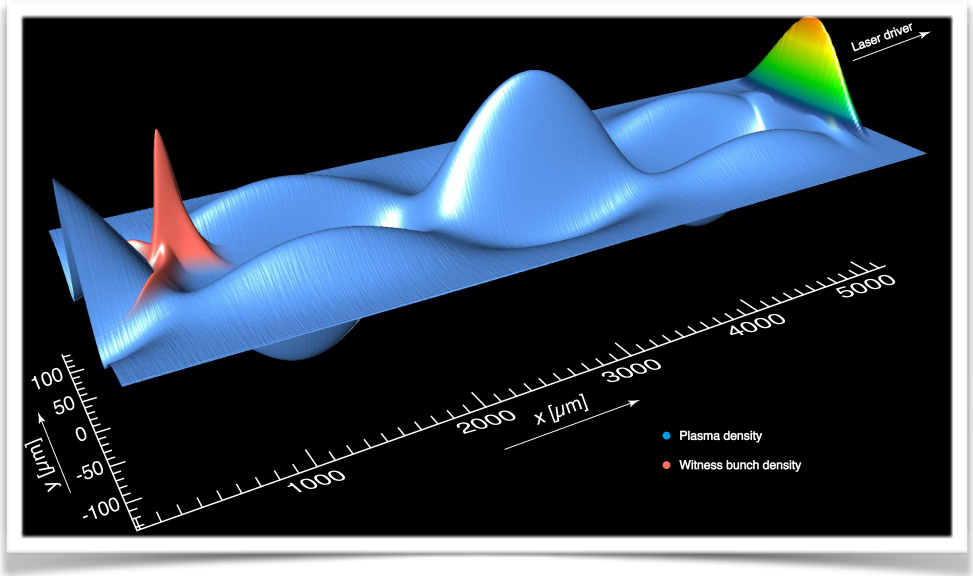
The Particle-In-Cell method: lesson 2

Elisabetta Boella

e.boella@lancaster.ac.uk
https://www.lancaster.ac.uk/staff/boella/
@EliBASEtta
Physics Department
Lancaster University (Lancaster, UK)
Cockcroft Institute
Sci-Tech Daresbury (Warrington, UK)



Acknowledgements

- Ricardo Fonseca (Instituto Superior Técnico, Lisbon, Portugal)
- Maria Elena Innocenti (KU Leuven, Leuven, Belgium)
- Giovanni Lapenta (KU Leuven, Leuven, Belgium)

Higher order interpolation functions reduce numerical noise

For simplicity 1D electrostatic case

Fourier transform of the force applied to one macroparticle: $F(k) = \int_{-\infty}^{\infty} F(x) \mathrm{e}^{-ikx}\, dx$

Force applied to particle i: $F_i = q_i \Delta x \sum_j E_j W(x_j - x_i)$

electric field @grid node j interpolation function from grid node j to particle i

Inserting the expression for the force into the expression for the Fourier transform, we get

$$F(k) = q \left[\Delta x \sum_j E_j \mathrm{e}^{-ikx_j} \right] \int_{-\infty}^{\infty} W(x_j - x) \mathrm{e}^{ik(x_j - x)}\, dx = qE(k)W(-k)$$

interpolation function

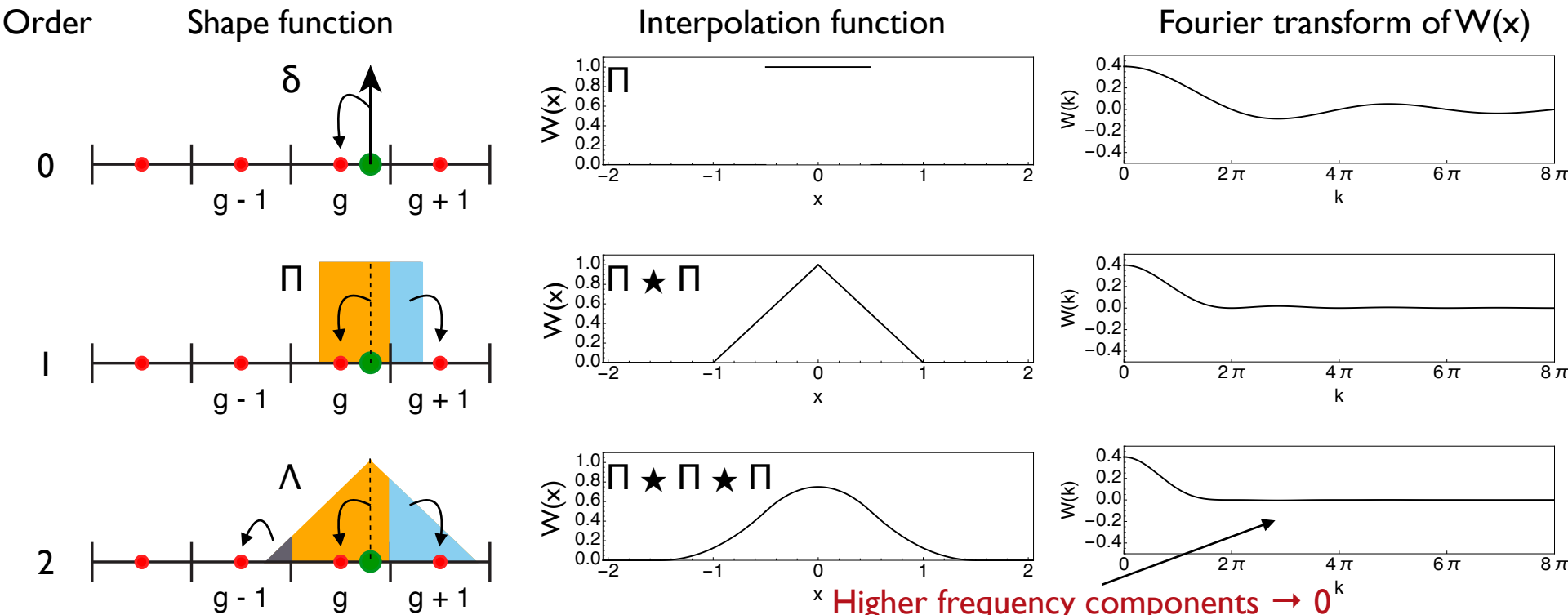
Fourier transform

Discrete Fourier transform of E_j

Periodic

Importance of high k-modes depend of how fast W(k) → 0 for k → ± ∞

Higher order interpolation functions reduce numerical noise



Numerical heating due to aliasing can be reduce resolving the Debye length

For simplicity 1D electrostatic case

Let us define the density of particle centres as $n(x) = \sum_{i=1}^{N_p} \delta(x - x_i)$

Then the charge density on the grid will be $\rho(x_g) = \frac{q}{\Delta x} \int n(x') W(x_g - x')\, dx'$

Similarly the charge density in the continuum will be $\rho'(x) = \frac{q}{\Delta x} \int n(x') W(x - x')\, dx' = \frac{q}{\Delta x} n(x) \star W(x)$

with Fourier transform $[\mathcal{F}\rho'(x)](k) = \hat{\rho}'(k) = \frac{q}{\Delta x} \hat{n}(k) \hat{W}(k)$

where the following definitions are valid $\hat{\rho}'(k) = \int_{-\infty}^{\infty} \rho'(x) \mathrm{e}^{-ikx}\, dx$ and $\rho'(x) = \int_{-\infty}^{\infty} \frac{1}{2\pi} \hat{\rho}'(k) \mathrm{e}^{ikx}\, dk$

Numerical heating due to aliasing can be reduce resolving the Debye length

The Fourier transform of the discrete charge density is $[\mathcal{F}\rho(x_g)](k) = \hat{\rho}(k) = \Delta x \sum_{g=-\infty}^{\infty} \rho(x_g) \mathrm{e}^{-ikx_g}$

with $\rho(x_g) = \int_{-\pi/\Delta x}^{\pi/\Delta x} \frac{1}{2\pi} \hat{\rho}(k) \mathrm{e}^{ikx_g}\, dk$ (1)

At x_g $\rho(x_g) = \rho'(x_g) \Rightarrow$ $\rho(x_g) = \int_{-\infty}^{\infty} \frac{1}{2\pi} \hat{\rho'}(k) \mathrm{e}^{ikx_p}\, dk = \int_{-\pi/\Delta x}^{\pi/\Delta x} \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} \hat{\rho'}(k') \mathrm{e}^{ik'x_g}\, dk'$ (2)

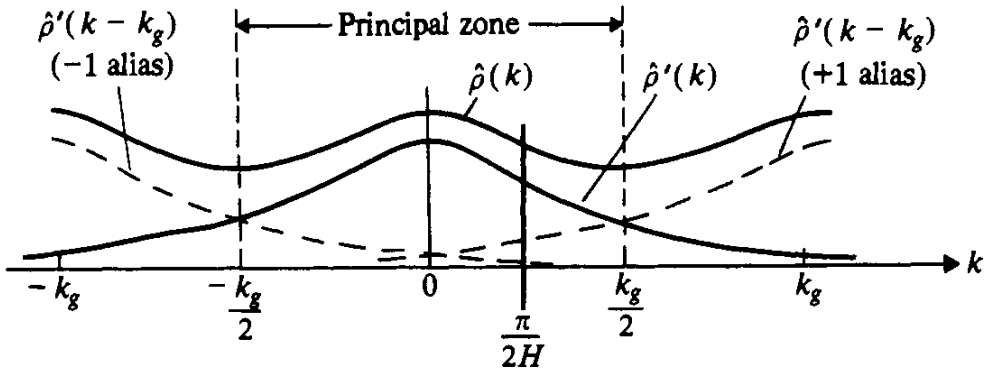
splitting the integral into segments of length $k_g = 2\pi/\Delta x$

changing variable $k' = k - nk_g$

Comparing (1) and (2) we get $\hat{\rho}(k) = \sum_{n=-\infty}^{\infty} \hat{\rho'}(k - nk_g) = \frac{q}{\Delta x} \sum_{n=-\infty}^{\infty} \hat{n}(k - nk_g) \hat{W}(k - nk_g)$

The Fourier transform of the discrete charge density is given by the sum of all the replicas of the Fourier transform of the continuous charge density. This gives rise to a phenomena called aliasing for not properly selected values $k_g = 2\pi/\Delta x$.

Numerical heating due to aliasing can be reduce resolving the Debye length



Aliasing derives from the fact that we are trying to squeeze continuous information into a discrete grid

The aliasing effects in k-space is fed back into the simulation producing numerical noise in the real space

Things become worse and worse if we do not sample enough, e.g. if we consider Δx bigger and bigger
Things become better with higher order interpolation functions whose Fourier transform is 0 for high ks

Aliasing becomes a serious problem if $\Delta x >> \lambda_D$ causing the development of the finite grid instability, which results in unphysical electron heating. In the case of linear interpolation, the instability is avoided if $\Delta x/\lambda_D << \pi$

The described PIC algorithm conserves total momentum

For simplicity 1D electrostatic case

$$\frac{dP}{dt} = \sum_i F_i = \sum_i q_i \Delta x \sum_j E_j W(x_j - x_i) = \Delta x \sum_j E_j \sum_i q_i W(x_j - x_i) = \Delta x \sum_j \rho_j E_j$$

electric field interpolated at particle position x_i

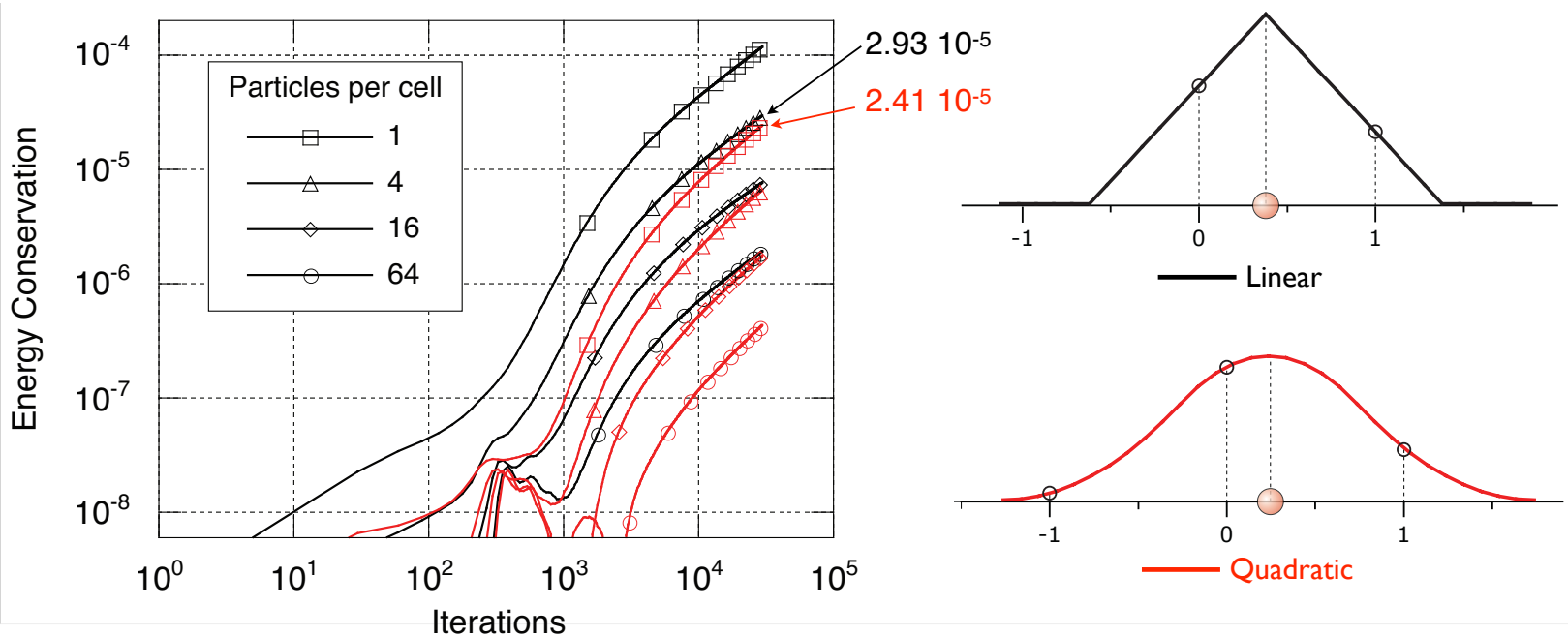
density on the grid at x_j

In a periodic system with left-right symmetry $\Delta x \sum_j \rho_j E_j = 0$

Therefore momentum is conserved!

When momentum is conserved, the total energy is not conserved.
Therefore, it is important to always monitor the evolution of the total energy of your simulation.

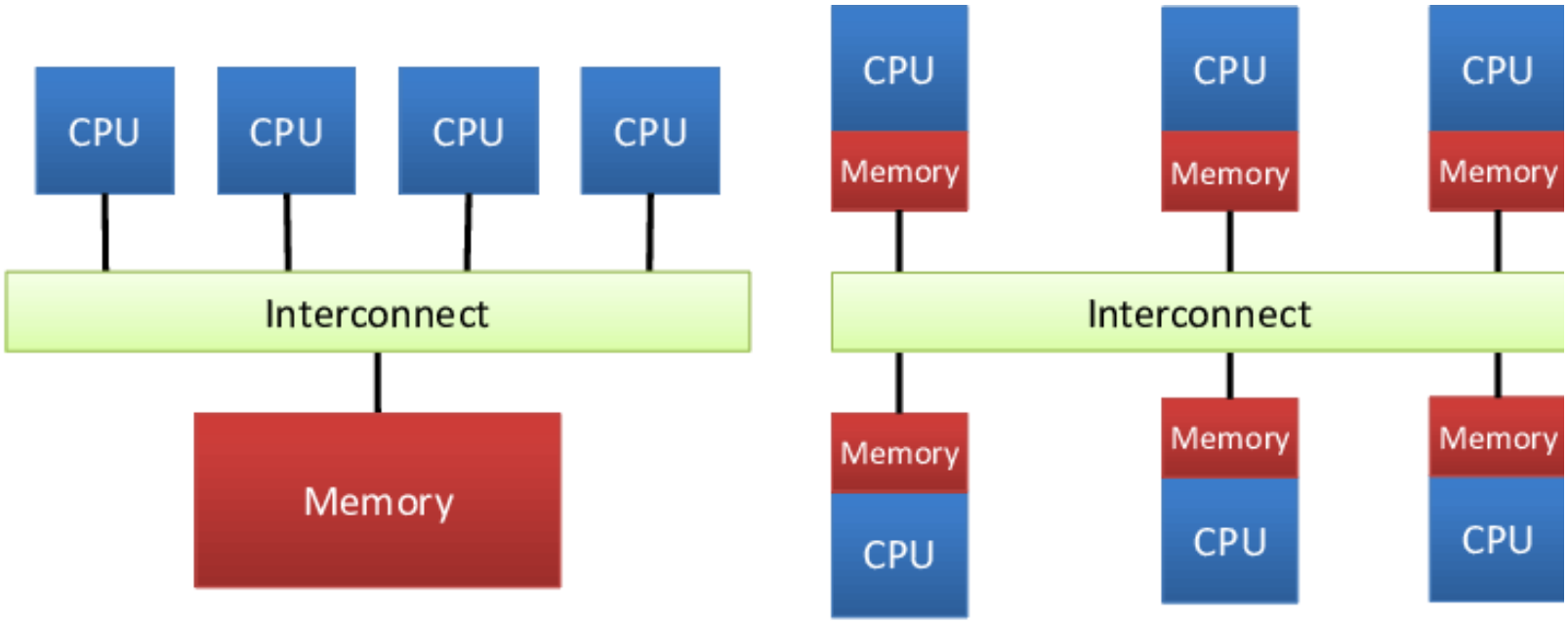
Higher order interpolation and many particle per cell improve energy conservation



Realistic PIC simulations need supercomputers

TOP 500
The List.

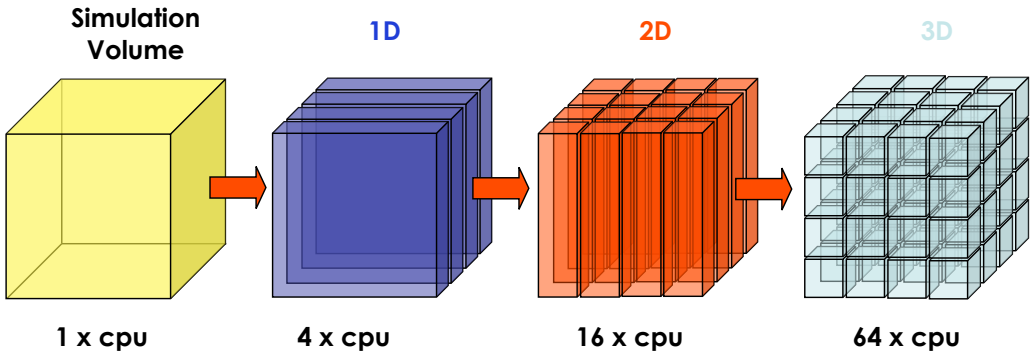
Different parallelism paradigms



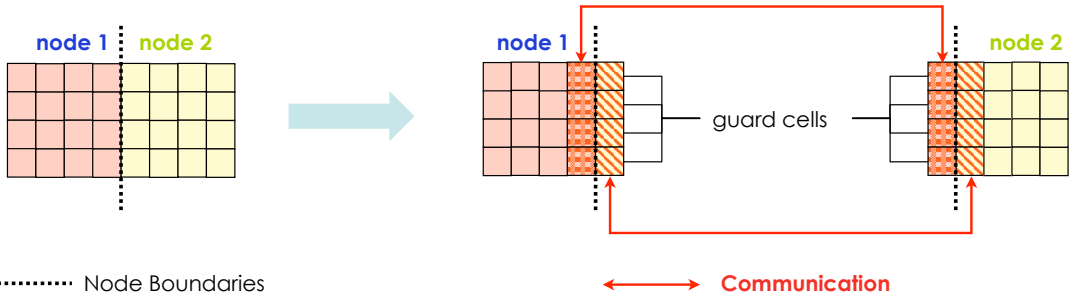
shared memory

distributed memory

Domain decomposition: simulation space is shared among several CPUs



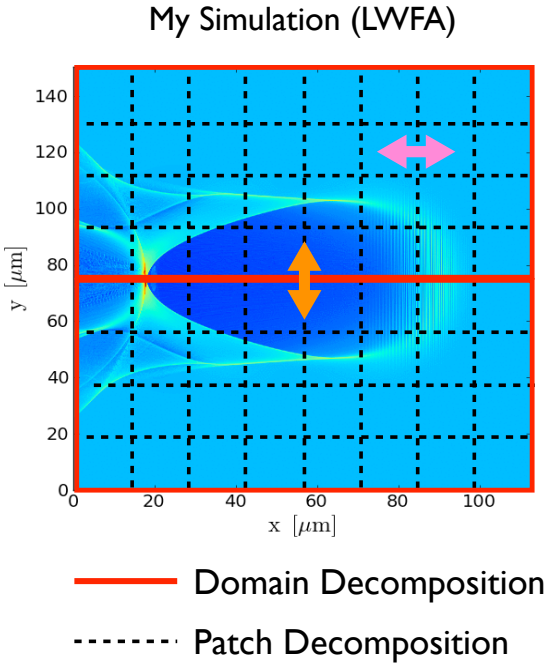
Guard cells are used to communicate among the processors

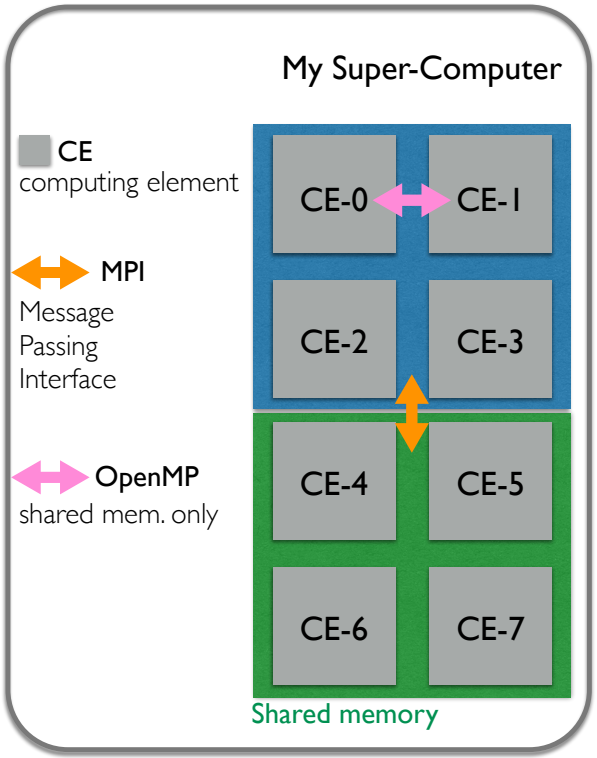


Usually field solver is local
Communication only with neighbouring nodes

Usually the same decomposition for particles and fields is used
Each computing node acts as a section of the global physical space

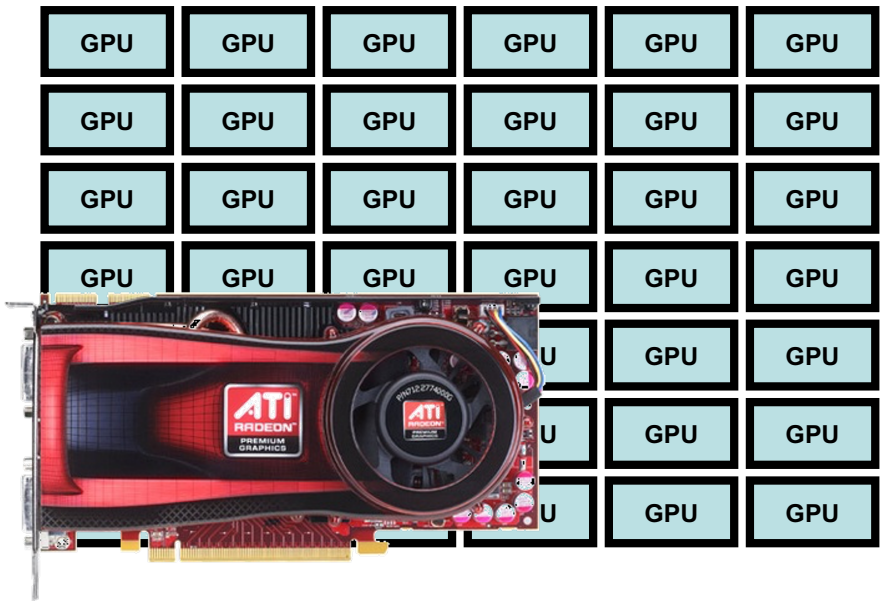
Hybrid parallelism allows for better taking advantage of HPC resources

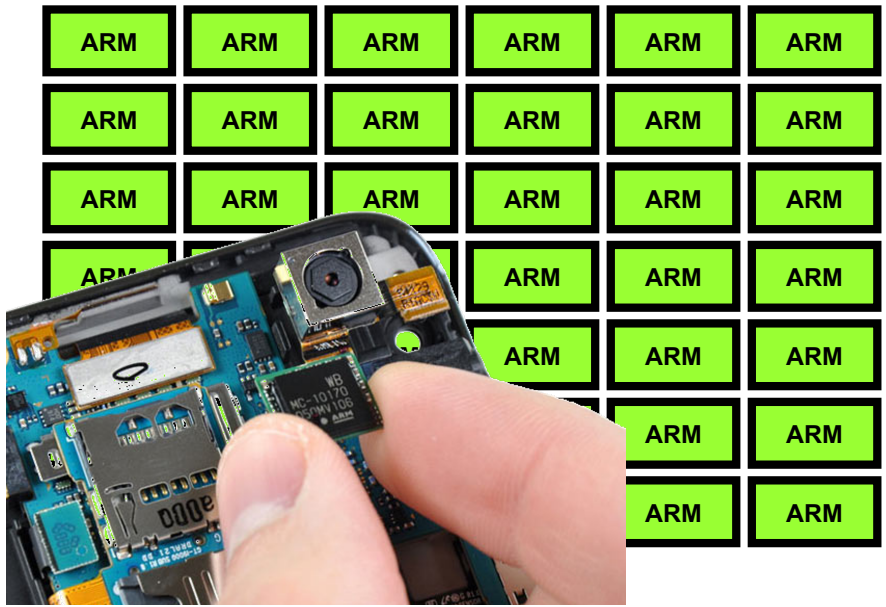




M. Grech, The Particle-In-Cell (PIC) simulation of plasmas, Les Houches (2019)

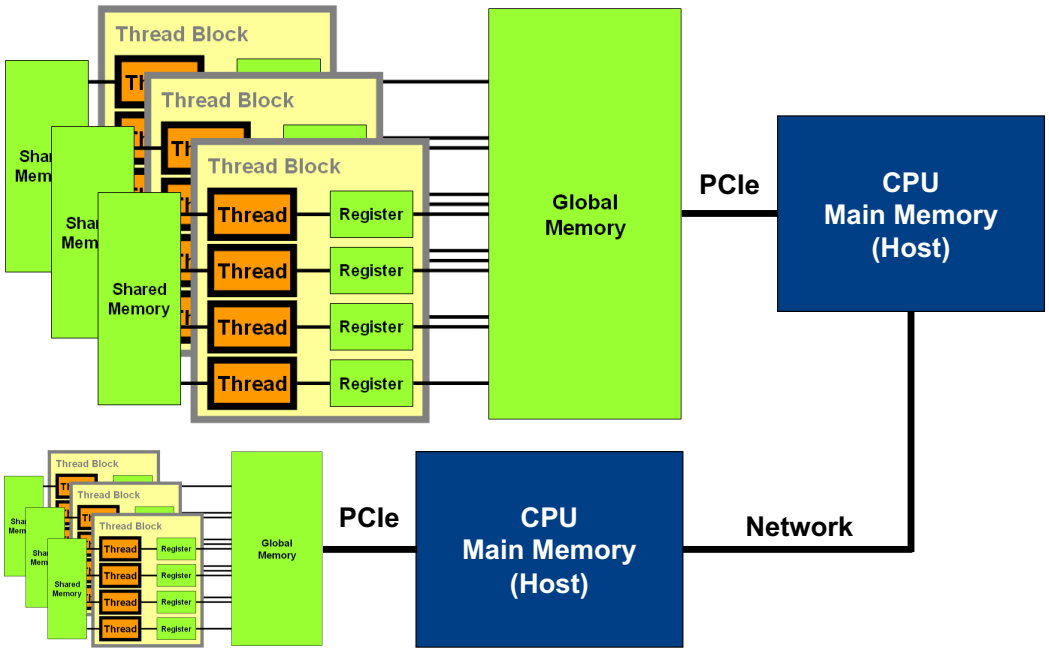
Towards the exascale





M. Bussmann et al., Frames per Second Laser Plasma Simulations, Austin (2012)

Memory management becomes even more critical: it is important to keep data locality



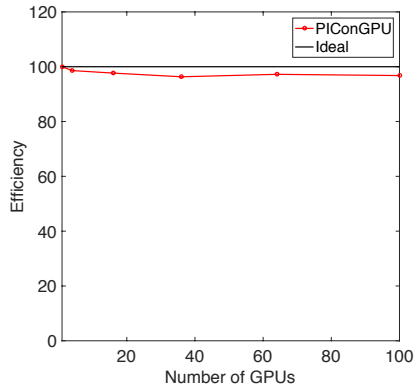
Structure of arrays vs **array of structures**

M. Bussmann et al., Frames per Second Laser Plasma Simulations, Austin (2012)

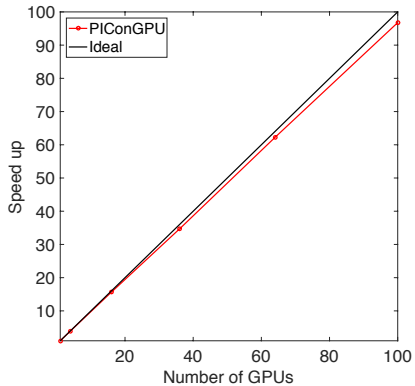
Testing the performances: strong scaling

Scaling of the code PIConGPU on the supercomputer D.A.V.I.D.E. @CINECA (Italy)

Grid size: 1200x1200
Super cell size: 16x16
Time steps: 1000
Particles/cell: 8x8
Interpolation: linear



Efficiency @10x10 GPUs: 96.77%

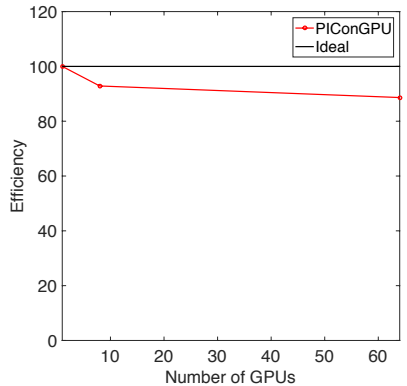


Speed up @10x10 GPUs: 96.77

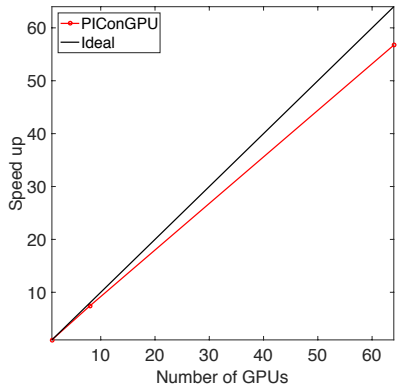
Testing the performances: weak scaling

Scaling of the code PIConGPU on the supercomputer D.A.V.I.D.E. @CINECA (Italy)

Grid size: 64x64x64 —> 320x320x320
Super cell size: 8x8x4
Time steps: 1000
Particles/cell: 4x2x2
Interpolation: quadratic

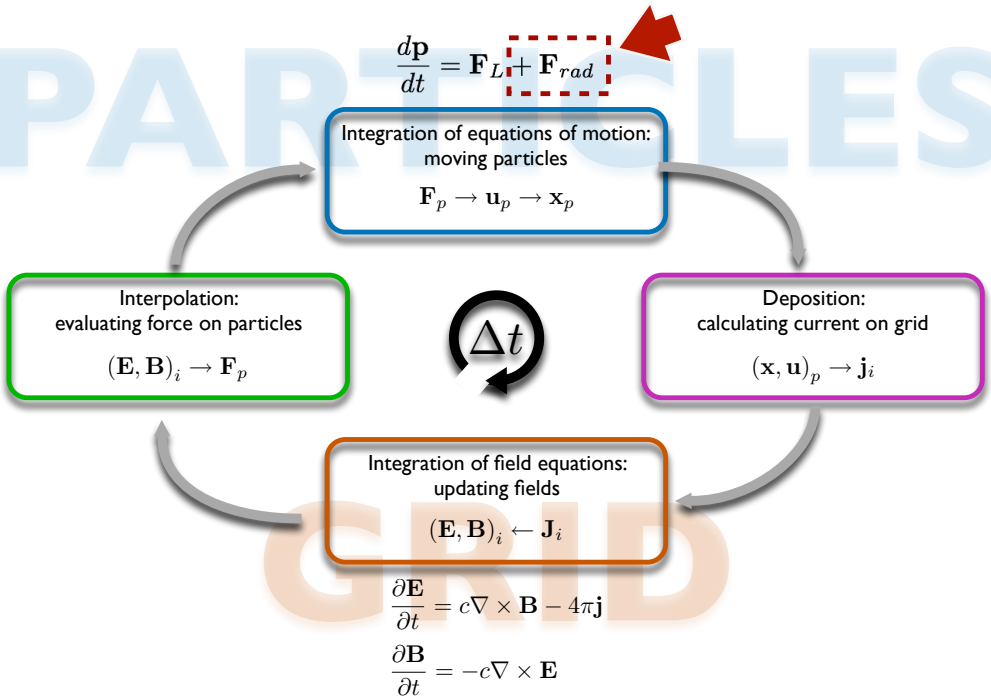


Efficiency @4x4x4 GPUs: 88.64%



Speed up @4x4x4 GPUs: 56.73

Beyond the classical PIC algorithm: modelling classical and quantum radiation reaction



Beyond the classical PIC algorithm: modelling classical and quantum radiation reaction

Different types of Radiation reaction models

$$\frac{d\mathbf{p}}{dt} = \mathbf{F}_L + \begin{cases} \mathbf{F}_{rad} & \text{Continuous damping rate} \\ \frac{d^2P}{dtd\chi_\gamma} & \text{QED probabilistic approach} \end{cases}$$

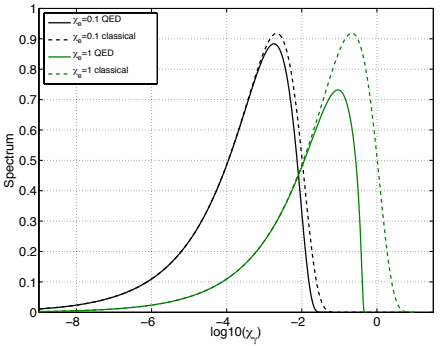
Implementation in PIC codes

- Continuous damping rate: particle pusher with $\mathbf{F}_{rad}$

- QED probabilistic approach: particle pusher + Monte Carlo module
  every Δt : probability of photon emission
  Select a photon in QED synchrotron spectrum
  Update particle momentum due to quantum recoil

- The QED approach can be generalized to any external EM fields under the conditions:
  - quasi-static fields $t_{carac}(\vec{E},\vec{B}) \gg t_{coh} \implies \gamma \gg 1$
  - weak fields $\chi_e^2 \gg \mathrm{Max}(f,g) \quad (f,g) \ll 1$

Synchrotron Spectrum



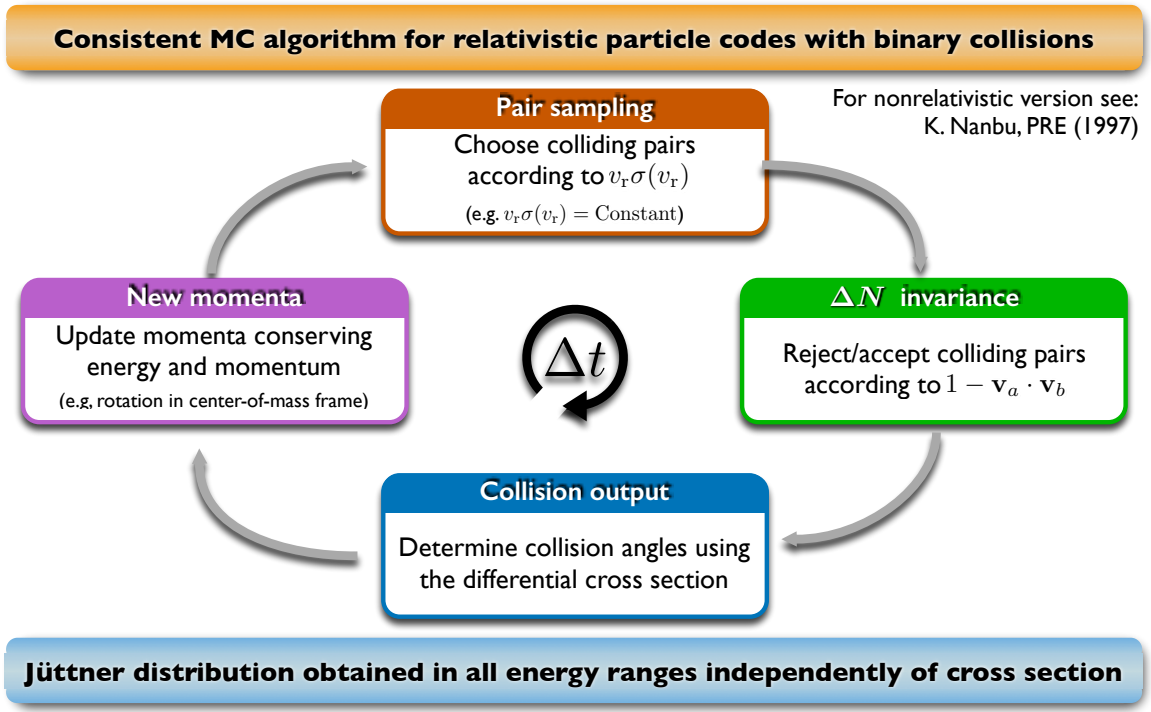
$$f = F_{\mu\nu}^2 / E_{crit}^2$$

$$g = F_{\mu\nu}^* F_{\mu\nu} / E_{crit}^2$$

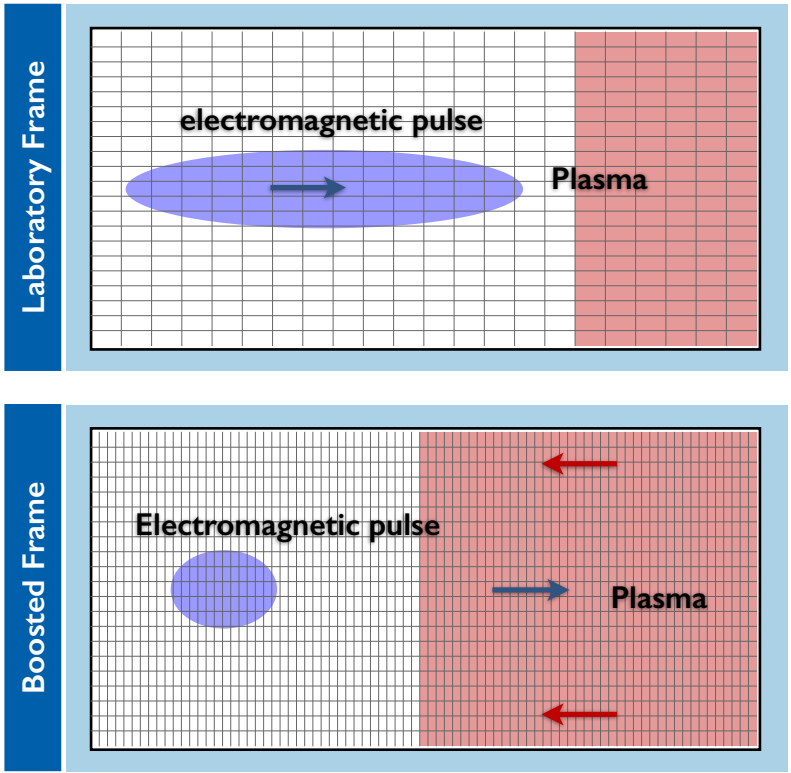
$$E_{crit} = m^2c^3/e\hbar$$

$$\chi_{e,\gamma} = \frac{|F_{\mu\nu}p_{e,\gamma}^\nu|}{E_{crit}mc}$$

Beyond the classical PIC algorithm: modelling Coulomb collision



Beyond the classical PIC algorithm: modelling LWFA in a boosted frame



Hands-on: electromagnetic PIC

Download the zPIC code from the GitHub repo
git clone https://github.com/ricardo-fonseca/zpic.git
or download from https://github.com/ricardo-fonseca/zpic

Go to the folder python and compile the code using make

Launch the Jupyter notebook from the source folder:
> jupyter notebook LWFA1D.ipynb

We are going to model plasma-based accelerators

Hands-on: electromagnetic PIC

Choose the normalisation

Plasma sets reference

Plasma density is unity

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

Example

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

Laser sets reference

Reference laser frequency is unity

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

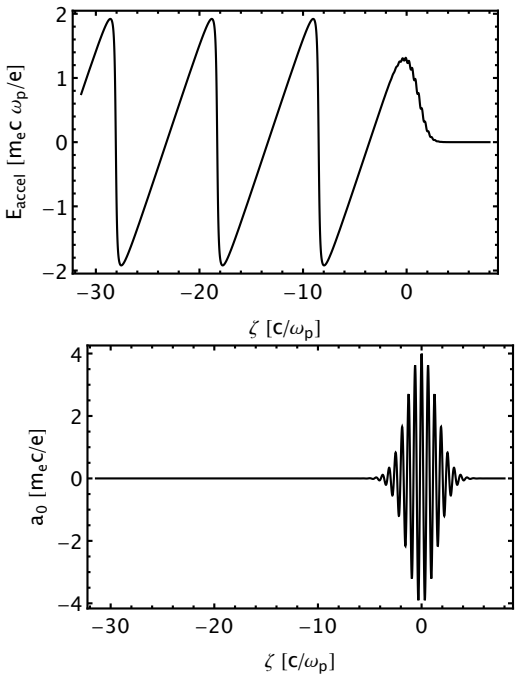
Example

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

Hands-on: electromagnetic PIC

LWFA simulation

- 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
  $\lambda_0 \sim 1\ \mu m \sim 0.18\ (n_p\ [10^{18} cm^{-3}])^{1/2}\ c/\omega_p$

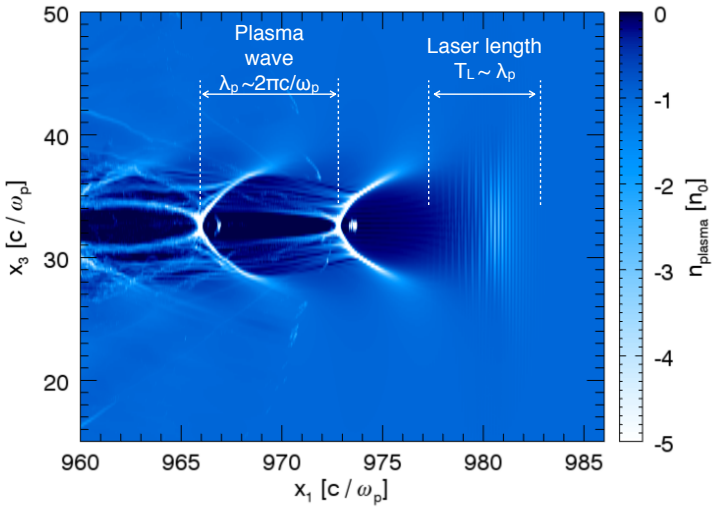


Longitudinal spatial Resolution: $\Delta x \sim \lambda_0/\# \sim 0.18/\#\ (n_p\ [10^{18} cm^{-3}])^{1/2}\ c/\omega_p$
> 20-30 (number of cells per laser wavelength)

Hands-on: electromagnetic PIC

Determine simulation box size

- 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 λp ~ 25 c/ωp
  Transversely (2D)
  - Laser pulse waist / transverse bubble size
  - > 4 λp ~ 25 c/ωp



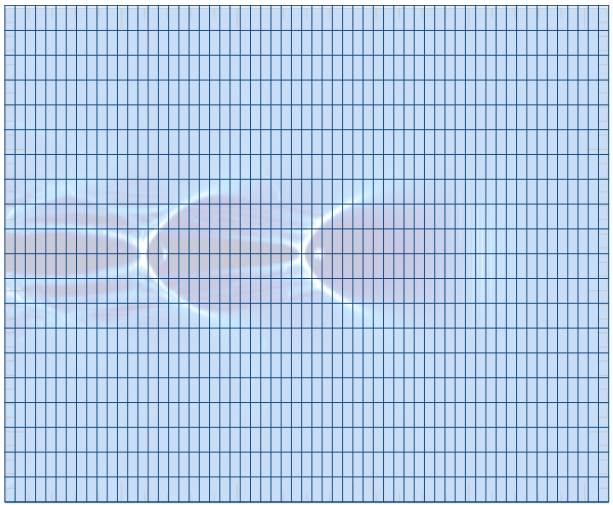
Hands-on: electromagnetic PIC

Set 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 ~ $L / \Delta x \sim 2800$ cells

Choose simulation particles

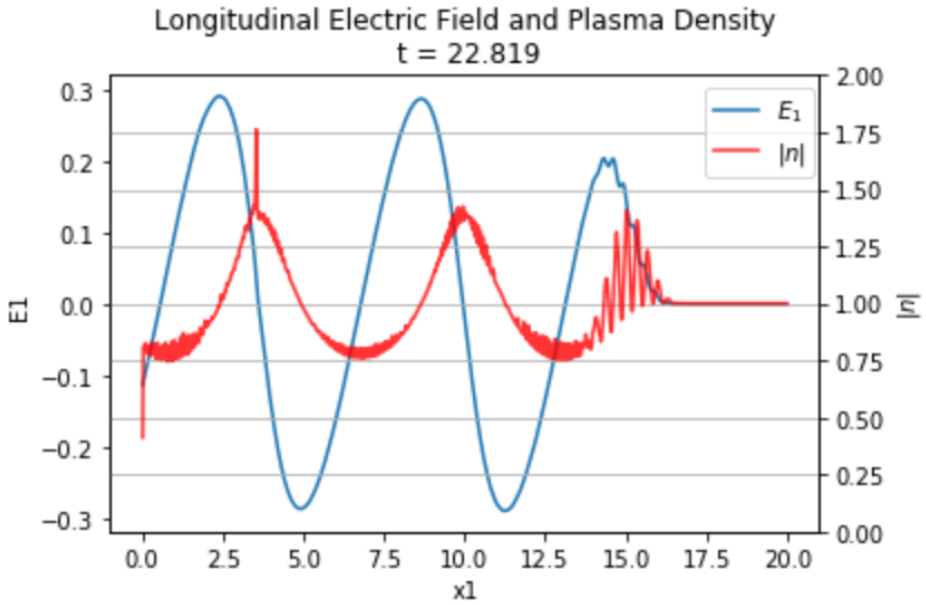
- Number of particles per cell must resolve local phasespace
- $\gg 1$ in 1D (e.g. 64)
- ~ 10 in 2D
- Higher numbers improve phasespace resolution (detailed distribution tails) and reduces simulation noise



Hands-on: electromagnetic PIC

Choose diagnostic

- 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 u1/x1
  - Wave structure and particle acceleration



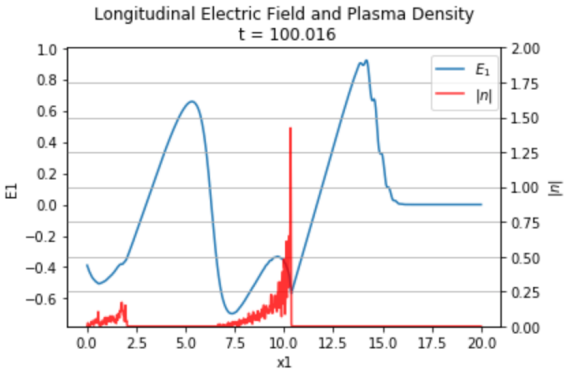
Hands-on: electromagnetic PIC

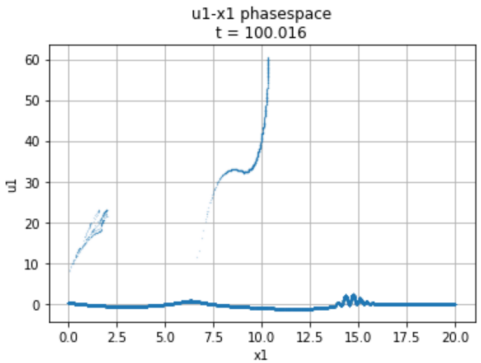
Simulate a laser wakefield accelerator:

- Add an ultra-intense laser beam as a driver (a_0~2)
- 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!





Hands-on: electromagnetic PIC

Laser Wakefield Accelerator

1D simulation of a laser wakefield accelerator.

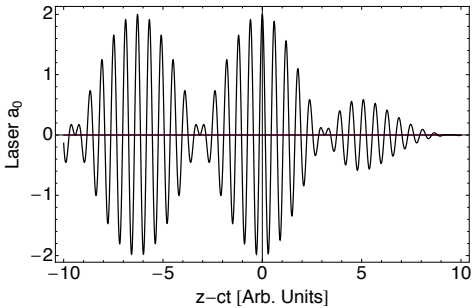
In [1]:

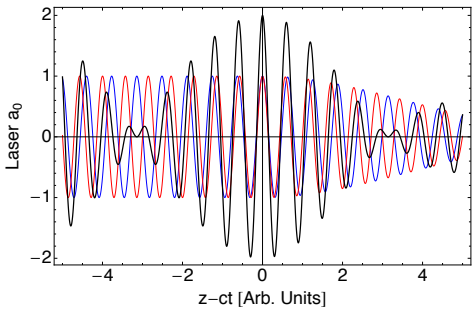
```
import emld
import numpy

# 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 = emld.Species( "electrons", -1.0, ppc,
                          density = emld.Density( type = "step", start = 20.0))

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

# Add laser pulse
sim.add_laser( emld.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( emld.Smooth(xtype = "compensated", xlevel = 4) )

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

Hands-on: electromagnetic PIC

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?

Hands-on: electromagnetic PIC

Laser Plasma Beatwave Accelerator

1D simulation of a laser plasma beatwave accelerator.

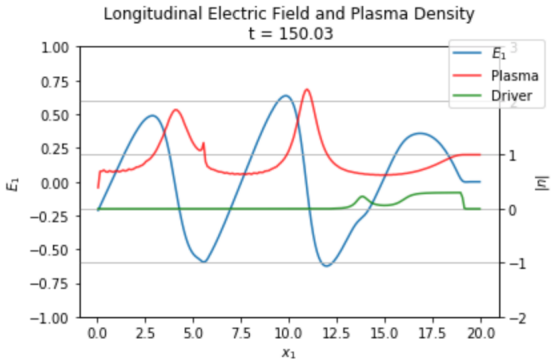
In [1]:
```
import emld
import numpy

# Time step
dt = 0.019
# Simulation time
tmax = 100.
# Number of cells
nx  = 1250
# Simulation box size
box = 50.0
# Particles per cell
ppc = 128

# Use a step density profile
electrons = emld.Species( "electrons", -1.0, ppc,
                          density = emld.Density( type = "step", start = 50.0))

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

# Add laser pulses
sim.add_laser( emld.Laser( start = 50.0, rise = 20.0 , flat = 40.0 ,  fall = 1.0 , a0 = 0.5, omega0 = 10.0, polarizatio
sim.add_laser( emld.Laser( start = 50.0, rise = 20.0 , flat = 40.0 , fall = 1.0, a0 = 0.5, omega0 = 11.0, polarization

# Set moving window
sim.set_moving_window()
# Set current smoothing
sim.set_smooth( emld.Smooth(xtype = "compensated", xlevel = 4) )

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

```
Running simulation up to t = 100 ...
n = 5264, t = 100.016
Done.
```

Hands-on: electromagnetic PIC

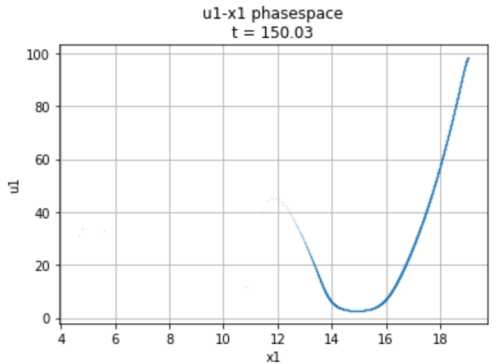
Simulate a plasma wakefield accelerator:

- Use an ultra-relativistic particle beam as a driver
  - e.g. u_{fl} = 100, length ~ 10 c/ω_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?



Hands-on: electromagnetic PIC

Plasma Wakefield Accelerator

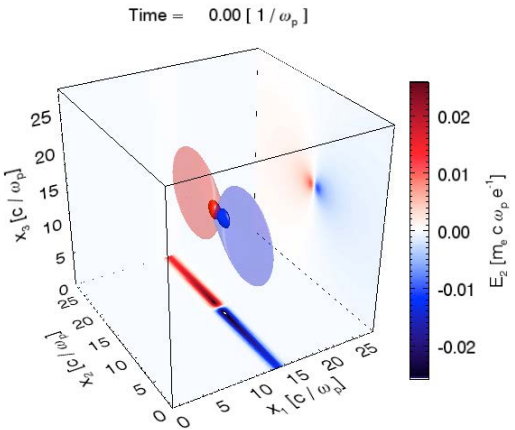
1D simulation of a plasma wakefield accelerator.

In [63]:

```
import emld
import numpy

# Time step
dt = 0.09
# Simulation time
tmax = 150.
# Number of cells
nx  = 200
# Simulation box size
box = 20.0

# Particles per cell
ppc = 128

# Use a step density profile
electrons = emld.Species( "electrons", -1.0, ppc,
                          density = emld.Density( type = "step", start = 100.0))

driver = emld.Species( "driver", -1.0, ppc,
                          density = emld.Density( n = 0.3 , type = "slab", start = 8.0 , end = 19.0 ),
                          ufl=[100.0,0.0,0.0])

# Initialize simulation
sim = emld.Simulation( nx, box, dt, species = [electrons,driver] )

# Set moving window
sim.set_moving_window()

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

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

```
Running simulation up to t = 150 ...
n = 1667, t = 150.03
Done.
```

A beam means violating charge and current neutrality: it requires computing the initial electromagnetic fields



Fields are calculated in the beam reference frame

Deposit beam charge in simulation frame

Boost to beam frame

Calculate electric field (no magnetic field in this frame)

Boost back to simulation frame, calculating magnetic field

Electric field calculations

Are performed using Coulomb's law • This assumes open boundaries

In the beam frame each charge cell becomes elongated along propagation direction

Assume each cell is an infinite slab (2D) or rod (3D) • Only transverse fields!