

Modelling & Simulation

Hands on I/II



Instituto Universitário de Lisboa

R. A. Fonseca^{1,2}, Jorge Vieira¹

¹GoLP/IPFN, Instituto Superior Técnico, Lisboa, Portugal

² DCTI, ISCTE-Instituto Universitário de Lisboa, Portugal



Committed to open science

Open-access model

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

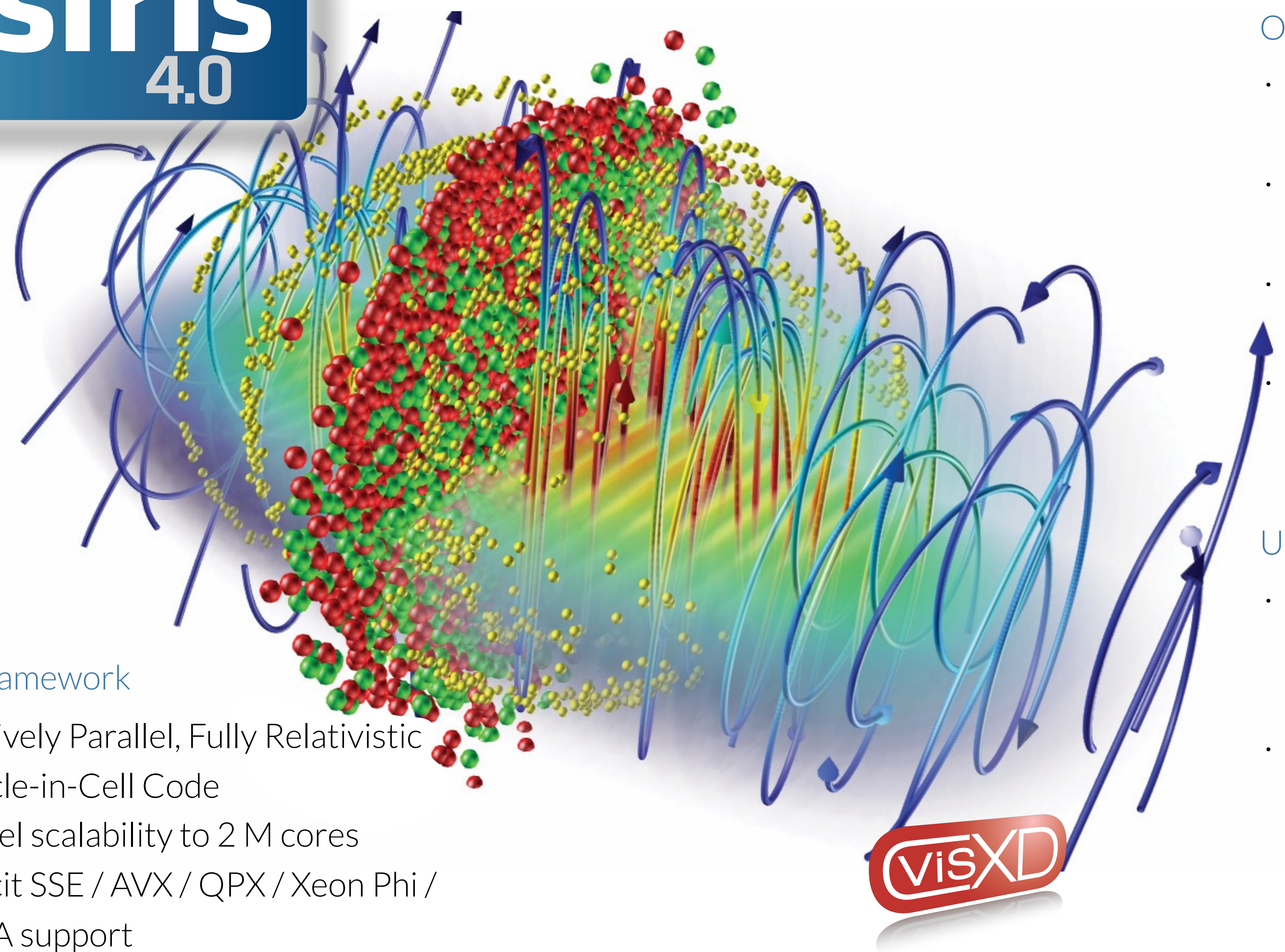
Using OSIRIS 4.0

- The code can be used freely by research institutions after signing an MoU
- Find out more at:

<http://epp.tecnico.ulisboa.pt/osiris>

OSIRIS framework

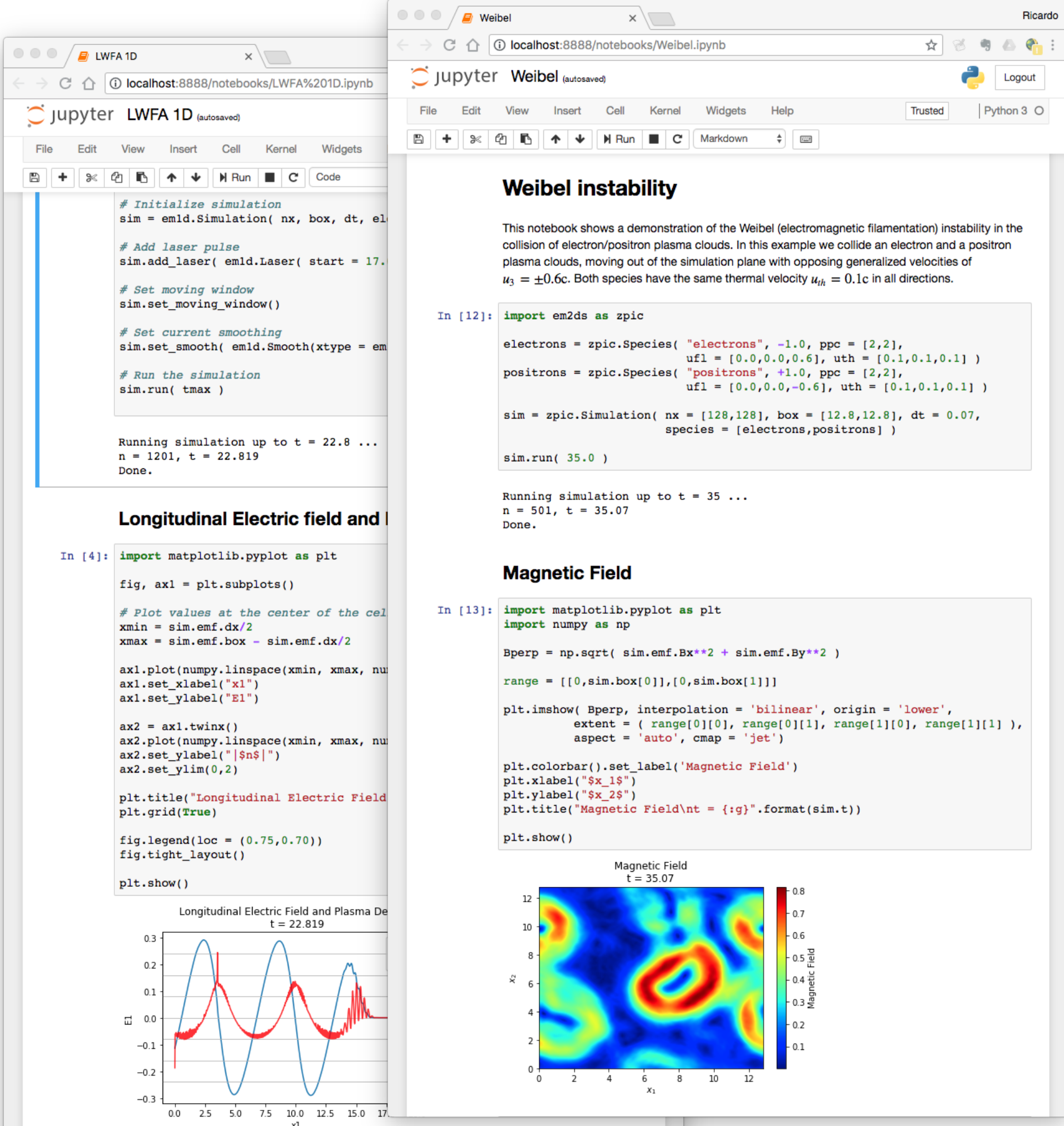
- Massively Parallel, Fully Relativistic Particle-in-Cell Code
- Parallel scalability to 2 M cores
- Explicit SSE / AVX / QPX / Xeon Phi / CUDA support
- Extended physics/simulation models



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
 - Electrostatic 1D/2D PIC algorithm
- **Requirements**
 - No external dependencies, requires only C99 compiler
 - Optional Python interface
- **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**



LWFA 1D

```
# Initialize simulation
sim = em1d.Simulation( nx, box, dt, el

# Add laser pulse
sim.add_laser( em1d.Laser( start = 17.

# Set moving window
sim.set_moving_window()

# Set current smoothing
sim.set_smooth( em1d.Smooth(xtype = em

# Run the simulation
sim.run( tmax )

Running simulation up to t = 22.8 ...
n = 1201, t = 22.819
Done.
```

Longitudinal Electric field and Plasma Density

```
In [4]: import matplotlib.pyplot as plt
fig, ax1 = plt.subplots()
ax1.plot(numpy.linspace(xmin, xmax, nu
ax1.set_xlabel("x1")
ax1.set_ylabel("E1")

ax2 = ax1.twinx()
ax2.plot(numpy.linspace(xmin, xmax, nu
ax2.set_ylabel("$n$")
ax2.set_ylim(0,2)

plt.title("Longitudinal Electric Field and Plasma Density")
plt.grid(True)

fig.legend(loc = (0.75,0.70))
fig.tight_layout()
plt.show()
```

Weibel instability

This notebook shows a demonstration of the Weibel (electromagnetic filamentation) instability in the collision of electron/positron plasma clouds. In this example we collide an electron and a positron plasma clouds, moving out of the simulation plane with opposing generalized velocities of $u_z = \pm 0.6c$. Both species have the same thermal velocity $u_{th} = 0.1c$ in all directions.

```
In [12]: import em2ds as zpzc
electrons = zpzc.Species( "electrons", -1.0, ppc = [2,2],
                        ufl = [0.0,0.0,0.6], uth = [0.1,0.1,0.1] )
positrons = zpzc.Species( "positrons", +1.0, ppc = [2,2],
                          ufl = [0.0,0.0,-0.6], uth = [0.1,0.1,0.1] )

sim = zpzc.Simulation( nx = [128,128], box = [12.8,12.8], dt = 0.07,
                      species = [electrons,positrons] )

sim.run( 35.0 )

Running simulation up to t = 35 ...
n = 501, t = 35.07
Done.
```

Magnetic Field

```
In [13]: import matplotlib.pyplot as plt
import numpy as np

Bperp = np.sqrt( sim.emf.Bx**2 + sim.emf.By**2 )

range = [[0,sim.box[0]],[0,sim.box[1]]]

plt.imshow( Bperp, interpolation = 'bilinear', origin = 'lower',
            extent = ( range[0][0], range[0][1], range[1][0], range[1][1] ),
            aspect = 'auto', cmap = 'jet' )

plt.colorbar().set_label('Magnetic Field')
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.title("Magnetic Field\nt = {t:g}".format(sim.t))

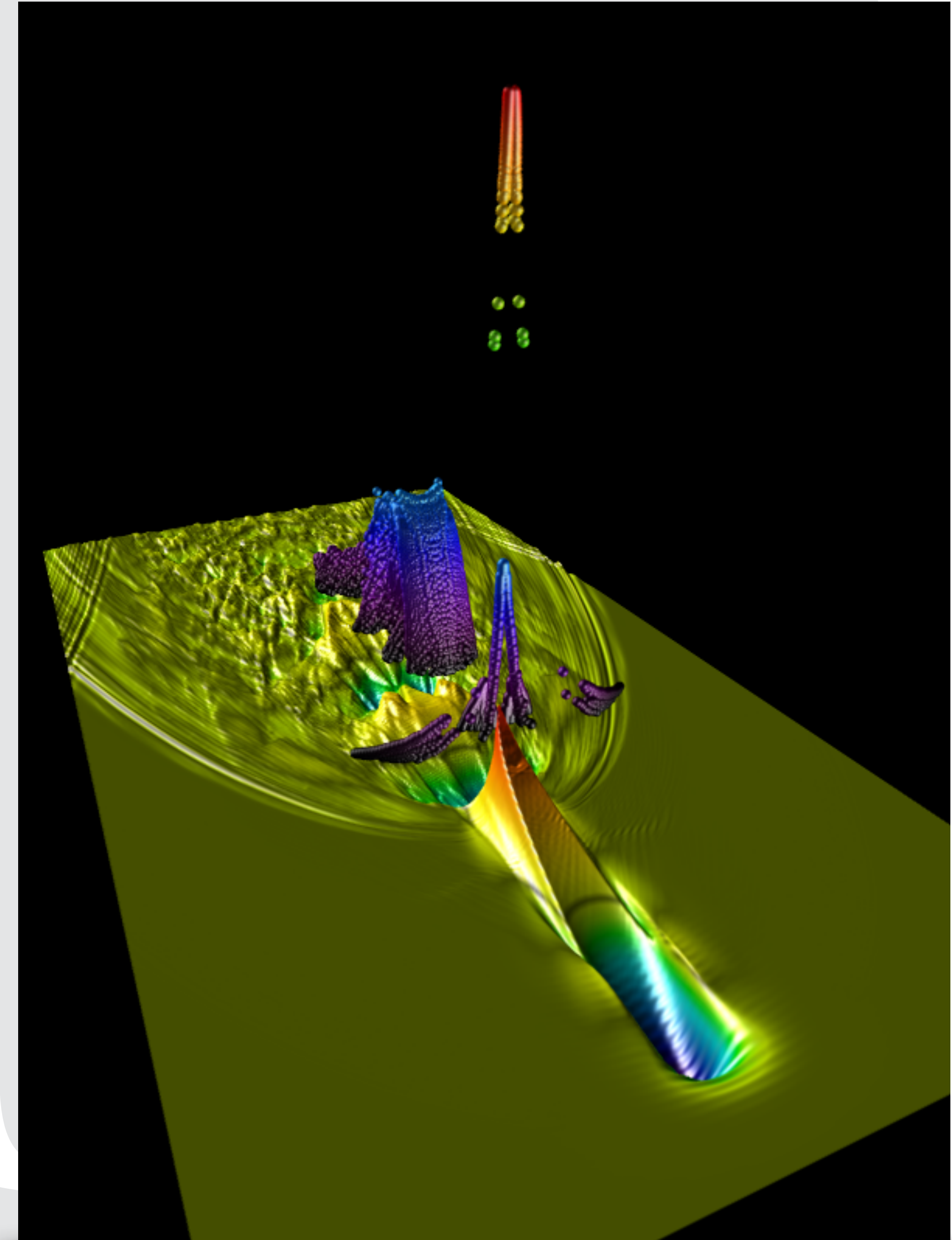
plt.show()
```

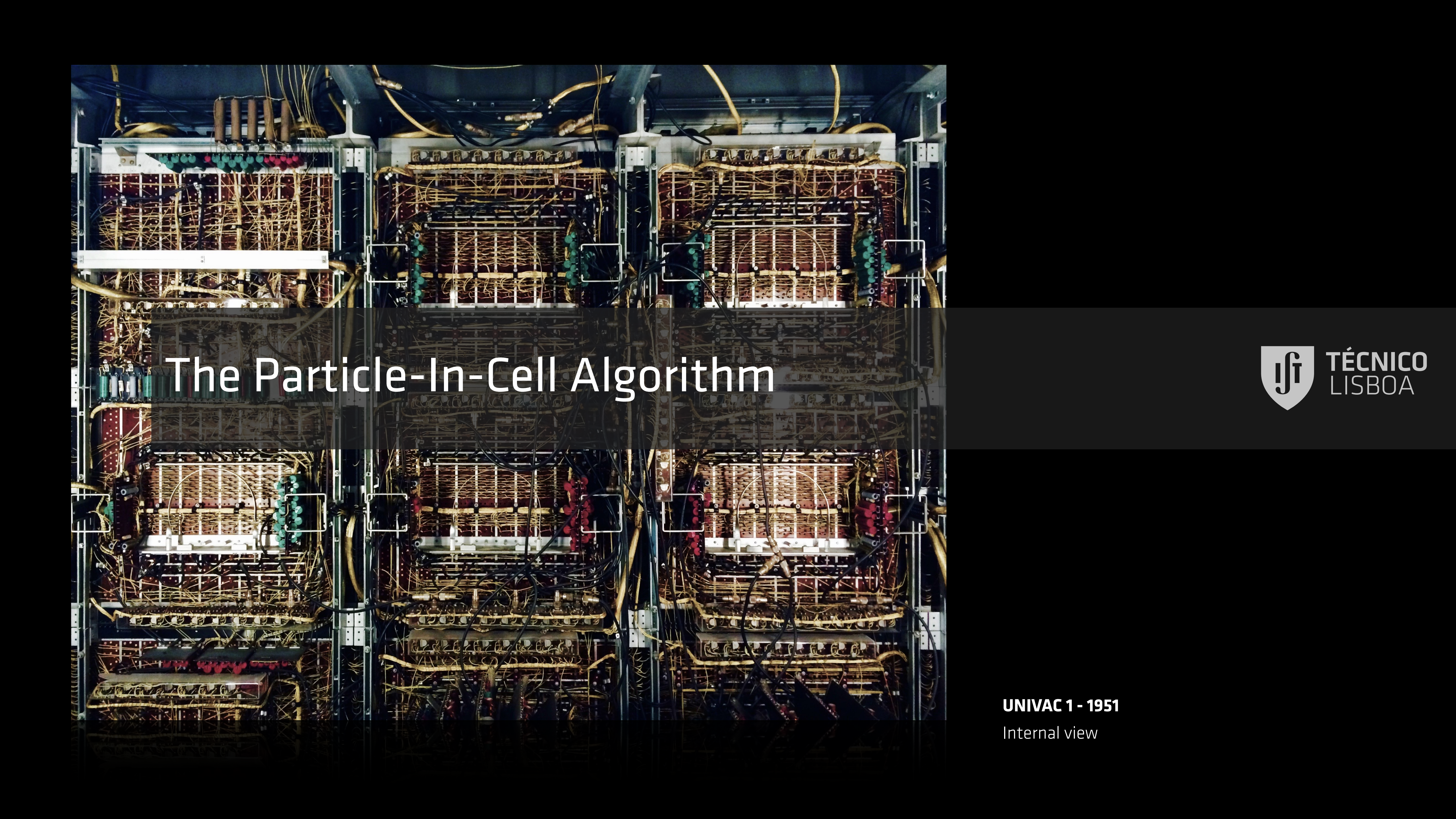
zpic@edu



Come find us on GitHub
github.com/zambzamb/zpic

- **Review of the Particle-In Cell Algorithm**
 - Plasma simulation using particles
 - The Particle-In-Cell algorithm
 - Units and Normalization
 - Time-step considerations
- **Modelling LWFA with PIC Codes**
 - Choice of normalization units
 - Resolution and box size
 - Simulation Particles
 - Useful diagnostics
- **Running ZPIC on your Computer**
 - Compiling from source
 - Using a Docker image
 - Hands-on examples





The Particle-In-Cell Algorithm

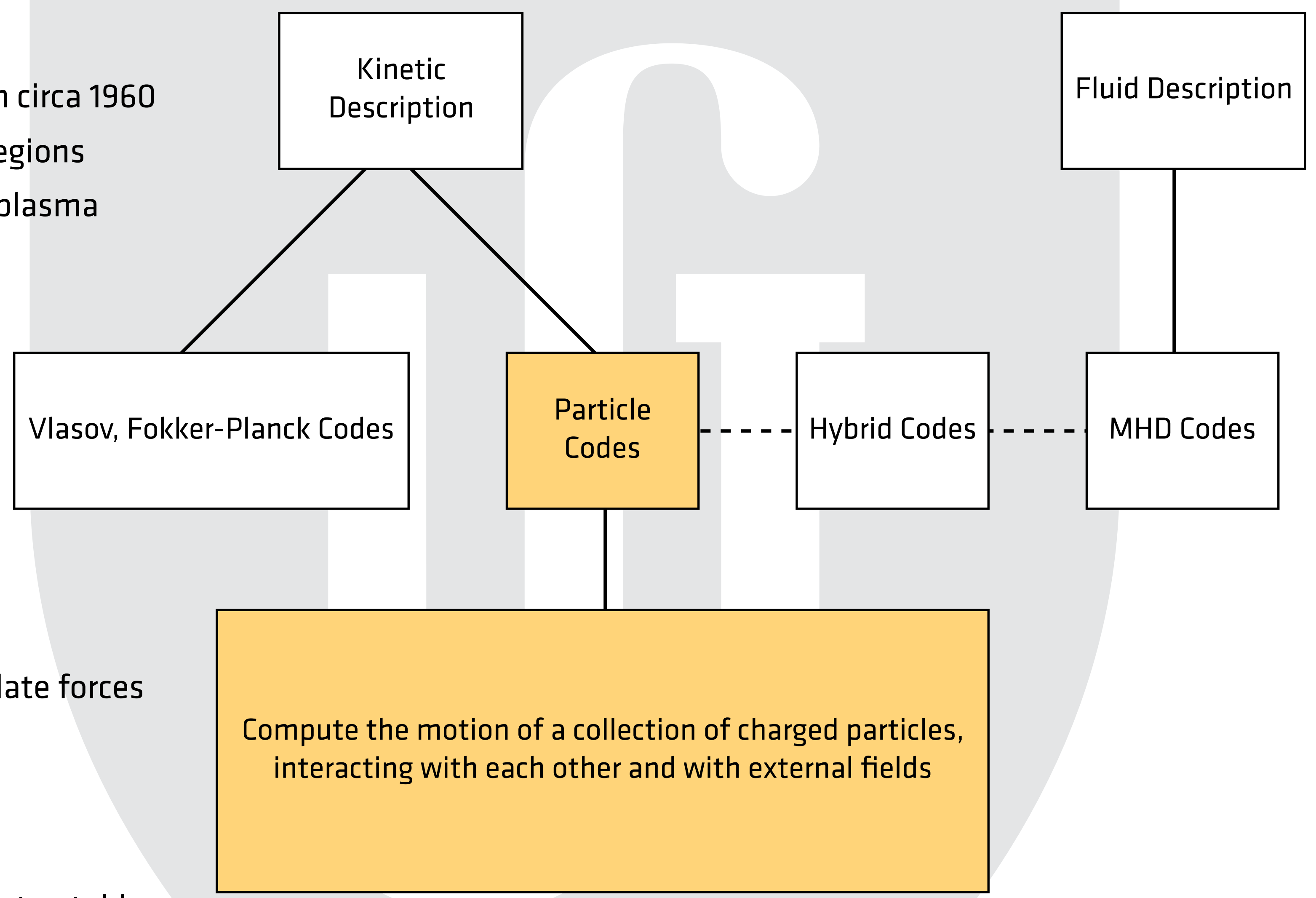
Overview of Plasma Simulation Algorithms

- **Plasma Simulations Using Particles**

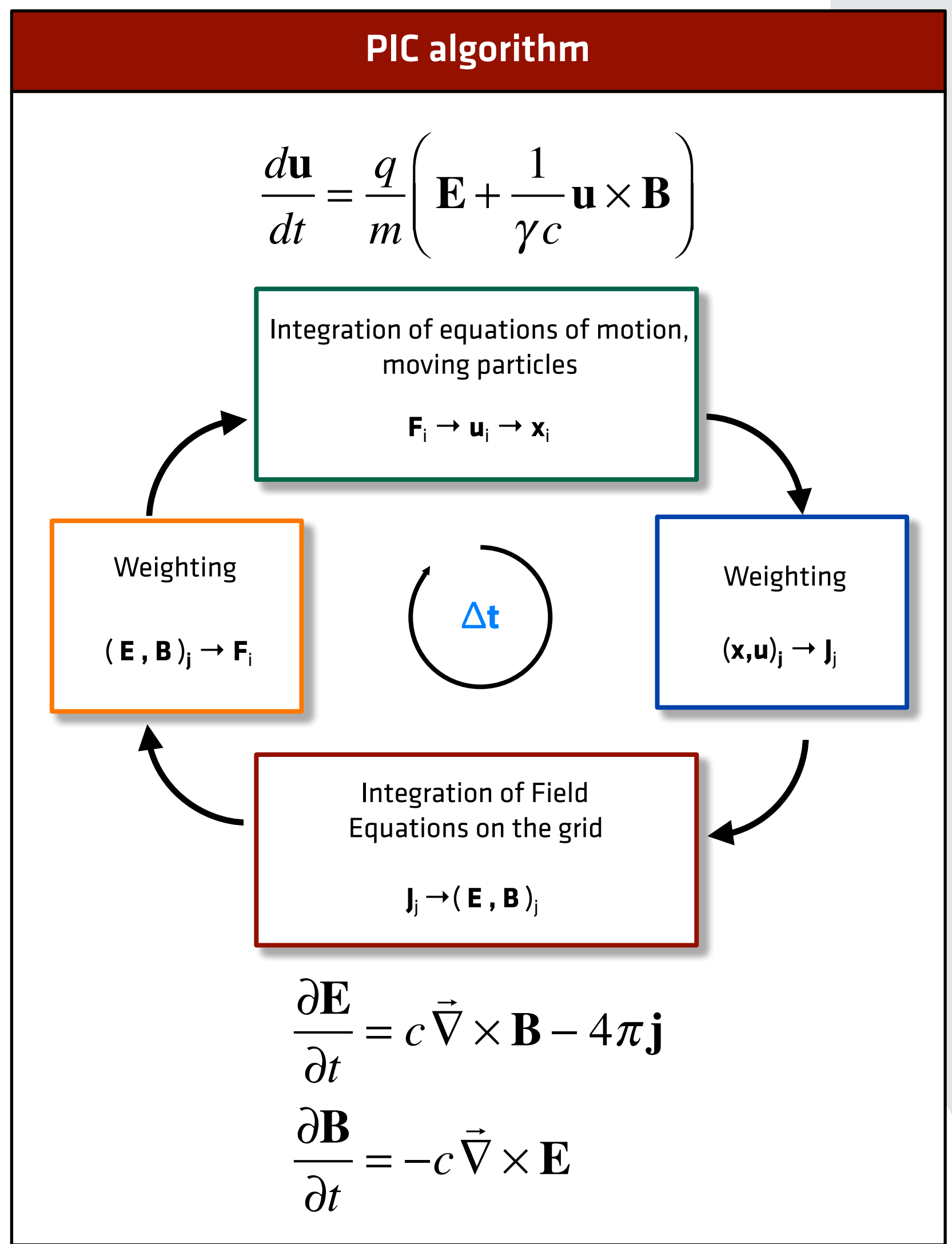
- Pioneered by John Dawson and Oscar Buneman circa 1960
- Use macro particles to simulate large spatial regions
 - 1 simulation particle corresponds to many plasma particles
- Particle-Particle simulations
 - Computations go with $O(N_p^2)$
 - Computationally very demanding

- **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



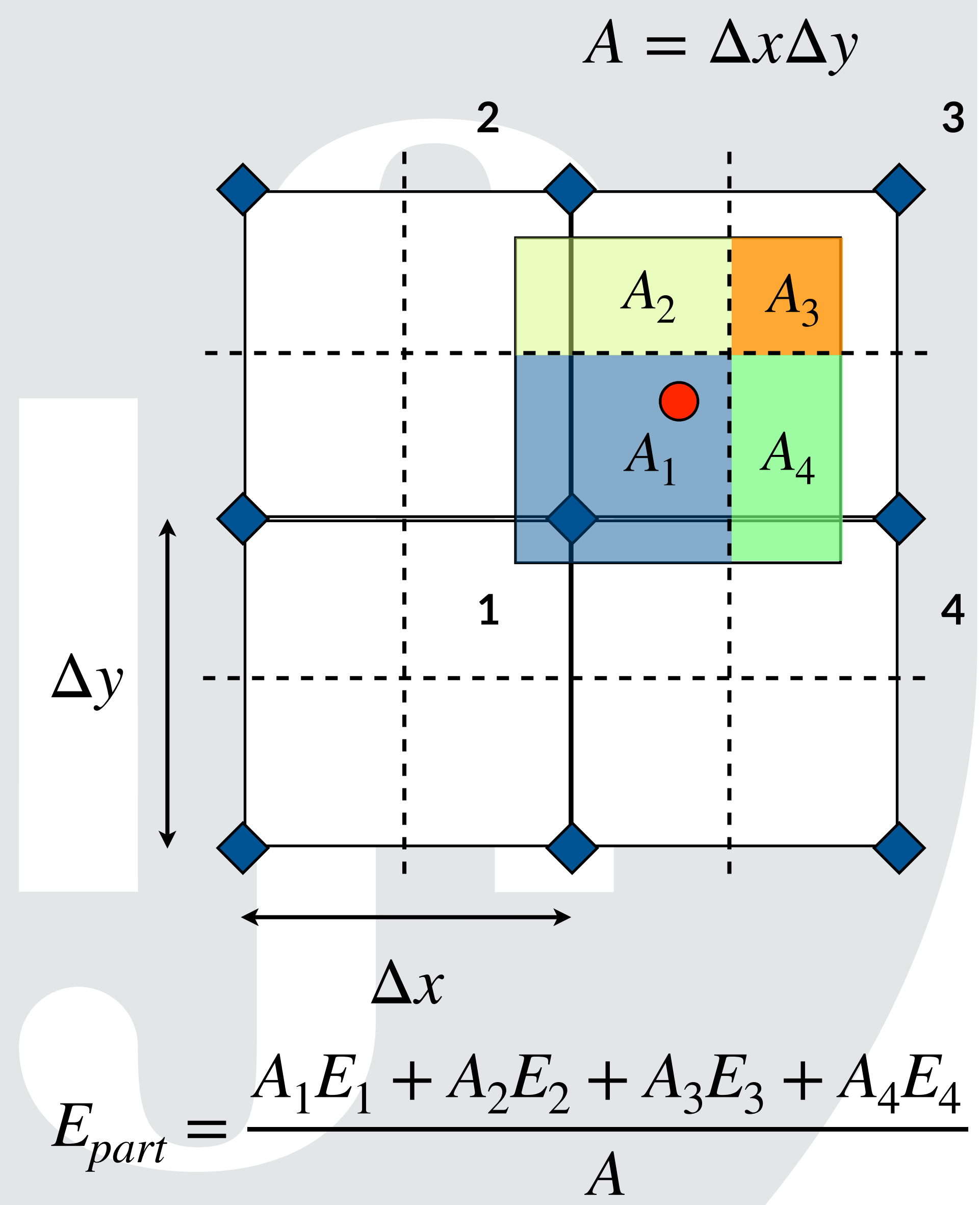
The particle-in-cell (PIC) Algorithm



- **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 $nD-3V$ phasespace
 - 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 \rightarrow particles
 - Current deposition connects particles \rightarrow fields
- **Four major steps**
 - Field interpolation
 - Particle advance
 - Current deposition
 - Field advance

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



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 $t^{n+1/2}$
- Second-order accuracy in time

- **Velocities are integrated using a relativistic Boris pusher**

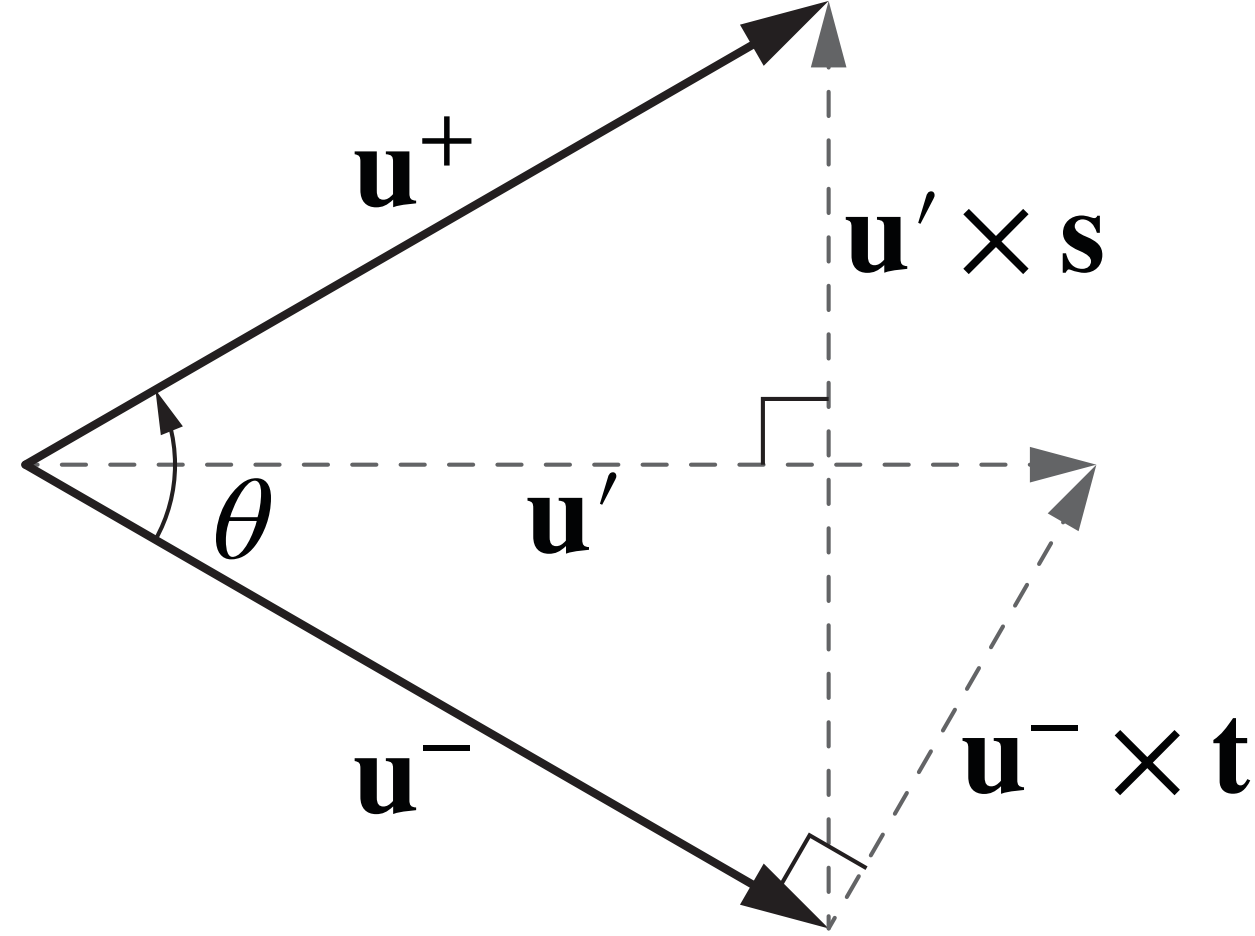
- Separate \mathbf{E} and \mathbf{B} contributions
 - Accelerate with $1/2$ electric impulse
 - Full magnetic field rotation
 - Add remaining $1/2$ electric impulse
- Fully relativistic, second order time accurate
- By construction, no work from \mathbf{B} field

- **Position advance is straightforward**

- ZPIC stores cell index and position inside cell

Advance momenta

- $\mathbf{u}^- = \mathbf{u}^{n-1/2} + \frac{q\mathbf{E}^n \Delta t}{m} \frac{\Delta t}{2}$
- $\mathbf{u}' = \mathbf{u}^- + \mathbf{u}^- \times \mathbf{t}$
- $\mathbf{u}^+ = \mathbf{u}^- + \mathbf{u}^- \times \mathbf{s}$
- $\mathbf{u}^{n+1/2} = \mathbf{u}^+ + \frac{q\mathbf{E}^n \Delta t}{m} \frac{\Delta t}{2}$



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

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

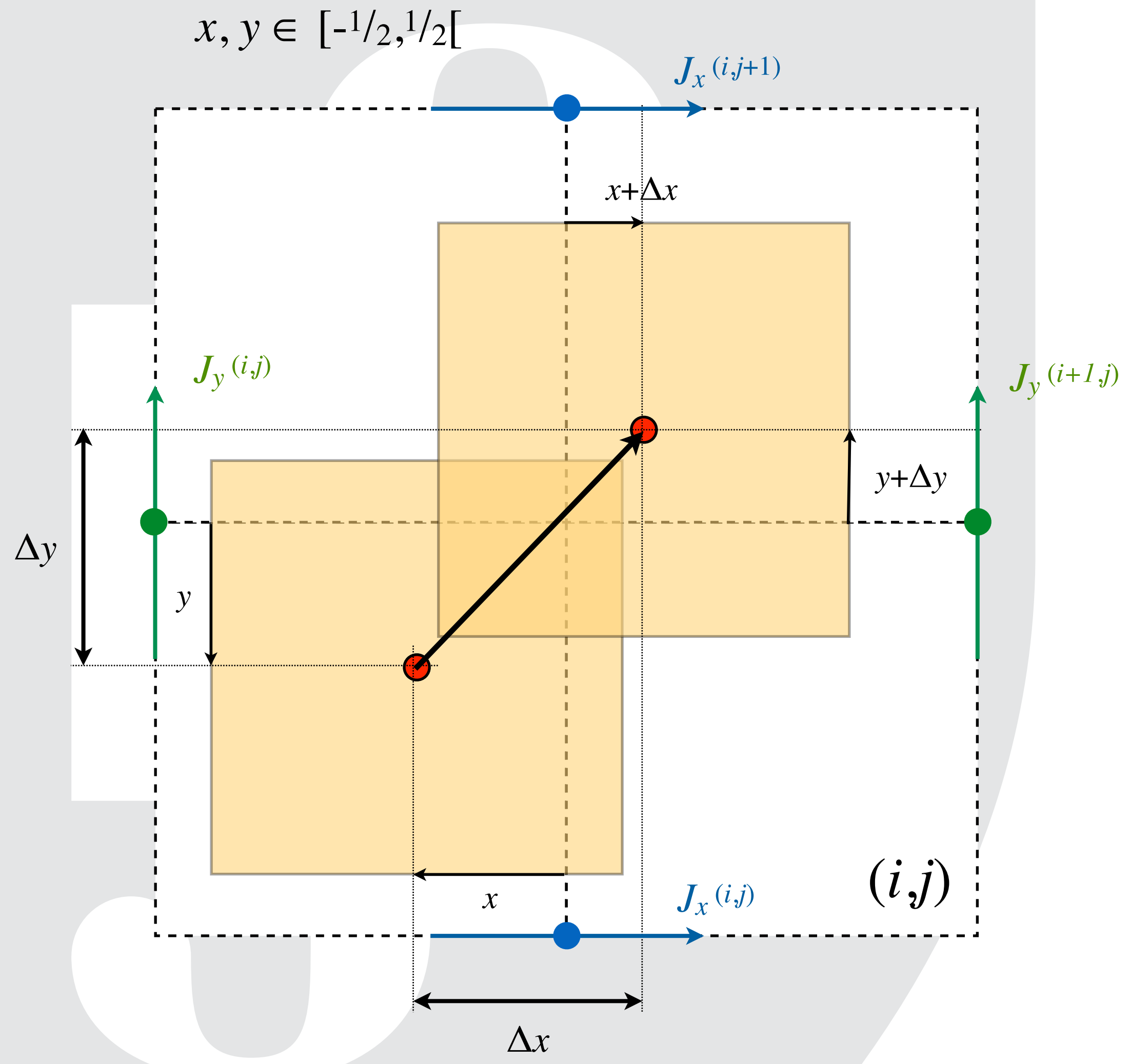
Advance positions

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

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



Advancing the EM fields

- **EM Fields are advanced in time using Maxwell's equations using the deposited current as source terms**

- Rearrange Ampère'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

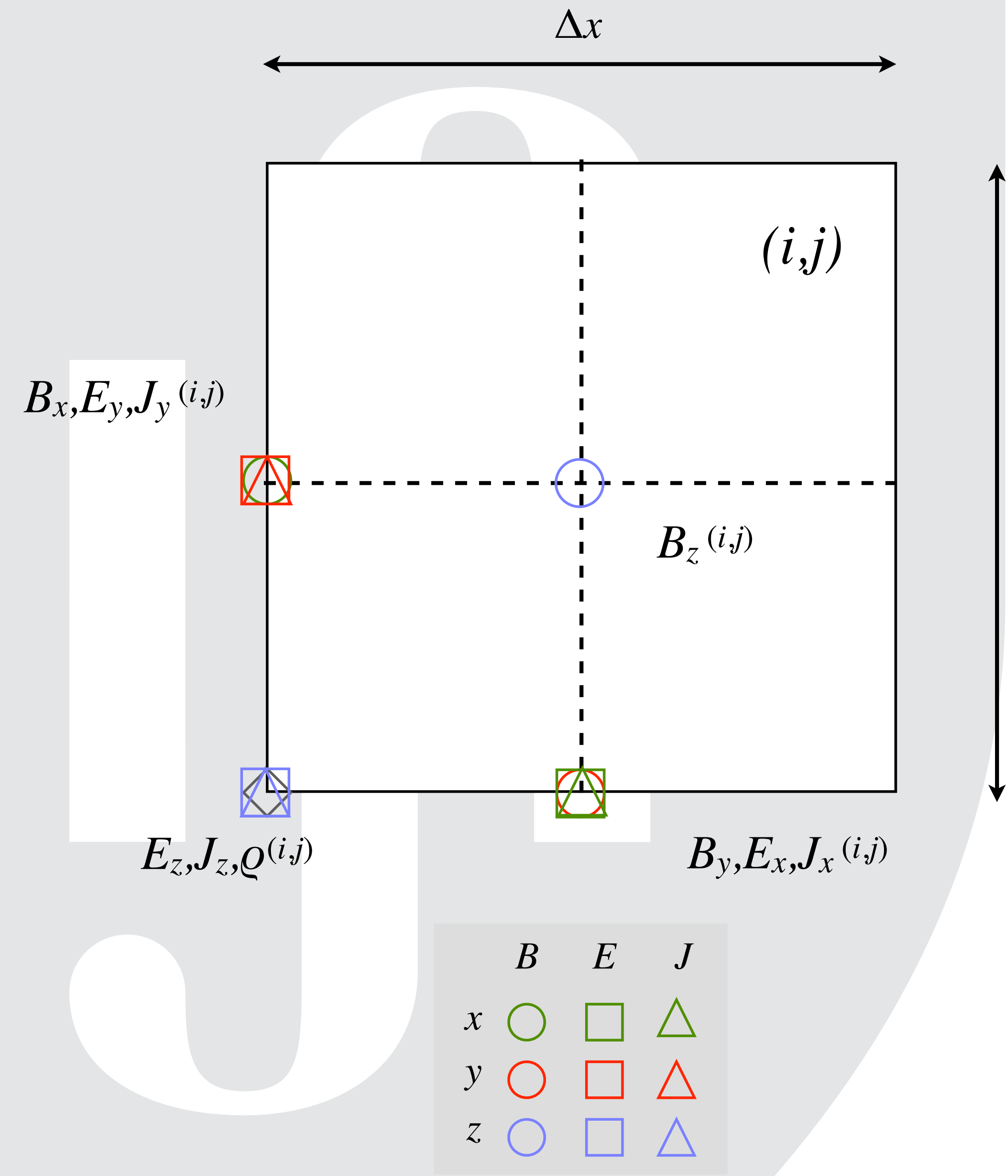
- **E** is defined at times t^n

- **B** and **j** are defined at times $t^{n+1/2}$

- **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

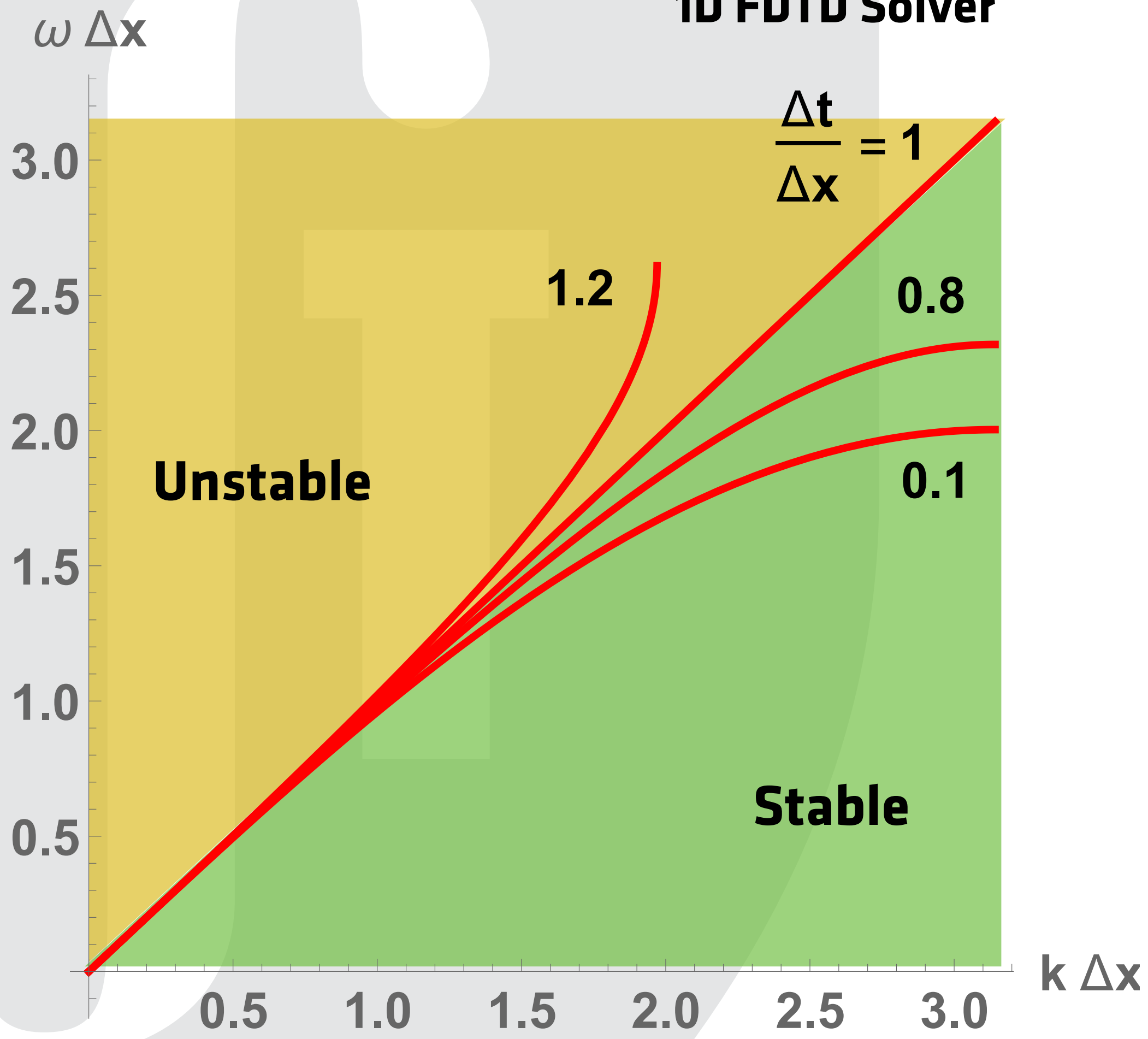
- Choice of time-step is dominated by the FDTD solver (in sim. units):

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

Numerical dispersion relation for 1D FDTD Solver



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 density (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



Modelling Laser Wakefield Acceleration

Laser Wakefield Acceleration

3D Simulation using the OSIRIS code

Choose the normalization

Plasma sets reference

Plasma density is unity

- Normalize lengths to plasma skin depth and frequency to plasma frequency

Example

- Plasma density $n_p = 10^{18} \text{ cm}^{-3}$
- Plasma frequency $\omega_p \sim 5.64 \times 10^{13} \text{ rad s}^{-1}$
- Laser wavelength $\lambda_0 = 1 \mu\text{m}$
- Laser frequency $\omega_0 \sim 2.34 \times 10^{15} \text{ rad s}^{-1}$
- Normalised laser frequency is $\omega_0/\omega_p \sim 41.5$

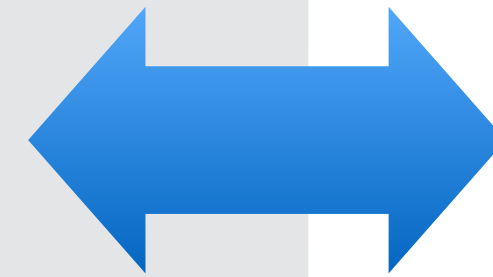
Laser sets reference

Reference laser frequency is unity

- Normalize plasma density to critical density; length to inverse laser wavenumber

Example

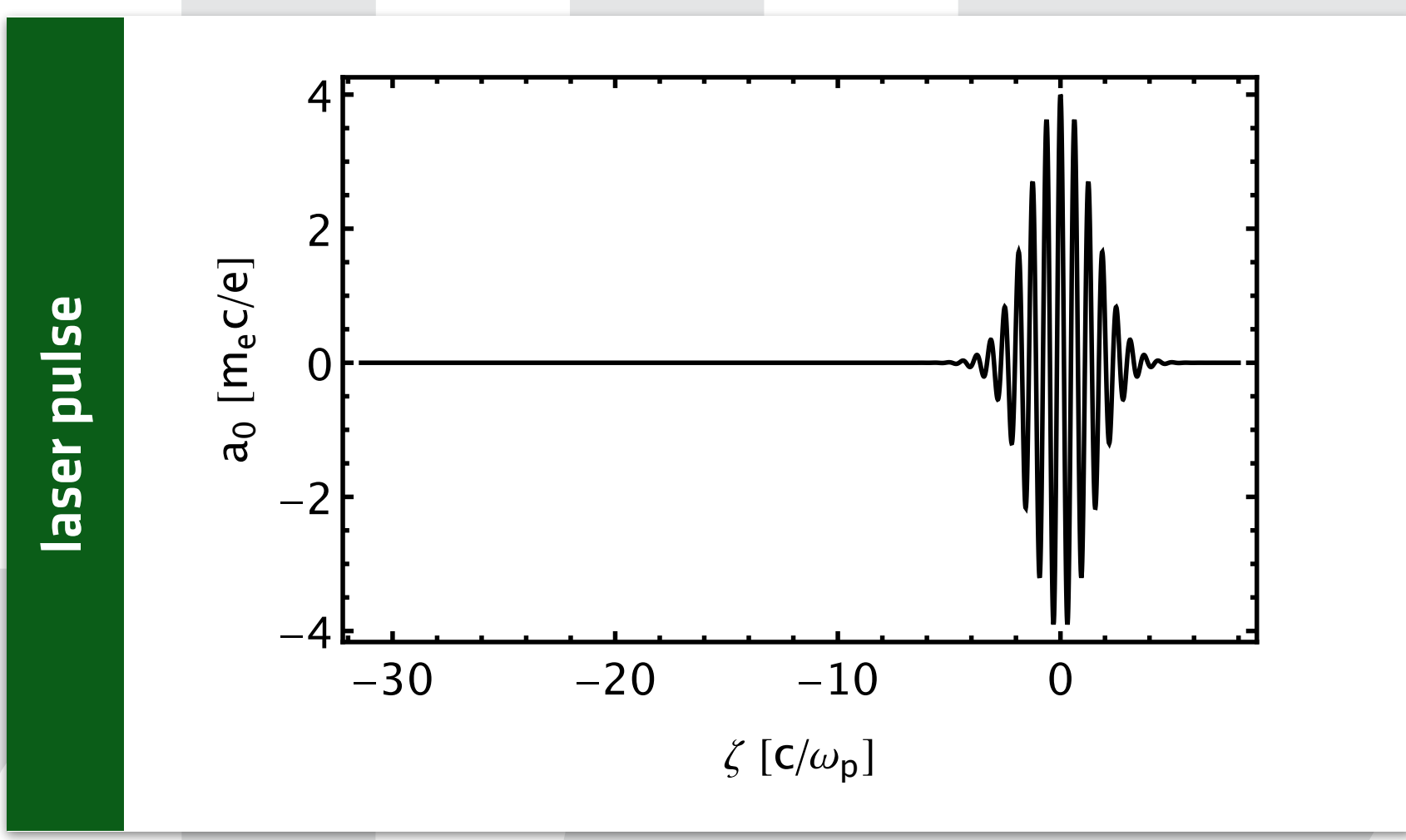
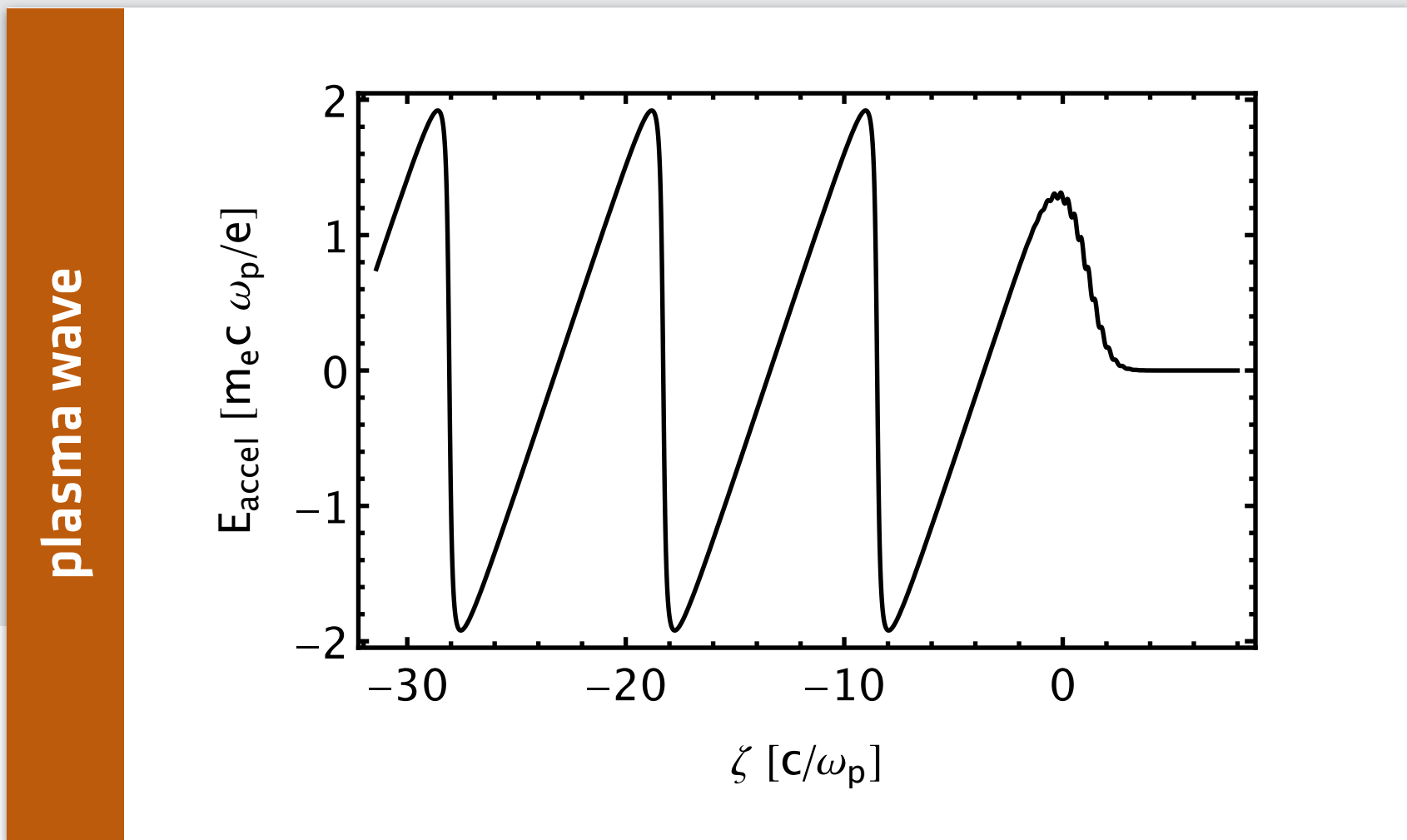
- Laser wavelength $\lambda_0 = 1 \mu\text{m}$
- Laser frequency $\omega_0 \sim 2.34 \times 10^{15} \text{ rad s}^{-1}$
- Critical frequency $n_{\text{crit}} \sim 1.72 \times 10^{21} \text{ cm}^{-3}$
- Plasma density $n_p = 10^{18} \text{ cm}^{-3}$
- Normalized plasma density $n_p/n_{\text{crit}} \sim 5.8 \times 10^{-4}$



Both (and other) normalizations are possible. In this session we will use the plasma as the reference!

Choosing the spatial resolution

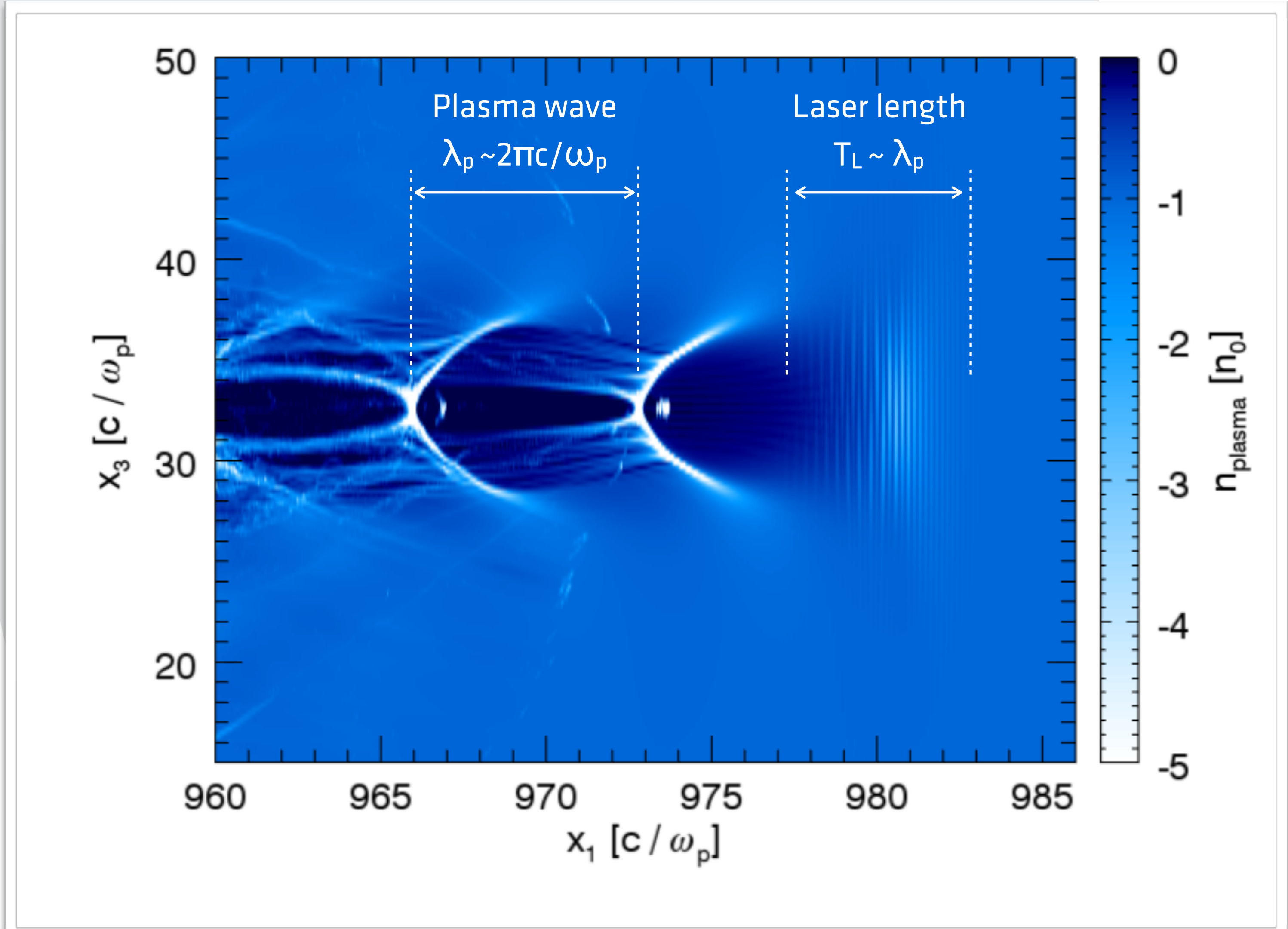
- **Laser propagates in an underdense plasma**
 - $n_p \ll n_{crit} \mid \lambda_0 \ll \lambda_p \mid \omega_p \ll \omega_0$
- **Need to resolve the smallest scale length**
 - > 20 - 30 cells per waveleng
- **Plasma wave**
 - Skin depth sets the plasma scale length
 - $c/\omega_p \sim 5.3 \mu\text{m} / (n_p [10^{18} \text{cm}^{-3}])^{1/2}$
- **Laser**
 - laser wavelength sets the laser scale length
 - $\lambda_0 \sim 1 \mu\text{m} \sim 0.18 (n_p [10^{18} \text{cm}^{-3}])^{1/2} c/\omega_p$



Longitudinal spatial Resolution: $\Delta x \sim \lambda_0 / \# \sim 0.18 / \# (n_p [10^{18} \text{cm}^{-3}])^{1/2} c / \omega_p$
 $\# > 20-30$ (number of cells per laser wavelength)

Simulation box dimensions

- **Simulations are done in a moving window moving at the speed of light**
 - The simulation box does not need to hold the entire propagation length
- **Simulation box needs only to model the relevant structures in the accelerator**
 - Laser driver and initial trailing buckets of accelerating structure
- **Box size determined by largest relevant structures**
 - Longitudinally
 - a few plasma wavelengths long
 - $> 4 \lambda_p \sim 25 c/\omega_p$
 - Transversely (2D)
 - Laser pulse waist / transverse bubble size
 - $> 4 \lambda_p \sim 25 c/\omega_p$



Setting up the simulation: cells, particles

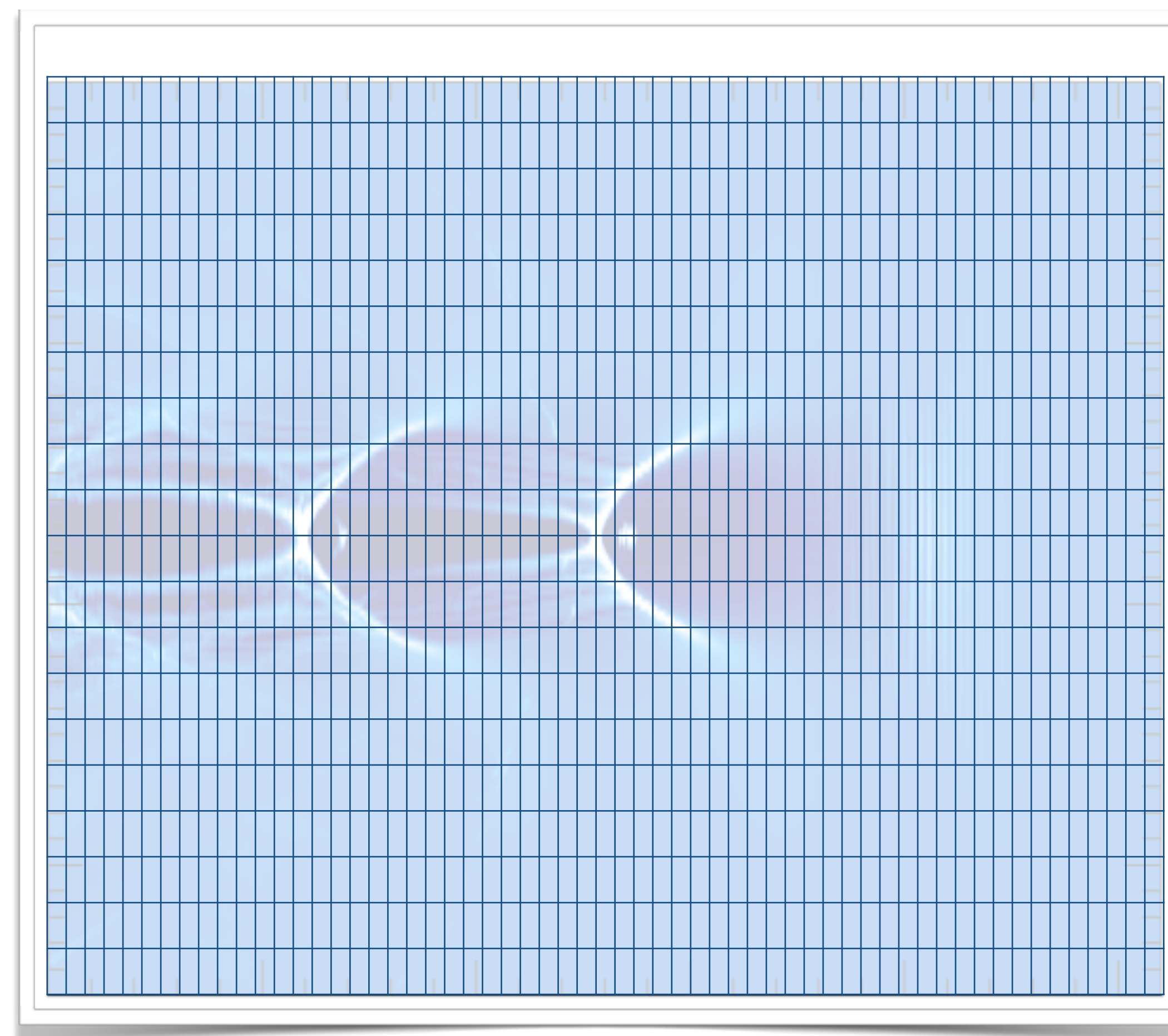
Particles per cell: $\gg 1$ in 1D (e.g. 64) and around 10 in 2D

- **Simulation grid**

- Box length: $L = 4 \lambda_p$
- 20 points per laser wavelength
- $\Delta x \sim \lambda_0/20 \sim 0.18/20 = 0.009 c/\omega_p$
- Number of cells $\sim L / \Delta x \sim 2800$ cells

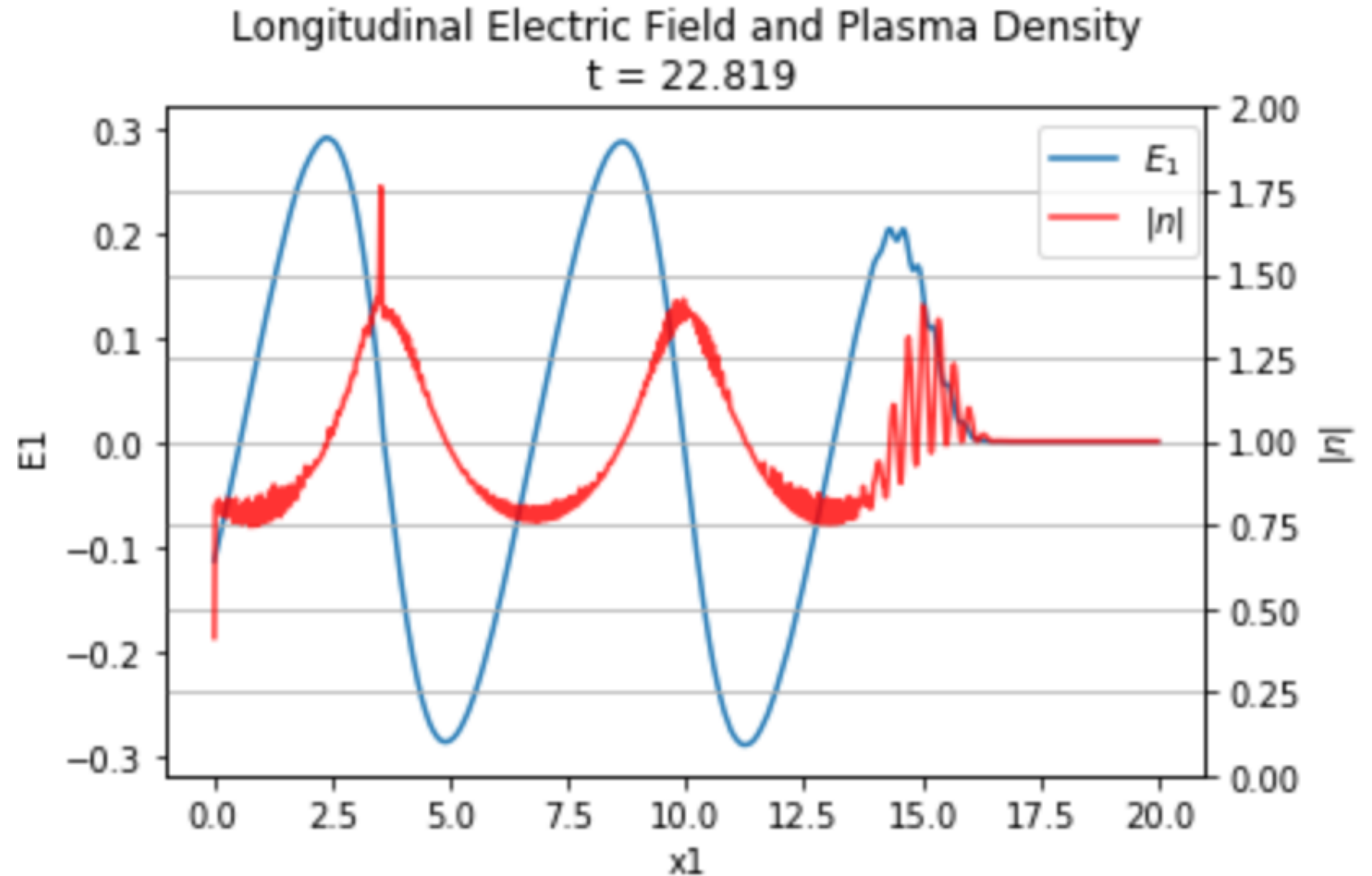
- **Simulation particles**

- Number of particles per cell must resolve local phase space
- $\gg 1$ in 1D (e.g. 64)
- ~ 10 in 2D
- Higher numbers improve phase space resolution (detailed distribution tails)
- Also reduces simulation noise



Longitudinal cells: $4 \lambda_p / (0.18/20 c/\omega_p) \sim 2800$ cells ($n_p = 10^{18} \text{ cm}^{-3}$)

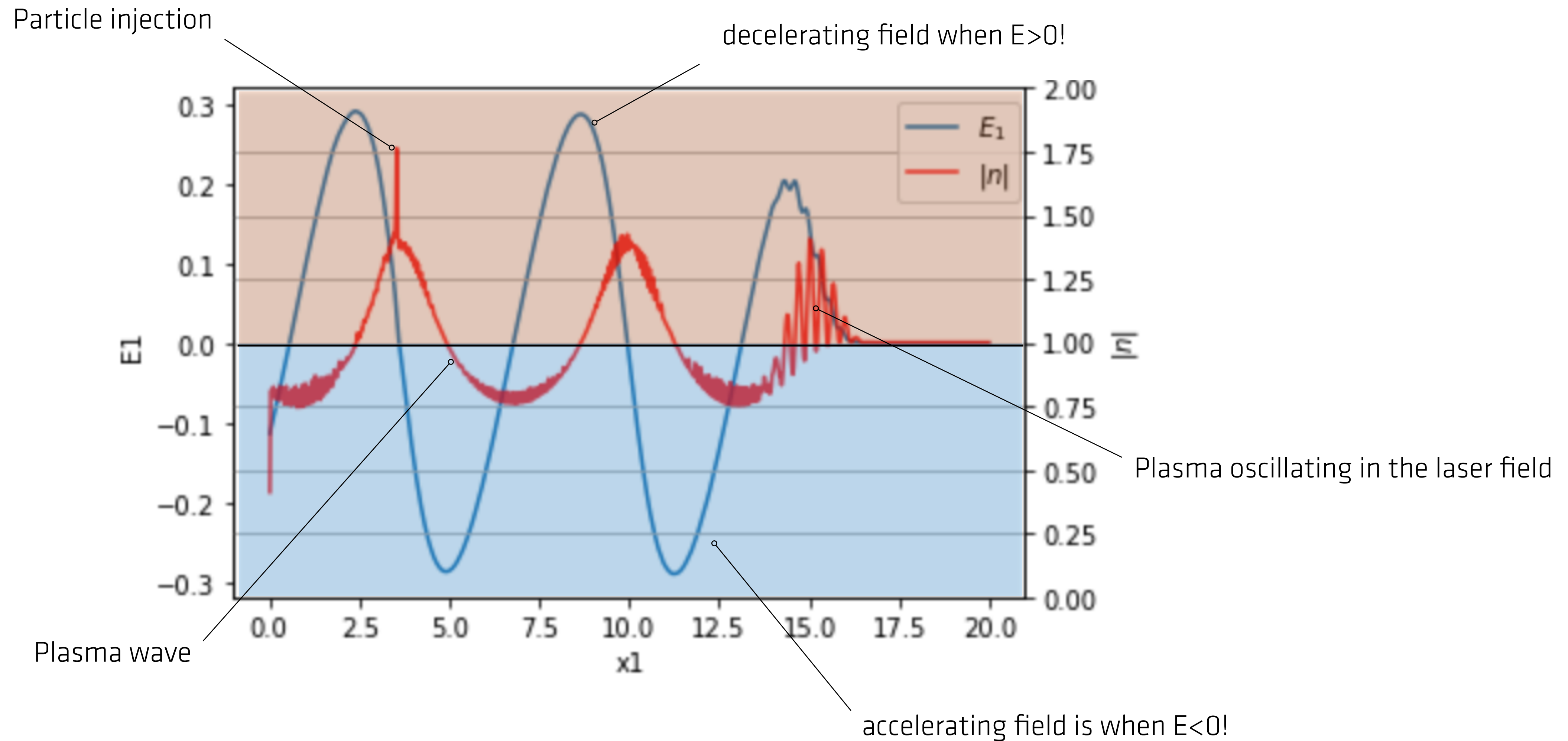
- **Plasma density**
 - Charge density of the background plasma
 - Wave structure and particle loading
- **Longitudinal electric field**
 - Accelerating / decelerating fields
- **Transverse electric field**
 - Laser field
 - Focusing / defocusing fields (2D)
- **Particle phasespace**
 - Show particle momenta as a function of position
 - Most common is u_1/x_1
 - Wave structure and particle acceleration



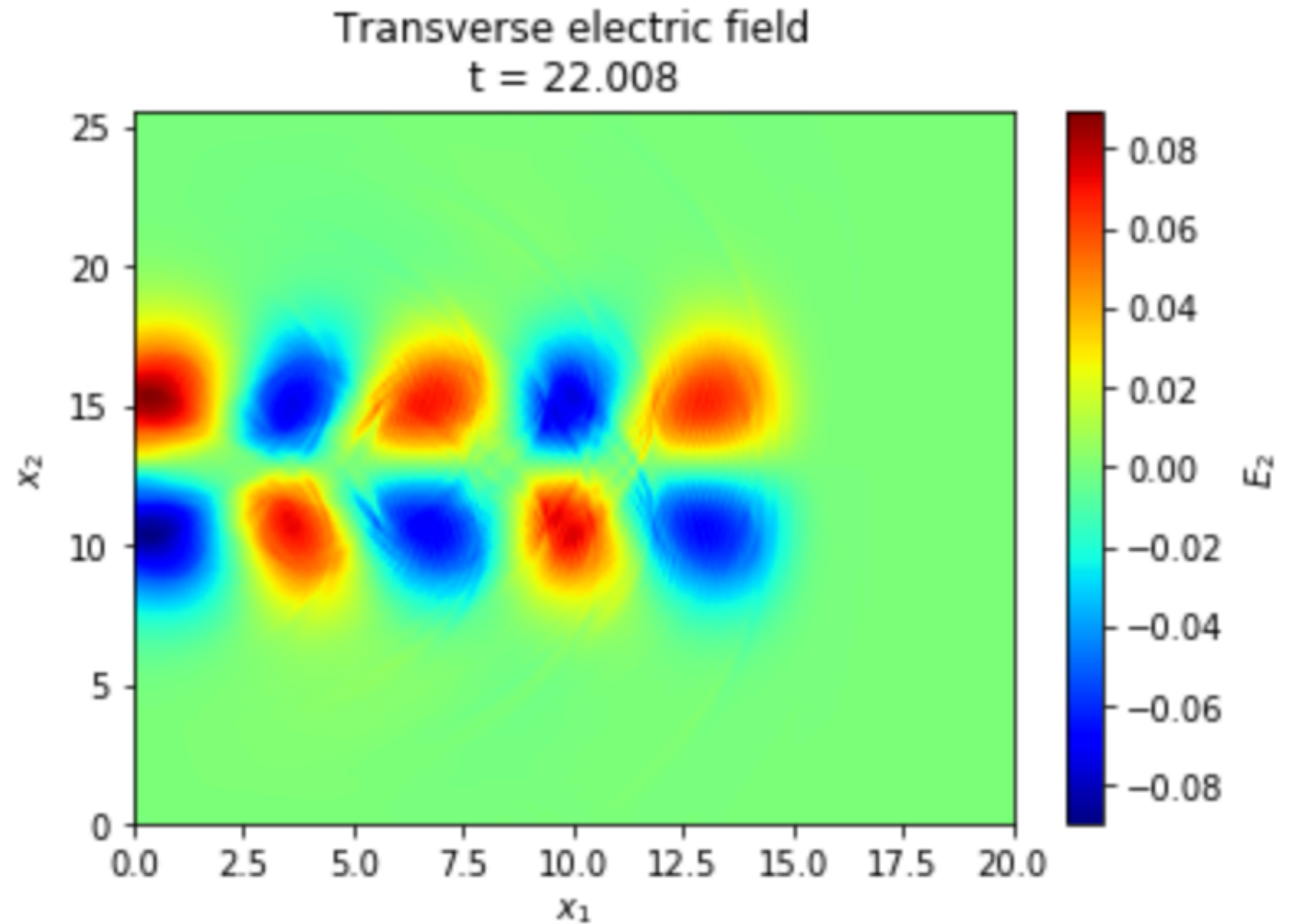
Simulations performed in a moving window that travels at c

Plasma density and longitudinal electric field

Simulations performed in a moving window that travels at c



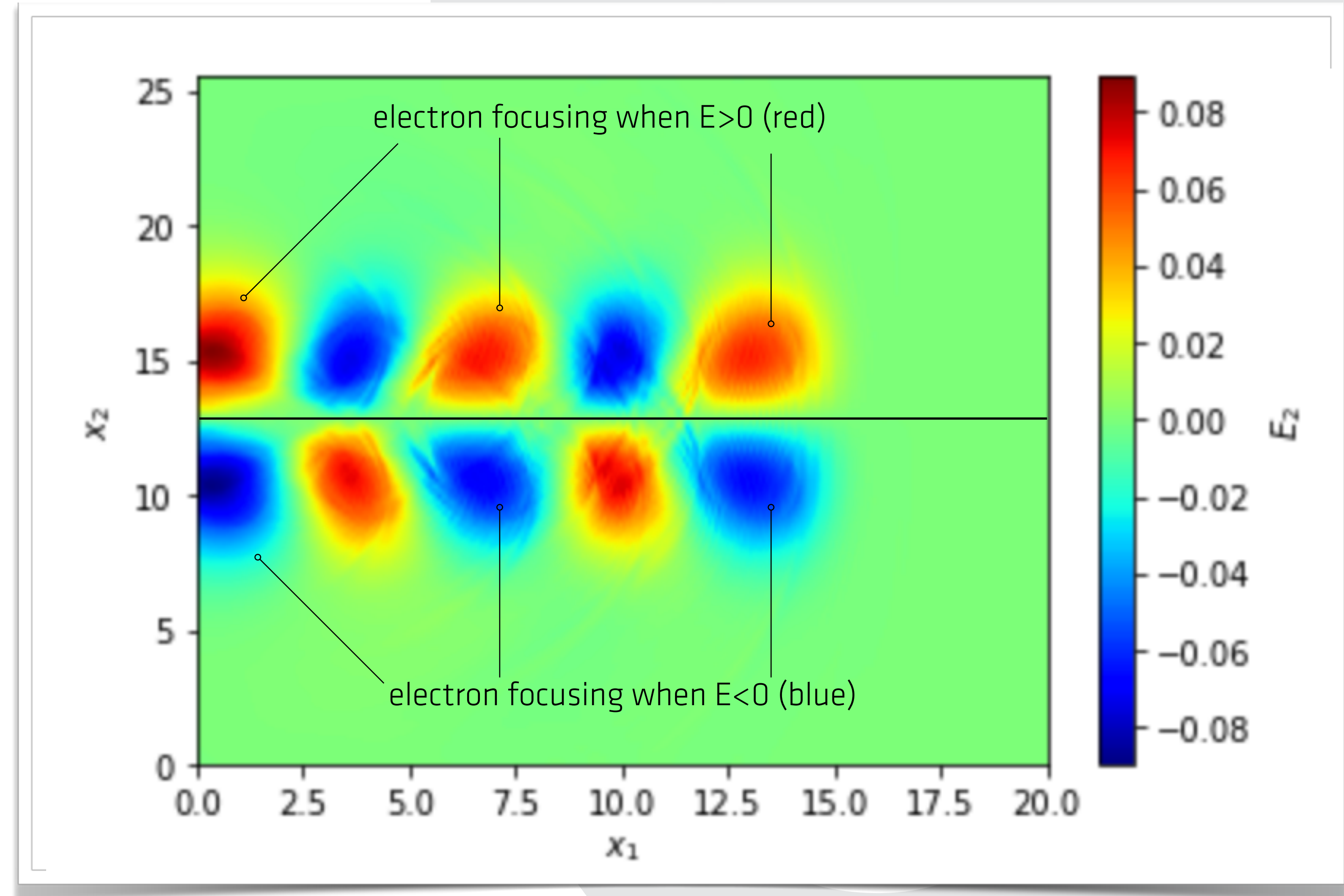
- **Transverse electric fields**
 - Laser pulse
 - Transverse wave structure
- **Also laser pulse**
 - In this example the laser was polarized out of the plane



Simulations performed in a moving window that travels at c

Focusing fields: non relativistic particles

E_{\perp} is the focusing force for non-relativistic particles $[(\mathbf{v} \times \mathbf{B} / c)_{\perp} \ll E_{\perp}]$

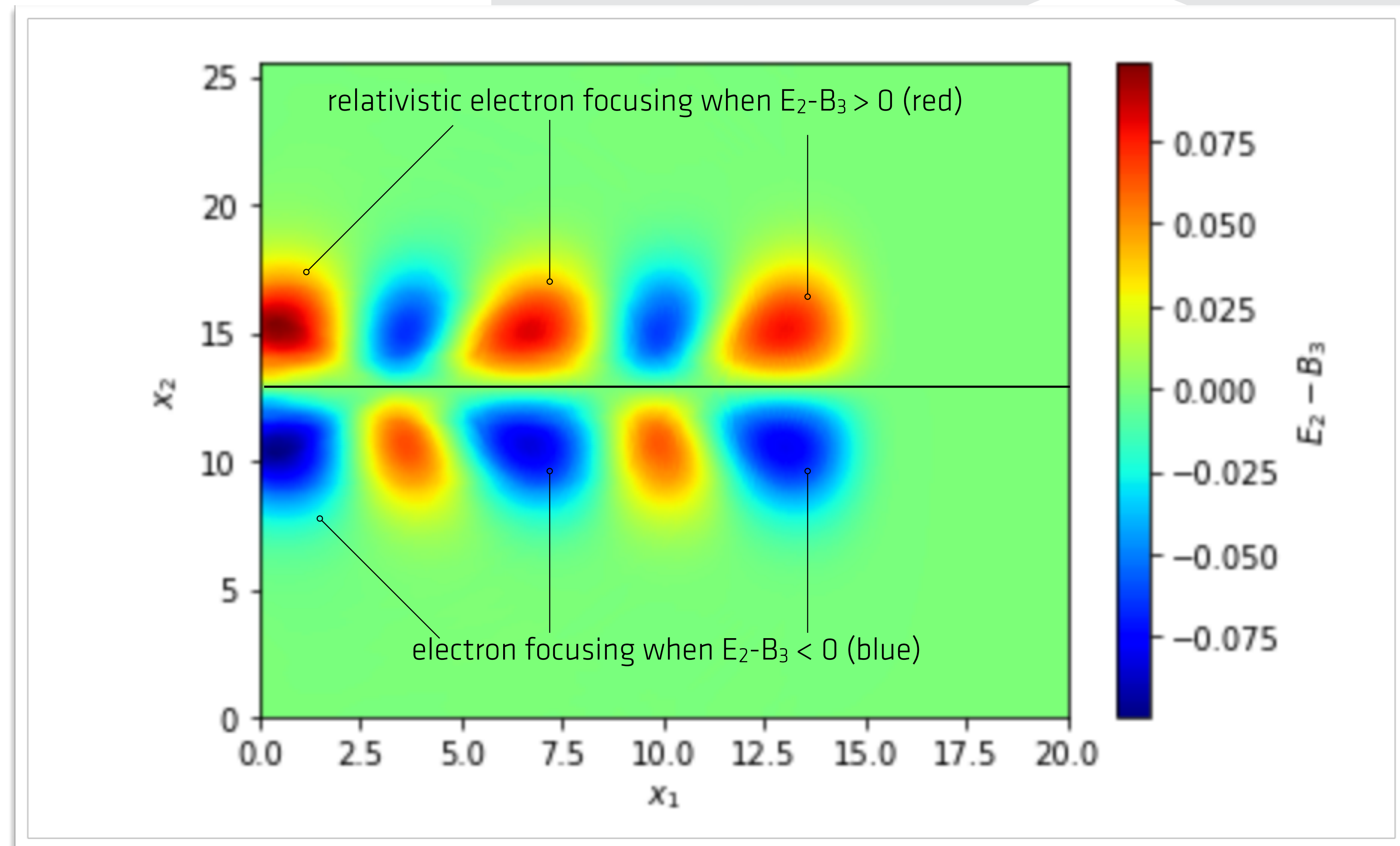


Simulations performed in a moving window that travels at c

Focusing fields: Ultra-relativistic particles

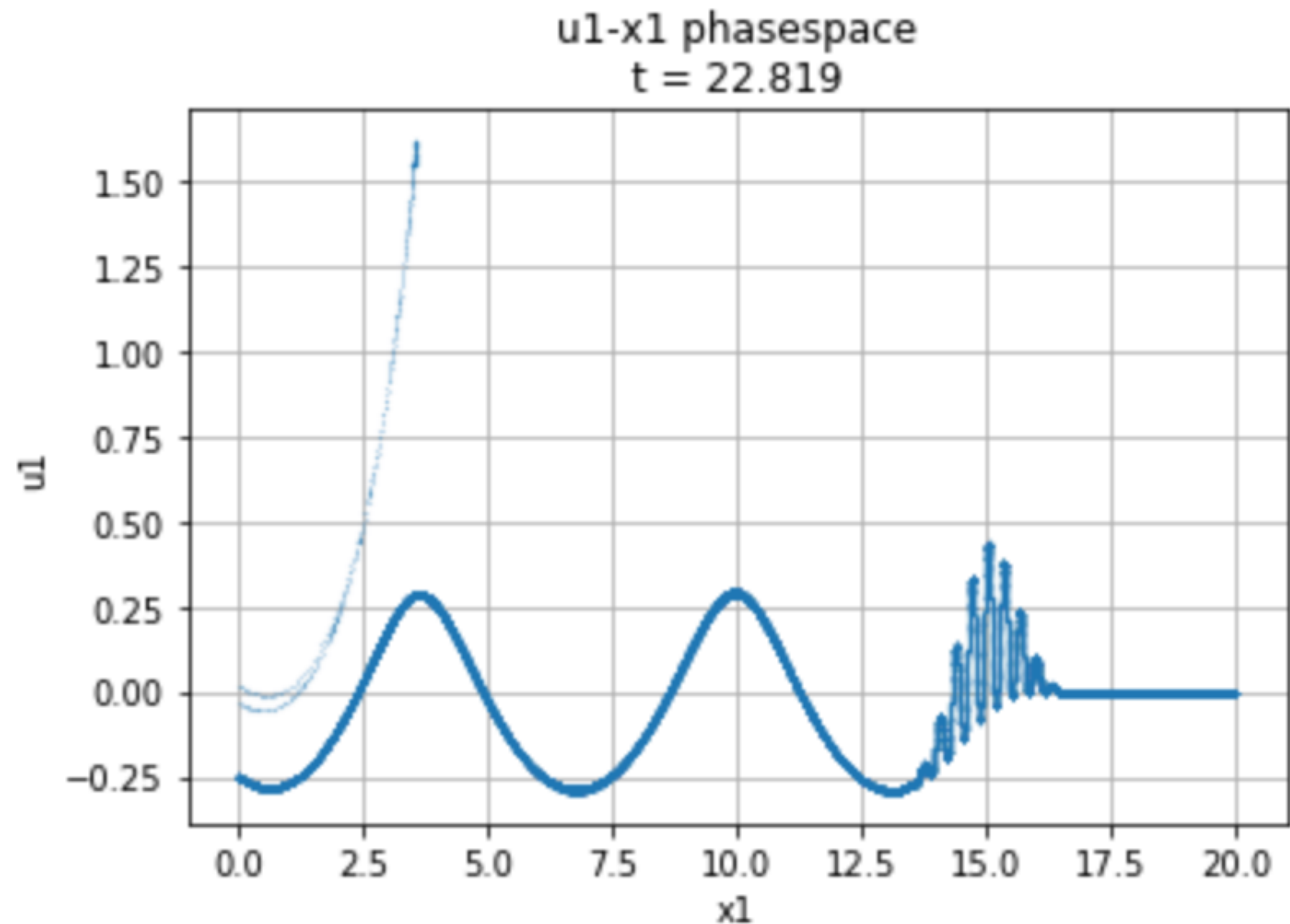
Focusing force for a ultra-relativistic particle:

$$\mathbf{E}_r + \mathbf{v}_{||} \times \mathbf{B}_\theta / c \approx \mathbf{E}_r - B_\theta$$



- **Longitudinal particle phase-space**

- Plots momenta vs. position
- Phasespace density or 1 point per particle
- Most common is longitudinal momenta vs. longitudinal position i.e. u_1-x_1



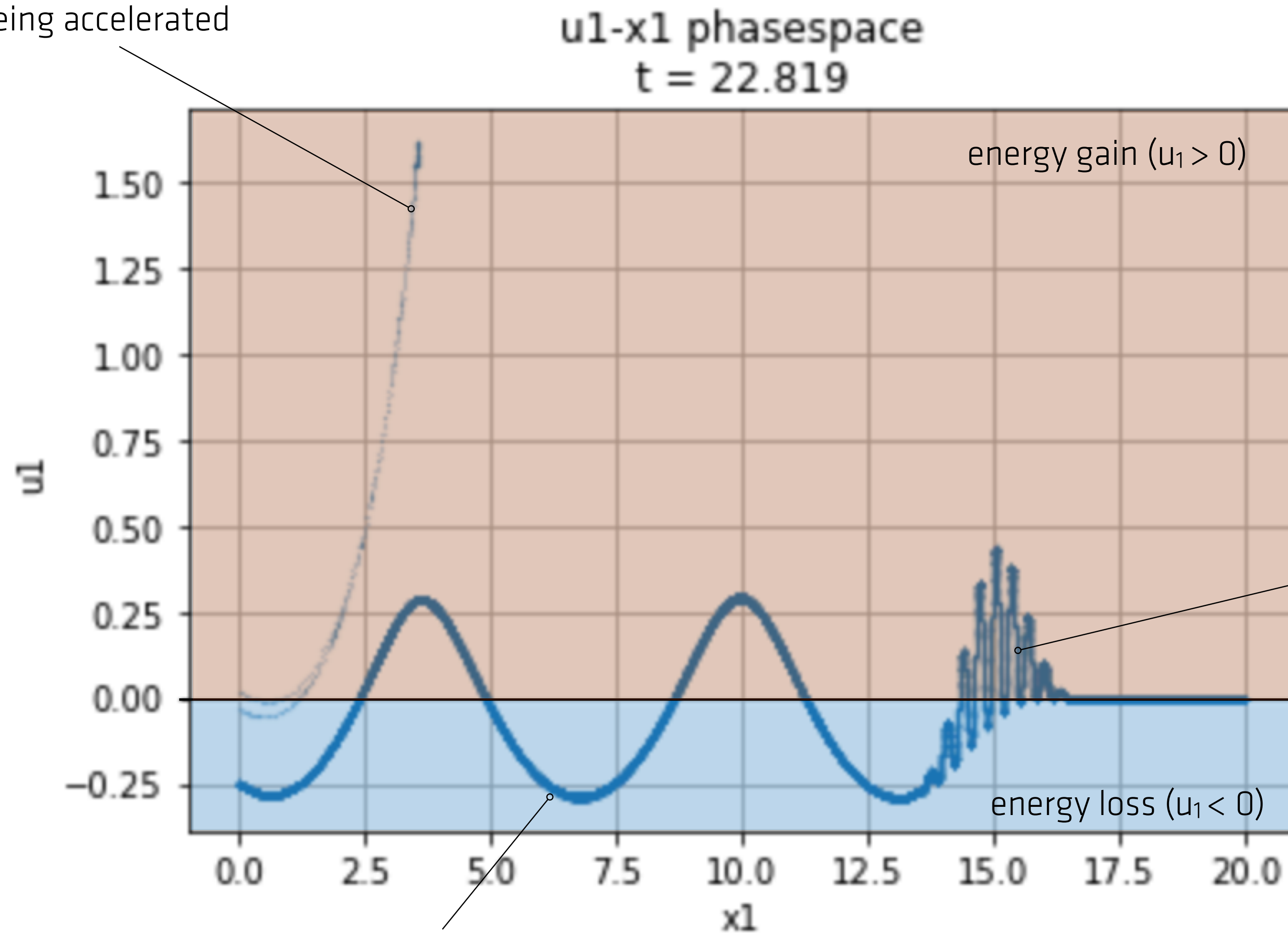
Simulations performed in a moving window that travels at c

Particle acceleration and deceleration



In plasma based acceleration: Energy [$m_e c^2$] = $\gamma - 1 \approx \gamma \approx u_1$ [me c]

Trapped particles being accelerated



Particles oscillating in the laser field

Plasma oscillating in the wakefield



Running ZPIC on your computer

Harvard Mark I - 1944

Rear view of Computing Section

Running ZPIC - Option 1 - compile from source

- **Build from ZPIC source**

- ZPIC itself has no external dependencies, and requires only a C99 compliant C compiler
 - **gcc, clang** and **intel** tested
- The code is open-source and hosted on GitHub
 - <https://github.com/zambzamb/zpic>

- **Build Python interface**

- The Python interface requires a Python3 installation
 - We recommend the Intel Python distribution
- The interface also requires NumPy and Cython packages to be installed
- Just use the Makefile in the python subfolder of the ZPIC distribution
 - This will also compile all of the ZPIC codes

- **Using the Jupyter notebooks**

- Requires a working Jupyter + Python installation
 - Again, we recommend the Intel Python distribution
- Launch Jupyter and open one of the example notebooks

```
python — fish /Users/zamb/Source/zpic/python — -fish
zamb@zamb-lap-2 ~/S/z/python> make
python3 setup.py build_ext -if
Compiling em1d.pyx because it changed.
Compiling em2d.pyx because it changed.
Compiling es1d.pyx because it changed.
Compiling em1ds.pyx because it changed.
Compiling em2ds.pyx because it changed.
[1/5] Cythonizing em1d.pyx
[2/5] Cythonizing em1ds.pyx
[3/5] Cythonizing em2d.pyx
[4/5] Cythonizing em2ds.pyx

pes -I/opt/intel/intelpython3/include -I/opt/intel/intelpython3/include -std=c99 -I. -I/opt/intel/intelpython3/include/python3.6m -c ../em2ds/zdf.c -o build/temp.macosx-10.6-x86_64-3.6/./em2ds/zdf.o
/usr/bin/clang -bundle -undefined dynamic_lookup -L/opt/intel/intelpython3/lib -L/opt/intel/intelpython3/lib -arch x86_64 build/temp.macosx-10.6-x86_64-3.6/em2ds.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/charge.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/current.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/emf.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/fft.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/filter.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/grid2d.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/particles.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/random.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/simulation.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/timer.o build/temp.macosx-10.6-x86_64-3.6/./em2ds/zdf.o -L/opt/intel/intelpython3/lib -o /Users/zamb/Source/zpic/python/em2ds.cpython-36m-darwin.so
zamb@zamb-lap-2 ~/S/z/python> 
```

```
python — jupyter /Users/zamb/Source/zpic/python — jupyter-notebook LWFA 2D.ipynb • python
zamb@zamb-lap-2 ~/S/z/python> jupyter notebook LWFA_2D.ipynb
[I 17:13:47.845 NotebookApp] JupyterLab extension loaded from /opt/intel/intelpython3/lib/python3.6/site-packages/jupyterlab
[I 17:13:47.845 NotebookApp] JupyterLab application directory is /opt/intel/intelpython3/share/jupyter/lab
[I 17:13:47.850 NotebookApp] Serving notebooks from local directory: /Users/zamb/Source/zpic/python
[I 17:13:47.850 NotebookApp] 0 active kernels
[I 17:13:47.850 NotebookApp] The Jupyter Notebook is running at:
[I 17:13:47.850 NotebookApp] http://localhost:8888/?token=676ee830df601408ba79a6ecf0c0db560784fc654521b963
[I 17:13:47.850 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 17:13:47.854 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=676ee830df601408ba79a6ecf0c0db560784fc654521b963
[I 17:13:48.855 NotebookApp] Accepting one-time-token-authenticated connection from ::1
[W 17:13:49.797 NotebookApp] 404 GET /static/components/moment/locale/en-gb.js?v=20190314171347 (:::1) 9.97ms referer=http://localhost:8888/notebooks/LWFA%202D.ipynb
[I 17:13:50.348 NotebookApp] Kernel started: 96761370-79fb-4e91-bf01-6c6f0143cea5
[I 17:13:51.092 NotebookApp] Adapting to protocol v5.1 for kernel 96761370-79fb-4e91-bf01-6c6f0143cea5

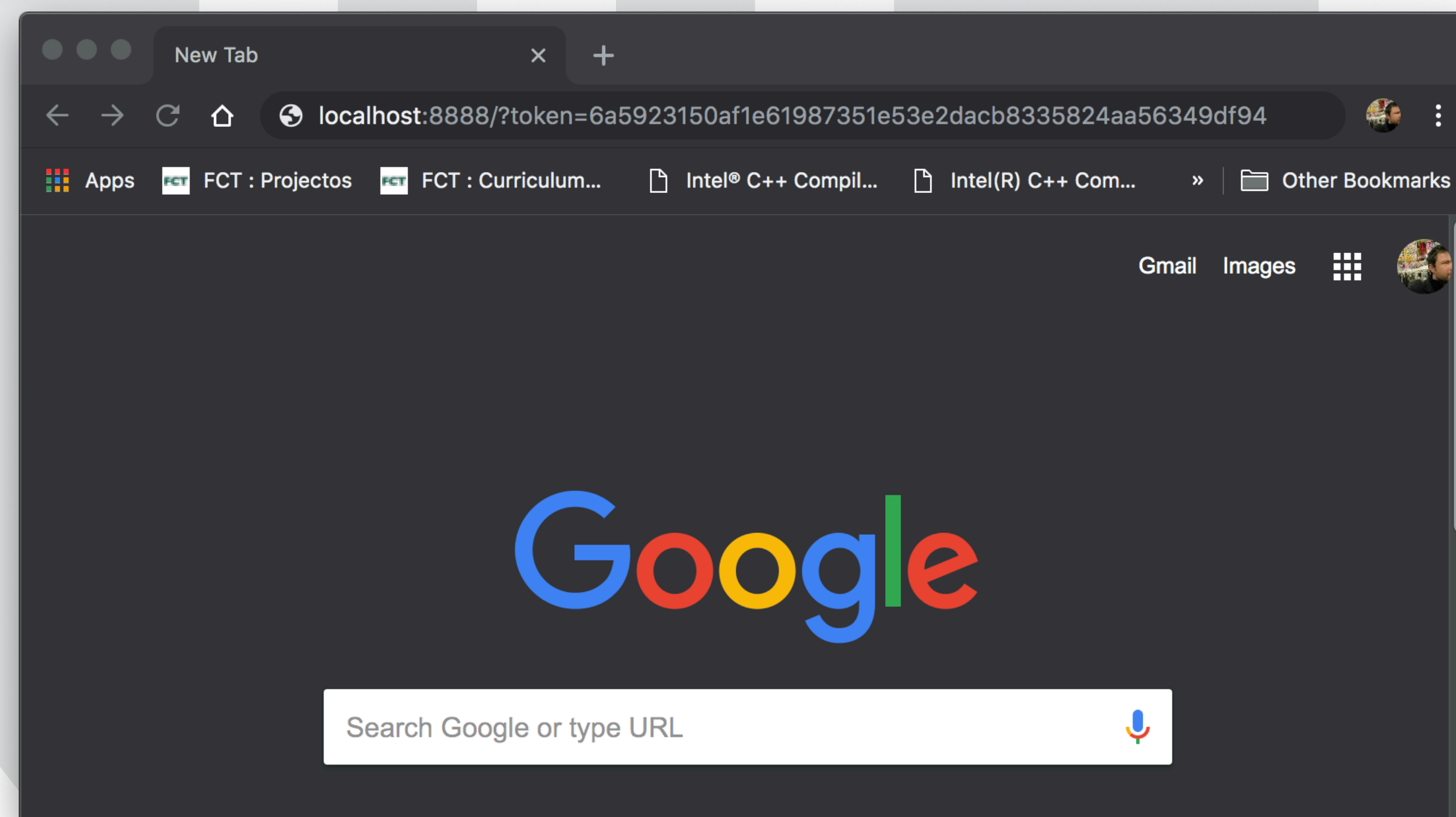
```


Running ZPIC - Option 2 - use a Docker container

- **Install Docker desktop on your computer**
 - Available for free at:
 - <https://www.docker.com/products/docker-desktop>
- **Run the ZPIC image**
 - The ZPIC container image is hosted on DockerHub
 - Open a terminal window and type the following command
 - `> docker run -p 8888:8888 -t zamb/zpic`
 - The first time you do it, it will download the ZPIC container image. This can take a little time.
- **Open a web browser on your computer and point it to the appropriate port**
 - Type in the following as the address
 - `localhost:8888/?token=[TOKEN]`
 - Get the [TOKEN] value from the output of the docker run command
 - The port number must match the docker run command

```
zamb@zamlap-2 ~> docker run -p 8888:8888 -t --rm zamb/zpic
Executing the command: jupyter notebook
[I 17:06:34.455 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
[I 17:06:34.668 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.7/site-packages/jupyterlab
[I 17:06:34.668 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 17:06:34.670 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 17:06:34.670 NotebookApp] The Jupyter Notebook is running at:
[I 17:06:34.670 NotebookApp] http://(d02798c226cc or 127.0.0.1):8888/?token=0dd946005de0e6db9083ca039ea66faffd24cd51bdd8d55d
[I 17:06:34.671 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 17:06:34.671 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://(d02798c226cc or 127.0.0.1):8888/?token=0dd946005de0e6db9083ca039ea66faffd24cd51bdd8d55d
```



Using ZPIC Notebooks

- **Jupyter notebooks**
 - Similar to Mathematica notebooks but for Python
 - Run in a web browser
 - Organized in a sequence of cells
 - Each cell can contain Python code or annotations
- **The code is runs inside the notebook**
 - Initialize the simulation
 - Run to specified time
 - Access simulation data directly to visualize output
 - Several examples provided
- **Saving simulation output not necessary**
 - Example simulations run in ~ 1 minute
 - Visualize results in the notebook
 - Interactively modify simulation parameters
 - If required (e.g. for longer simulations) the code can save simulation results to disk
 - Files are saved in the ZDF format
 - a Python module is provided to read these files

The screenshot shows a Jupyter notebook interface with the following code in a cell:

```
w = np.linspace(1e-6, wC - 1e-6, num = 512)
k = w * np.sqrt( 1.0 - 1.0/(w**2 * (1-Bx0(w) ) ) )
plt.plot( k, w, label = "R-wave")

plt.ylim(0,12)
plt.xlim(0,12)
plt.xlabel("$k$ [$\omega_n/c$]")
plt.ylabel("$\omega$ [$\omega_n$]")
plt.title("R/L-waves dispersion relation")

plt.legend()

plt.show()
```

Below the code, a plot titled "R/L-waves dispersion relation" is displayed. The x-axis is labeled $k [\omega/c]$ and the y-axis is labeled $\omega [\omega_n]$. The plot shows two curves: a blue line for "L-wave" and a red line for "R-wave". Both curves show a non-linear relationship between k and ω .

The screenshot shows a Jupyter notebook interface with the following code in a cell:

```
In [3]: import matplotlib.pyplot as plt

range = [[0,sim.box[0]],[0,sim.box[1]]]

plt.imshow( sim.emf.Ex, interpolation = 'bilinear', origin = 'lower',
            extent = ( range[0][0], range[0][1], range[1][0], range[1][1] ),
            aspect = 'auto', cmap = 'jet')

plt.colorbar().set_label('$E_1$')
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.title("Accelerating Field\n t = {:g}".format(sim.t))

plt.show()
```

Below the code, a heatmap titled "Accelerating Field" is displayed. The title also includes "t = 22.008". The x-axis is labeled x_1 and the y-axis is labeled x_2 . The plot shows a color map of the electric field E_1 with a color bar on the right ranging from -0.2 to 0.2. The plot shows a central region of high field intensity (red/yellow) surrounded by lower intensity regions (green/blue).



Hands-on



CERN Large Hadron Collider
Accelerator Tunnel

- **Option 1 - Compile from source**

- i. Compile the code

- ii. Launch the Jupyter notebook from the source folder:

```
> jupyter notebook LWFA1D.ipynb
```

- **Option 2 - Use a Docker Container**

- i. Install Docker

- ii. Launch the zpic container

```
> docker run -p 8888:8888 -t -v $PWD:/home/jovyan/work zamb/zpic
```

- This mounts the directory **\$PWD** on the directory **work** on your container so you can save changes to the existing notebooks or create new ones

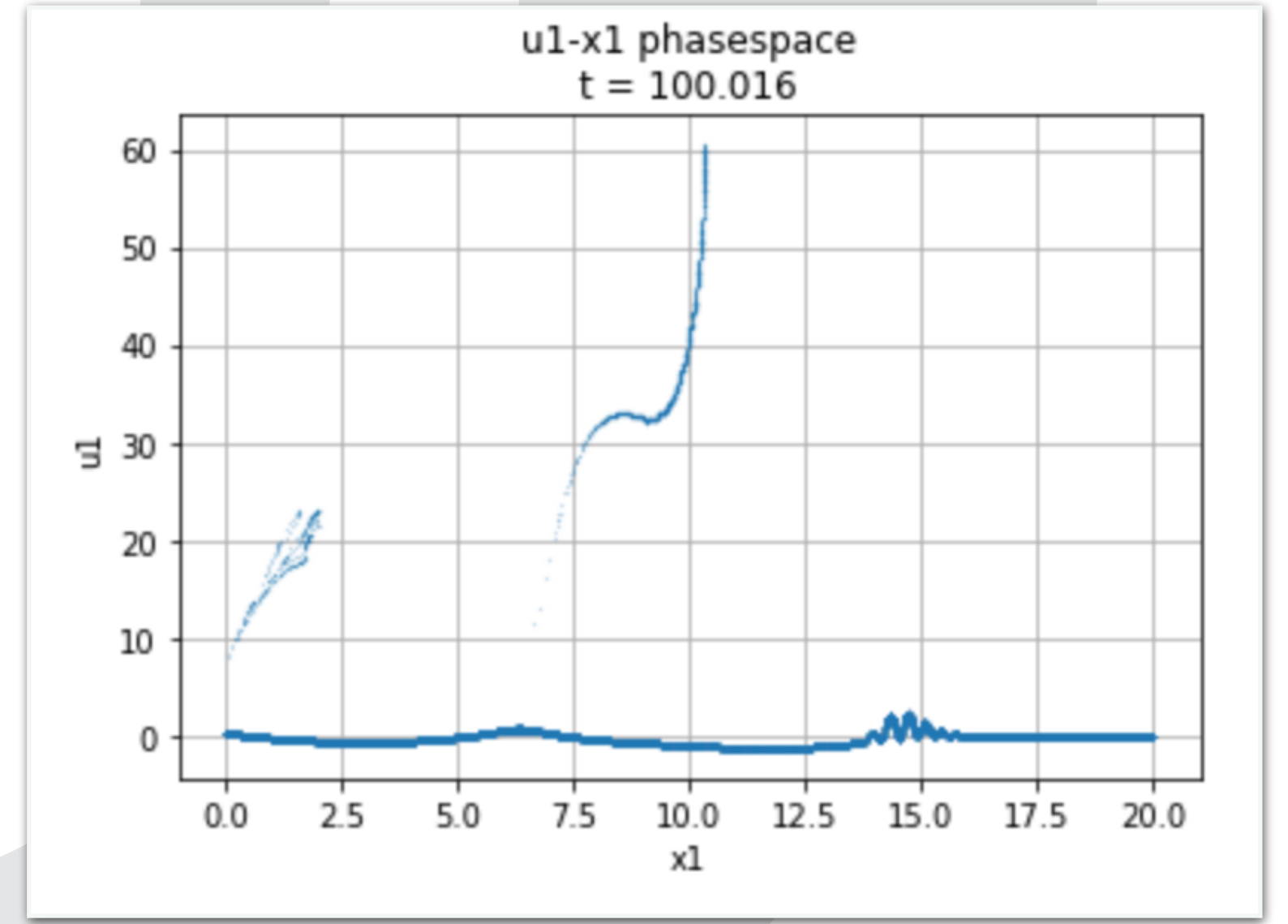
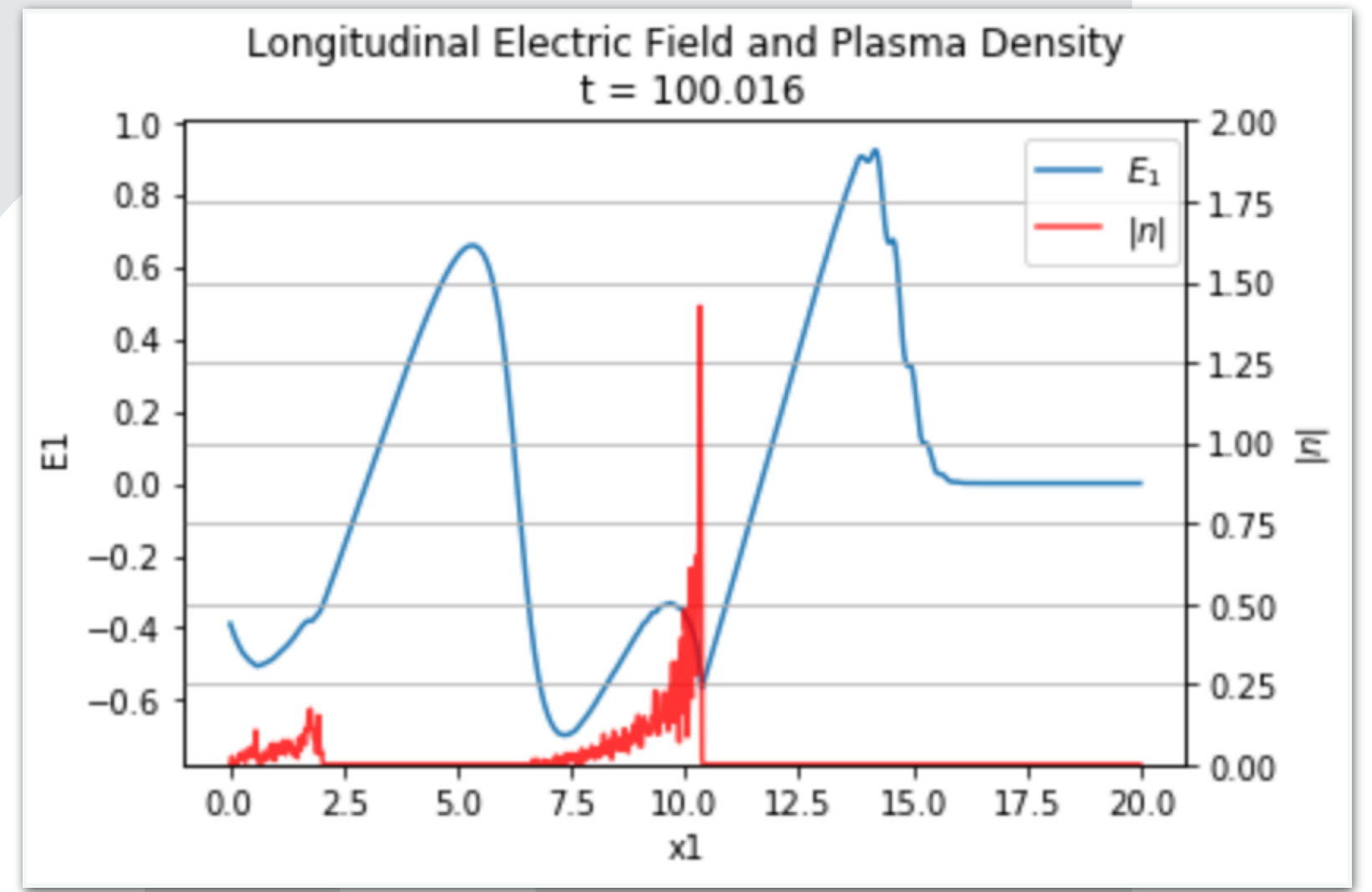
Example: Laser Wakefield Accelerator

Simulate a laser wakefield accelerator:

- Add an ultra-intense laser beam as a driver ($a_0 \sim 2$)
- Choose laser length smaller than λ_p

Questions:

1. can you observe particle injection and trapping?
2. is the energy gain consistent with the longitudinal electric field values?
3. describe and justify the shape for the plasma electric field in the region where particles accelerate
4. could you accelerate positrons in this plasma wave? where would you place them and with what initial velocity/energy? try to simulate!



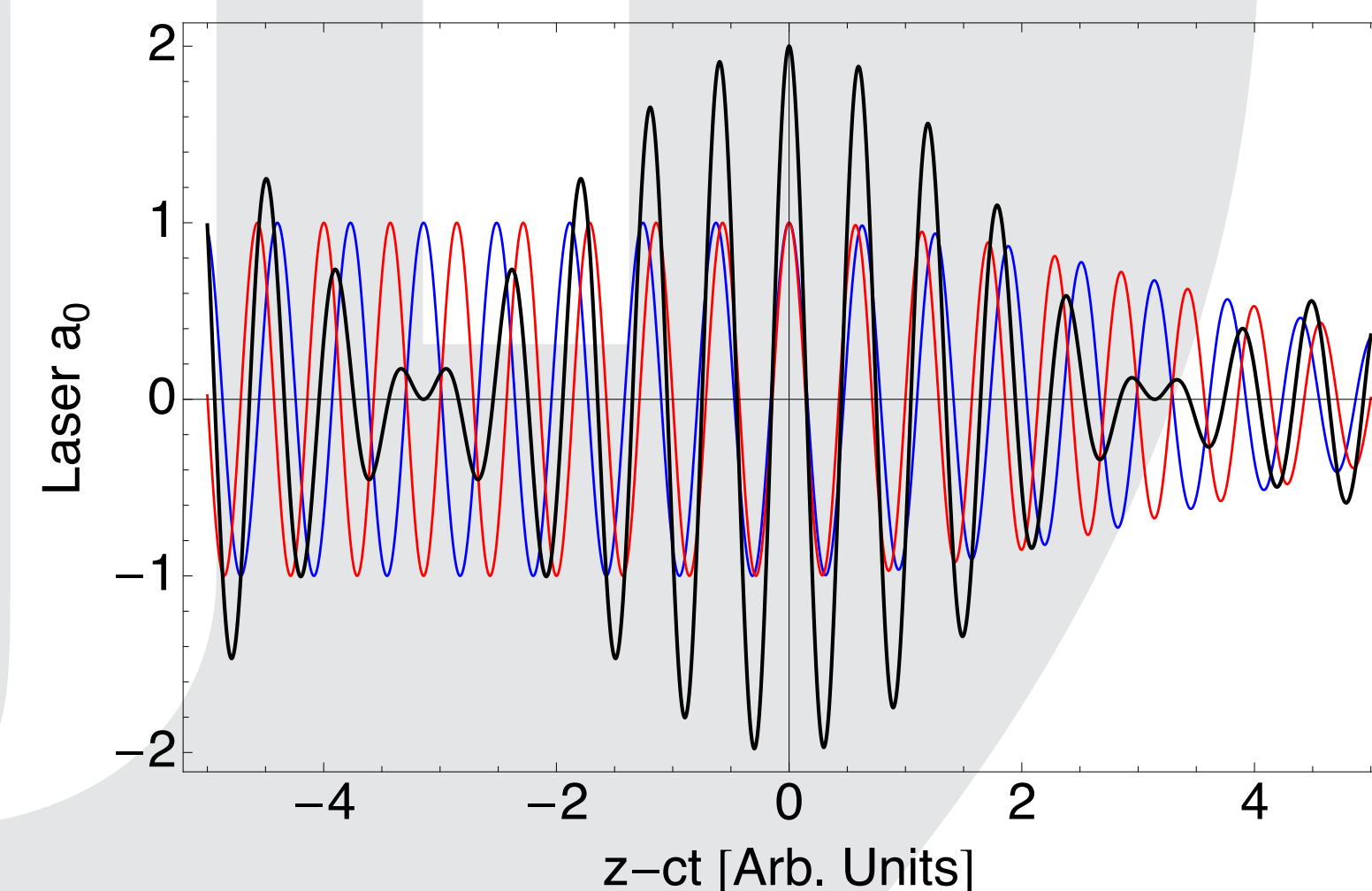
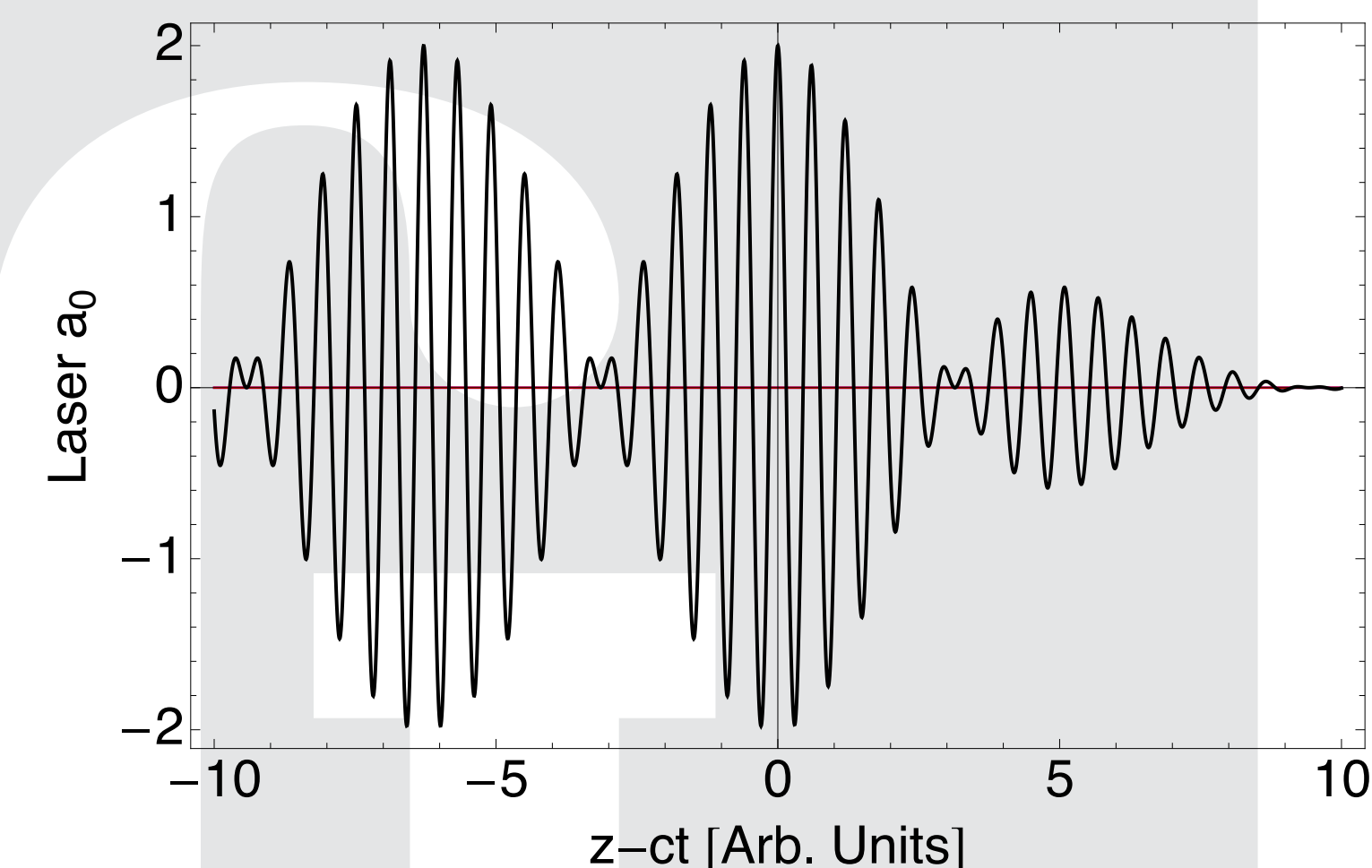
Example: plasma beat wave accelerator

Simulate a plasma beat-wave accelerator:

- Super-impose three laser modes with frequencies differing by ω_p (e.g. $\omega_0 = 10, 11 \omega_p$)
- Choose Laser length $\gg \lambda_p$

Questions:

1. why does the plasma wave amplitude increase along the pulse?
2. what happens if the the initial frequencies of the lasers are not separated by the plasma frequency? Why?
3. compare the trapping threshold, as a function of the peak laser a_0 , for a standard LWFA (with pulse length smaller than λ_p) with the beat-wave accelerator. Which one is the lowest?
4. Decrease the amplitude of the laser side bands and run the simulation for longer times. What happens to the laser?



Example: plasma wakefield accelerator

Simulate a plasma wakefield accelerator:

- Use an ultra-relativistic particle beam as a driver
 - e.g. $u_{fl} = 100$, length $\sim 10 c/\omega_p$, density ~ 0.3
- Choose plasma length $\gg \lambda_p$

Questions:

1. Why does the head of the driver loose energy?
2. What happens to the energy of the driver if it has a length comparable to λ_p ?
3. What is the phase velocity of the plasma wave?
4. Can you observe plasma electron trapping and acceleration?

