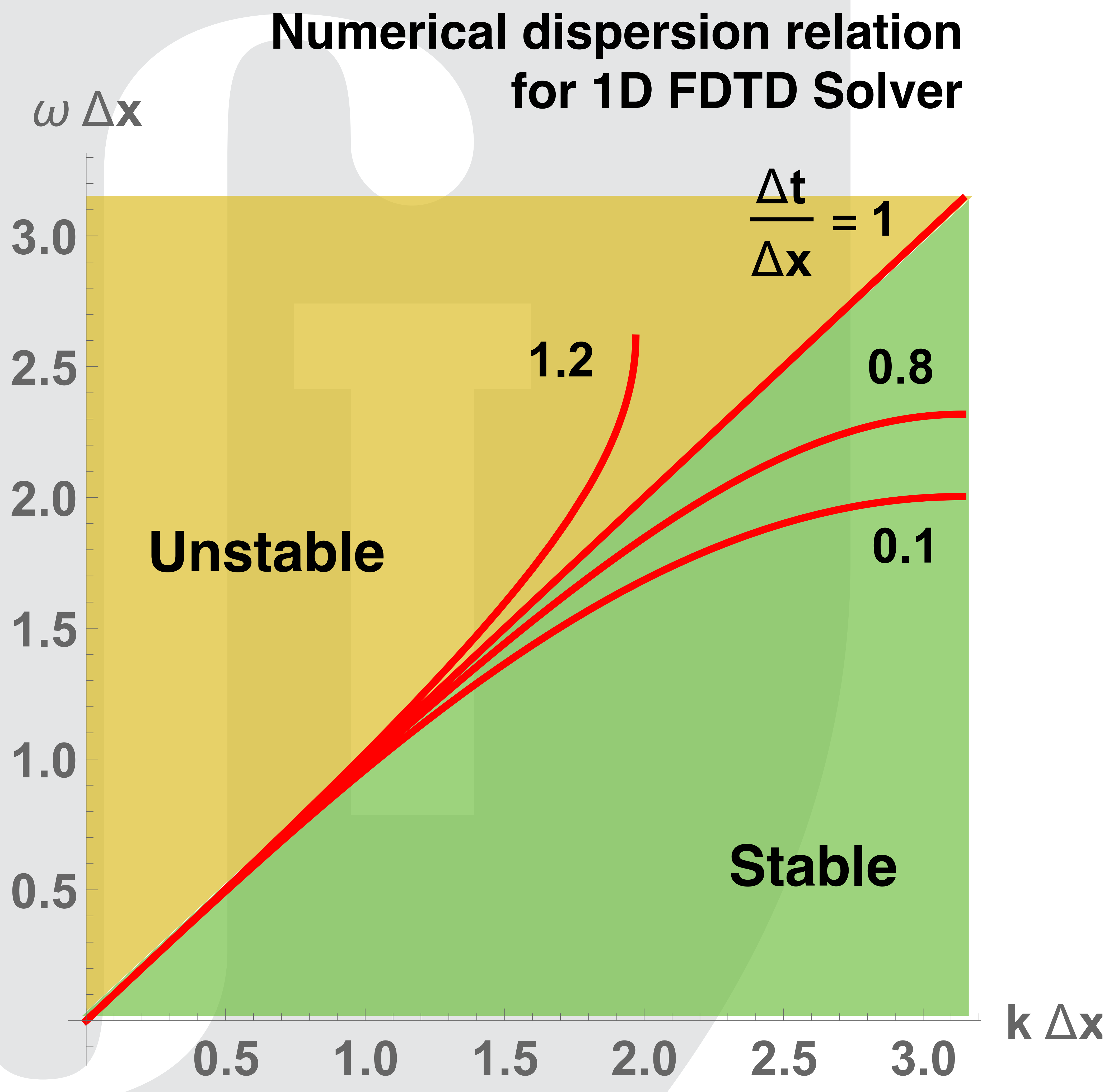
Choice of time-step

- Choice of time-step is dominated by the FDTD solver (in sim. units):

1D $$\Delta t \leq \Delta x$$

2D $$\Delta t \leq (\Delta x^{-2} + \Delta y^{-2})^{-\frac{1}{2}}$$

- If time step is larger than Courant condition the field solver becomes unstable
- If time step is much smaller than courant condition for large k, v_{ph} drops as low as $2/\pi = 0.637$ c
- Relativistic particles may have $v > v_0$
  – Numerical Cherenkov



Units and Normalization in ZPIC

- **Careful choice of units and normalization is critical**
  - Avoids multiplication by several constants (e.g. m_e, e and c) improving performance and numerical accuracy.
  - By expressing the simulation quantities in terms of fundamental plasma quantities the results are general and not bound to some specific units we may choose

- **Units and normalization in ZPIC**
  - The frequencies are normalized to a normalization frequency, ω_n. Time is normalized to ω_n^{-1}.
  - Proper velocities are normalized to the speed of light, c. Space is normalized to c/ω_n.
  - Charge and mass are normalized to the absolute electron charge, e, and the electron mass, m_e. The fields are then normalized appropriately.
  - The density is normalized to ω_n^2 (the normalization frequency squared). So if the density is 1 at a given location then the normalization frequency is the plasma frequency at that location.
  - If the laser frequency is 1, then the normalization frequency is the laser frequency and the density is normalized to the critical densify (for that laser frequency).

zpic units

$$\mathbf{x}' = \frac{\omega_n}{c}\mathbf{x}$$

$$\mathbf{p}' = \frac{\mathbf{p}}{m_{sp}c} = \frac{\gamma \mathbf{v}}{c} = \frac{\mathbf{u}}{c}$$

$$\mathbf{E}' = e\frac{c/\omega_n}{m_e c^2}\mathbf{E}$$

$$\mathbf{B}' = e\frac{c/\omega_n}{m_e c^2}\mathbf{B}$$

m_{sp} is the mass of the species

zpic@edu