Modelling of ultra-intense laser propagation in plasmas and laser-plasma accelerators: fundamentals

Laserlab-Europe

Jorge Vieira[1], **R. A. Fonseca**[1,2]

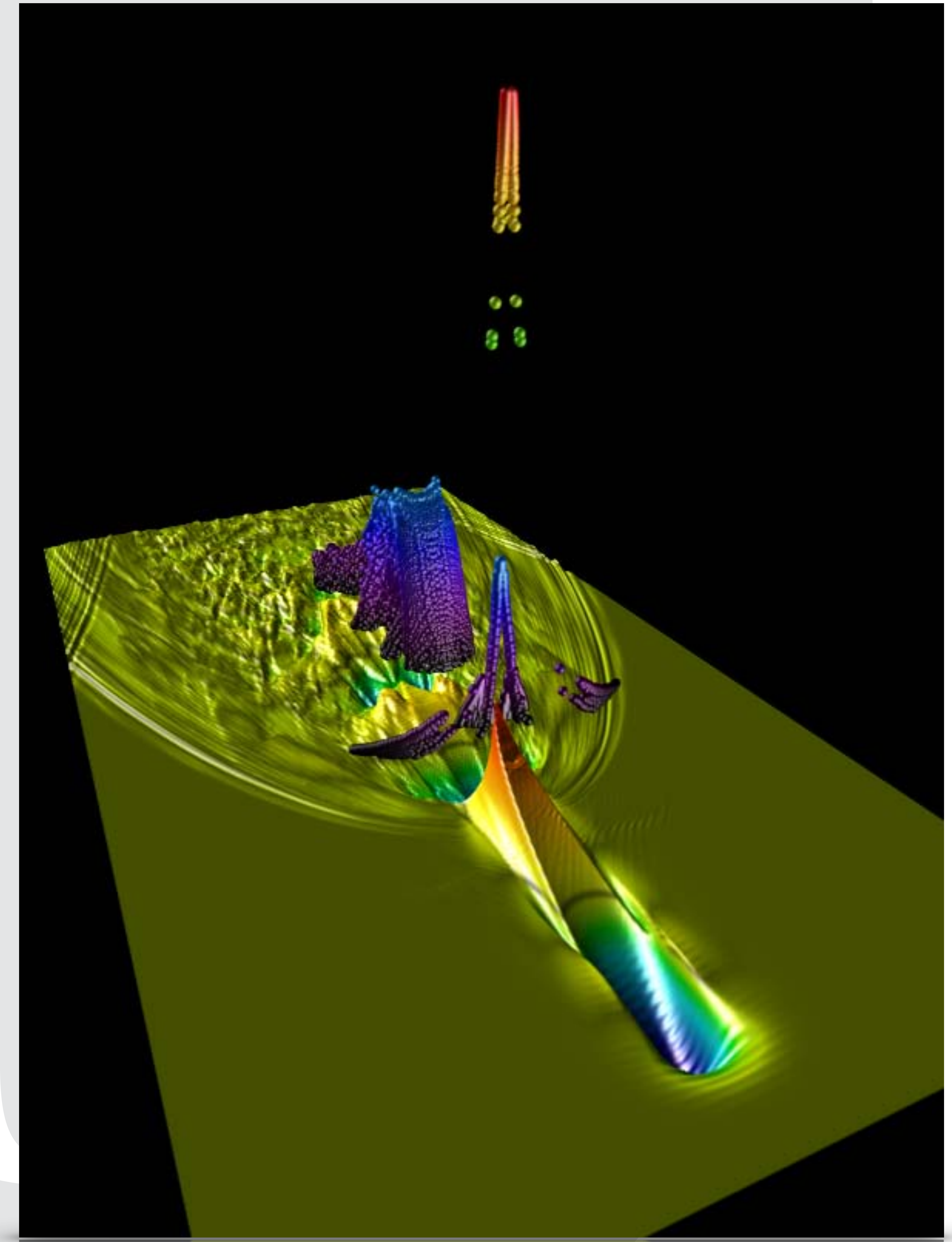
[1]GoLP/IPFN, Instituto Superior Técnico, Lisboa, Portugal
[2] DCTI, ISCTE-Instituto Universitário de Lisboa, Portugal

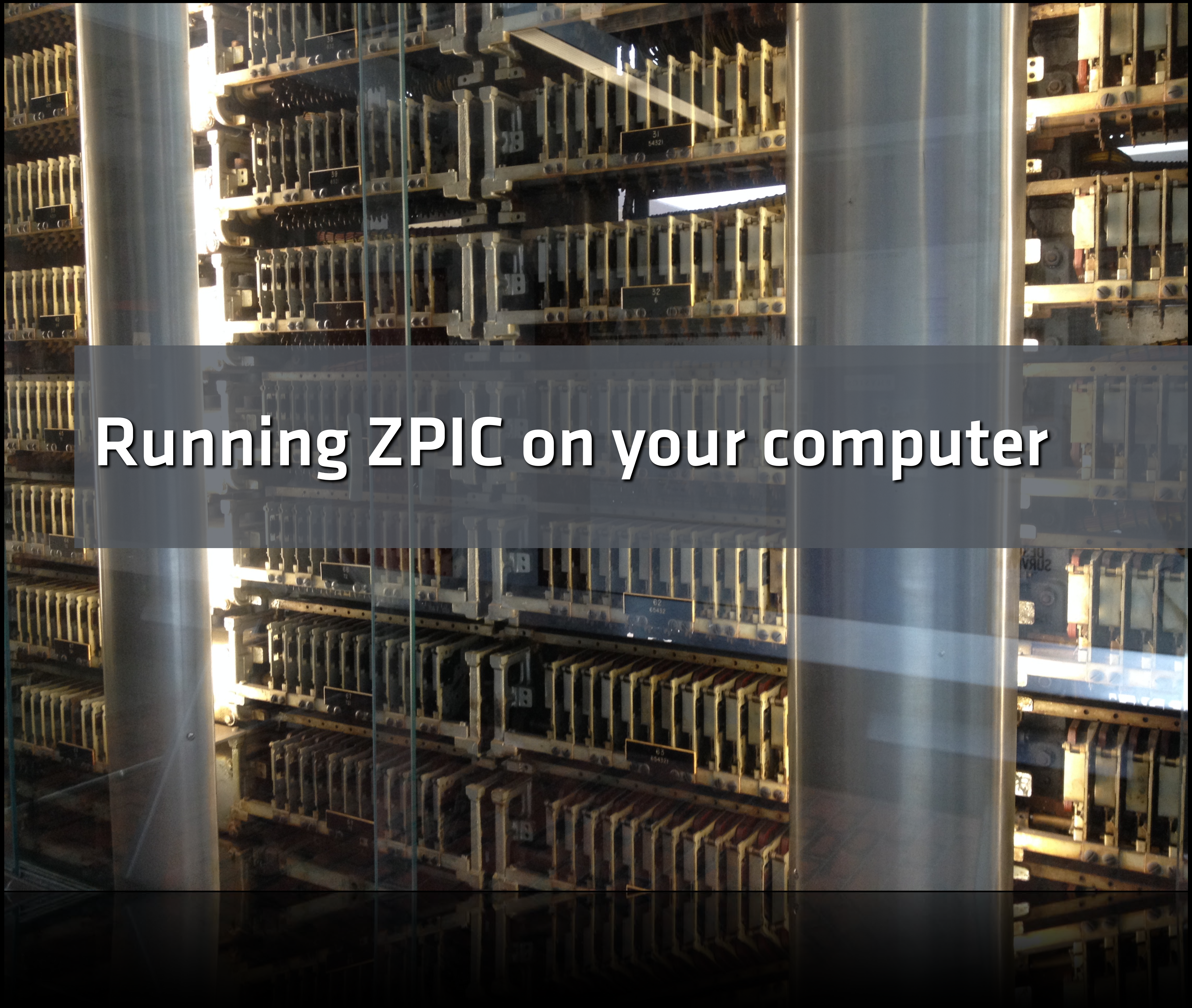
TÉCNICO
LISBOA

ISCTE IUL
Instituto Universitário de Lisboa

Outline

- **Running ZPIC in your computer**
  - Recall installation notes
- **ZPIC toolkit**
  - Educational notebooks
  - Landmark papers
- **(Quick) introduction to LWFA**
  - What a laser wakefield accelerator is
  - Why is it interesting?
- **Modelling LWFA with PIC Codes**
  - Choice of normalization units
  - Resolution and box size
  - Simulation Particles
  - Useful diagnostics





Running ZPIC on your computer

Harvard Mark I - 1944

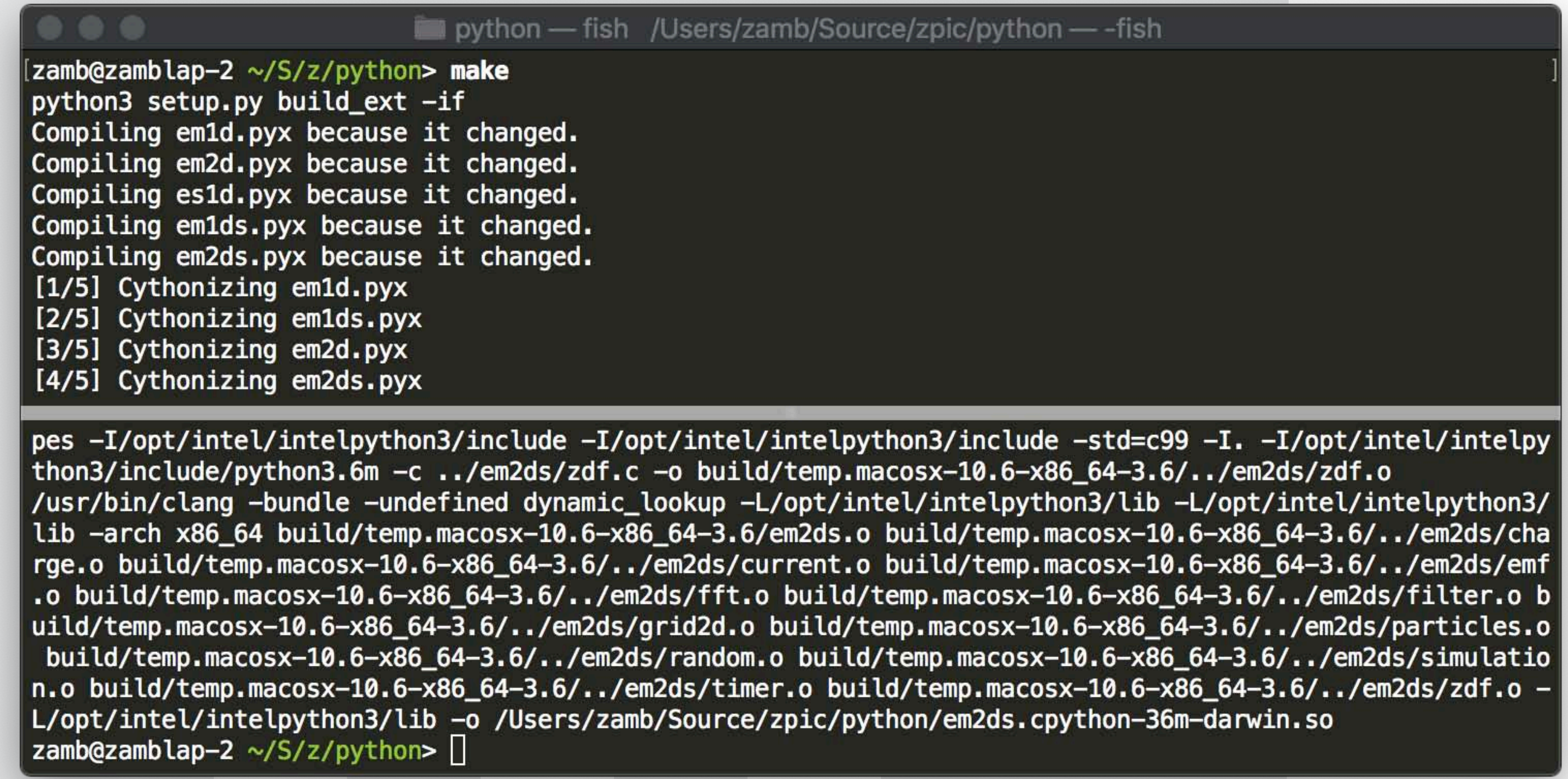
Rear view of Computing Section

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/ricardo-fonseca/zpic

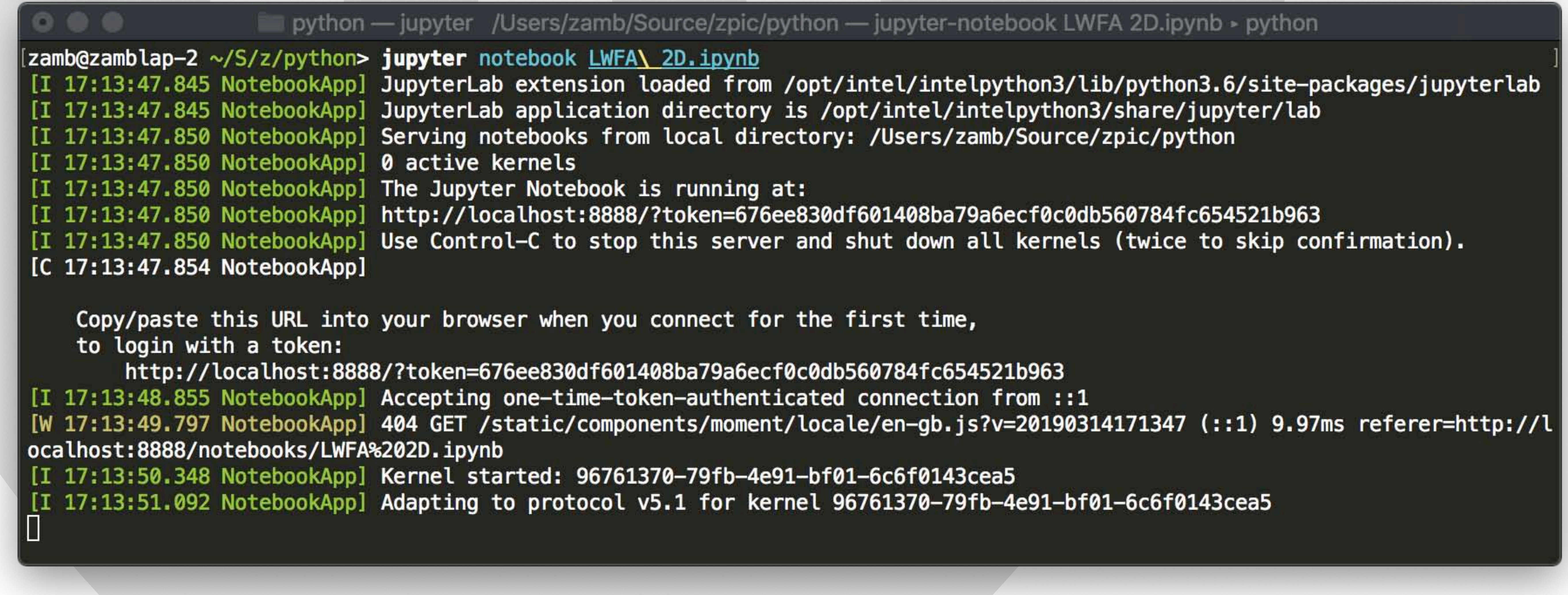
- **Build Python interface**
  - The Python interface requires a Python3 installation
  - The interface also requires NumPy and Cython packages to be installed
  - Just use the Makefile in the python subfolder of the ZPIC distribution
    - This will also compile all of the ZPIC codes

- **Using the Jupyter notebooks**
  - Requires a working Jupyter + Python installation
  - Launch Jupyter and open one of the example notebooks
  - Use either a browser of Visual Studio Code



```
zamb@zamblap-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/intelpy
thon3/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/cha
rge.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 b
uild/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/simulatio
n.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@zamblap-2 ~/S/z/python> 
```



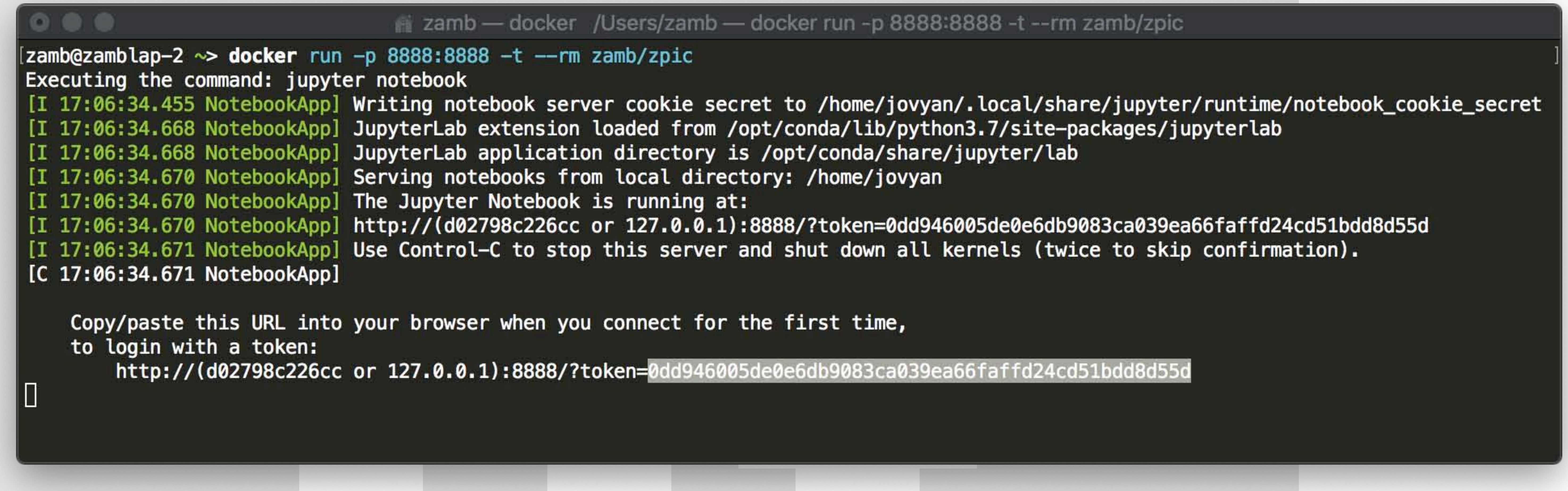
```
zamb@zamblap-2 ~/S/z/python> jupyter notebook LWFA\ 2D.ipynb
[I 17:13:47.845 NotebookApp] JupyterLab extension loaded from /opt/intel/intelpython3/lib/python3.6/site-packages/jupyterlab
[I 17:13:47.845 NotebookApp] JupyterLab application directory is /opt/intel/intelpython3/share/jupyter/lab
[I 17:13:47.850 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://l
ocalhost: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

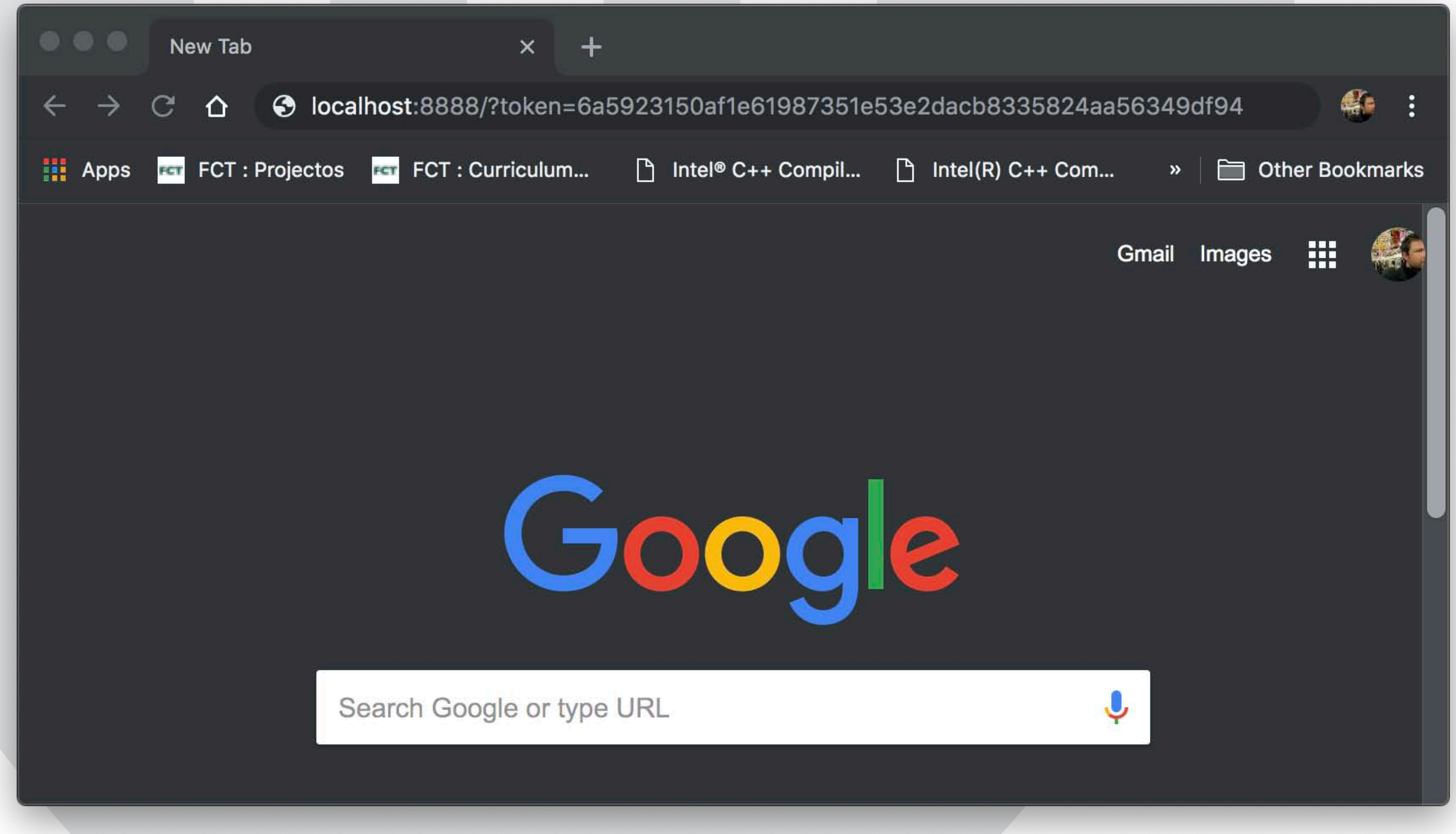
- **Install Docker desktop on your computer**
  - Available for free at:
    - https://www.docker.com/products/docker-desktop

- **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 --rm 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`
  - Get the [TOKEN] value from the output of the docker run command
  - The port number must match the docker run command



```
zamb@zamblap-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
```



Launch a ZPIC notebook

- **Option 1 - Compile from source**
  1. Compile the code
  2. Launch the Jupyter notebook from the source folder:

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

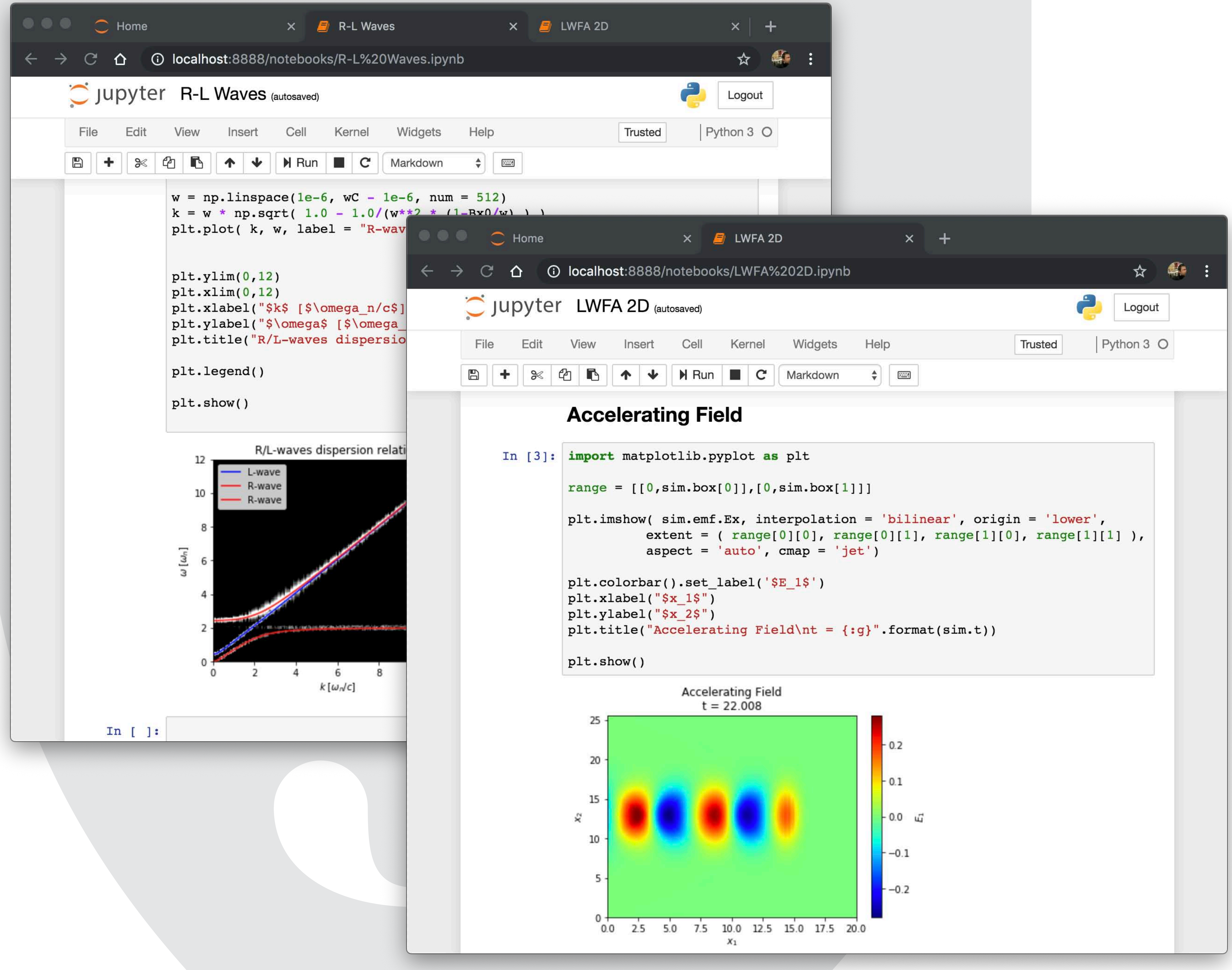
- **Option 2 - Use a Docker Container**
  1. Install Docker
  2. Launch the zpic container

```
> 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

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

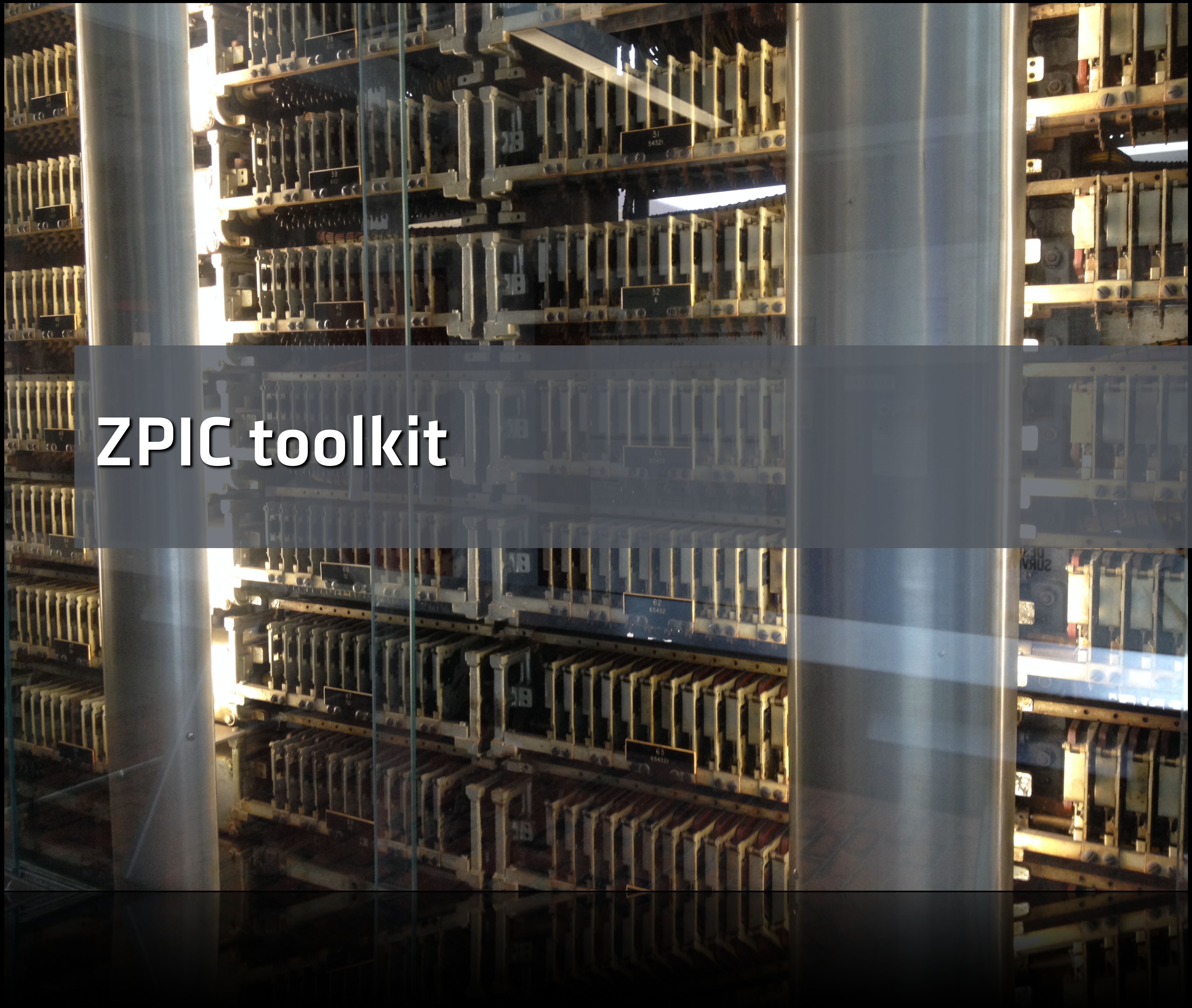


```
w = np.linspace(1e-6, wC - 1e-6, num = 512)
k = w * np.sqrt( 1.0 - 1.0/(w**2 * (
plt.plot( k, w, label = "R-wav


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

plt.legend()

plt.show()
```

Accelerating Field

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\nt = {:g}".format(sim.t))

plt.show()
```



ZPIC toolkit

Harvard Mark I - 1944

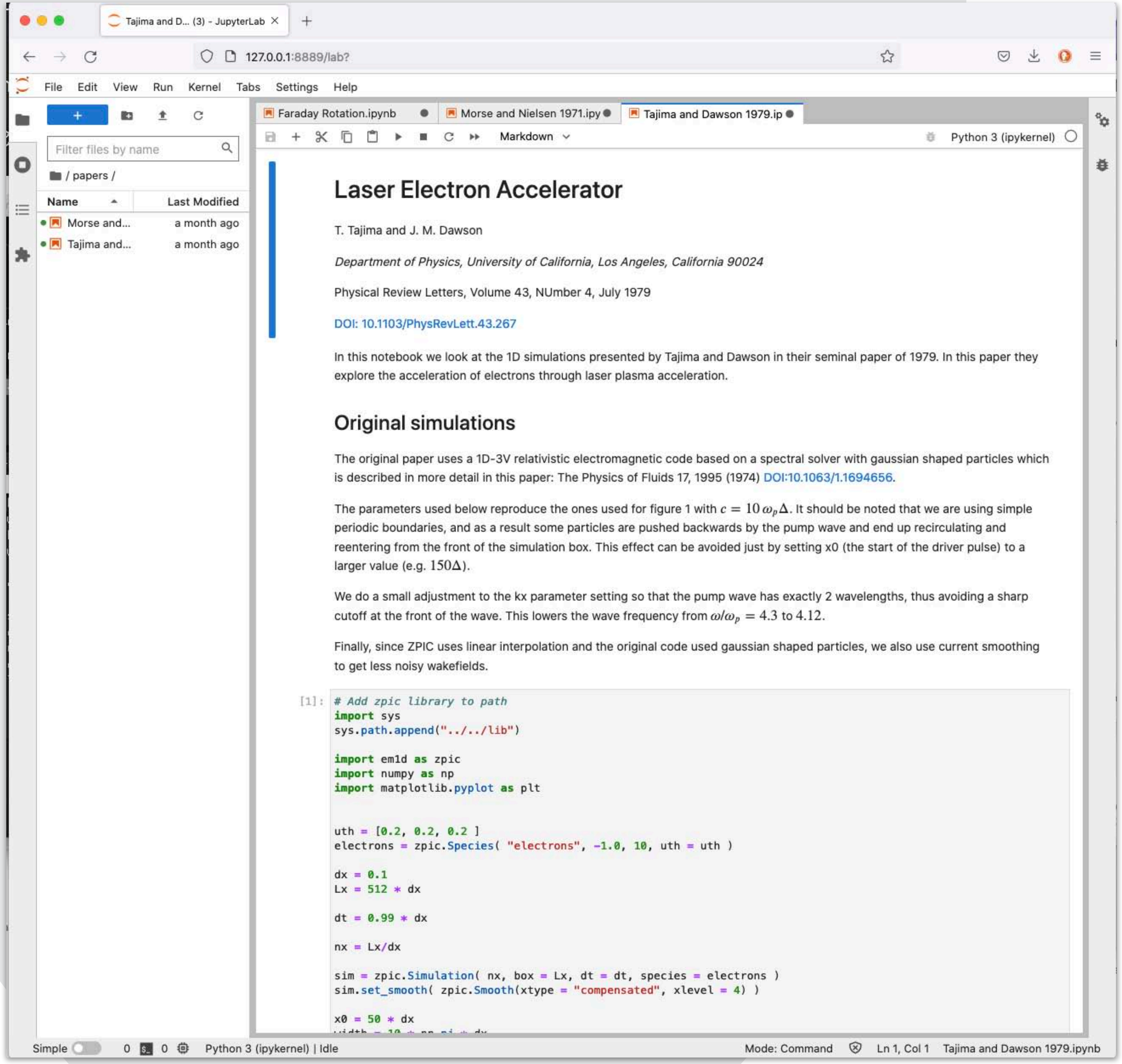
Rear view of Computing Section

Classroom notebooks

- **Since particle and fluid drifts**
  - ExB
  - Diamagnetic
  - Magnetic bottle
  - ...

- **Wave propagation**
  - Electrostatic and electromagnetic waves
  - Magnetized and un-magnetized plasmas
  - Faraday rotation
  - ...

- **Instabilities**
  - Two-stream
  - Weibel
  - ...



Laser Electron Accelerator

T. Tajima and J. M. Dawson

Department of Physics, University of California, Los Angeles, California 90024

Physical Review Letters, Volume 43, NUmber 4, July 1979

DOI: 10.1103/PhysRevLett.43.267

In this notebook we look at the 1D simulations presented by Tajima and Dawson in their seminal paper of 1979. In this paper they explore the acceleration of electrons through laser plasma acceleration.

Original simulations

The original paper uses a 1D-3V relativistic electromagnetic code based on a spectral solver with gaussian shaped particles which is described in more detail in this paper: The Physics of Fluids 17, 1995 (1974) DOI:10.1063/1.1694656.

The parameters used below reproduce the ones used for figure 1 with $c = 10\,\omega_p\Delta$. It should be noted that we are using simple periodic boundaries, and as a result some particles are pushed backwards by the pump wave and end up recirculating and reentering from the front of the simulation box. This effect can be avoided just by setting x0 (the start of the driver pulse) to a larger value (e.g. 150Δ).

We do a small adjustment to the kx parameter setting so that the pump wave has exactly 2 wavelengths, thus avoiding a sharp cutoff at the front of the wave. This lowers the wave frequency from $\omega/\omega_p = 4.3$ to 4.12.

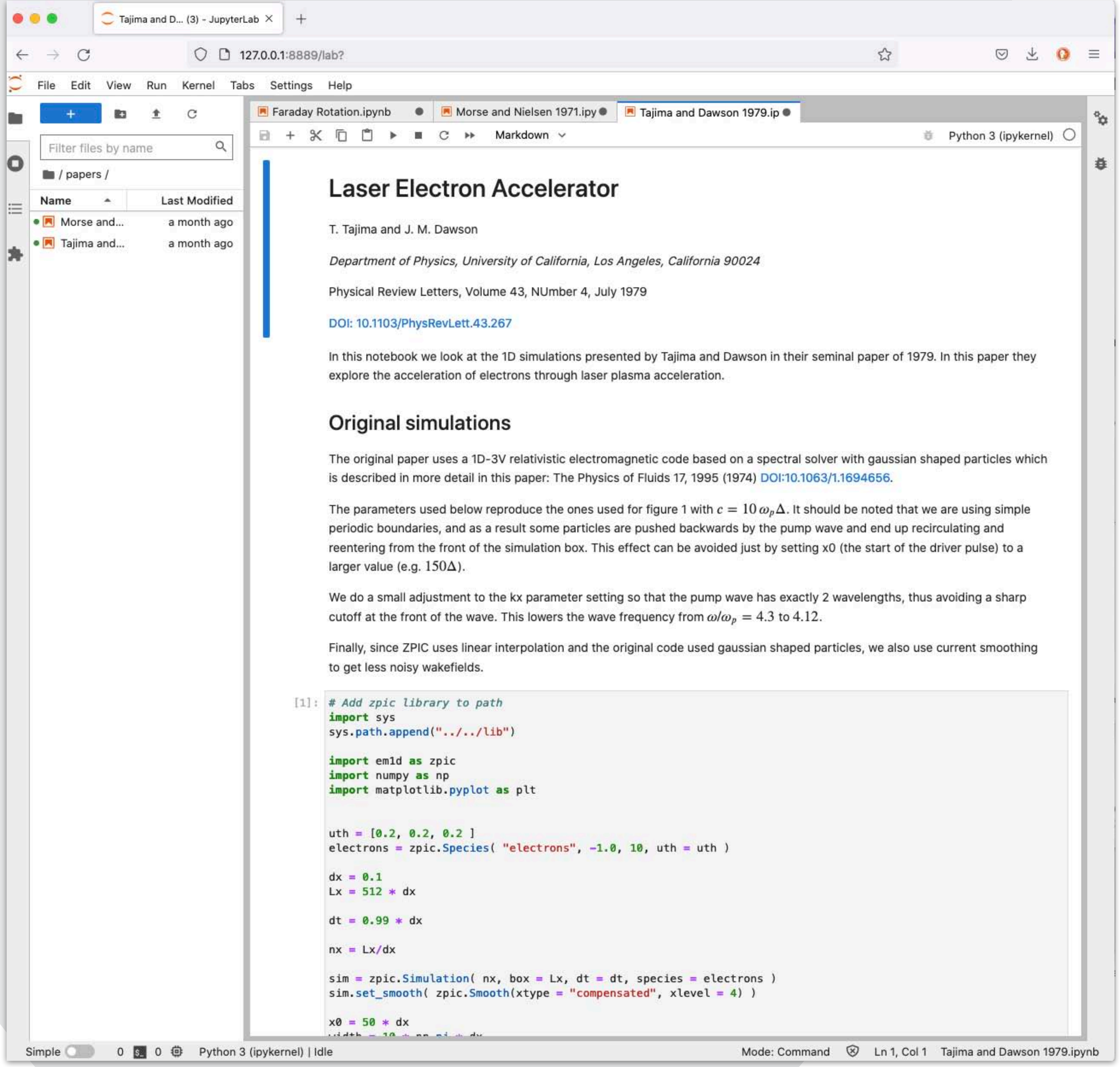
Finally, since ZPIC uses linear interpolation and the original code used gaussian shaped particles, we also use current smoothing to get less noisy wakefields.

[1]:

```
# Add zpic library to path
import sys
sys.path.append("../../lib")

import em1d as zpic
import numpy as np
import matplotlib.pyplot as plt


uth = [0.2, 0.2, 0.2 ]
electrons = zpic.Species( "electrons", -1.0, 10, uth = uth )

dx = 0.1
Lx = 512 * dx

dt = 0.99 * dx

nx = Lx/dx

sim = zpic.Simulation( nx, box = Lx, dt = dt, species = electrons )
sim.set_smooth( zpic.Smooth(xtype = "compensated", xlevel = 4) )

x0 = 50 * dx
```

Landmark papers

- **Laser wakefield accelerator**
  - Tajima and Dawson's paper
- **Morse and Nielsen**
  - Weibel instability



Laser Electron Accelerator

T. Tajima and J. M. Dawson

Department of Physics, University of California, Los Angeles, California 90024

Physical Review Letters, Volume 43, NUmber 4, July 1979

DOI: 10.1103/PhysRevLett.43.267

In this notebook we look at the 1D simulations presented by Tajima and Dawson in their seminal paper of 1979. In this paper they explore the acceleration of electrons through laser plasma acceleration.

Original simulations

The original paper uses a 1D-3V relativistic electromagnetic code based on a spectral solver with gaussian shaped particles which is described in more detail in this paper: The Physics of Fluids 17, 1995 (1974) DOI:10.1063/1.1694656.

The parameters used below reproduce the ones used for figure 1 with $c = 10\,\omega_p \Delta$. It should be noted that we are using simple periodic boundaries, and as a result some particles are pushed backwards by the pump wave and end up recirculating and reentering from the front of the simulation box. This effect can be avoided just by setting x0 (the start of the driver pulse) to a larger value (e.g. 150Δ).

We do a small adjustment to the kx parameter setting so that the pump wave has exactly 2 wavelengths, thus avoiding a sharp cutoff at the front of the wave. This lowers the wave frequency from $\omega/\omega_p = 4.3$ to 4.12.

Finally, since ZPIC uses linear interpolation and the original code used gaussian shaped particles, we also use current smoothing to get less noisy wakefields.

[1]:

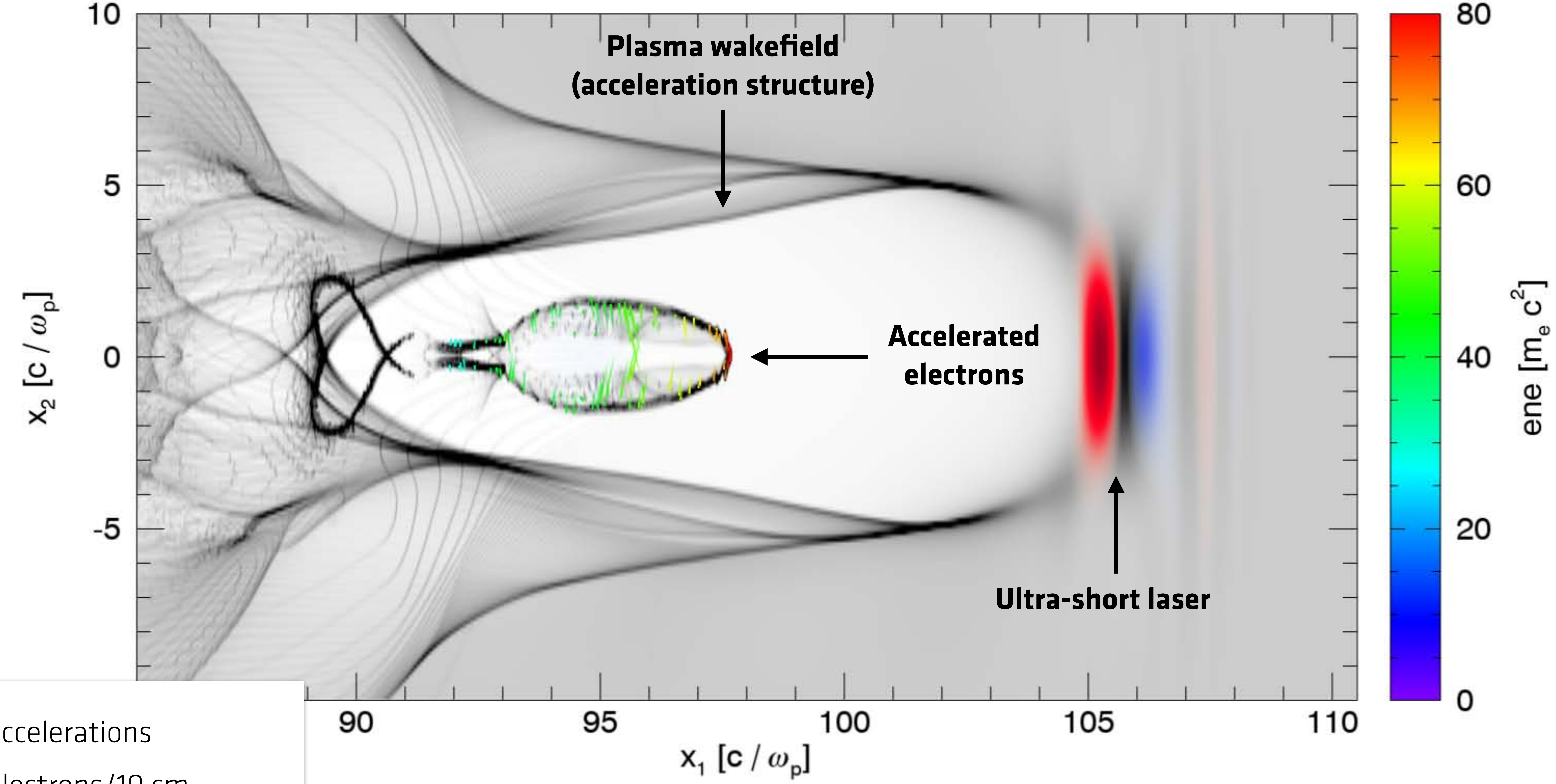
```python
# Add zpic library to path
import sys
sys.path.append("../../lib")

import em1d as zpic
import numpy as np
import matplotlib.pyplot as plt


uth = [0.2, 0.2, 0.2 ]
electrons = zpic.Species( "electrons", -1.0, 10, uth = uth )

dx = 0.1
Lx = 512 * dx

dt = 0.99 * dx

nx = Lx/dx

sim = zpic.Simulation( nx, box = Lx, dt = dt, species = electrons )
sim.set_smooth( zpic.Smooth(xtype = "compensated", xlevel = 4) )

x0 = 50 * dx
```

Modelling Laser Wakefield Acceleration

Laser Wakefield Acceleration

3D Simulation using the OSIRIS code

A quick introduction to laser wakefield acceleration



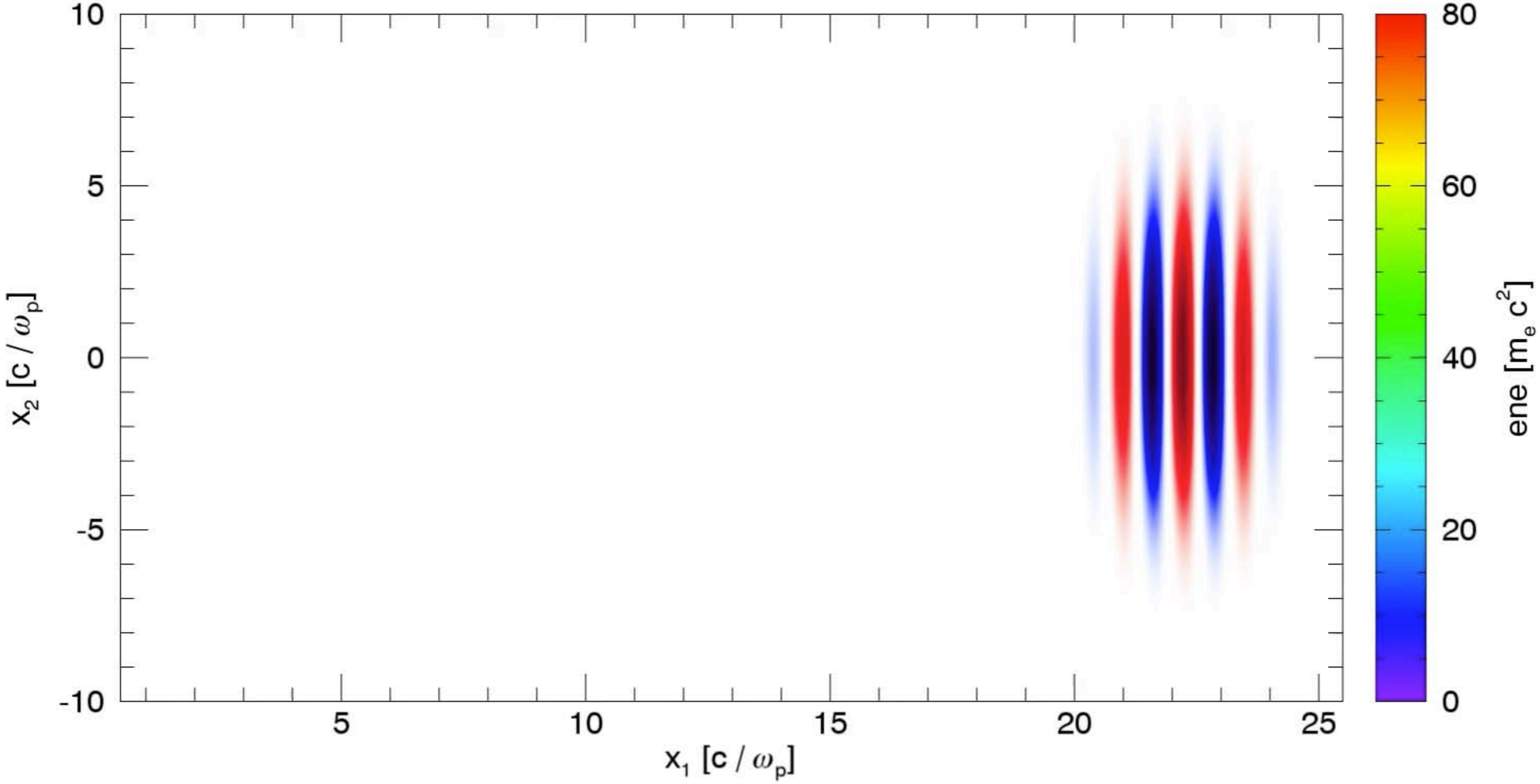
Plasma wakefield (acceleration structure)

Accelerated electrons

Ultra-short laser

- GV/cm accelerations
- 10 GeV electrons/10 cm plasma (demonstrated)

A quick introduction to laser wakefield acceleration



LWFA first experiments

30 September 2004

International weekly journal of science

nature

www.nature.com/nature

Dream beam

The dawn of compact particle accelerators

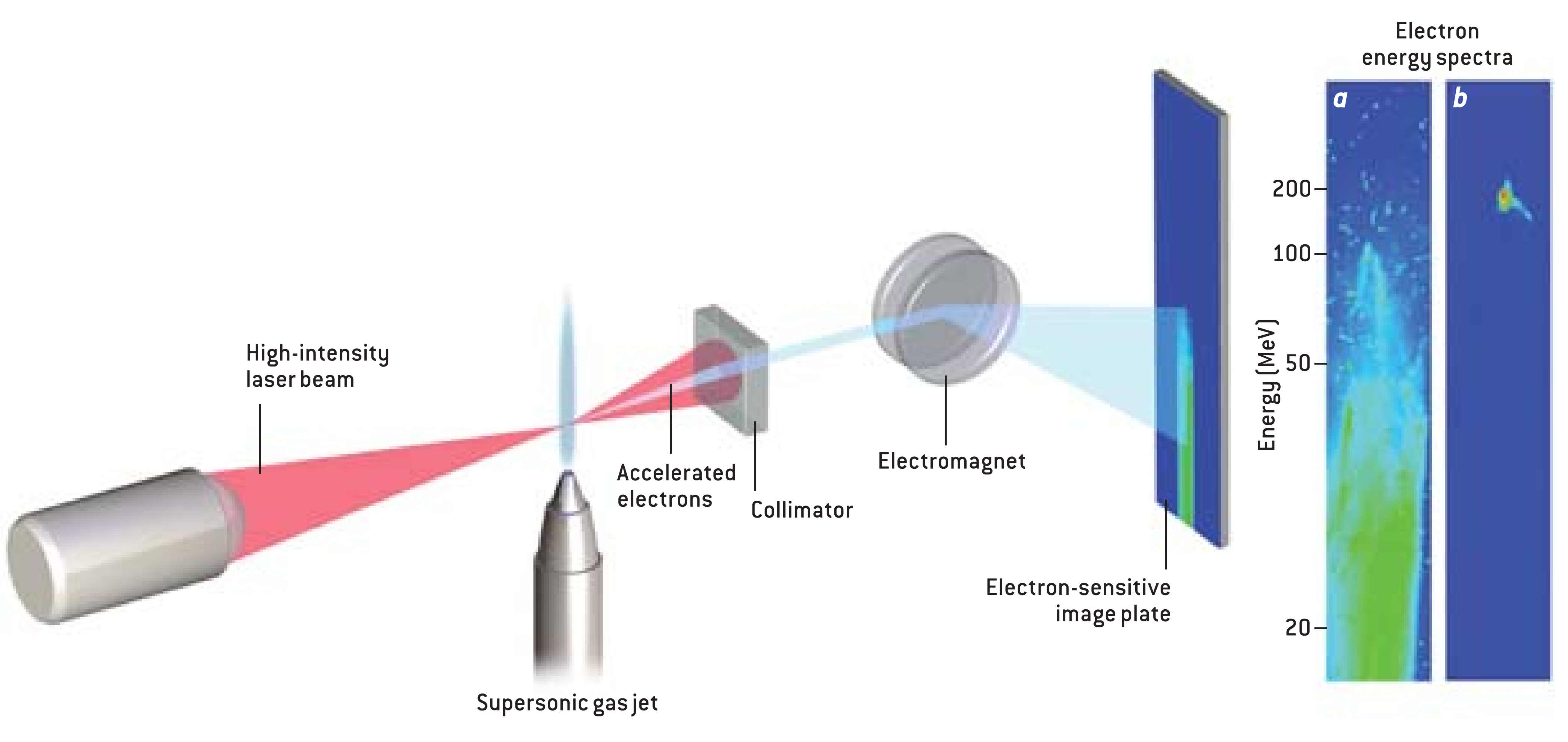
Offshore tuna ranches A threat to US waters?

The Earth's hum Sounds of air and sea

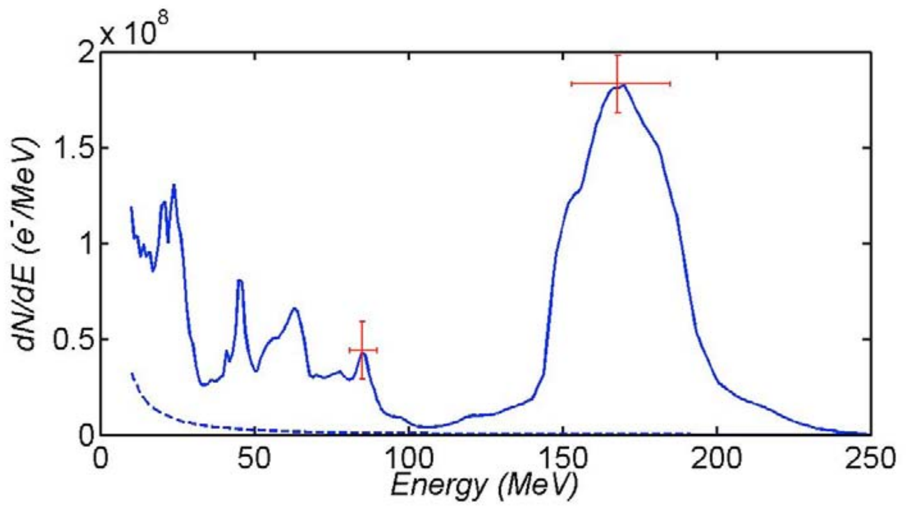
Protein folding Escape from the ribosome

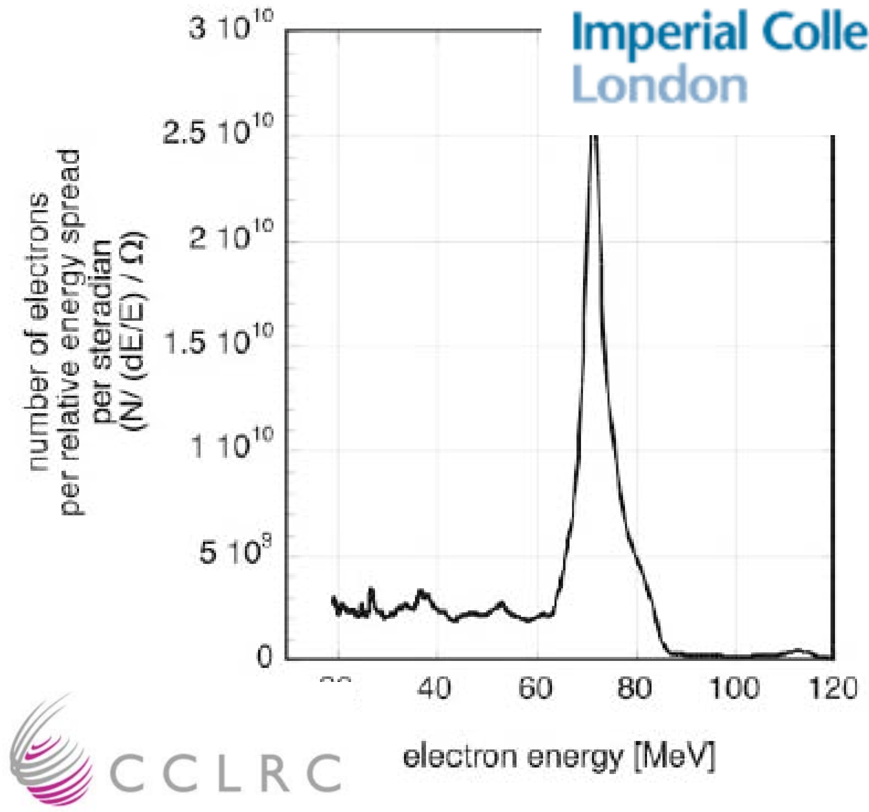
Human ancestry One from all and all from one

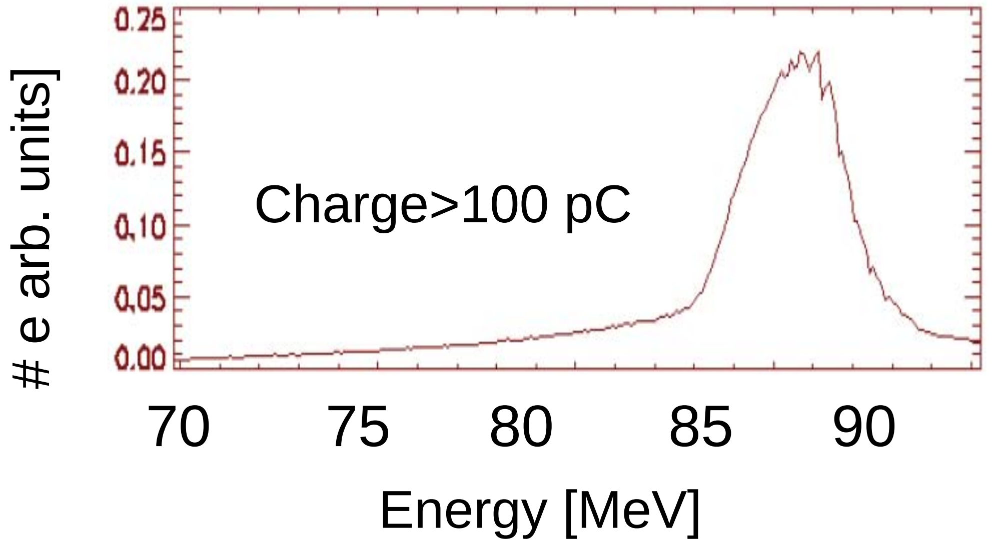
technology feature RNA interference



First experimental results



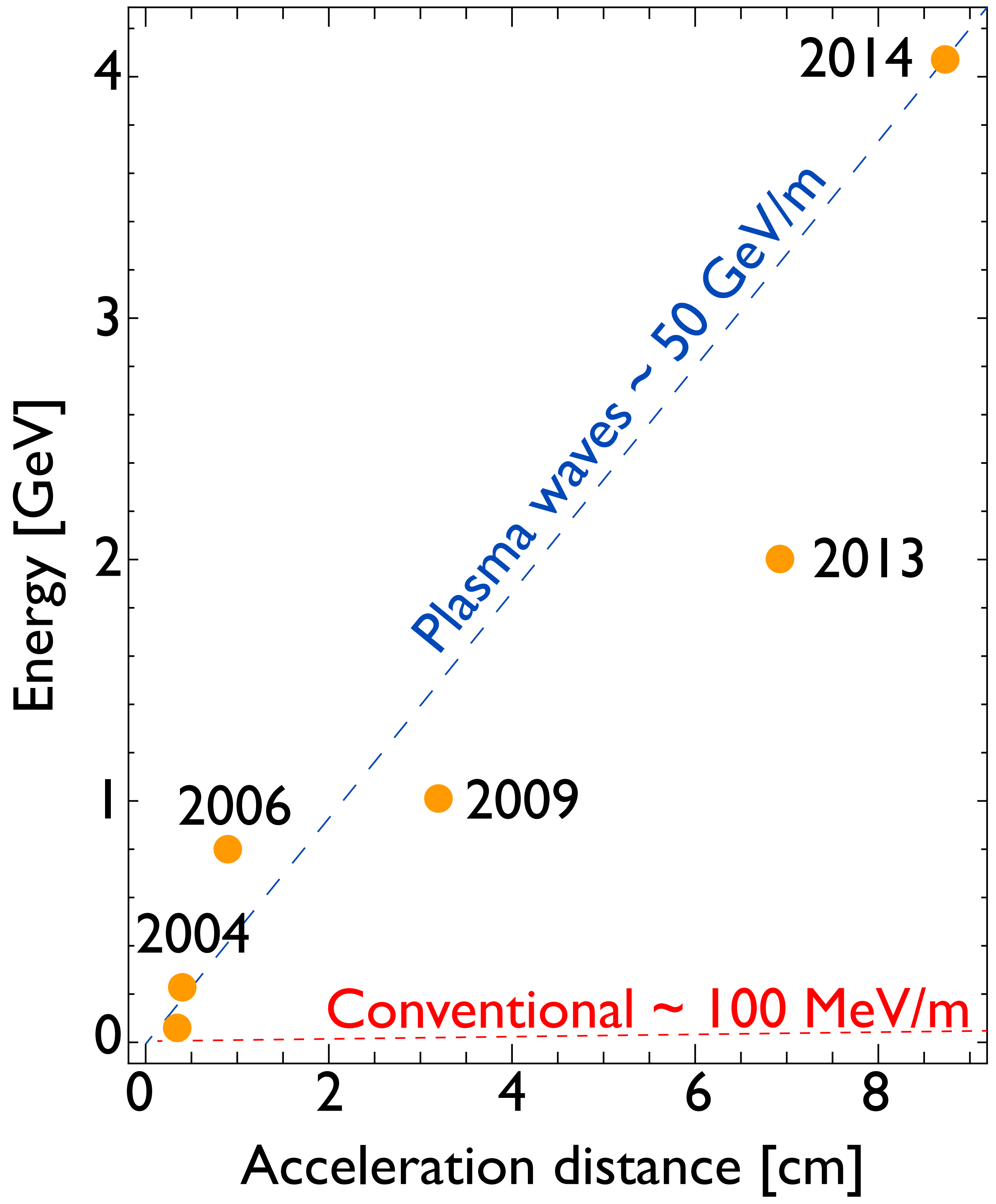




Courtesy: V. Malka (LOA), K. Krushelnick (IC/RAL), W. Leemans (LBL)

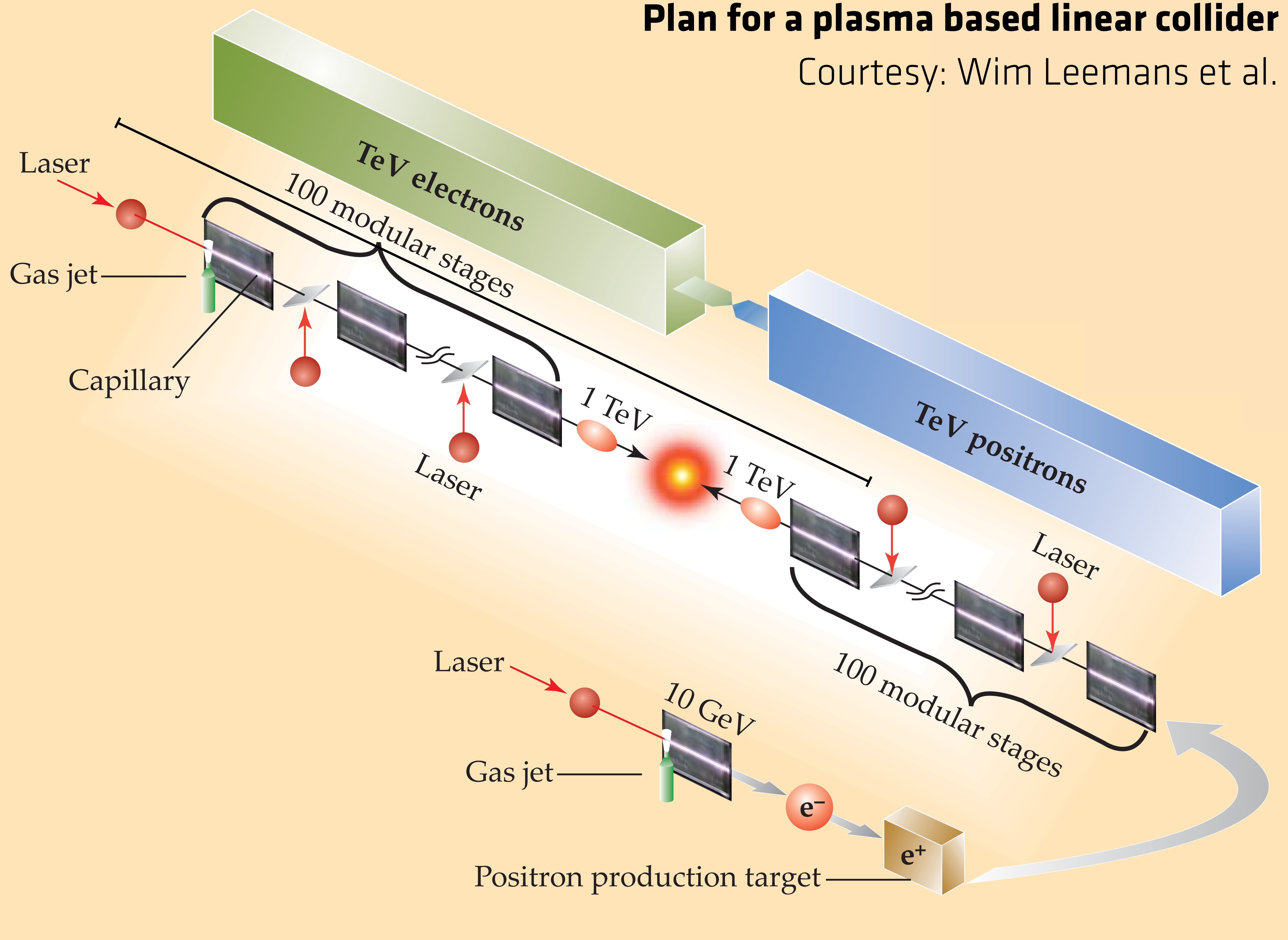
LWFA linear collider

Explosive energy gain



Plan for a plasma based linear collider

Courtesy: Wim Leemans et al.



Reference length and time

Choose the normalization

Plasma sets reference

Plasma density is unity

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

Example

- Plasma density $n_p = 10^{18}\ \mathrm{cm}^{-3}$
- Plasma frequency $\omega_p \sim 5.64 \times 10^{13}\ \mathrm{rad\ s}^{-1}$
- Laser wavelength $\lambda_0 = 1\ \mu\mathrm{m}$
- Laser frequency $\omega_0 \sim 2.34 \times 10^{15}\ \mathrm{rad\ s}^{-1}$
- 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 $\lambda_0 = 1\mu\mathrm{m}$
- Laser frequency $\omega_0 \sim 2.34 \times 10^{15}\ \mathrm{rad\ s}^{-1}$
- Critical frequency $n_{crit} \sim 1.72 \times 10^{21}\ \mathrm{cm}^{-3}$
- Plasma density $n_p = 10^{18}\ \mathrm{cm}^{-3}$
- Normalized plasma density $n_p/n_{crit} \sim 5.8 \times 10^{-4}$

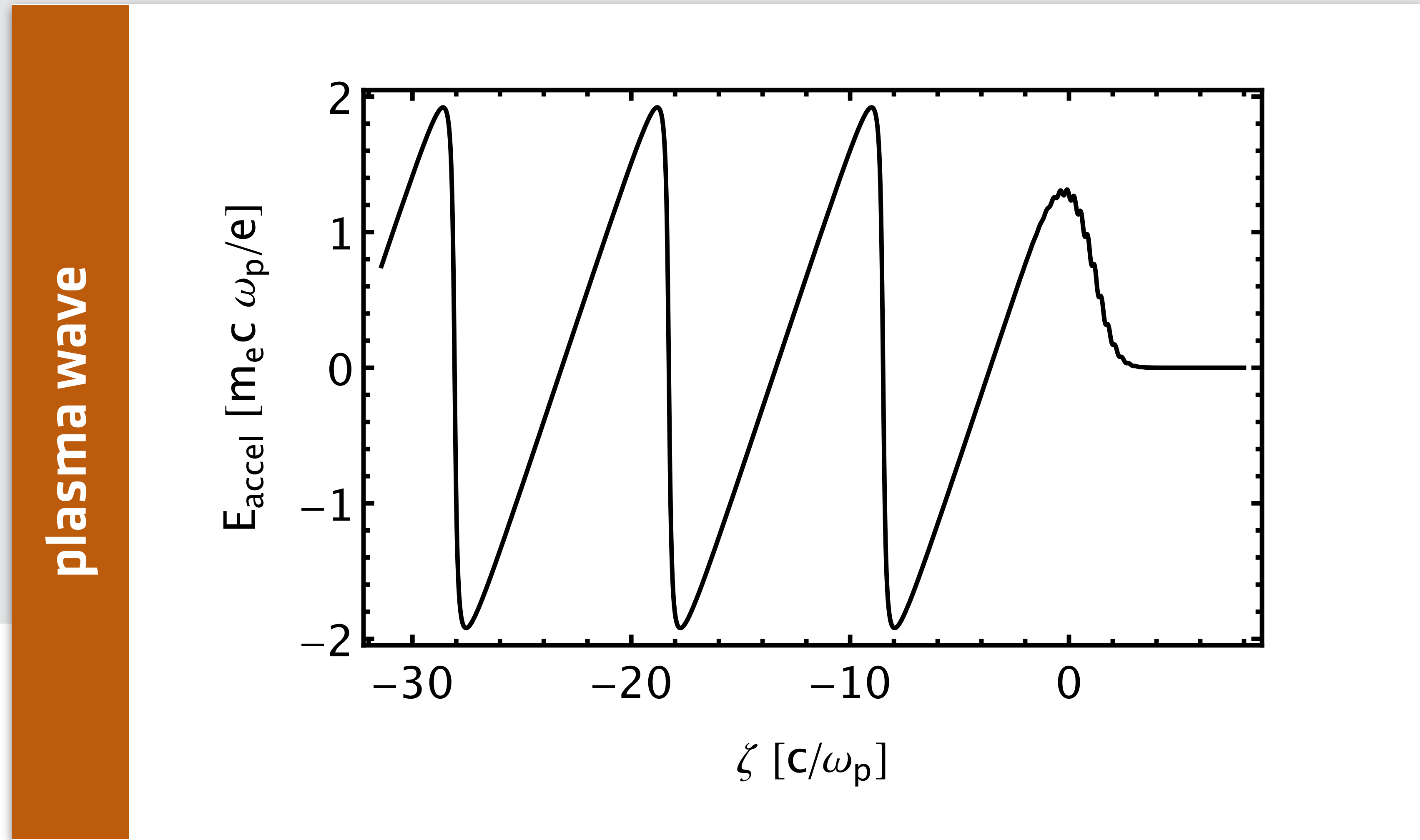
Both (and other) normalizations are possible. In this session we will use the plasma as the reference!

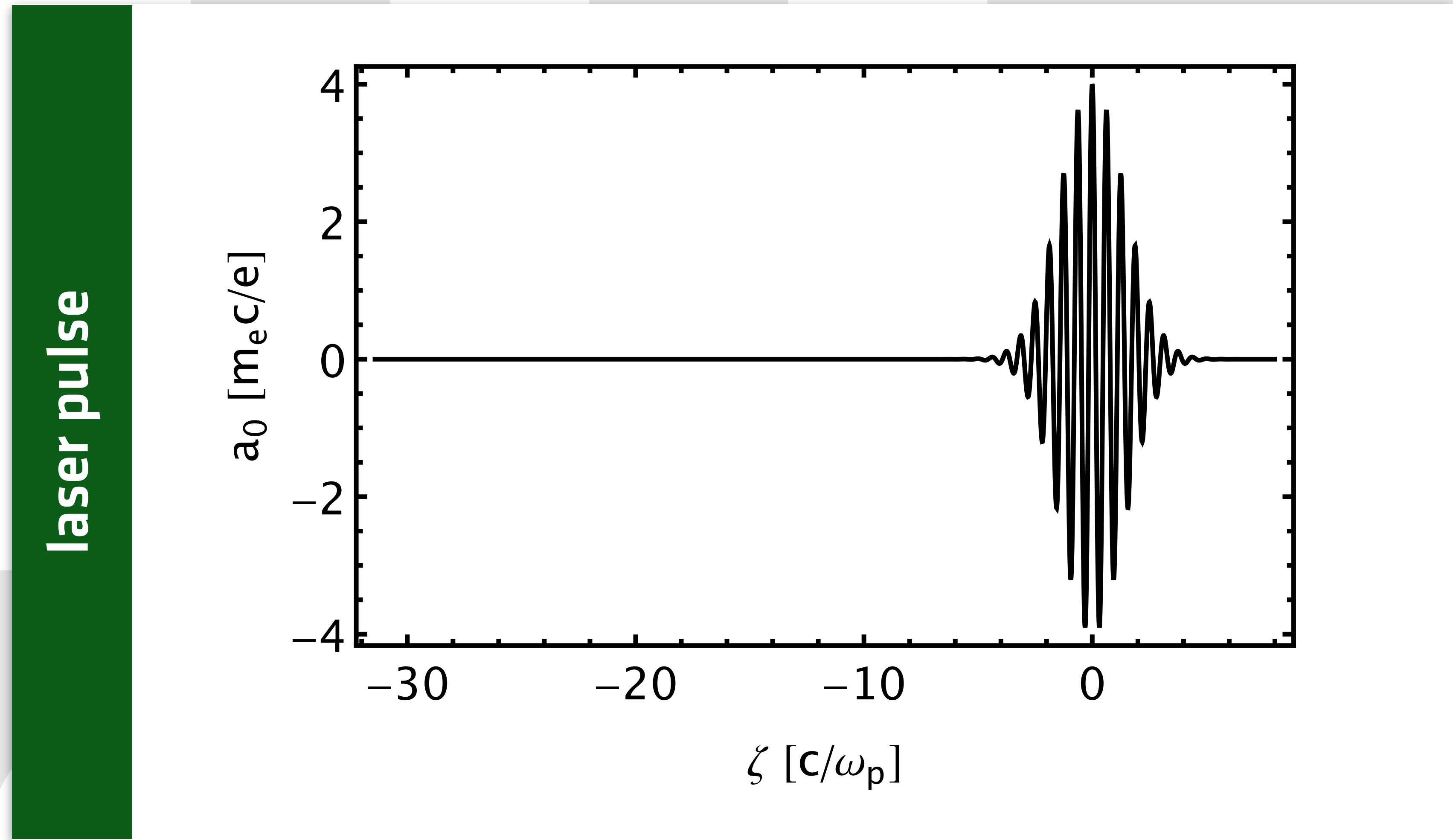
Choosing the spatial resolution

Spatial resolution

need to resolve the smallest scale length

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





Longitudinal spatial Resolution: $\Delta x \sim \lambda_0/\# \sim 0.18/\#\ (\ n_p\,[10^{18} cm^{-3}]\ )^{1/2}\ c/\omega_p$

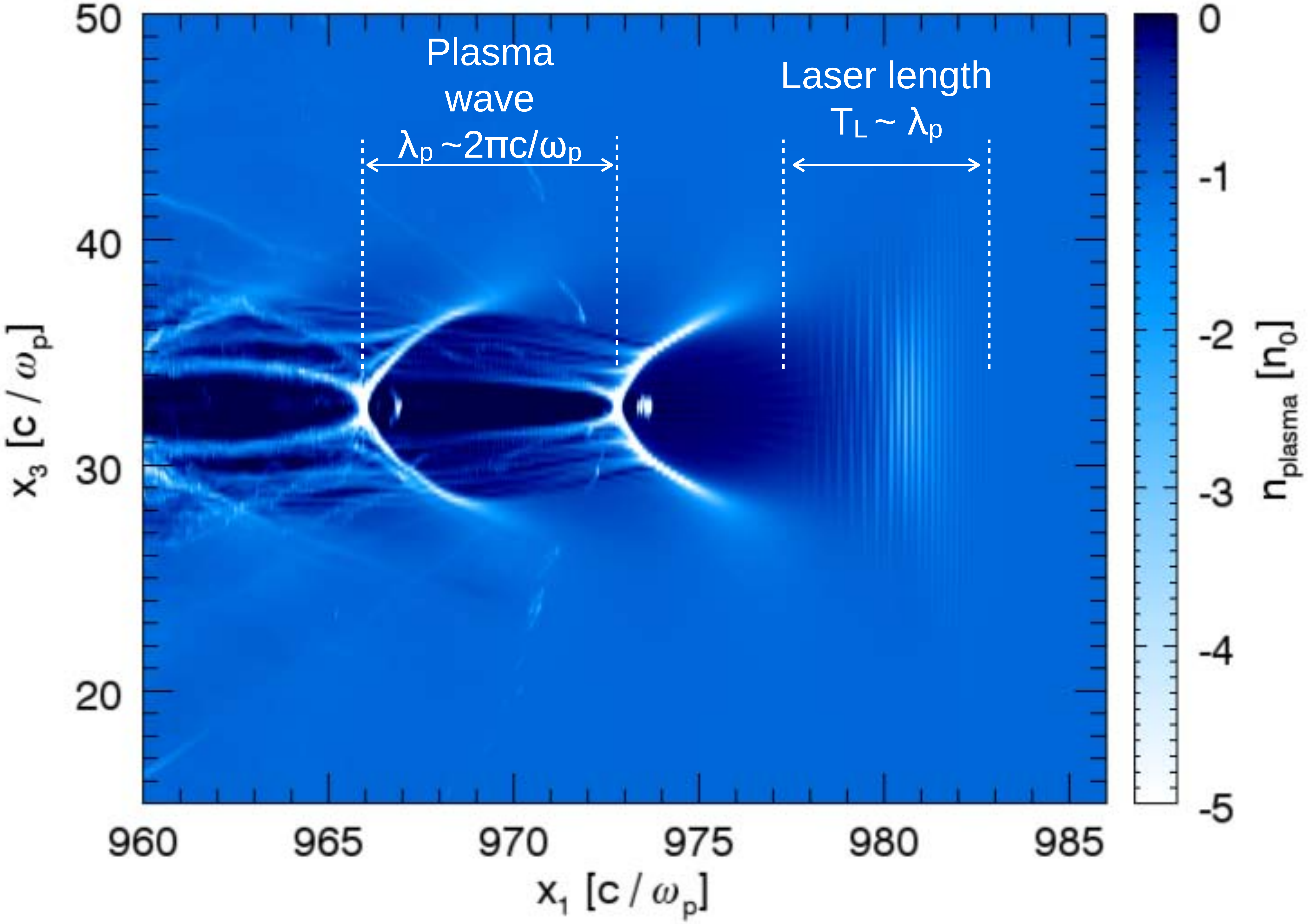
$\# > 20\text{-}30$ (number of cells per laser wavelength)

Simulation box dimensions

Box size

needs to be larger than the largest structure

- **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
    - $> 4 \lambda_p \sim 25\ c/\omega_p$

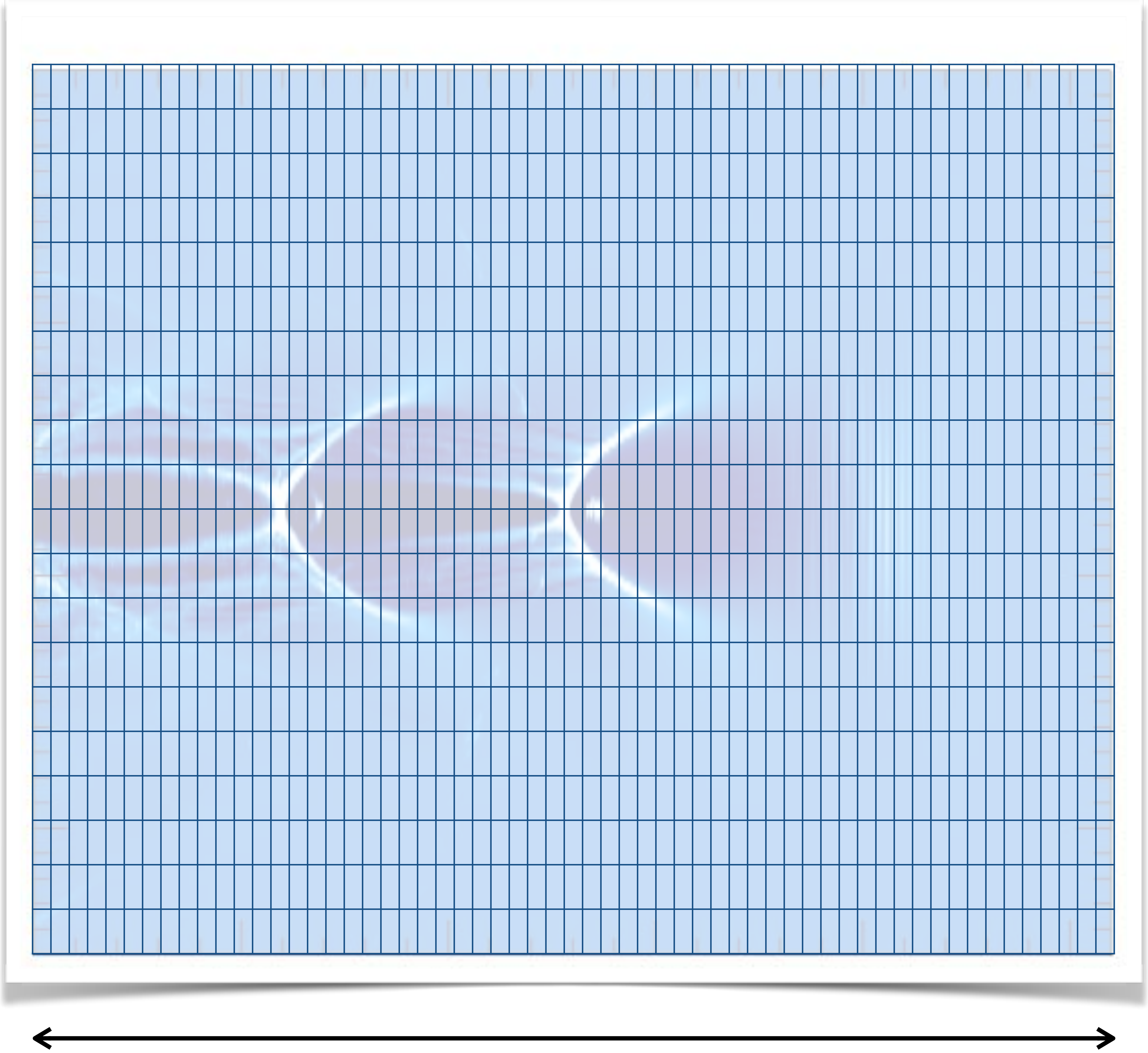


Setting up the simulation: cells, particles

- **Simulation grid**
  - Box length: L = 4 λ_p
  - 20 points per laser wavelength
  - $\Delta x \sim \lambda_0/20 \sim 0.18/20 = 0.009\ c/\omega_p$
  - Number of cells ~ L / Δx ~ 2800 cells

- **Simulation particles**
  - Number of particles per cell must resolve local phasespace
  - ≫1 in 1D (e.g. 64)
  - ~ 10 in 2D
  - Higher numbers improve phasespace resolution (detailed distribution tails)
  - Also reduces simulation noise

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



Longitudinal cells: 4 λ_p /(0.18/20 c/ω_p) ~ 2800 cells ($n_p = 10^{18}$ cm^{-3})

Example box dimensions and resolution (normalized units)

Typical LWFA parameters

Quantity	Normalized laser vector potential (a0)	Background plasma density (n0) [1/cm^3]	Plasma skin depth (c/ωp) [microns]	Laser wavelength (λL) [nm]	Laser frequency (ωL) [rad/s]	Laser Pulse Duration (σt) [fs]	Laser spot-size (w0) [microns]
Dimensional	-	1,50E+19	1,68	800	2,35E+15	30	10
Normalized Units	4	1	1	0,476190476	10,75828707	6,55E+00	7,28E+00
Comment	-	-	ωp=4,64x10^4xsqrt(n0)	λL/(c/ωp)	ωL/ωp	st*ωp	w0*ωp/c

PIC simulation box size and resolutions

	Laser		Plasma	Simulation box					
	Spot size [c/wp]	Frequency (ωL/ωp)	λp [c/ωp]	Lx [c/ωp]	Ly [c/ωp]	Δx [c/ωp]	Δy [c/ωp]	#cells x	#cellx y
Criteria	-	-	-	Larger than plasma wavelength	More than 4x laser spot-size	at least 30 points per laser wavelength	at least 30 points per laser spot-size/plasma wavelength	Lx/Δx	Ly/Δy
Values	4,00	5,00	6,28	25	32	0,04	0,13	600	240

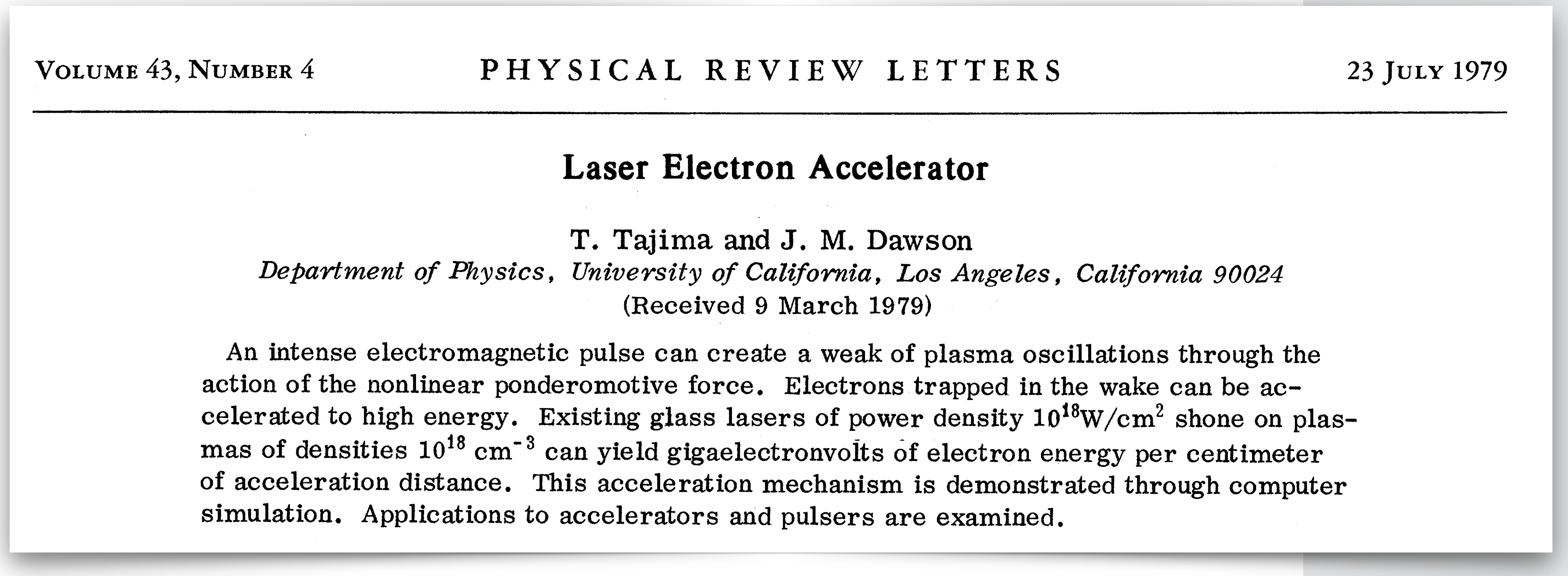
Tajima and Dawson’s LWFA paper

TÉCNICO
LISBOA

Laser Wakefield Acceleration

3D Simulation using the OSIRIS code

Simulation initialisation



VOLUME 43, NUMBER 4 PHYSICAL REVIEW LETTERS 23 JULY 1979

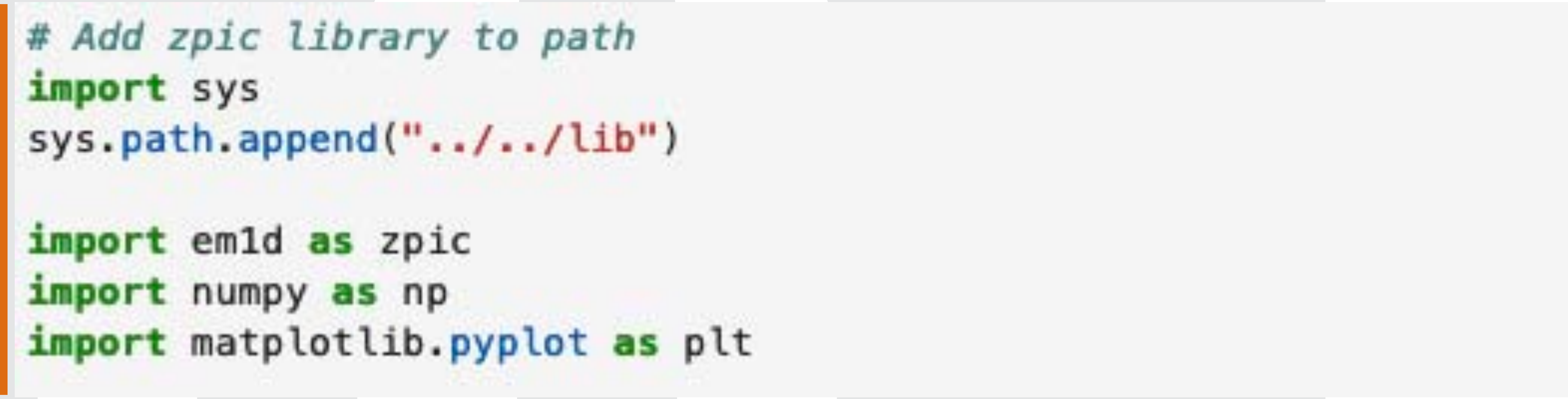
Laser Electron Accelerator

T. Tajima and J. M. Dawson
Department of Physics, University of California, Los Angeles, California 90024
(Received 9 March 1979)

An intense electromagnetic pulse can create a weak of plasma oscillations through the action of the nonlinear ponderomotive force. Electrons trapped in the wake can be accelerated to high energy. Existing glass lasers of power density 10^{18}W/cm^2 shone on plasmas of densities 10^{18} cm^{-3} can yield gigaelectronvolts of electron energy per centimeter of acceleration distance. This acceleration mechanism is demonstrated through computer simulation. Applications to accelerators and pulsers are examined.

- **Initializing a ZPIC simulation requires**
  - Selecting the code version
  - Setting up the particle species (sets of particles)
    - The number of species is arbitrary
  - Setting up the simulation
    - Grid / Box size
    - Time step
    - Add species

Code



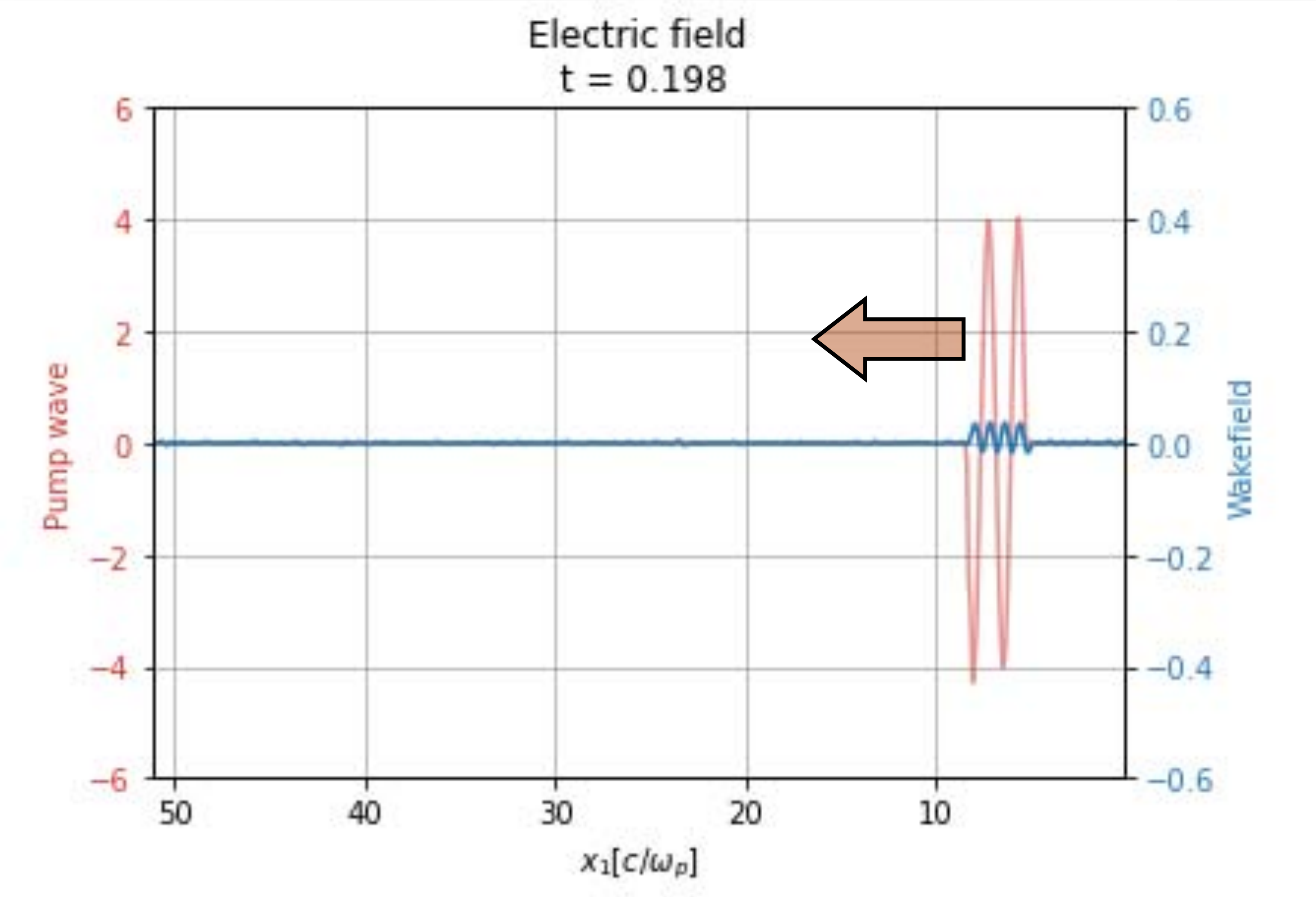
```python
# Add zpic library to path
import sys
sys.path.append("../../lib")

import em1d as zpic
import numpy as np
import matplotlib.pyplot as plt
```

Box and species

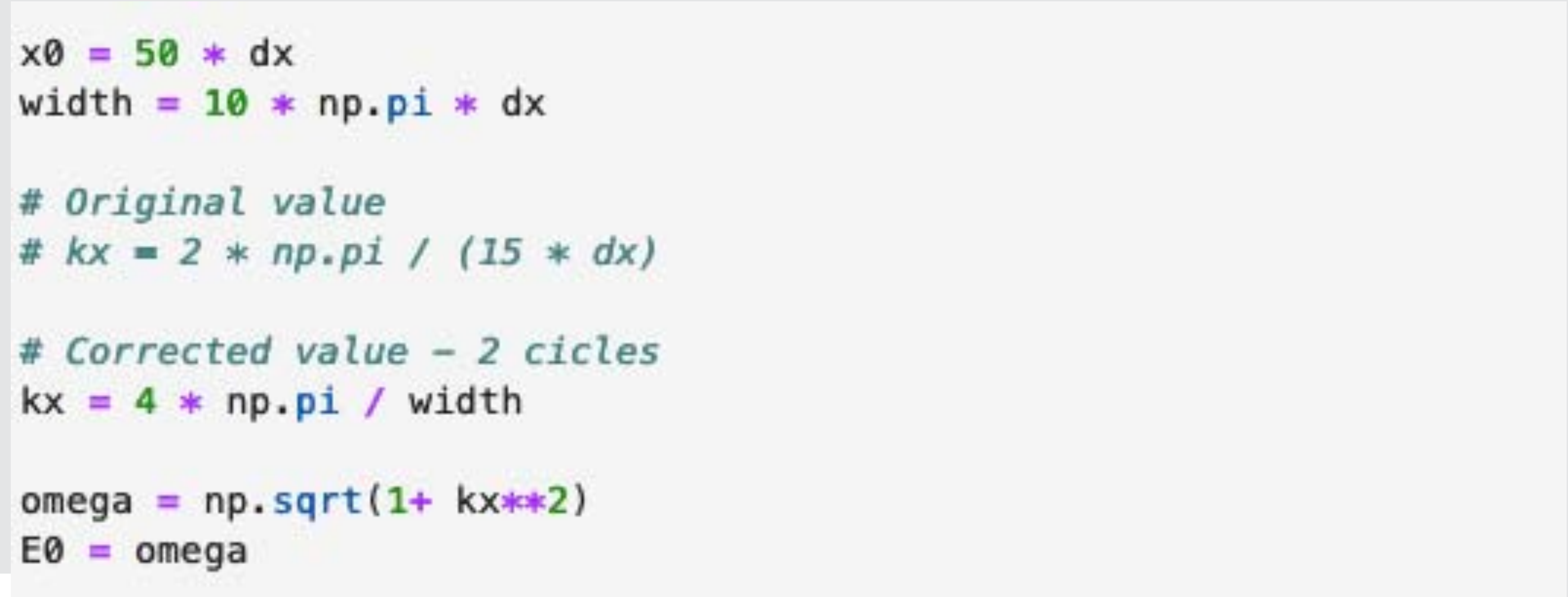
```python
uth = [0.2, 0.2, 0.2 ]
electrons = zpic.Species( "electrons", -1.0, 10, uth = uth )

dx = 0.1
Lx = 512 * dx

dt = 0.99 * dx

nx = Lx/dx

sim = zpic.Simulation( nx, box = Lx, dt = dt, species = electrons )
sim.set_smooth( zpic.Smooth(xtype = "compensated", xlevel = 4) )

x0 = 50 * dx
width = 10 * np.pi * dx
```

Simulation initialisation - external e.m. fields

- **Additional steps**
  - Adding laser pulse
  - External field init.

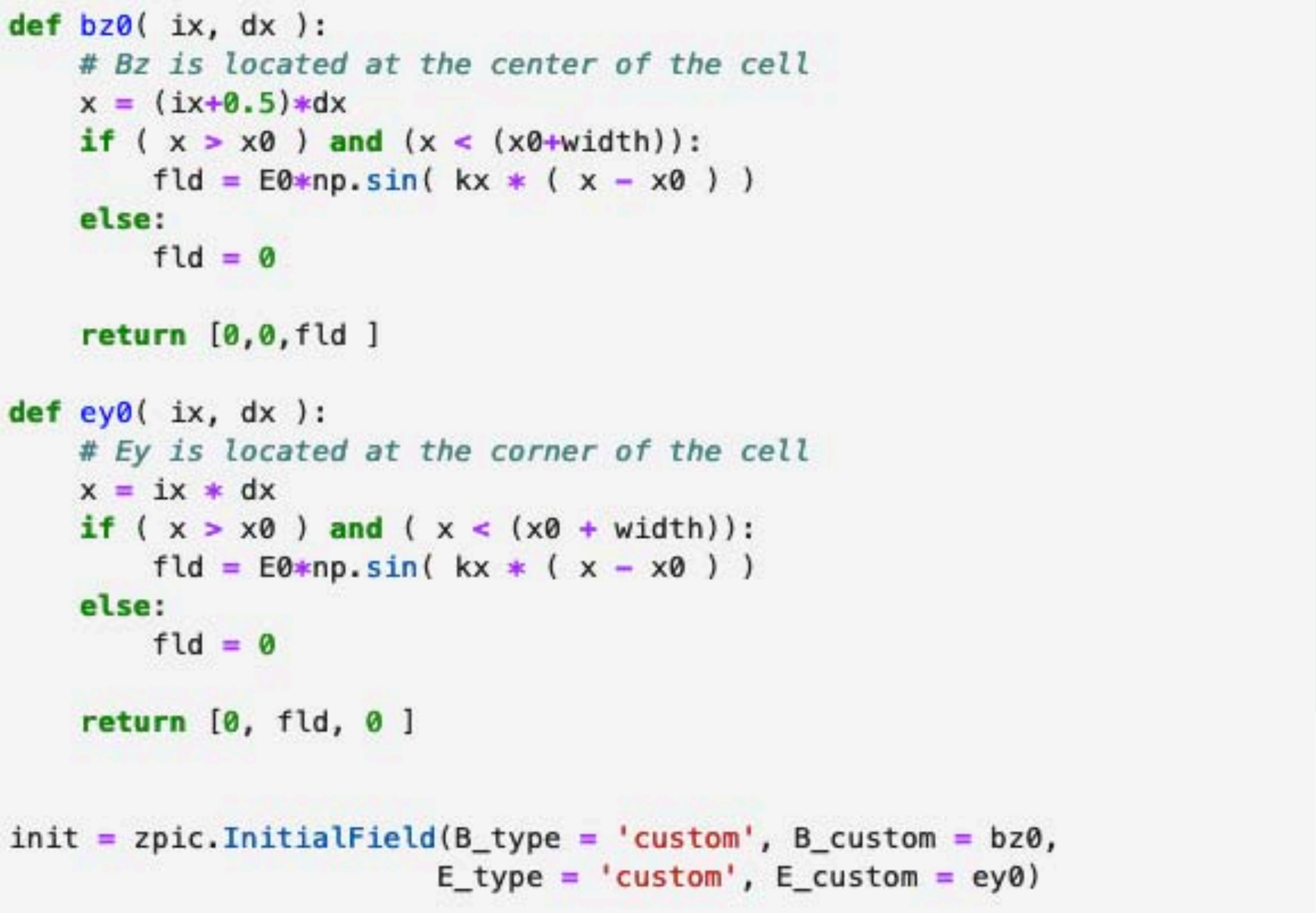


Laser definitions



```
x0 = 50 * dx
width = 10 * np.pi * dx

# Original value
# kx = 2 * np.pi / (15 * dx)

# Corrected value - 2 cicles
kx = 4 * np.pi / width

omega = np.sqrt(1+ kx**2)
E0 = omega
```

External fields



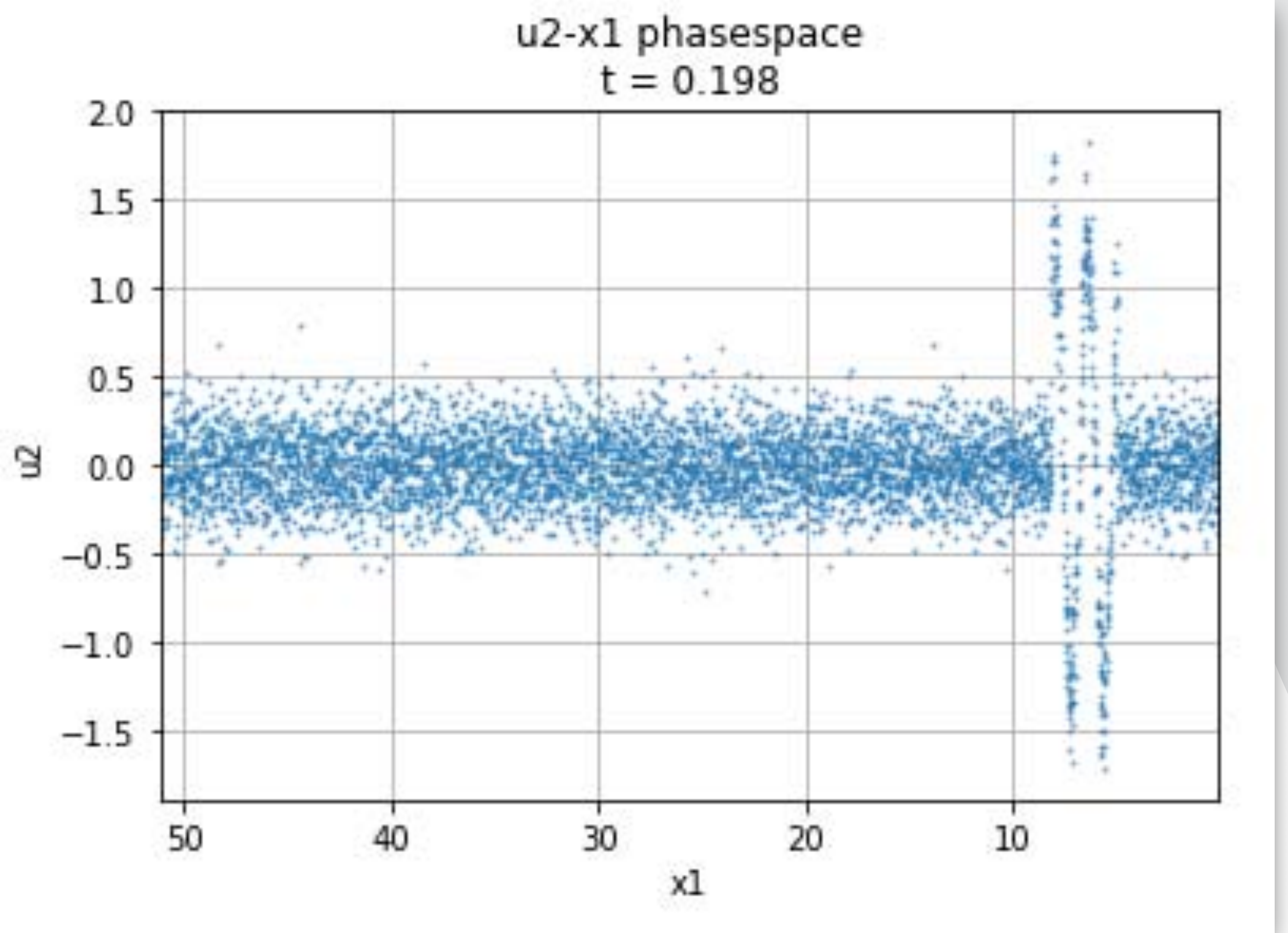
```
def bz0( ix, dx ):
    # Bz is located at the center of the cell
    x = (ix+0.5)*dx
    if ( x > x0 ) and (x < (x0+width)):
        fld = E0*np.sin( kx * ( x - x0 ) )
    else:
        fld = 0

    return [0,0,fld ]

def ey0( ix, dx ):
    # Ey is located at the corner of the cell
    x = ix * dx
    if ( x > x0 ) and ( x < (x0 + width)):
        fld = E0*np.sin( kx * ( x - x0 ) )
    else:
        fld = 0

    return [0, fld, 0 ]


init = zpic.InitialField(B_type = 'custom', B_custom = bz0,
                         E_type = 'custom', E_custom = ey0)
```

Run

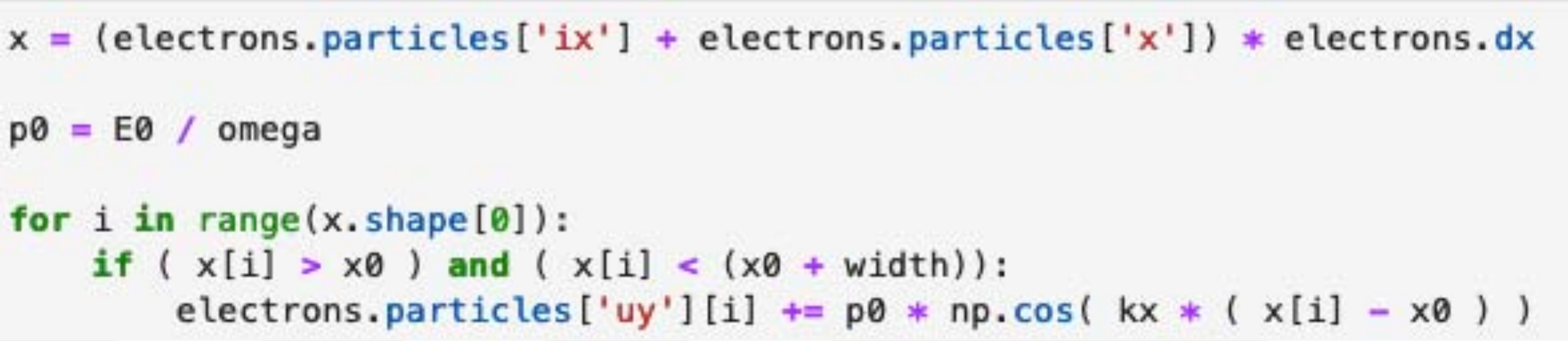
```
init = zpic.InitialField(B_type = 'custom', B_custom = bz0,
                         E_type = 'custom', E_custom = ey0)

sim.emf.init_fld( init )
```

Simulation initialisation - transverse electron momentum

- **Additional steps**
  - Laser injected on top of plasma
  - Canonical momentum conservation condition - plasma initially at rest
  - Set initial particle momentum in polarisation direction

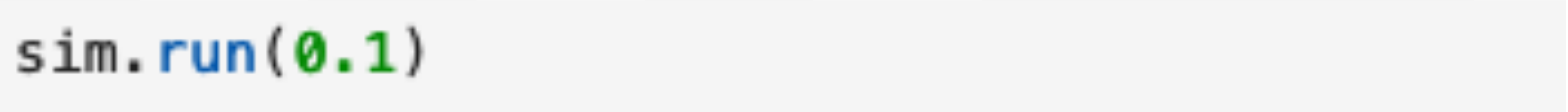


Custom p_y



```
x = (electrons.particles['ix'] + electrons.particles['x']) * electrons.dx

p0 = E0 / omega

for i in range(x.shape[0]):
    if ( x[i] > x0 ) and ( x[i] < (x0 + width)):
        electrons.particles['uy'][i] += p0 * np.cos( kx * ( x[i] - x0 ) )
```

Run



```
sim.run(0.1)
```

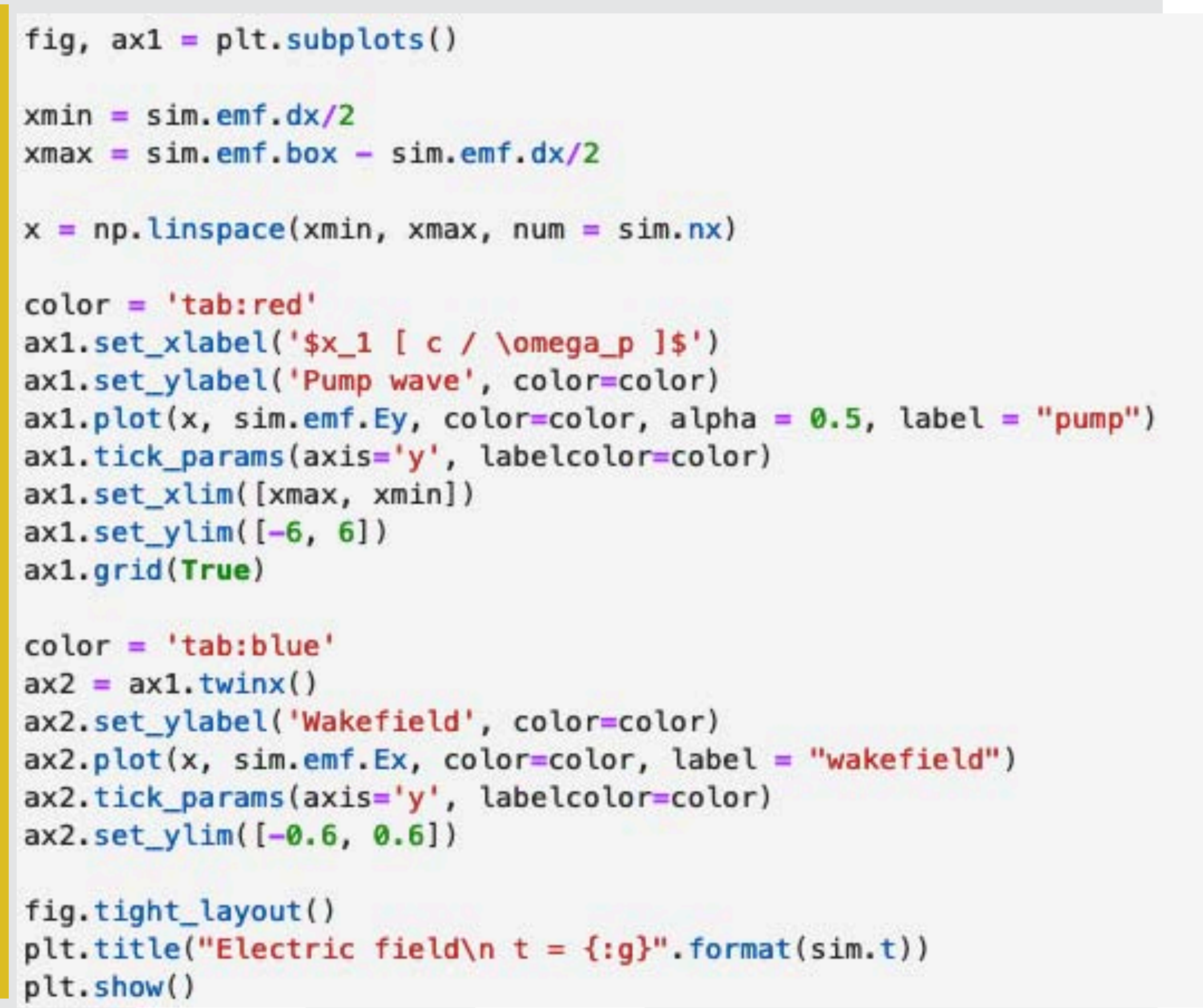
Plotting data - laser and wakefield

- **This data is available as properties of the sim.emf and sim.current objects**
  - Electric field
    - sim.emf.E[x|y|z]
  - Magnetic field
    - sim.emf.B[x|y|z]
  - Electric current
    - sim.current.J[x|y|z]

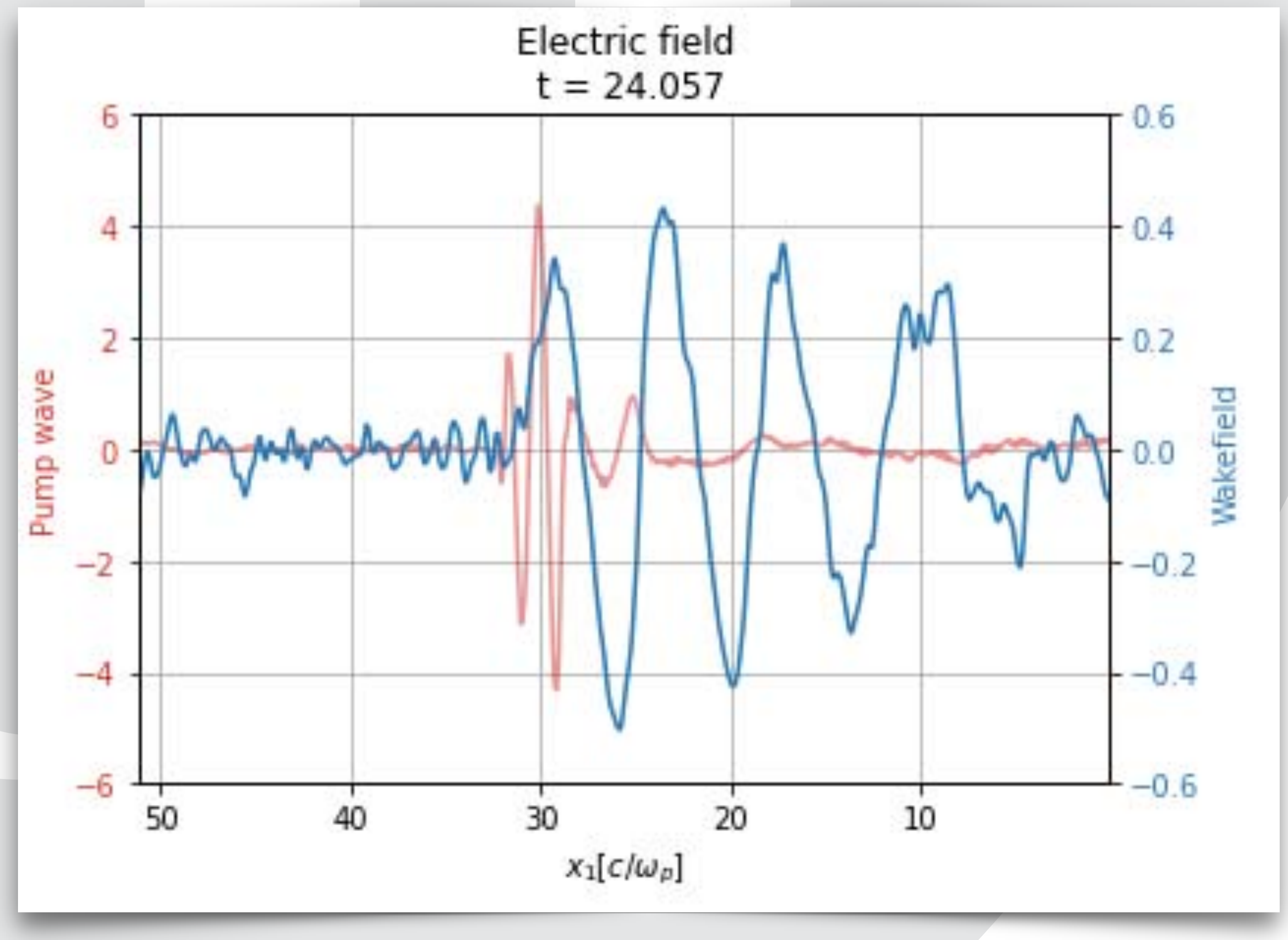
- **Each of these properties is available as a NumPy array**
  - The array dimensions are the same as the simulation grid

- **Data can be plotted using any Python tool**
  - Matplotlib works fine

Plot E_y (laser) and E_x wake field



```
fig, ax1 = plt.subplots()

xmin = sim.emf.dx/2
xmax = sim.emf.box - sim.emf.dx/2

x = np.linspace(xmin, xmax, num = sim.nx)

color = 'tab:red'
ax1.set_xlabel('$x_1 [ c / \omega_p ]$')
ax1.set_ylabel('Pump wave', color=color)
ax1.plot(x, sim.emf.Ey, color=color, alpha = 0.5, label = "pump")
ax1.tick_params(axis='y', labelcolor=color)
ax1.set_xlim([xmax, xmin])
ax1.set_ylim([-6, 6])
ax1.grid(True)

color = 'tab:blue'
ax2 = ax1.twinx()
ax2.set_ylabel('Wakefield', color=color)
ax2.plot(x, sim.emf.Ex, color=color, label = "wakefield")
ax2.tick_params(axis='y', labelcolor=color)
ax2.set_ylim([-0.6, 0.6])

fig.tight_layout()
plt.title("Electric field\n t = {:g}".format(sim.t))
plt.show()
```

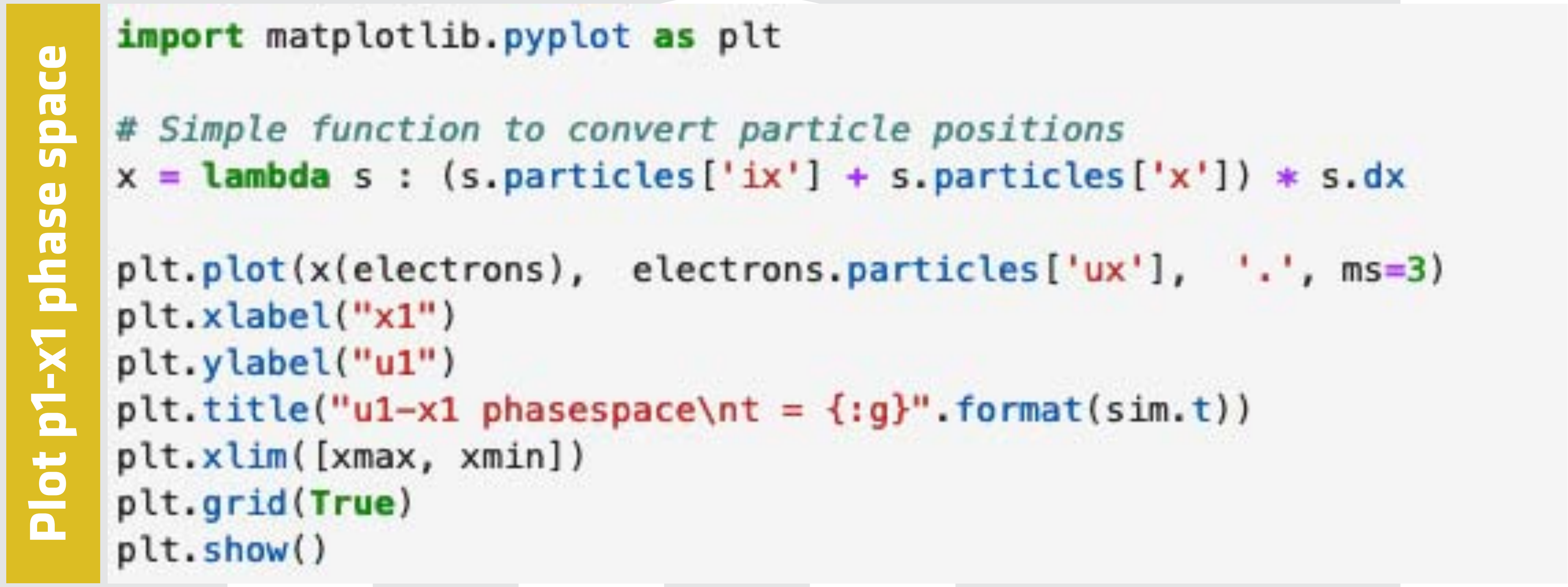


Plotting data - phase space

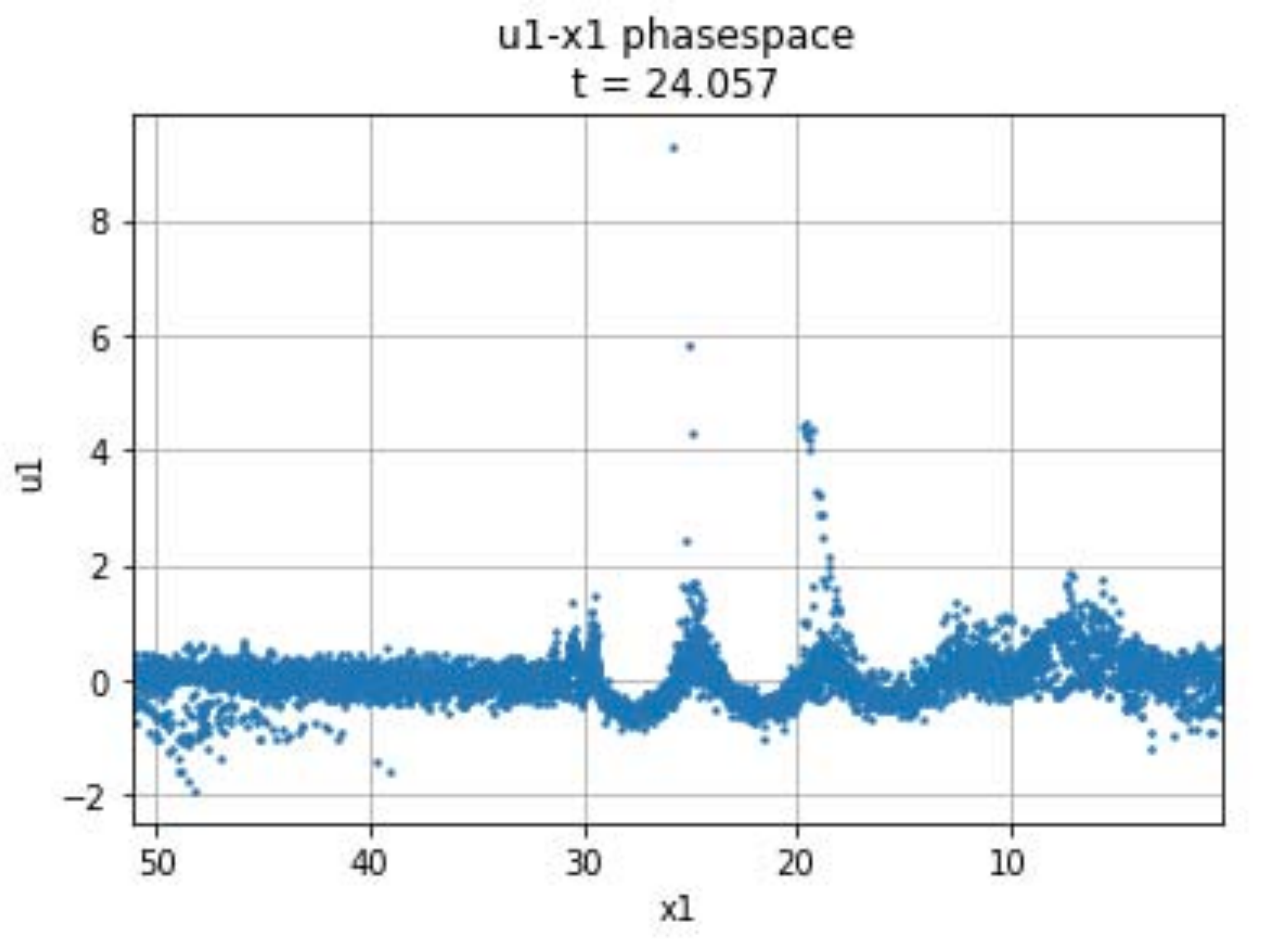
- **Particle data is available using the particles property of each species object**
  - This will be a NumPy array of structures containing
    - ix - the particle cell
    - x - the particle position inside the cell
    - ux, uy, uz - particle generalized velocities

- **These can be easily used to produce a phase space plot for the simulation**
  - Note that we have to convert the cell index / position to simulation position

Plot p1-x1 phase space



```
import matplotlib.pyplot as plt

# Simple function to convert particle positions
x = lambda s : (s.particles['ix'] + s.particles['x']) * s.dx

plt.plot(x(electrons),  electrons.particles['ux'],  '.', ms=3)
plt.xlabel("x1")
plt.ylabel("u1")
plt.title("u1-x1 phasespace\nt = {:g}".format(sim.t))
plt.xlim([xmax, xmin])
plt.grid(True)
plt.show()
```



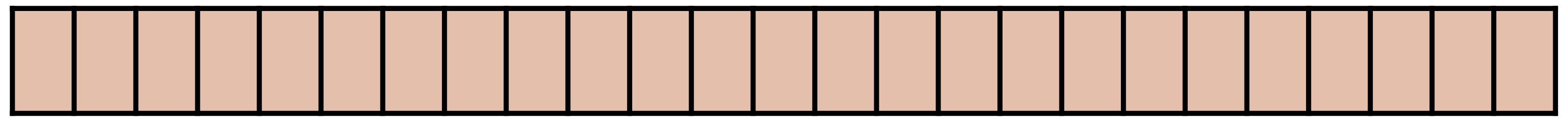
LWFA simulation design

Laser Wakefield Acceleration

3D Simulation using the OSIRIS code

Initialise ZPIC simulation of a LWFA (1D)

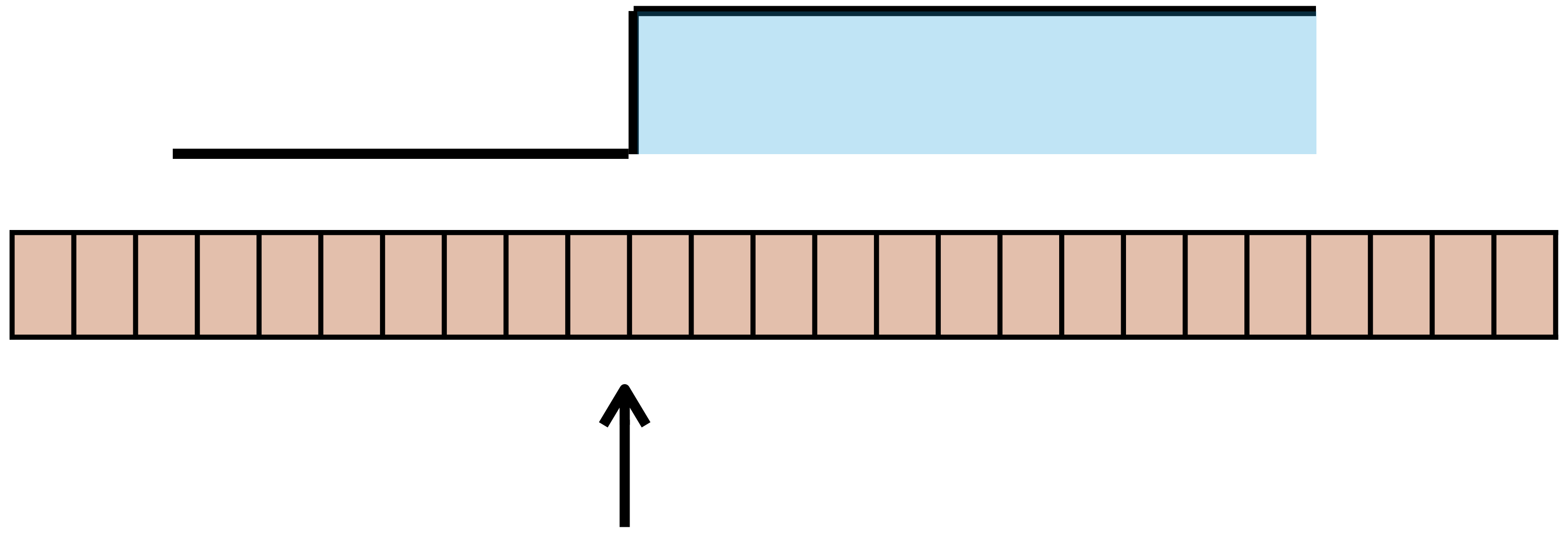
Box and resolution



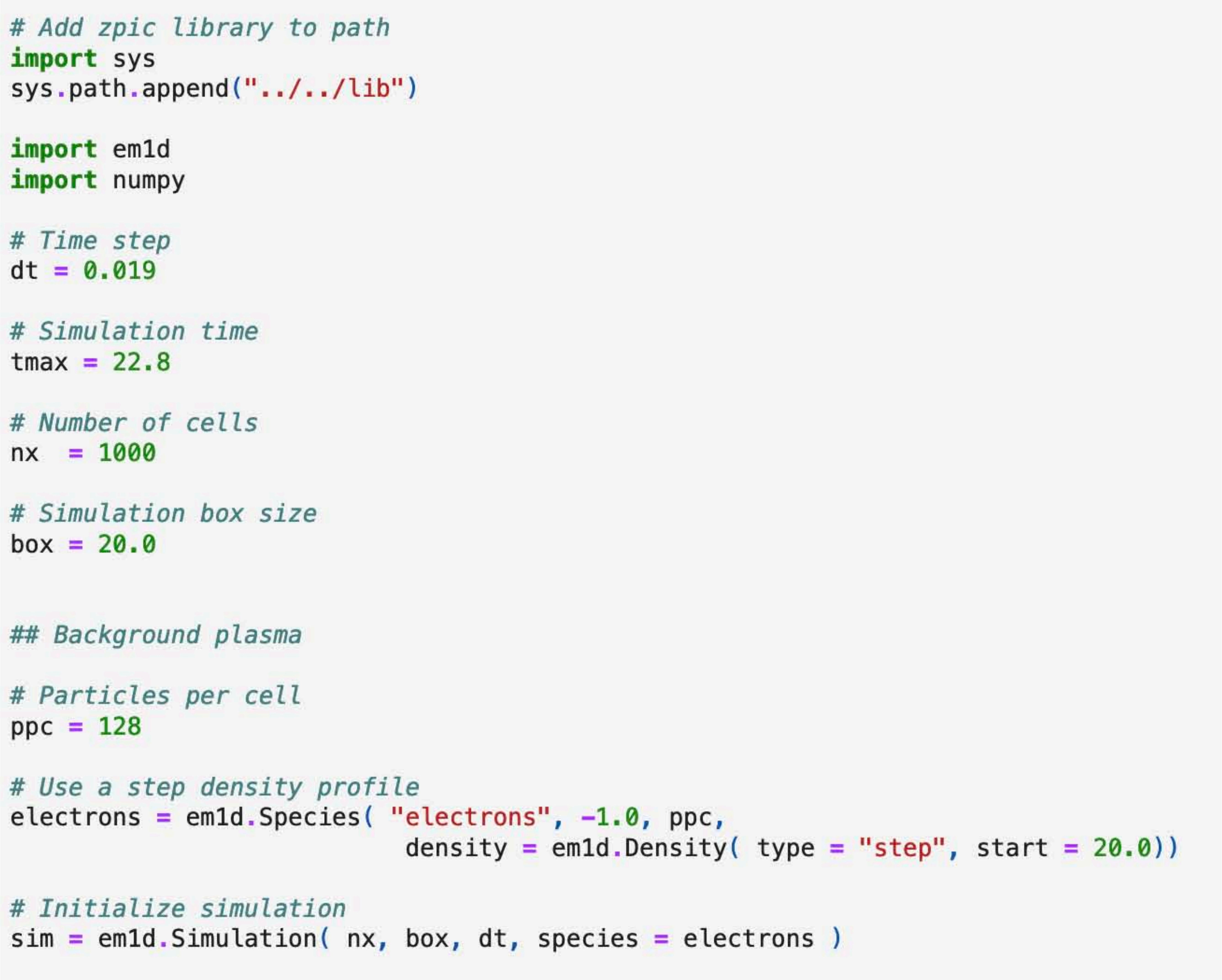
$$L = 20c/\omega_p$$

$$\Delta x = L/n_x = 20/1000 = 0{,}02$$

Species



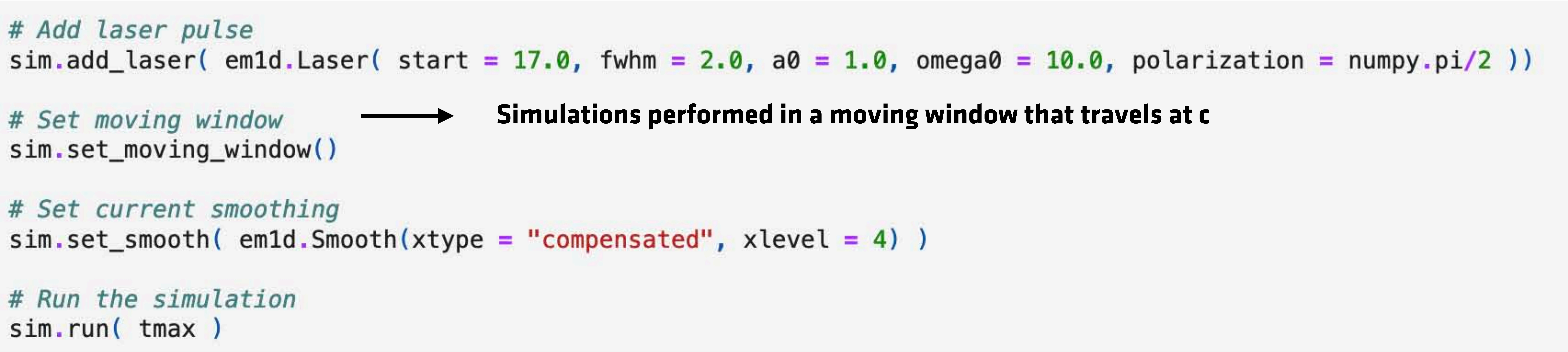
$$x_1 = 20c/\omega_p$$



```
# Add zpic library to path
import sys
sys.path.append("../../lib")

import em1d
import numpy

# Time step
dt = 0.019

# Simulation time
tmax = 22.8

# Number of cells
nx   = 1000

# Simulation box size
box = 20.0


## Background plasma

# Particles per cell
ppc = 128

# Use a step density profile
electrons = em1d.Species( "electrons", -1.0, ppc,
                           density = em1d.Density( type = "step", start = 20.0))

# Initialize simulation
sim = em1d.Simulation( nx, box, dt, species = electrons )
```

Initialise ZPIC simulation of a LWFA (1D)

Laser and moving window



```
# Add laser pulse
sim.add_laser( em1d.Laser( start = 17.0, fwhm = 2.0, a0 = 1.0, omega0 = 10.0, polarization = numpy.pi/2 ))

# Set moving window
sim.set_moving_window()

# Set current smoothing
sim.set_smooth( em1d.Smooth(xtype = "compensated", xlevel = 4) )

# Run the simulation
sim.run( tmax )
```

→ **Simulations performed in a moving window that travels at c**

Laser - physical parameters

$$\sigma_t[\text{fwhm}] = 2c/\omega_p$$

$$\omega_L = 10\omega_p$$

$$a_0 = 1$$

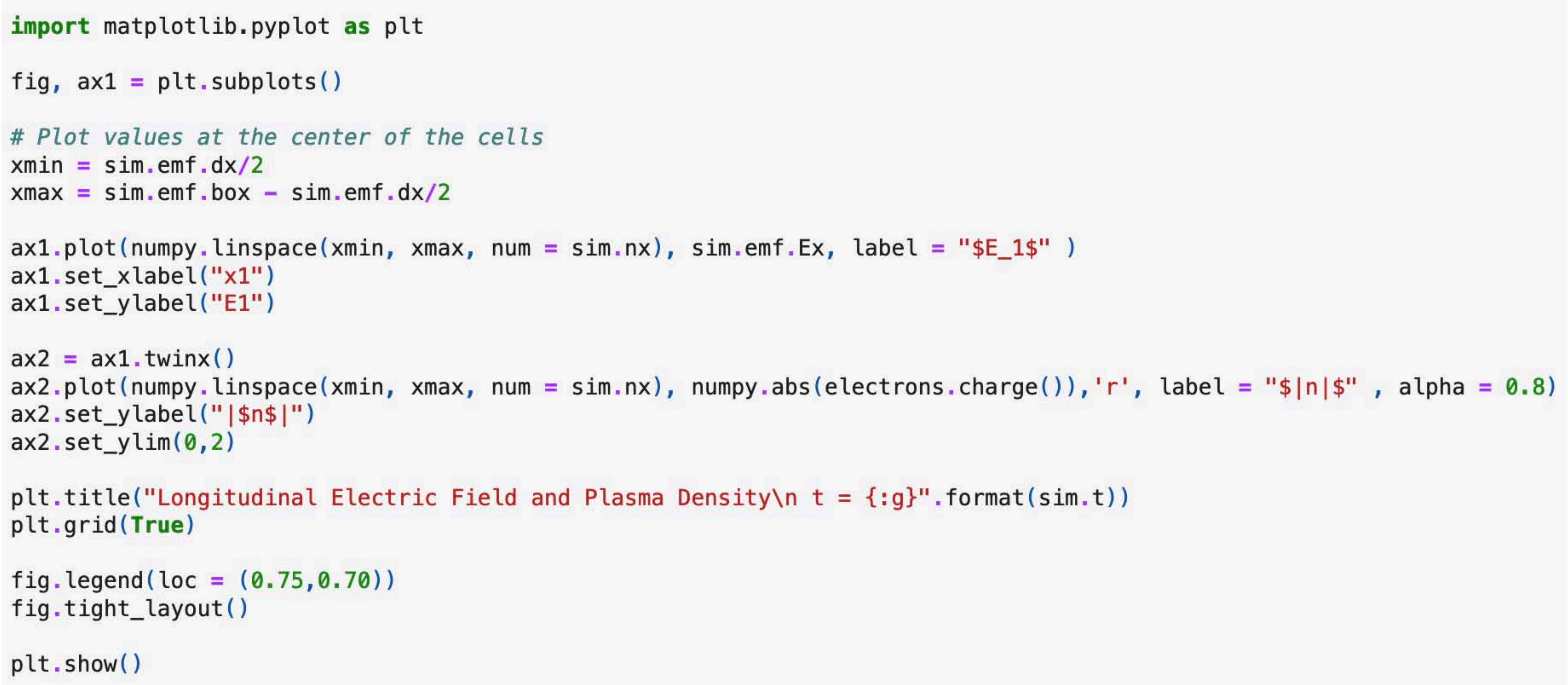
- **More lasers?**
  - add several sim.add_laser(...) sections
  - 1D only (so far)

Useful diagnostics

- **Plasma density**
  - Charge density of the background plasma
  - Wave structure and particle loading

- **Longitudinal electric field**
  - Accelerating / decelerating fields
  - Useful engineering formula for de-normalization:
    - Eaccel [V/m] = $0.96 \times n_0^{1/2}[\mathrm{cm}^{-3}] * E_{sim}$

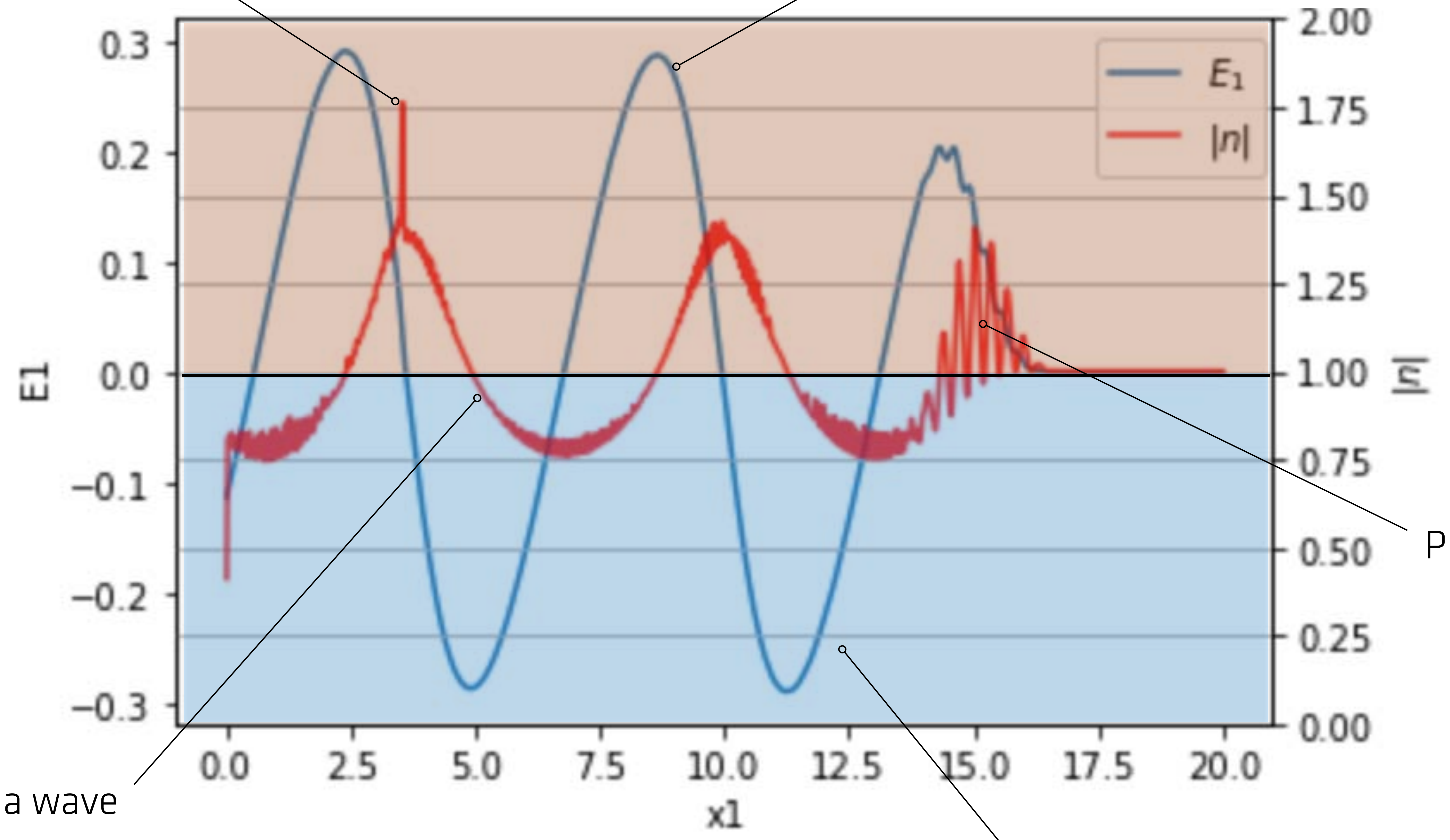
Plot E_x and n_e



```
import matplotlib.pyplot as plt

fig, ax1 = plt.subplots()

# Plot values at the center of the cells
xmin = sim.emf.dx/2
xmax = sim.emf.box - sim.emf.dx/2

ax1.plot(numpy.linspace(xmin, xmax, num = sim.nx), sim.emf.Ex, label = "$E_1$" )
ax1.set_xlabel("x1")
ax1.set_ylabel("E1")

ax2 = ax1.twinx()
ax2.plot(numpy.linspace(xmin, xmax, num = sim.nx), numpy.abs(electrons.charge()),'r', label = "$|n|$" , alpha = 0.8)
ax2.set_ylabel("|$n$|")
ax2.set_ylim(0,2)

plt.title("Longitudinal Electric Field and Plasma Density\n t = {:g}".format(sim.t))
plt.grid(True)

fig.legend(loc = (0.75,0.70))
fig.tight_layout()

plt.show()
```

Plasma density and longitudinal electric field

Simulations performed in a moving window that travels at c



Particle injection

decelerating field when E>0!

Plasma wave

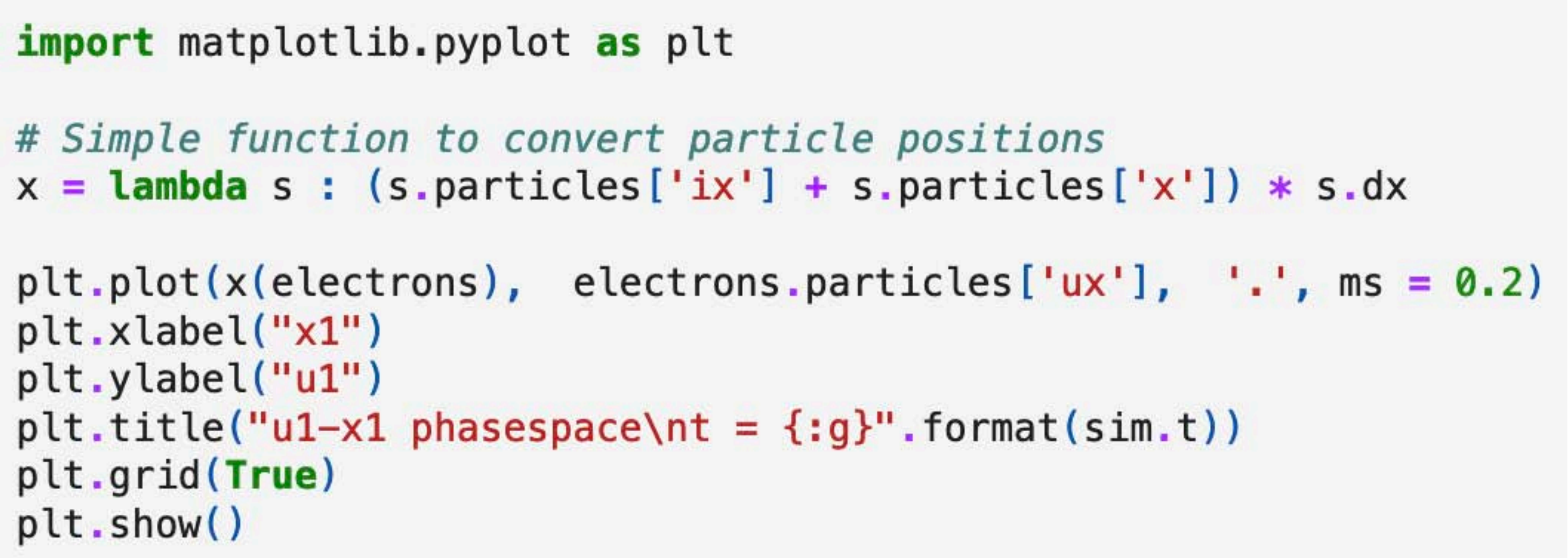
Plasma oscillating in the laser field

accelerating field is when E<0!

Useful diagnostics - Phasespace

- **Particle phasespace**
  - Show particle momenta as a function of position
  - Most common is u_1/x_1
  - Wave structure and particle acceleration
- **Useful de-normalization formula**
  - (ultra-relativistic) electron energy
    - E [MeV] = p_1 x 0.5 ~ gamma x 0.5

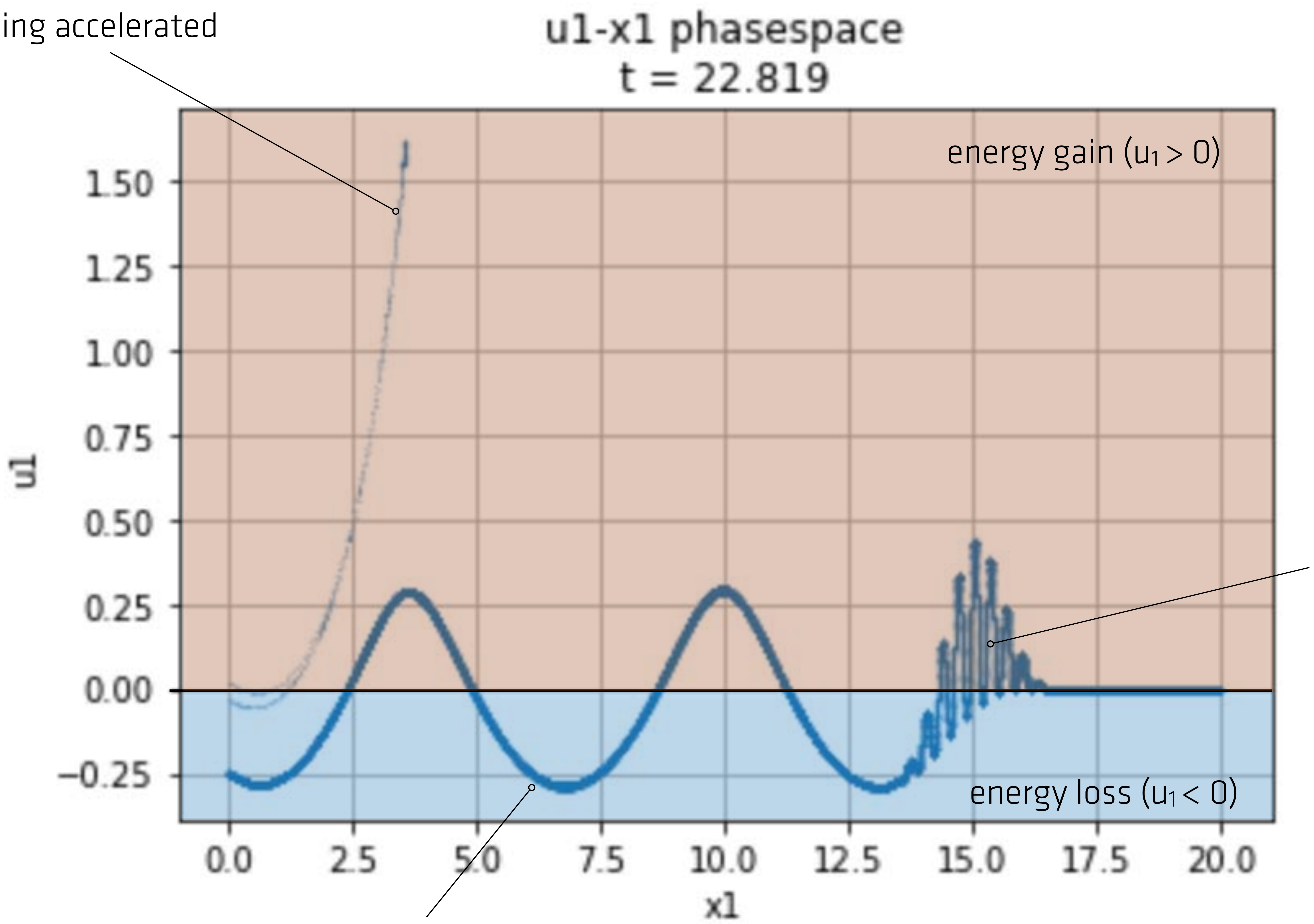
Plot p_1x_1 phasespace



```python
import matplotlib.pyplot as plt

# Simple function to convert particle positions
x = lambda s : (s.particles['ix'] + s.particles['x']) * s.dx

plt.plot(x(electrons),  electrons.particles['ux'],  '.', ms = 0.2)
plt.xlabel("x1")
plt.ylabel("u1")
plt.title("u1-x1 phasespace\nt = {:g}".format(sim.t))
plt.grid(True)
plt.show()
```

Particle acceleration and deceleration

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



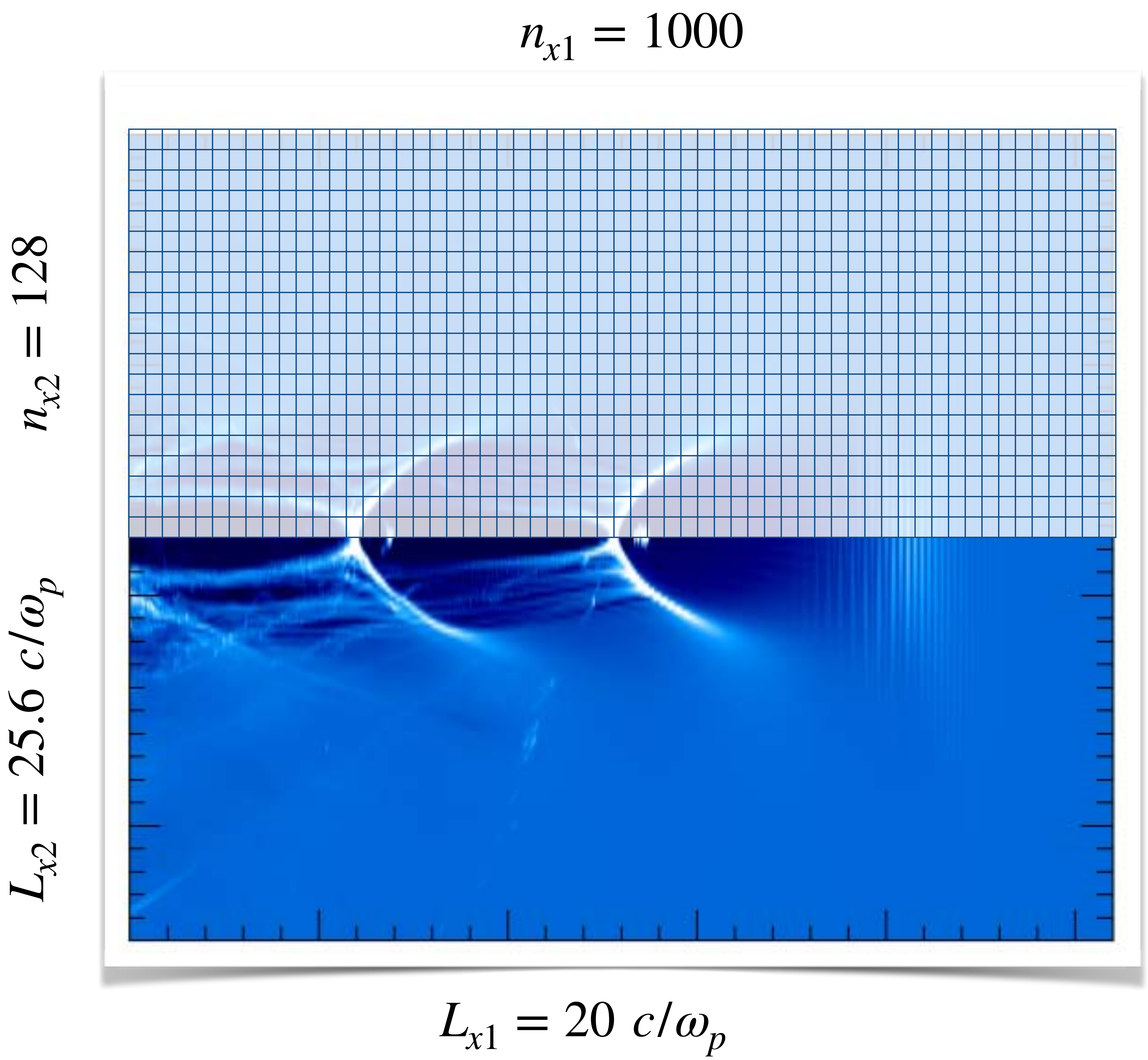
Trapped particles being accelerated

Particles oscillating in the laser field

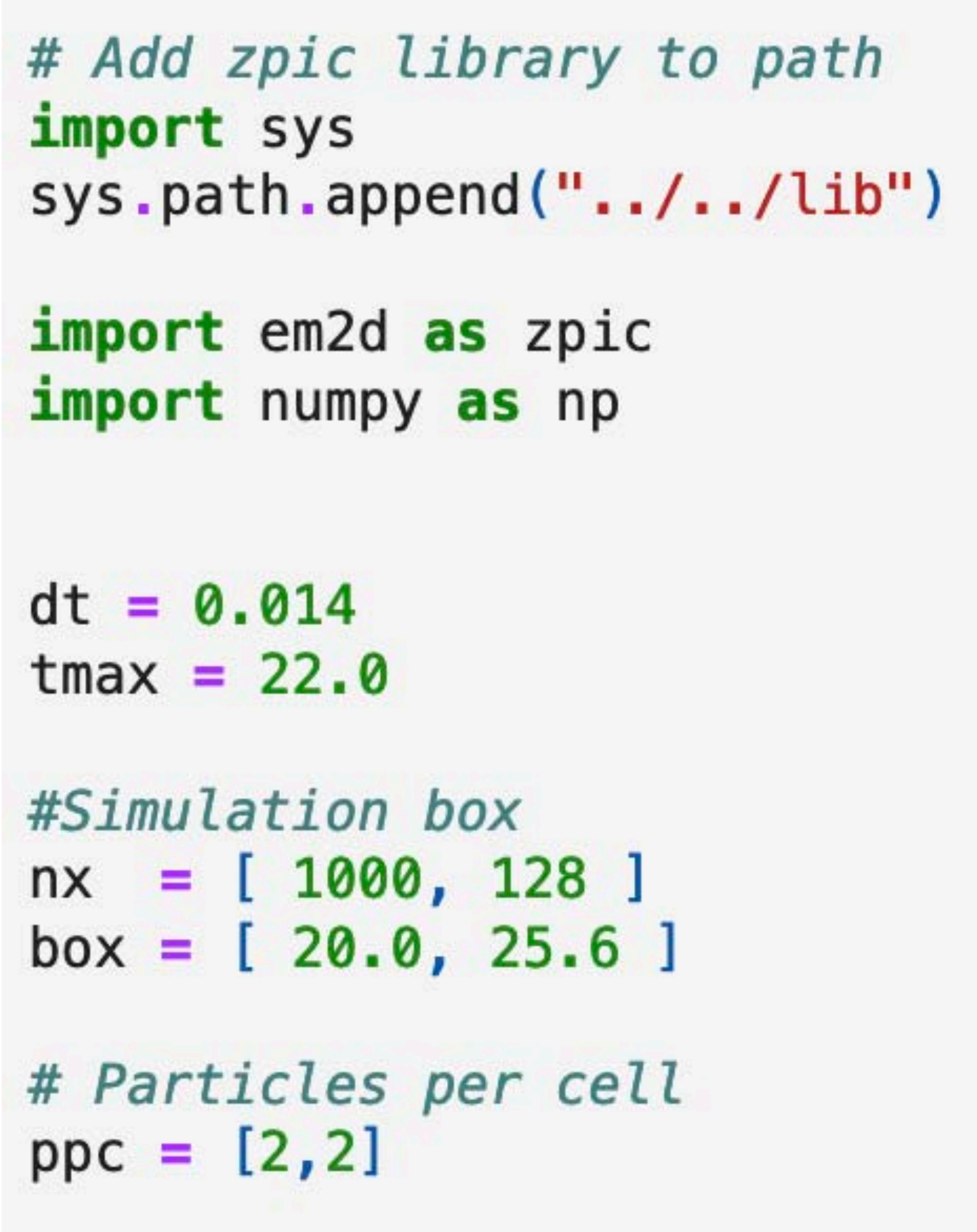
Plasma oscillating in the wakefield

LWFA simulation in 2D

2D simulation box

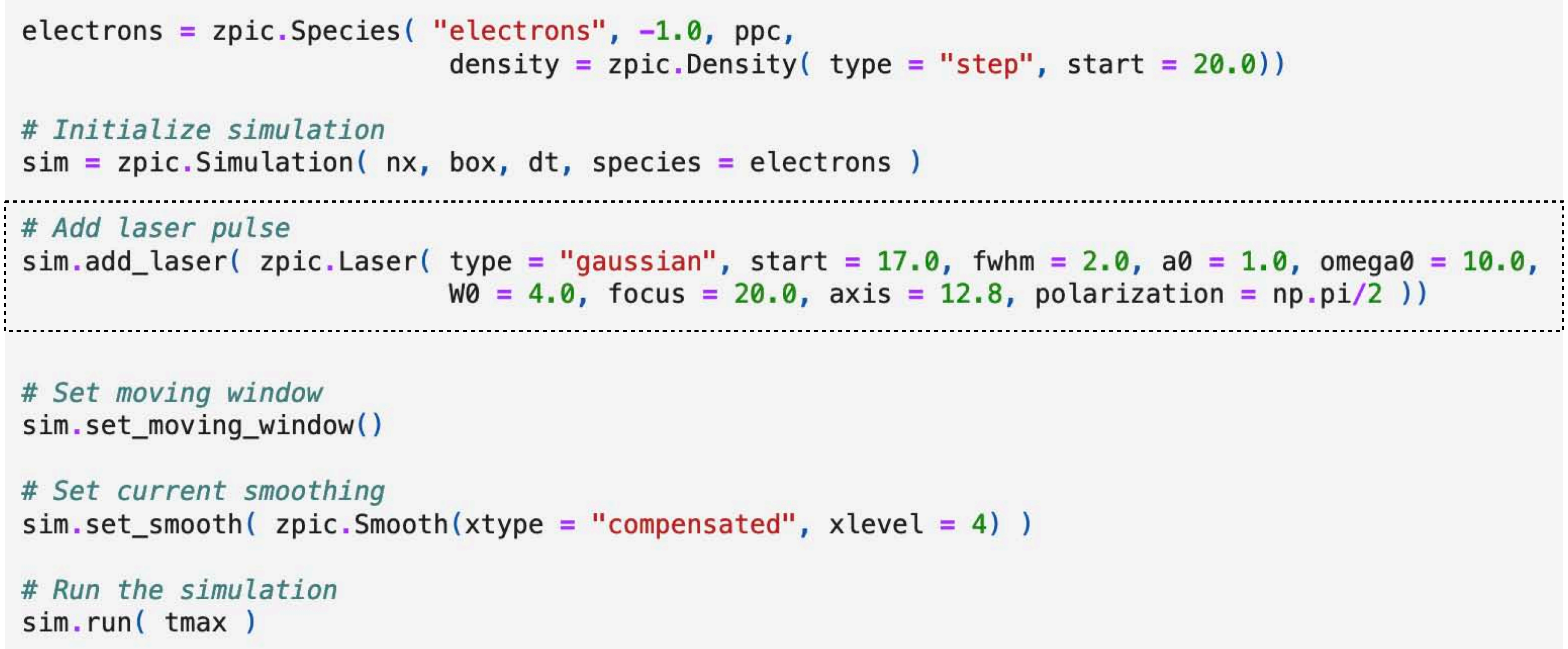


2D simulation initialisation



```
# Add zpic library to path
import sys
sys.path.append("../../lib")

import em2d as zpic
import numpy as np


dt = 0.014
tmax = 22.0

#Simulation box
nx  = [ 1000, 128 ]
box = [ 20.0, 25.6 ]

# Particles per cell
ppc = [2,2]
```

LWFA simulation in 2D

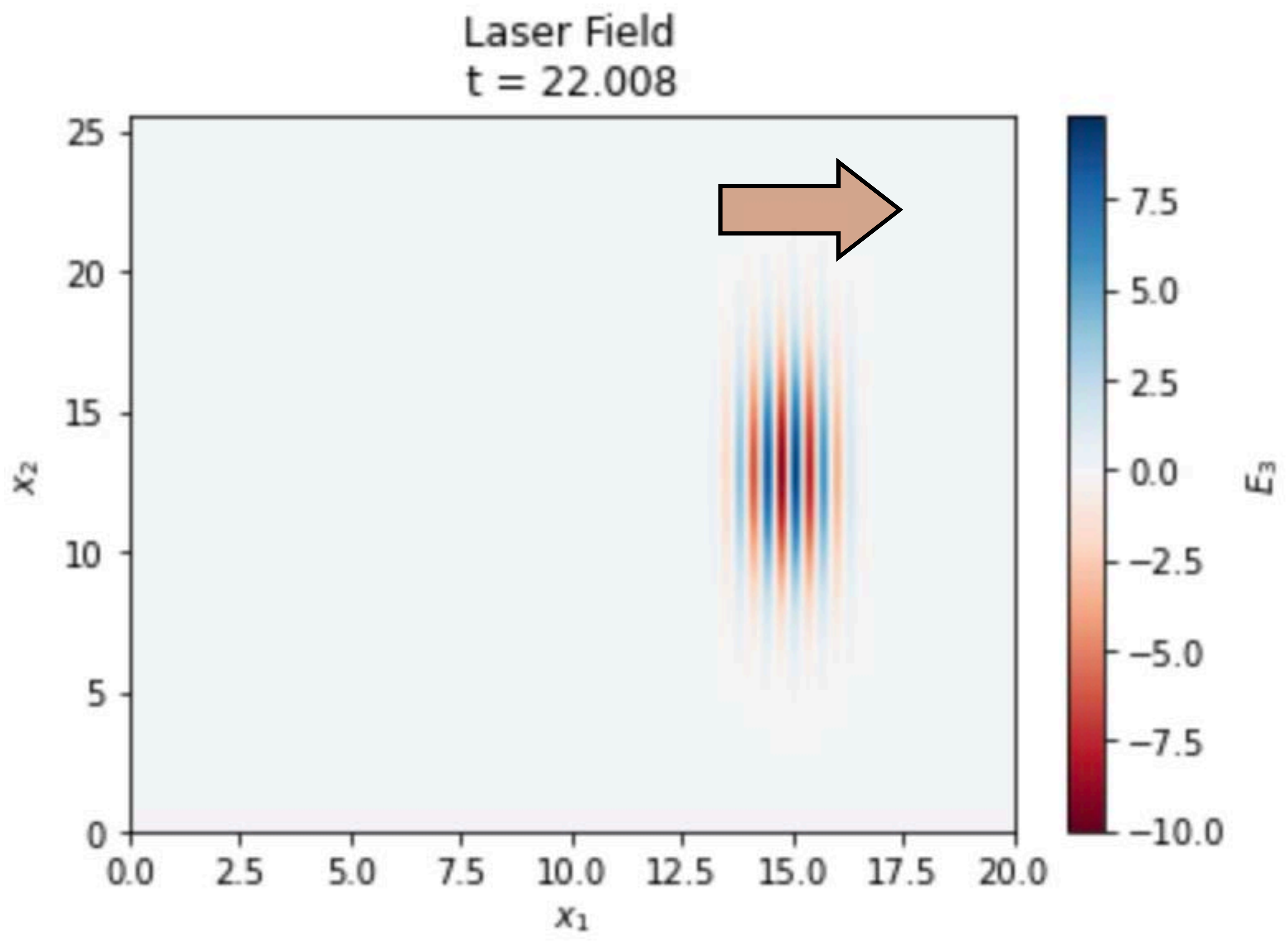


```python
electrons = zpic.Species( "electrons", -1.0, ppc,
                           density = zpic.Density( type = "step", start = 20.0))

# Initialize simulation
sim = zpic.Simulation( nx, box, dt, species = electrons )

# Add laser pulse
sim.add_laser( zpic.Laser( type = "gaussian", start = 17.0, fwhm = 2.0, a0 = 1.0, omega0 = 10.0,
                           W0 = 4.0, focus = 20.0, axis = 12.8, polarization = np.pi/2 ))


# Set moving window
sim.set_moving_window()

# Set current smoothing
sim.set_smooth( zpic.Smooth(xtype = "compensated", xlevel = 4) )

# Run the simulation
sim.run( tmax )
```

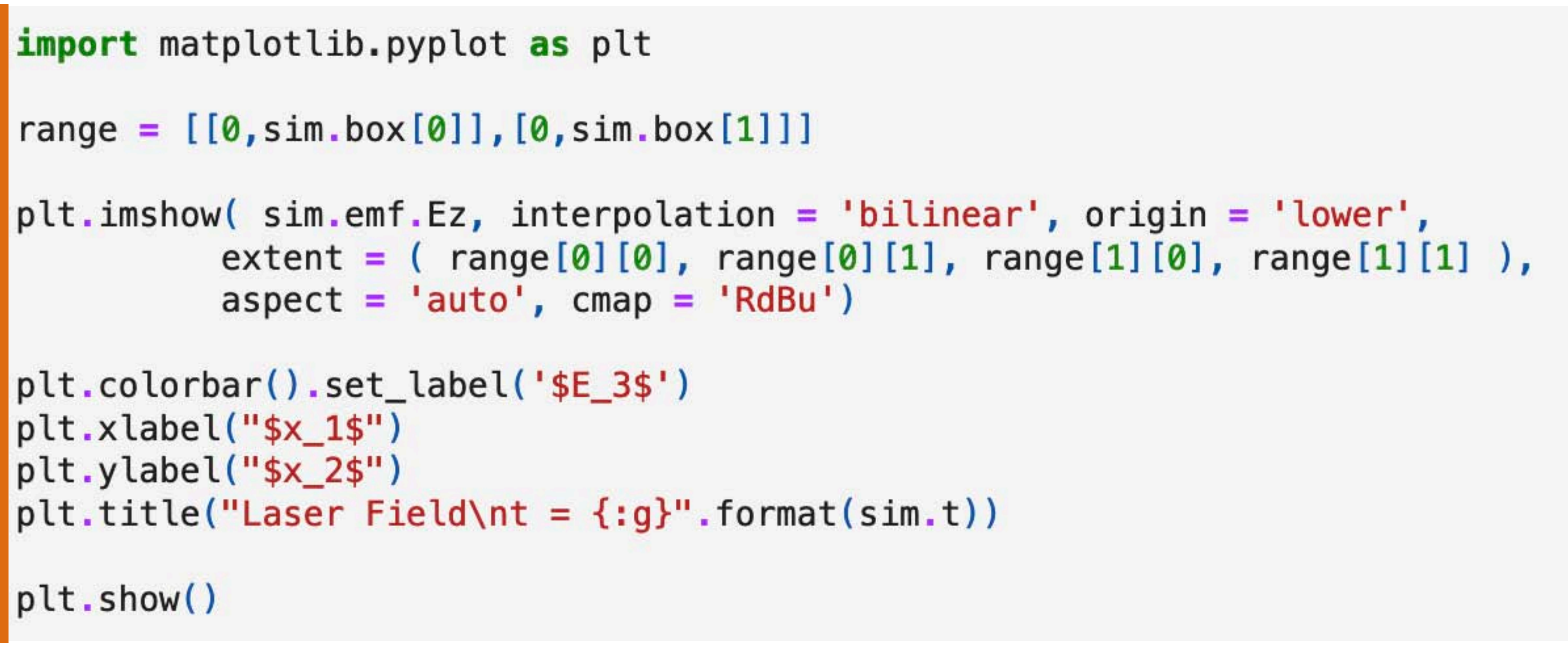
Useful diagnostics in 2D - Transverse electric fields

- **Transverse electric fields**
  - Laser pulse
  - Transverse wave structure

- **Also laser pulse**
  - In this example the laser was polarized out of the plane



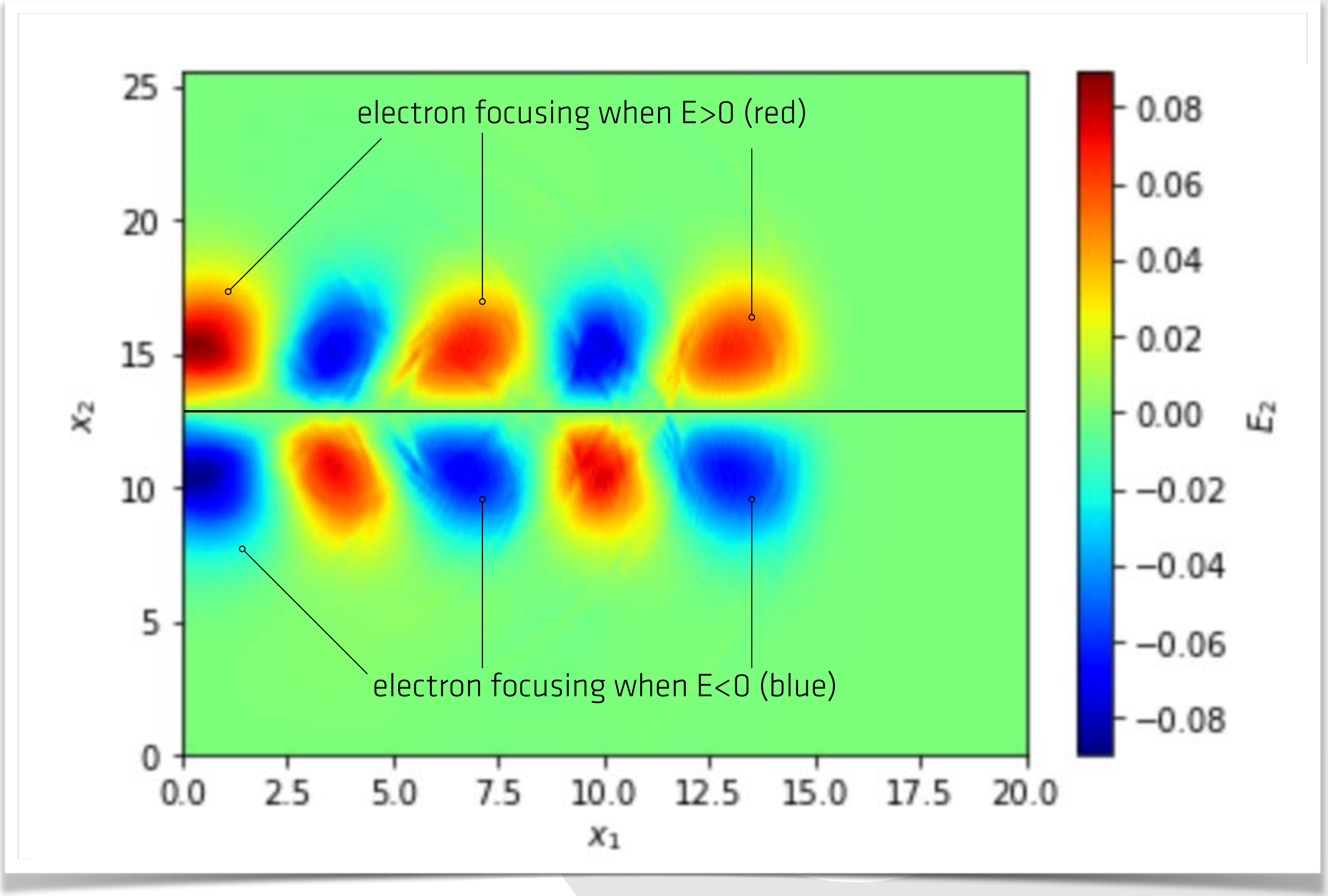
Plot 2D grid diagnostics



```
import matplotlib.pyplot as plt

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

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

plt.colorbar().set_label('$E_3$')
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.title("Laser Field\nt = {:g}".format(sim.t))

plt.show()
```

Focusing fields: non relativistic particles

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

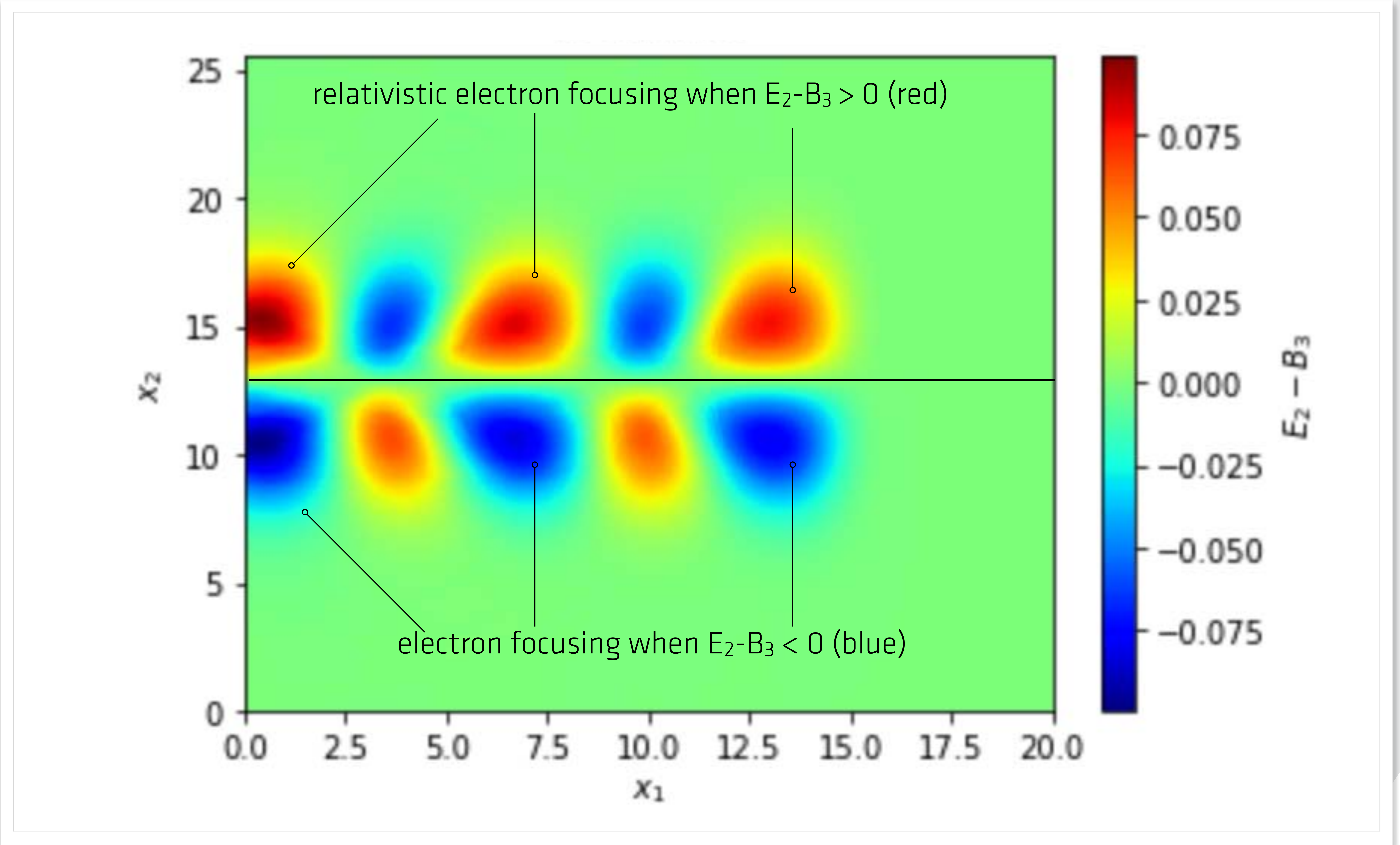


Simulations performed in a moving window that travels at c

Focusing fields: Ultra-relativistic particles

Focusing force for a ultra-relativistic particle:

$$\mathbf{E_r + v_{||} \times B_\Theta/c \simeq E_r - B_\Theta}$$



Suggestions

- **Increase laser a_0**
  - Observe that the amplitude of the plasma wave grows
  - What happens to the plasma density in 2D?
  - Presence of wavebreaking and electron injection

- **Decrease (normalized) laser frequency**
  - Equivalent to increase plasma density
  - When is the laser reflected by the plasma?

- **Dynamics of injected electron beam**
  - Add external e-/e+ beam (see PWFA notebook)
  - Where is the beam simultaneously focused and accelerated?

- **Use up-ramps and down-ramps**
  - Down-ramps can induce electron trapping and acceleration
  - Can you observe this mechanism?

Overview

CERN Large Hadron Collider

Accelerator Tunnel

At the end of this session, I should be able to

- **Understand the importance of laser plasma accelerators**
  - Compact accelerators
  - Applications related to light sources are on the way
- **Setup laser wakefield acceleration simulation**
  - Understand how to choose box size and resolutions
  - Setup simulations in 1D and 2D
  - Plot key diagnostics (fields, plasma, phasespace)
- **Interpret diagnostics**
  - Electro dynamics in electric fields
  - Use “denormalization” engineering formulas

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

ZPIC website
ricardo-fonseca.github.io/zpic