

Particle-In-Cell (PIC) simulation of Laser-Plasma Interaction

Frédéric Perez

LULI laboratory, CNRS, Ecole Polytechnique, France

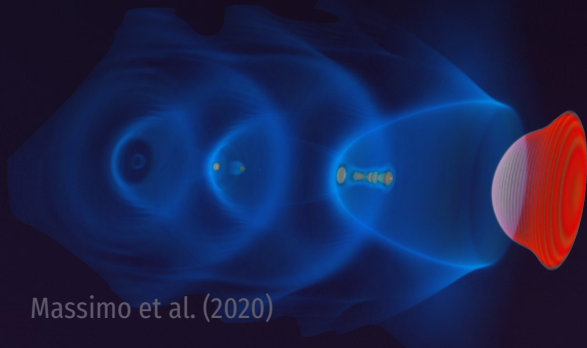


supported by Project ELI-RO/DFG/2025_013 IATP-NP 2.0
funded by the Institute of Atomic Physics, Romania

- Kinetic plasma physics
- PIC algorithms
- Limits of PIC
- Adding more physics
- High-performance computing
- Practical aspects

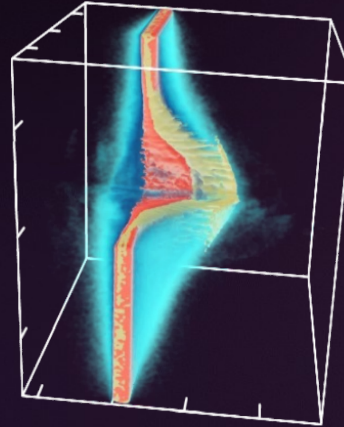
Electromagnetic PIC codes, central to laser-plasma interaction

Electron laser wakefield acceleration

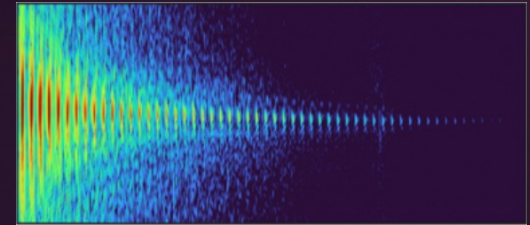


Proton acceleration

Shou et al.,
Nat. Comm.
16 (2025)

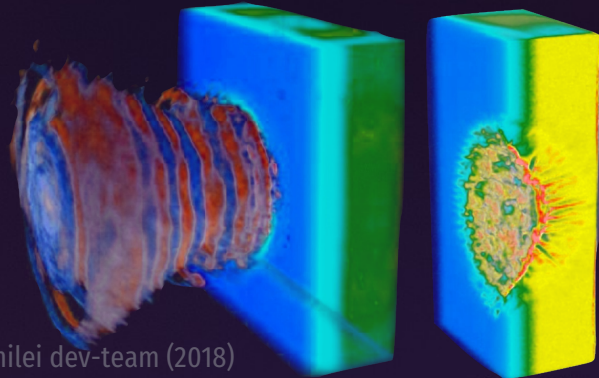


High-harmonic generation

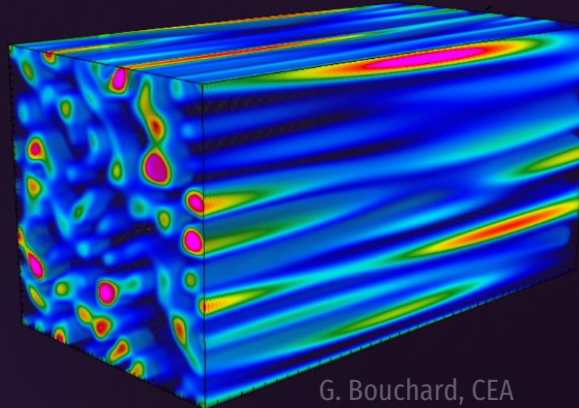


Kim et al., PRR 7 (2025)

Ultra-high intensity laser-solid interaction

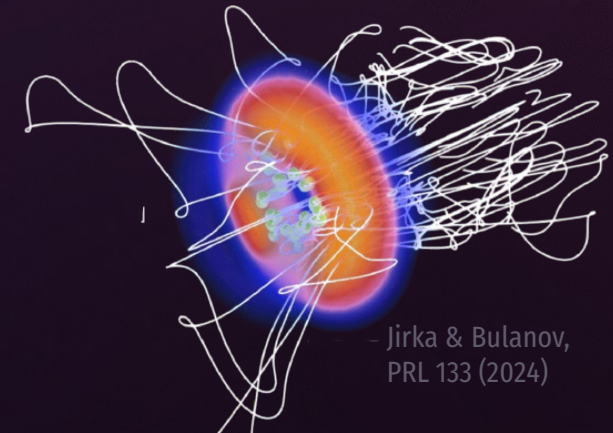


Parametric instabilities



G. Bouchard, CEA

e^+e^- pair production

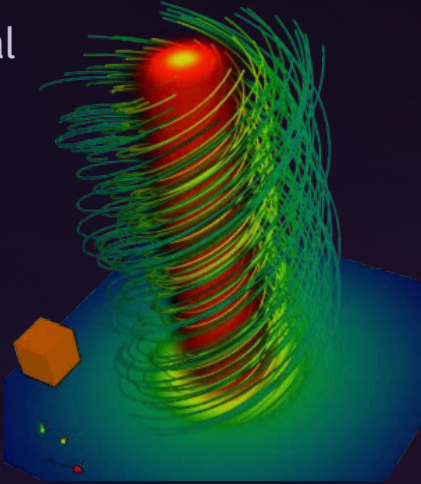


Jirka & Bulanov,
PRL 133 (2024)

... and in astrophysics

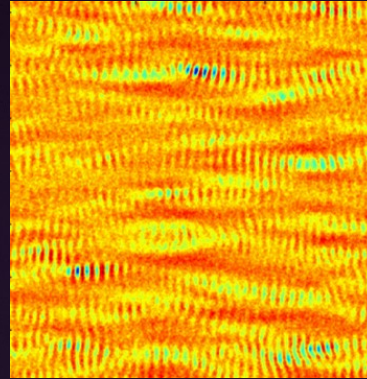
Fundamental plasma structures

Yoon et al.,
Comm. Phys. 7
(2024)



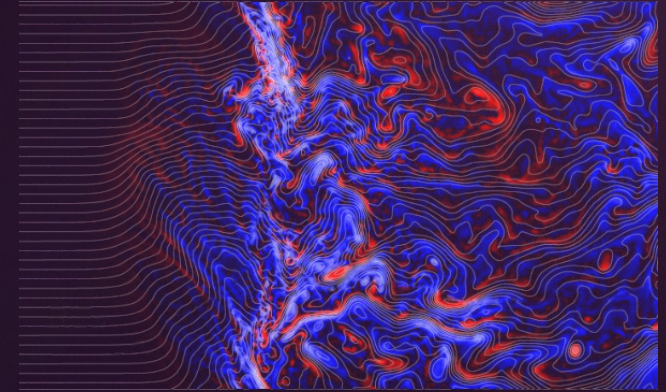
Wave couplings in the solar wind

Krafft & Savoini
ApJL 964 (2024)



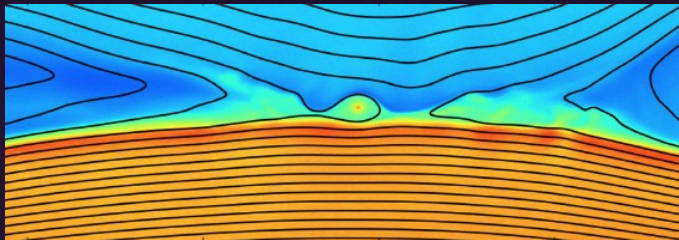
Shocks

Guo et al., ApJ 955 (2023)



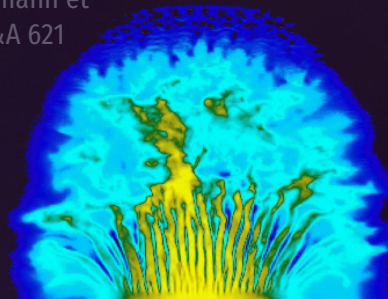
Magnetic reconnection

Dargent et al., GRL 47 (2020)



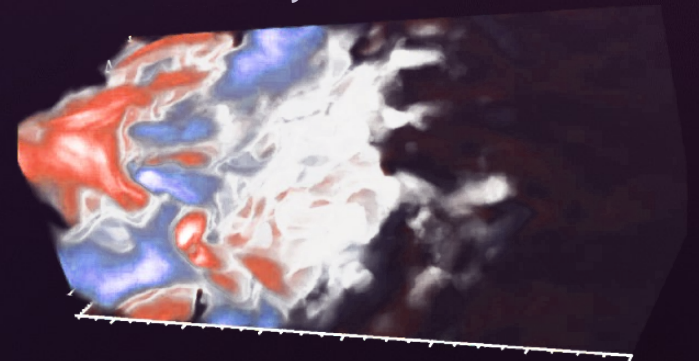
e⁺e⁻ jets

Dieckmann et
al., A&A 621
(2019)



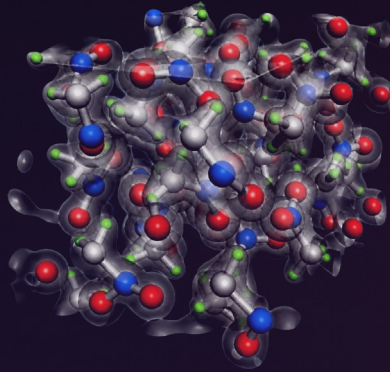
Weibel instability

Matsumoto et al., PRL119 (2017)

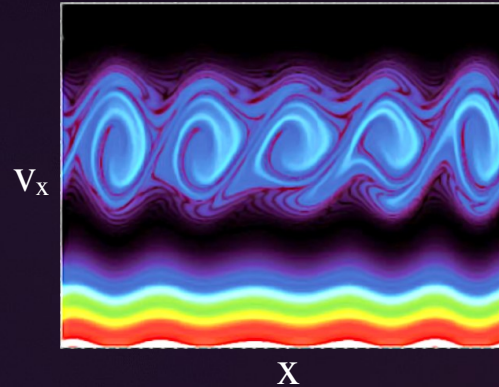


PIC codes are a type of *kinetic* code

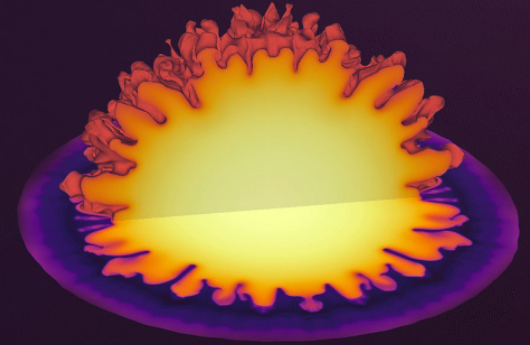
Molecular dynamics → Kinetic → hydrodynamics



Atoms, orbitals



distribution function



$\rho, \vec{v}, T$

“Kinetic” plasma physics $\approx$ use smooth distribution functions

The kinetic description of plasma

particles $\left\{ \begin{array}{l} f_s(t, \mathbf{x}, \mathbf{p}) = \text{distribution function} \\ \text{density} \quad \rho(t, \mathbf{x}) = \int d\mathbf{p} f_s(t, \mathbf{x}, \mathbf{p}) \\ \text{current density} \quad \mathbf{J}(t, \mathbf{x}) = q_s \int d\mathbf{p} \frac{\mathbf{p}}{m_s \gamma} f_s(t, \mathbf{x}, \mathbf{p}) \end{array} \right.$

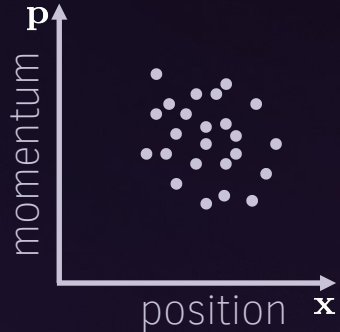
$\uparrow$
species

fields $\left\{ \begin{array}{l} \mathbf{E}(t, \mathbf{x}) \quad \text{Electric field} \\ \mathbf{B}(t, \mathbf{x}) \quad \text{Magnetic field} \end{array} \right.$

$\rightarrow$ force $\mathbf{F}_L = q_s \left(\mathbf{E} + \frac{\mathbf{p}}{m_s \gamma} \times \mathbf{B} \right)$

What equation governs the distribution function?

Klimontovich's exact picture



The exact state of the system is known at a time t_0

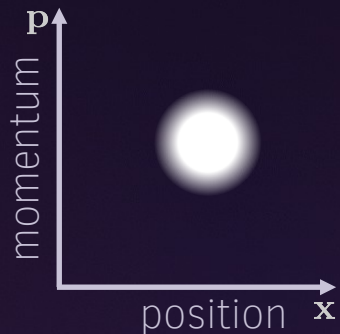
$$D_s(t_0, \mathbf{x}, \mathbf{p}) = \sum_p \delta(\mathbf{x} - \mathbf{x}_p(t_0)) \delta(\mathbf{p} - \mathbf{p}_p(t_0))$$

The evolution of the system at later times is known exactly

$$\partial_t D_s + \mathbf{p} \cdot \nabla D_s + q_s (\mathbf{E}_{\text{tot}} + \mathbf{v} \times \mathbf{B}_{\text{tot}}) \cdot \nabla_p D_s = 0.$$

Klimontovich's equation

Ensemble average : towards the Vlasov equation



total	smooth, average	microscopic fluctuations
$D_s(t, \mathbf{x}, \mathbf{p})$	$f_s(t, \mathbf{x}, \mathbf{p})$	$+ \delta f_s(t, \mathbf{x}, \mathbf{p})$
$\mathbf{E}_{\text{tot}}(t, \mathbf{x})$	$\mathbf{E}(t, \mathbf{x})$	$+ \delta \mathbf{E}(t, \mathbf{x})$
$\mathbf{B}_{\text{tot}}(t, \mathbf{x})$	$\mathbf{B}(t, \mathbf{x})$	$+ \delta \mathbf{B}(t, \mathbf{x})$

Plug this in Klimontovich's equation + ensemble average:

$$\partial_t f_s + \mathbf{p} \cdot \nabla f_s + q_s (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_p f_s = -q_s \langle (\delta \mathbf{E} + \mathbf{v} \times \delta \mathbf{B}) \delta f_s \rangle.$$

collective behavior

microscopic/collisions

Vlasov's equation

$$\partial_t f_s + \mathbf{p} \cdot \nabla f_s + q_s (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_p f_s = -q_s \langle (\delta \mathbf{E} + \mathbf{v} \times \delta \mathbf{B}) \delta f_s \rangle .$$

Average force

$$\partial_t f_s + \mathbf{v} \cdot \nabla f_s + \mathbf{F} \cdot \nabla_p f_s = 0$$

Vlasov's equation
(fluctuations neglected)

Distribution function



We have assumed a weakly coupled plasma !

Add Maxwell's equations

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}$$

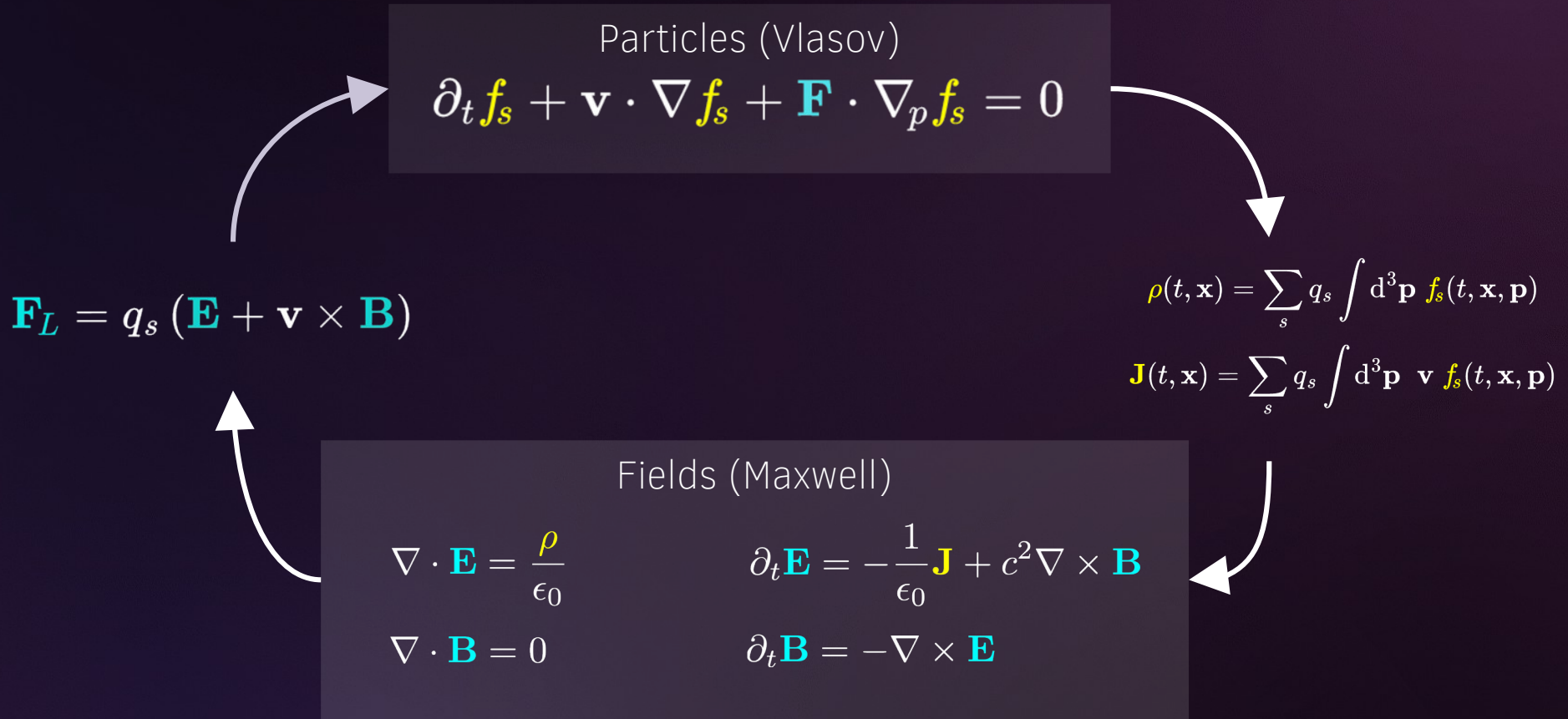
$$\nabla \cdot \mathbf{B} = 0$$

$$\partial_t \mathbf{E} = -\frac{1}{\epsilon_0} \mathbf{J} + c^2 \nabla \times \mathbf{B}$$

$$\partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$

How to combine with Vlasov?

The Maxwell-Vlasov system



There is a natural set of units

Velocity	c
Charge	e
Mass	m_e
Momentum	$m_e c$
Energy, Temperature	$m_e c^2$
Time	ω_r^{-1}
Length	c/ω_r
Number density	$n_r = \epsilon_0 m_e \omega_r^2 / e^2$
Current density	$e c n_r$
Pressure	$m_e c^2 n_r$
Electric field	$m_e c \omega_r / e$
Magnetic field	$m_e \omega_r / e$
Poynting flux	$m_e c^3 n_r / 2$

ω_r is the *reference angular frequency*.
The code does not need to know its value.
Results of the simulation can be scaled
a posteriori.

Maxwell's equations

$$\nabla \cdot \mathbf{E} = \rho \quad \partial_t \mathbf{E} = -\mathbf{J} + \nabla \times \mathbf{B}$$

$$\nabla \cdot \mathbf{B} = 0 \quad \partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$



The units of density is not
equal to $(c/\omega_r)^{-3}$

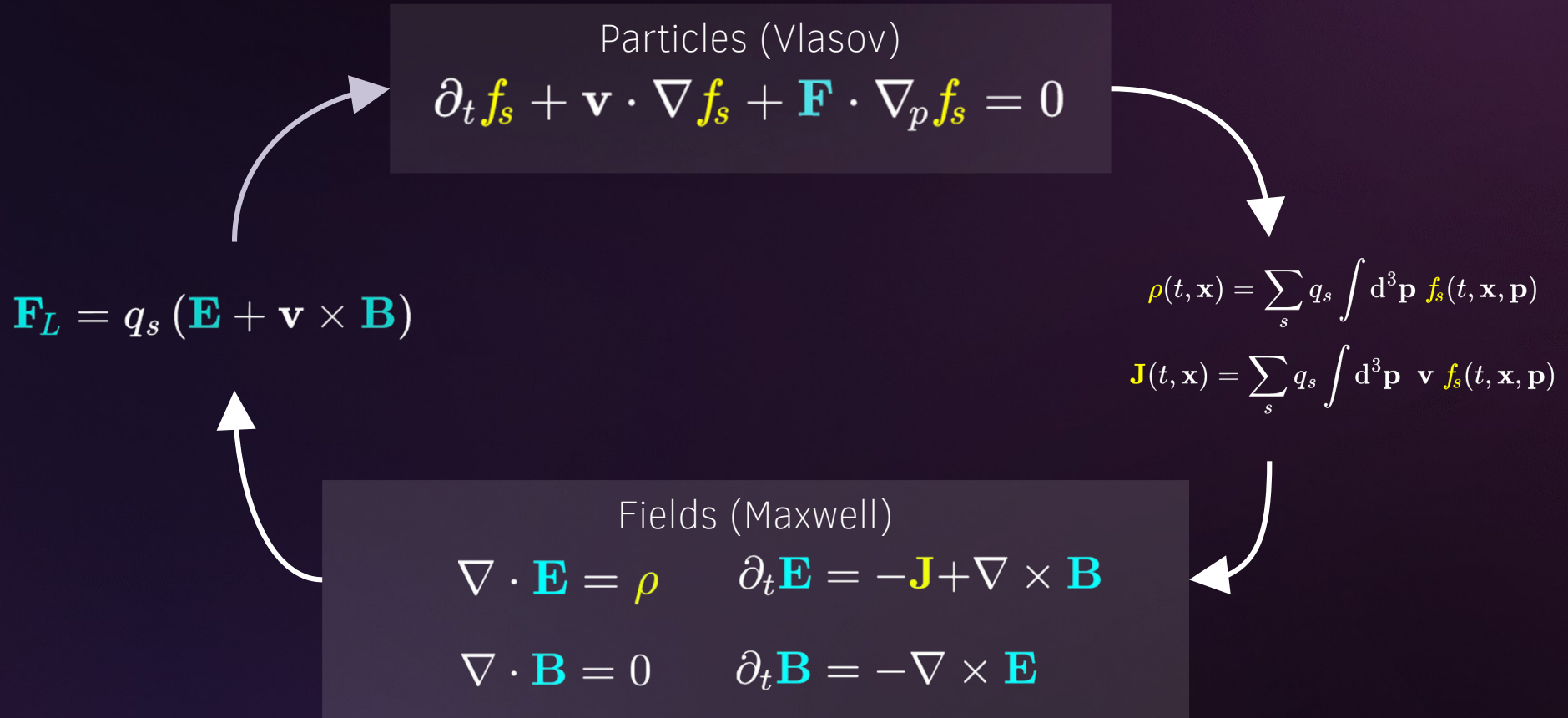
How to choose ω_r ?

ω_r is generally an important frequency of the problem

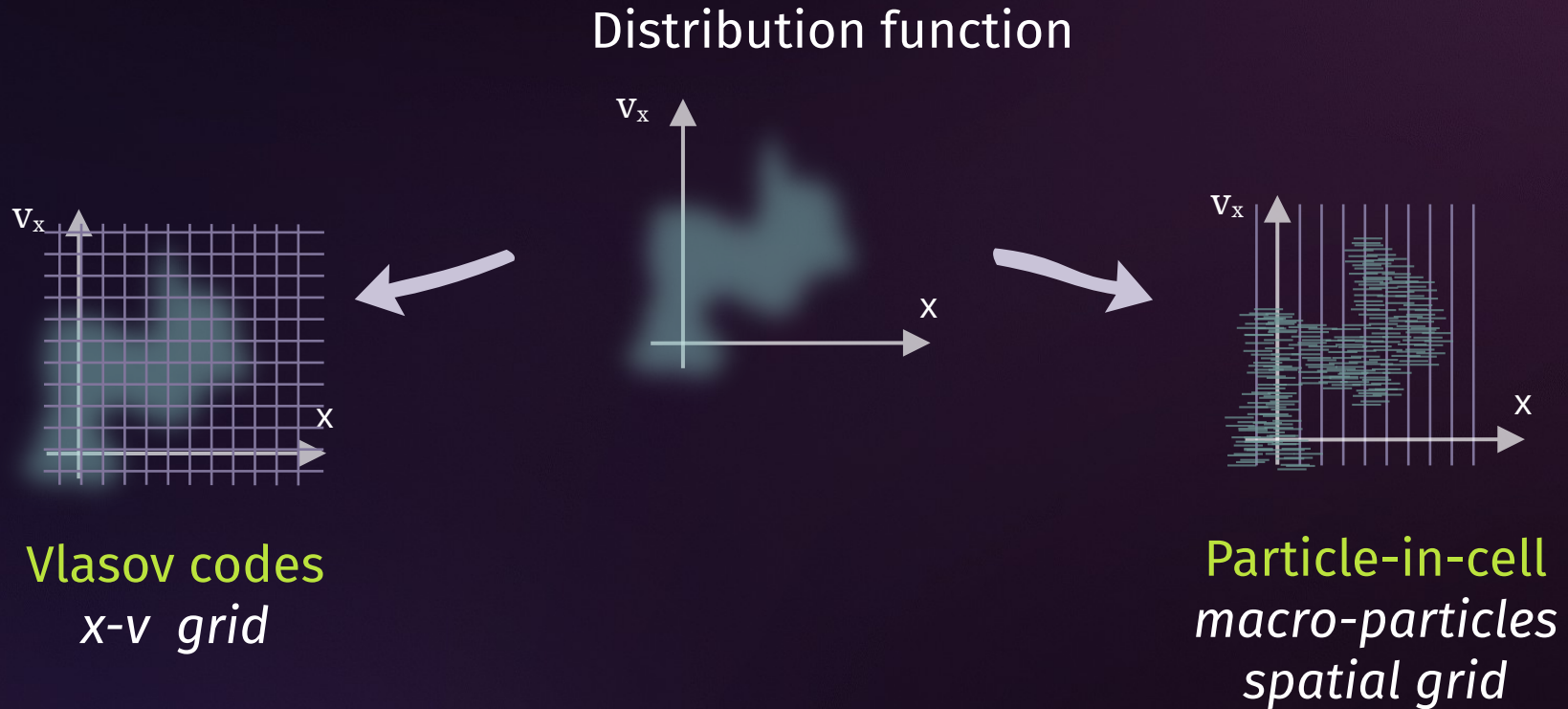
- laser frequency (in this case, $n_r = n_c$)
- plasma frequency (in this case, $n_r = n$)
- cyclotron frequency (in this case, $B_r = B/\gamma$)

Again, the code does not need to know its value
(unless you need collisions, ionisation, ...)

The Maxwell-Vlasov system



How to solve Maxwell-Vlasov on a computer?



How to solve Maxwell on a grid?

$$\nabla \cdot \mathbf{E} = \rho \quad \partial_t \mathbf{E} = -\mathbf{J} + \nabla \times \mathbf{B}$$

$$\nabla \cdot \mathbf{B} = 0 \quad \partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$

We only need 2 Maxwell equations

Maxwell-Ampere

$$\partial_t \mathbf{E} = \nabla \times \mathbf{B} - \mathbf{J}$$



divergence

$$\partial_t \nabla \cdot \mathbf{E} + \nabla \cdot \mathbf{J} = 0$$



Conservation of charge
 $\partial_t \rho + \nabla \cdot \mathbf{J} = 0$

$$\partial_t (\nabla \cdot \mathbf{E} - \rho) = 0$$



Maxwell-Poisson is conserved

Maxwell-Faraday

$$\partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$



divergence

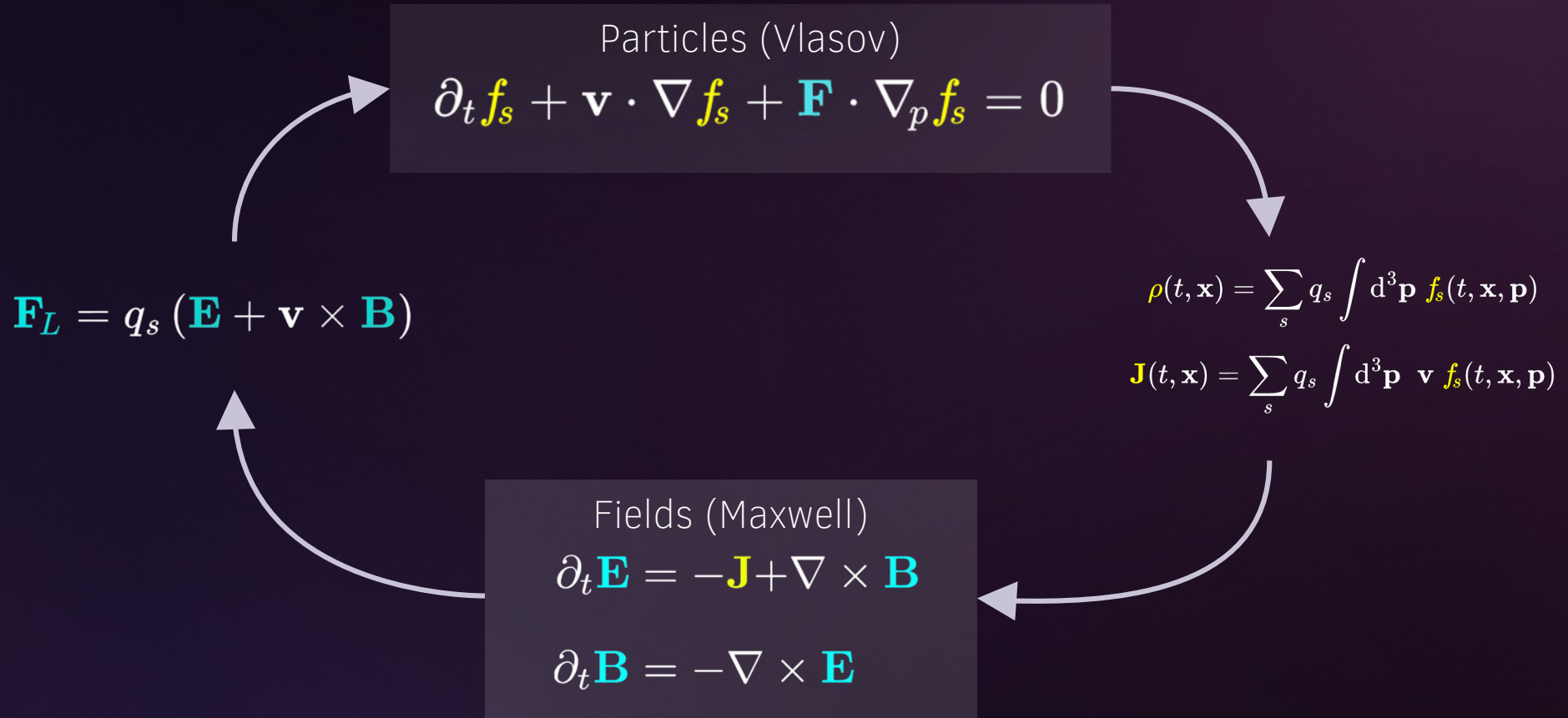
$$\partial_t \nabla \cdot \mathbf{B} = 0$$



Maxwell-Gauss is conserved

We do not need to solve Maxwell-Poisson and Maxwell-Gauss,
(provided they are satisfied initially)

The Maxwell-Vlasov system

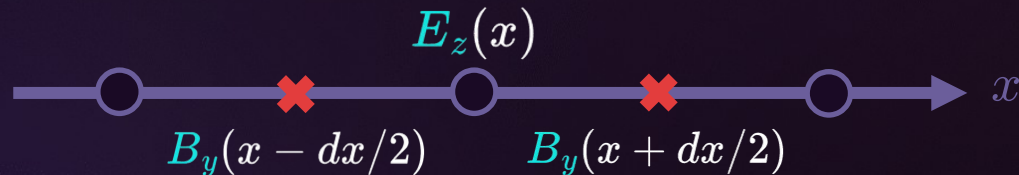


Fields are defined on a grid

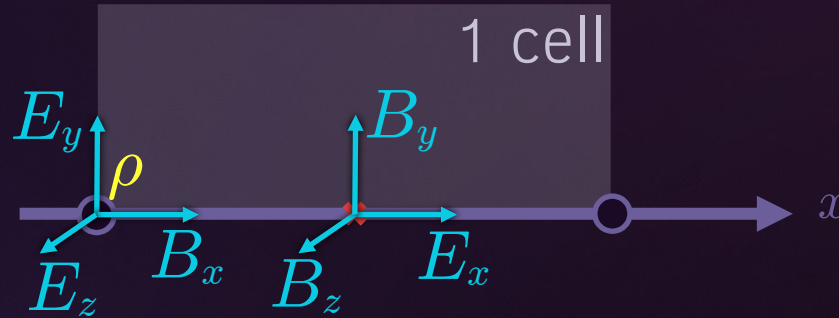
There are several ways to solve Maxwell on a grid.
Let us illustrate with the most common technique
“Finite Difference Time Domain” (FDTD)

Maxwell-Ampere in 1D:

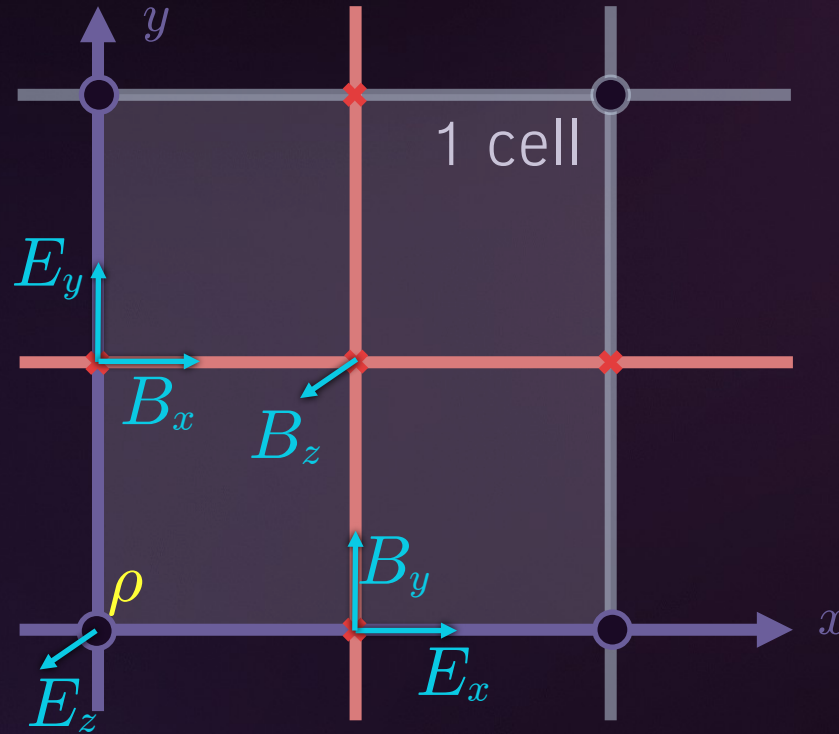
$$\partial_t \mathbf{E}_z = \partial_x \mathbf{B}_y - \mathbf{J}_z \quad \longrightarrow \quad (\partial_t \mathbf{E}_z)(x) = (\partial_x \mathbf{B}_y)(x) - \mathbf{J}_z(x)$$
$$\longrightarrow \quad \partial_t \mathbf{E}_z(x) = \frac{\underbrace{\mathbf{B}_y(x + \Delta x/2) - \mathbf{B}_y(x - \Delta x/2)}_{\text{offset in space}}}{\Delta x} - \mathbf{J}_z(x)$$



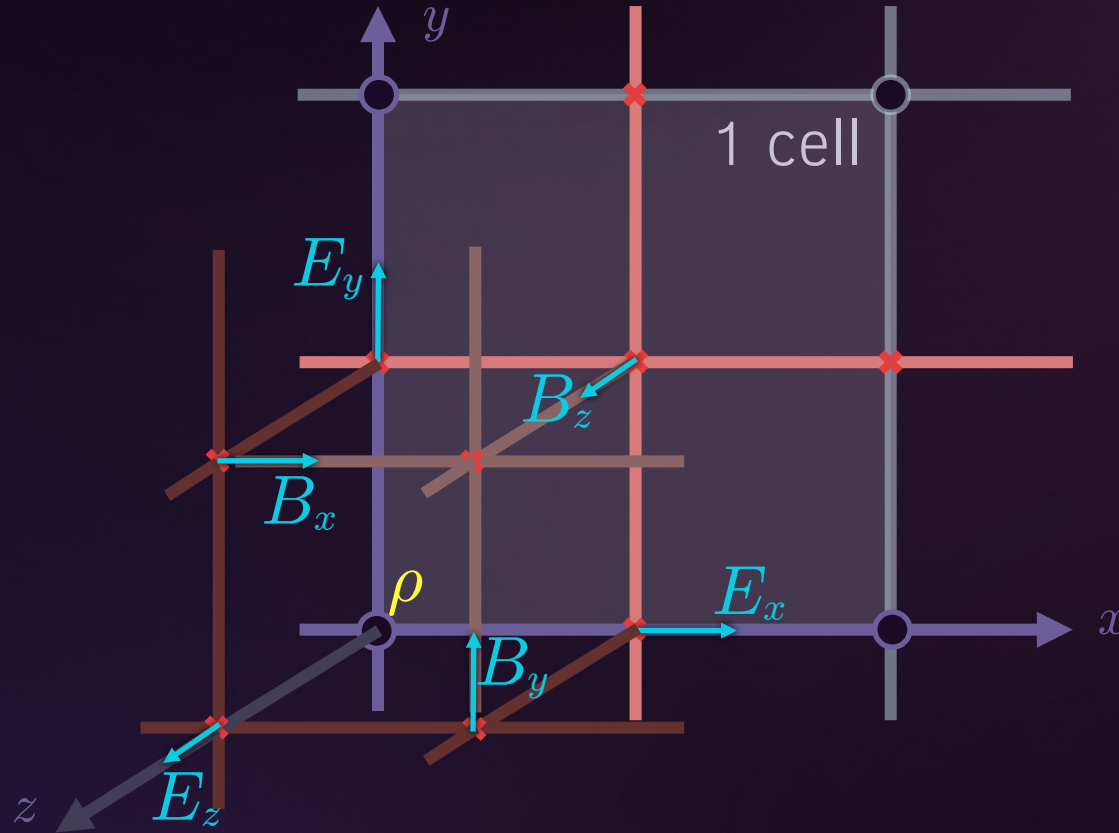
The grid is *staggered* (Yee's scheme)



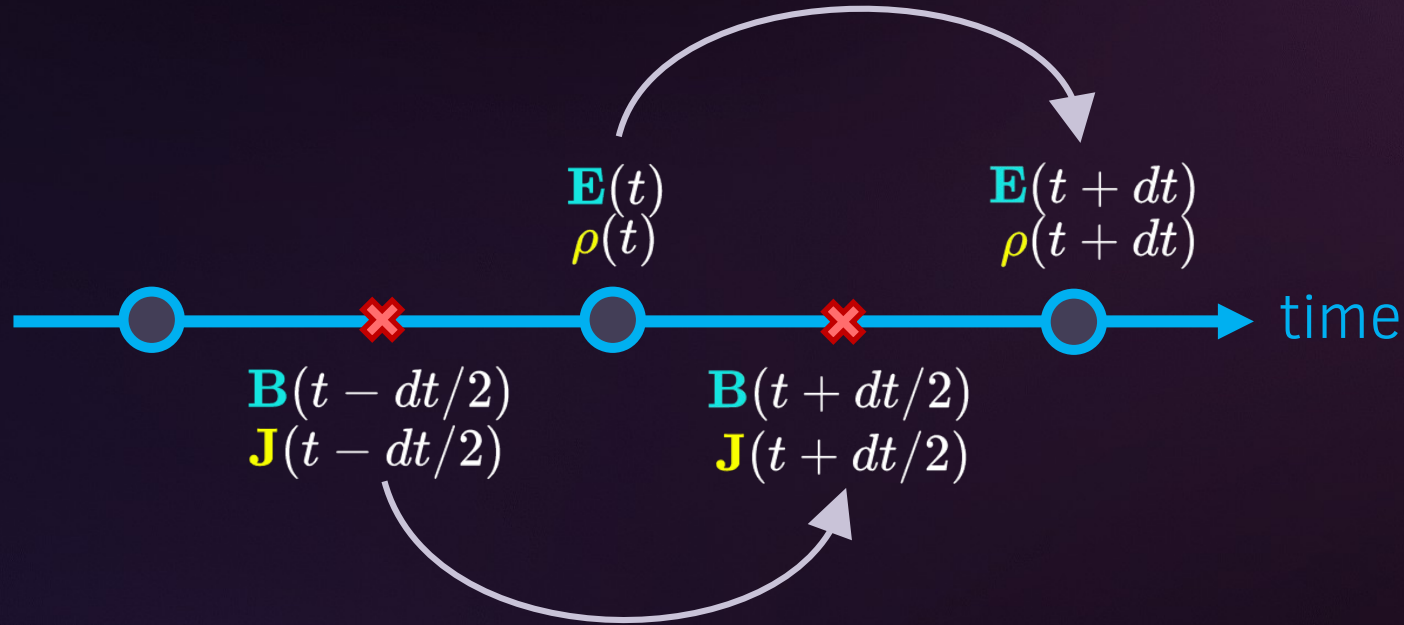
The grid is *staggered* (Yee's scheme)



The grid is *staggered* (Yee's scheme)

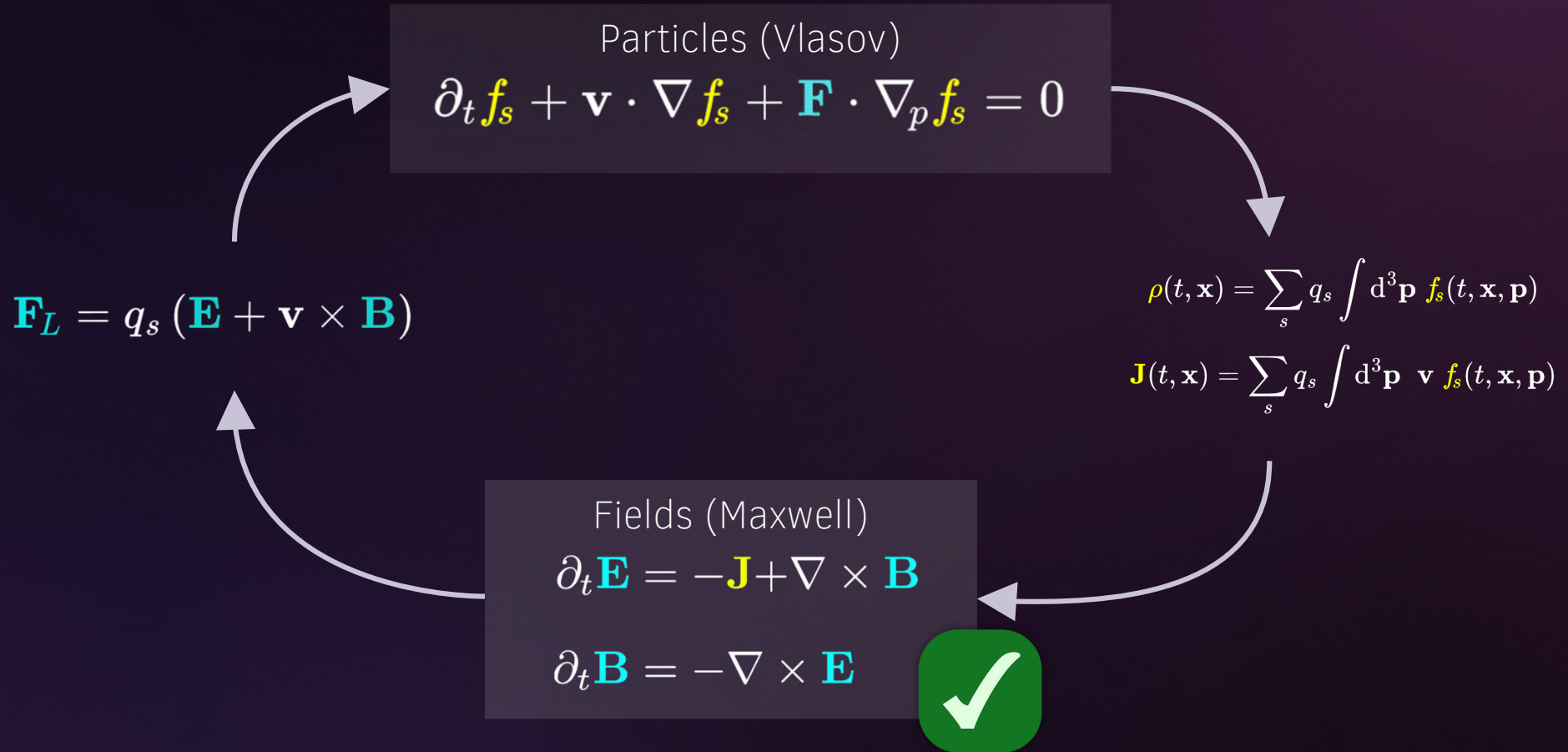


The scheme is also staggered in time!



“Leap-frog” scheme

The Maxwell-Vlasov system

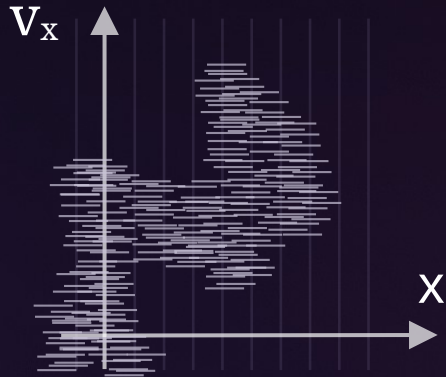


How to solve Vlasov with macro-particles?

$$\partial_t f_s + \mathbf{v} \cdot \nabla f_s + \mathbf{F} \cdot \nabla_p f_s = 0$$

The distribution function is split into macro-particles

One macro-particle = w_p $\mathbf{x}_p(t)$ $\mathbf{p}_p(t)$



$$f_s(t, \mathbf{x}, \mathbf{p}) = \sum_{p=1}^N w_p S(\mathbf{x} - \mathbf{x}_p(t)) \delta(\mathbf{p} - \mathbf{p}_p(t))$$

Statistical weight

Shape function
(makes a smoother profile)

How do we use these macro-particles to make sure they satisfy Vlasov?

$$\partial_t f_s + \mathbf{v} \cdot \nabla f_s + \mathbf{F} \cdot \nabla_p f_s = 0$$

Vlasov + macro-particles → ???

Vlasov's equation

$$\partial_t f_s + \mathbf{v} \cdot \nabla f_s + \mathbf{F} \cdot \nabla_p f_s = 0$$

& macro-particles

$$f_s(t, \mathbf{x}, \mathbf{p}) = \sum_{p=1}^N w_p S(\mathbf{x} - \mathbf{x}_p(t)) \delta(\mathbf{p} - \mathbf{p}_p(t))$$

$$\partial_t f_s = \sum_{p=1}^N w_p [-\dot{\mathbf{x}}_p \cdot \nabla - \dot{\mathbf{p}}_p \cdot \nabla_p] S(\mathbf{x} - \mathbf{x}_p) \delta(\mathbf{p} - \mathbf{p}_p)$$

$$\sum w_p [(\mathbf{v} - \dot{\mathbf{x}}_p) \cdot \nabla + (\mathbf{F} - \dot{\mathbf{p}}_p) \cdot \nabla_p] S(\mathbf{x} - \mathbf{x}_p) \delta(\mathbf{p} - \mathbf{p}_p) = 0$$

As $\mathbf{v}$ commutes with ∇ and $\mathbf{F}$ commutes with ∇_p , we obtain:

$$\begin{aligned} & \sum w_p \nabla \cdot [(\mathbf{v} - \dot{\mathbf{x}}_p) S(\mathbf{x} - \mathbf{x}_p) \delta(\mathbf{p} - \mathbf{p}_p)] \\ & + \sum w_p \nabla_p \cdot [(\mathbf{F} - \dot{\mathbf{p}}_p) S(\mathbf{x} - \mathbf{x}_p) \delta(\mathbf{p} - \mathbf{p}_p)] = 0 \end{aligned}$$

Vlasov + macro-particles → ???

Integrate over momenta

$$\begin{aligned}
 & \sum w_p \nabla \cdot \int d^3 \mathbf{p} (\mathbf{v} - \dot{\mathbf{x}}_p) \delta(\mathbf{p} - \mathbf{p}_p) S(\mathbf{x} - \mathbf{x}_p) \\
 & + \sum w_p \int d^3 \mathbf{p} \nabla_p \cdot [(\mathbf{F} - \dot{\mathbf{p}}_p) \delta(\mathbf{p} - \mathbf{p}_p)] S(\mathbf{x} - \mathbf{x}_p) = 0
 \end{aligned}$$

$= \mathbf{v}_p - \dot{\mathbf{x}}_p$

$= 0$

Only one term remains

$$\sum w_p \nabla \cdot [(\mathbf{v}_p - \dot{\mathbf{x}}_p) S(\mathbf{x} - \mathbf{x}_p)] = 0$$

Multiply by $\mathbf{x}$

and integrate over $d^3 \mathbf{x}$ by parts

$$\sum w_p (\mathbf{v}_p - \dot{\mathbf{x}}_p) = 0$$

$$\dot{\mathbf{x}}_p = \mathbf{v}_p$$



Vlasov + macro-particles → ???

Same calculation, but now
we first multiply by $\mathbf{p}$
then integrate over $d^3\mathbf{p}$

$$= \mathbf{p}_p(\mathbf{v}_p - \dot{\mathbf{x}}_p) = 0$$

$$\sum w_p \nabla \cdot \int d^3\mathbf{p} \mathbf{p} (\mathbf{v} - \dot{\mathbf{x}}_p) \delta(\mathbf{p} - \mathbf{p}_p) S(\mathbf{x} - \mathbf{x}_p)$$

$$+ \sum w_p \int d^3\mathbf{p} \mathbf{p} \nabla_p \cdot [(\mathbf{F} - \dot{\mathbf{p}}_p) \delta(\mathbf{p} - \mathbf{p}_p)] S(\mathbf{x} - \mathbf{x}_p) = 0$$

Integrate by parts:

$$\sum w_p \int d^3\mathbf{p} (\mathbf{F} - \dot{\mathbf{p}}_p) \delta(\mathbf{p} - \mathbf{p}_p) S(\mathbf{x} - \mathbf{x}_p) = 0$$

$$\sum w_p (q_s \mathbf{E} + q_s \mathbf{v}_p \times \mathbf{B} - \dot{\mathbf{p}}_p) S(\mathbf{x} - \mathbf{x}_p) = 0$$

Integrate over $d^3\mathbf{x}$

$$\sum w_p (q_s \mathbf{E}_p + q_s \mathbf{v}_p \times \mathbf{B}_p - \dot{\mathbf{p}}_p) = 0$$

where we have defined

$$\left\{ \begin{array}{l} \mathbf{E}_p = \int \mathbf{E}(\mathbf{x}) S(\mathbf{x} - \mathbf{x}_p) d^3\mathbf{x} \\ \mathbf{B}_p = \int \mathbf{B}(\mathbf{x}) S(\mathbf{x} - \mathbf{x}_p) d^3\mathbf{x} \end{array} \right.$$

$$\dot{\mathbf{p}}_p = q_s \mathbf{E}_p + q_s \mathbf{v}_p \times \mathbf{B}_p \quad !$$

Vlasov + macro-particles $\rightarrow$ motion

The motion of macro-particles is essentially that of real particles

$$\begin{cases} \dot{\mathbf{x}}_p = \mathbf{v}_p \\ \dot{\mathbf{p}}_p = q_s \mathbf{E}_p + q_s \mathbf{v}_p \times \mathbf{B}_p \end{cases} \quad \text{“Pusher”}$$

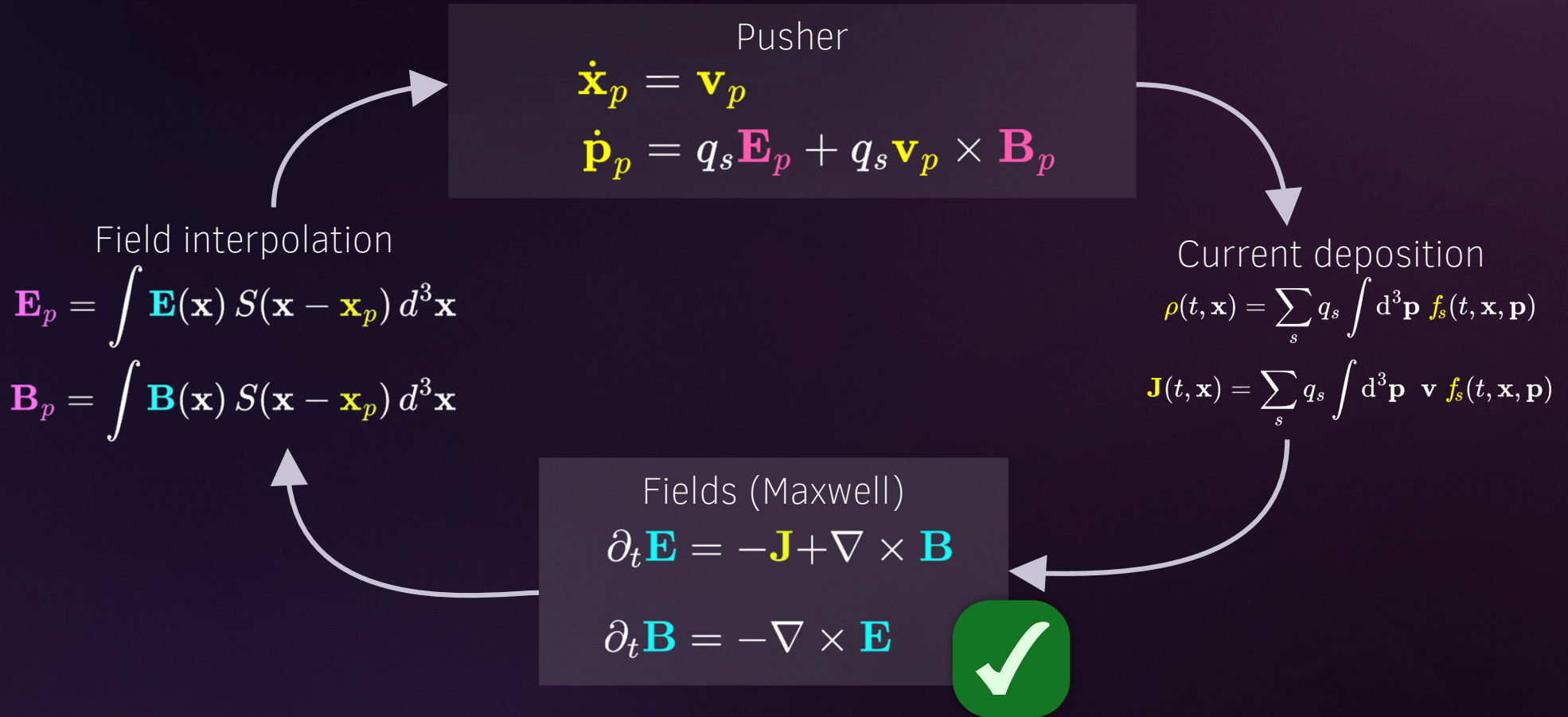
But ...

$$\begin{cases} \mathbf{E}_p = \int \mathbf{E}(\mathbf{x}) S(\mathbf{x} - \mathbf{x}_p) d^3\mathbf{x} \\ \mathbf{B}_p = \int \mathbf{B}(\mathbf{x}) S(\mathbf{x} - \mathbf{x}_p) d^3\mathbf{x} \end{cases}$$

The fields are “averaged”
in the vicinity of the particle.

“Field interpolation”

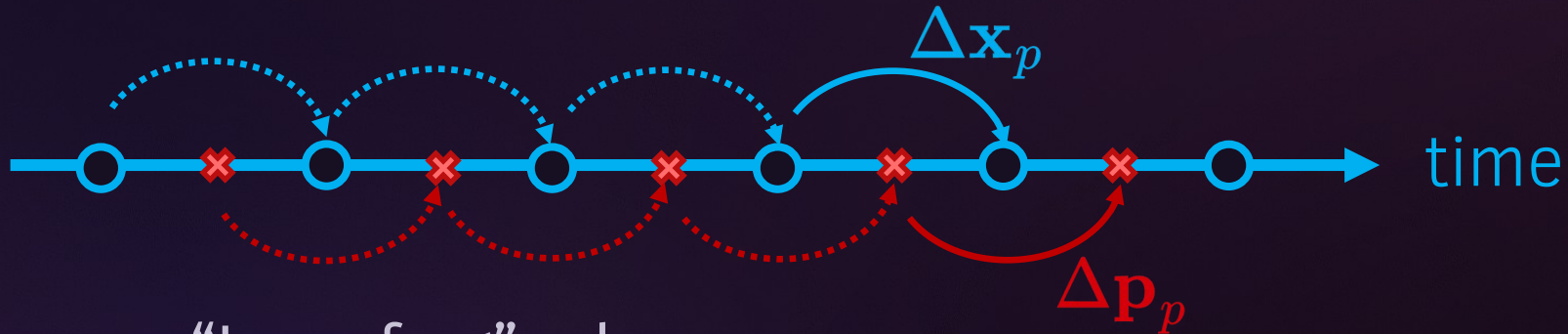
The Maxwell-Vlasov system



The pusher is also *staggered* in time

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p \longrightarrow \mathbf{x}_p^{n+1} - \mathbf{x}_p^n = \mathbf{v}_p^{n+1/2} \Delta t$$

$$\frac{d\mathbf{p}_p}{dt} = \mathbf{F}_p \longrightarrow \mathbf{p}_p^{n+1/2} - \mathbf{p}_p^{n-1/2} = \mathbf{F}_p^n \Delta t$$

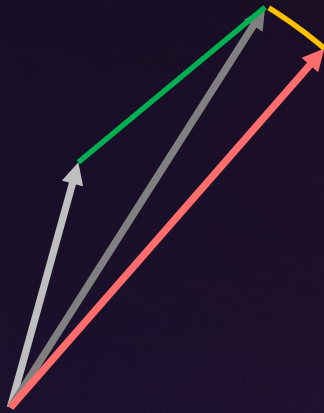


“Leap-frog” scheme

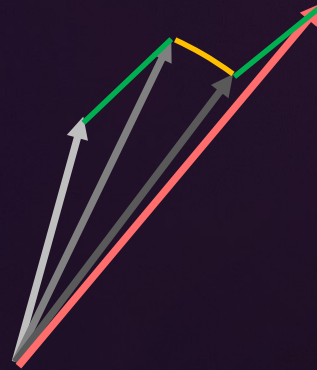
How to make an accurate pusher?

$$\frac{d\mathbf{p}_p}{dt} = q_s \mathbf{E}_p + q_s \mathbf{v}_p \times \mathbf{B}_p \quad \rightarrow \quad \frac{\mathbf{p}^{n+1/2} - \mathbf{p}^{n-1/2}}{\Delta t} = q_s \mathbf{E}^n + q_s \left(\frac{\mathbf{v}^{n+1/2} + \mathbf{v}^{n-1/2}}{2} \right) \times \left(\frac{\mathbf{B}^{n+1/2} + \mathbf{B}^{n-1/2}}{2} \right)$$

addition + rotation



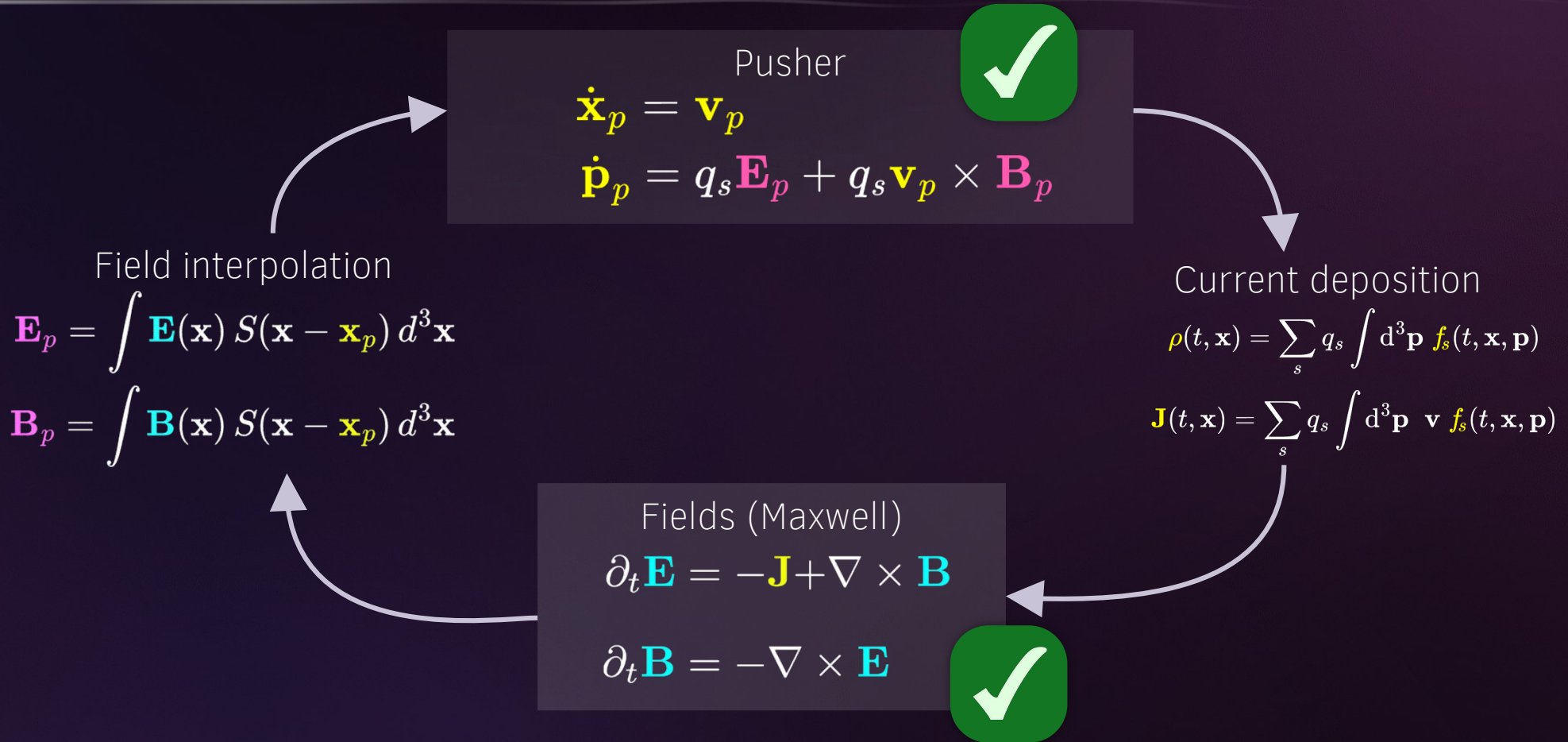
Naïve method



Boris' method
(more accurate)

Other methods ...
(depending on calculation order)

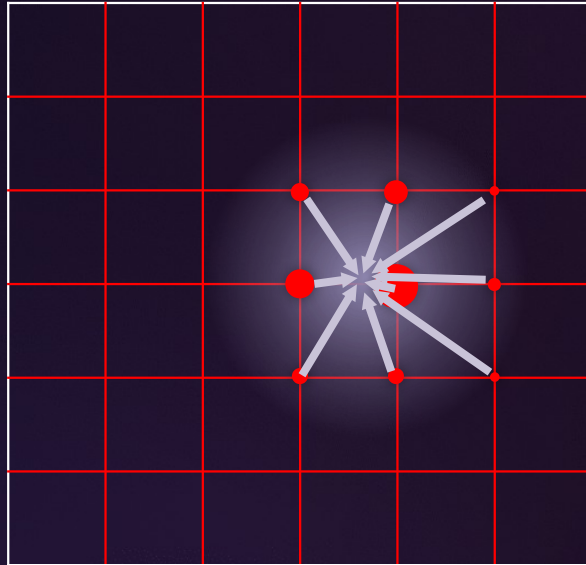
The Maxwell-Vlasov system



Interpolating fields on macro-particles ...

$$\begin{cases} \mathbf{E}_p = \int \mathbf{E}(\mathbf{x}) S(\mathbf{x} - \mathbf{x}_p) d^3\mathbf{x} \\ \mathbf{B}_p = \int \mathbf{B}(\mathbf{x}) S(\mathbf{x} - \mathbf{x}_p) d^3\mathbf{x} \end{cases}$$

Fields have values on grid points
We must express the integrals as a sum

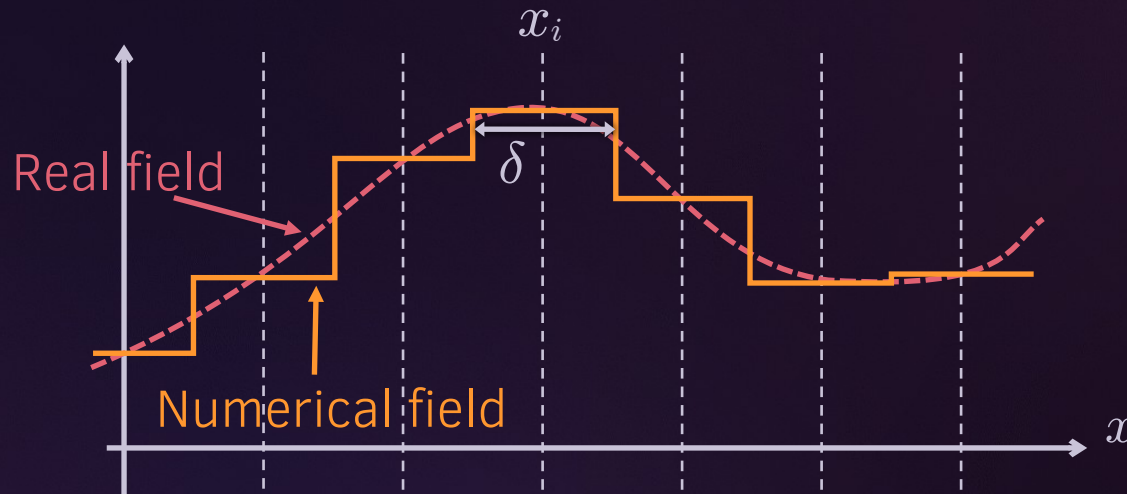


Each grid point surrounding
the macro-particle
contributes to the field it sees

Interpolating fields on macro-particles ...

$$\mathbf{E}_p = \int \mathbf{E}(x) S^{(m)}(x - \mathbf{x}_p) dx = \sum_{\text{cells } i} \mathbf{E}(x_i) \underbrace{\int_{x_i-\delta}^{x_i+\delta} S^{(m)}(x - \mathbf{x}_p) dx}_{\text{shape function of order } m+1}$$

We define this integral as the
shape function of order $m+1$
 $S^{(m+1)}(\mathbf{x}_i - \mathbf{x}_p)$



Interpolating fields on macro-particles ...

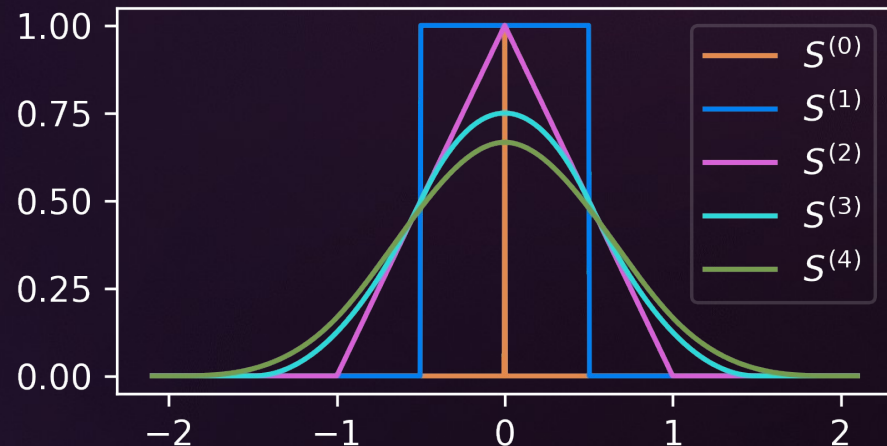
Finally, we obtain the field interpolation in the form:

$$\mathbf{E}_p = \sum_{\text{cells } i} \mathbf{E}(\mathbf{x}_i) S^{(m+1)}(\mathbf{x}_i - \mathbf{x}_p)$$

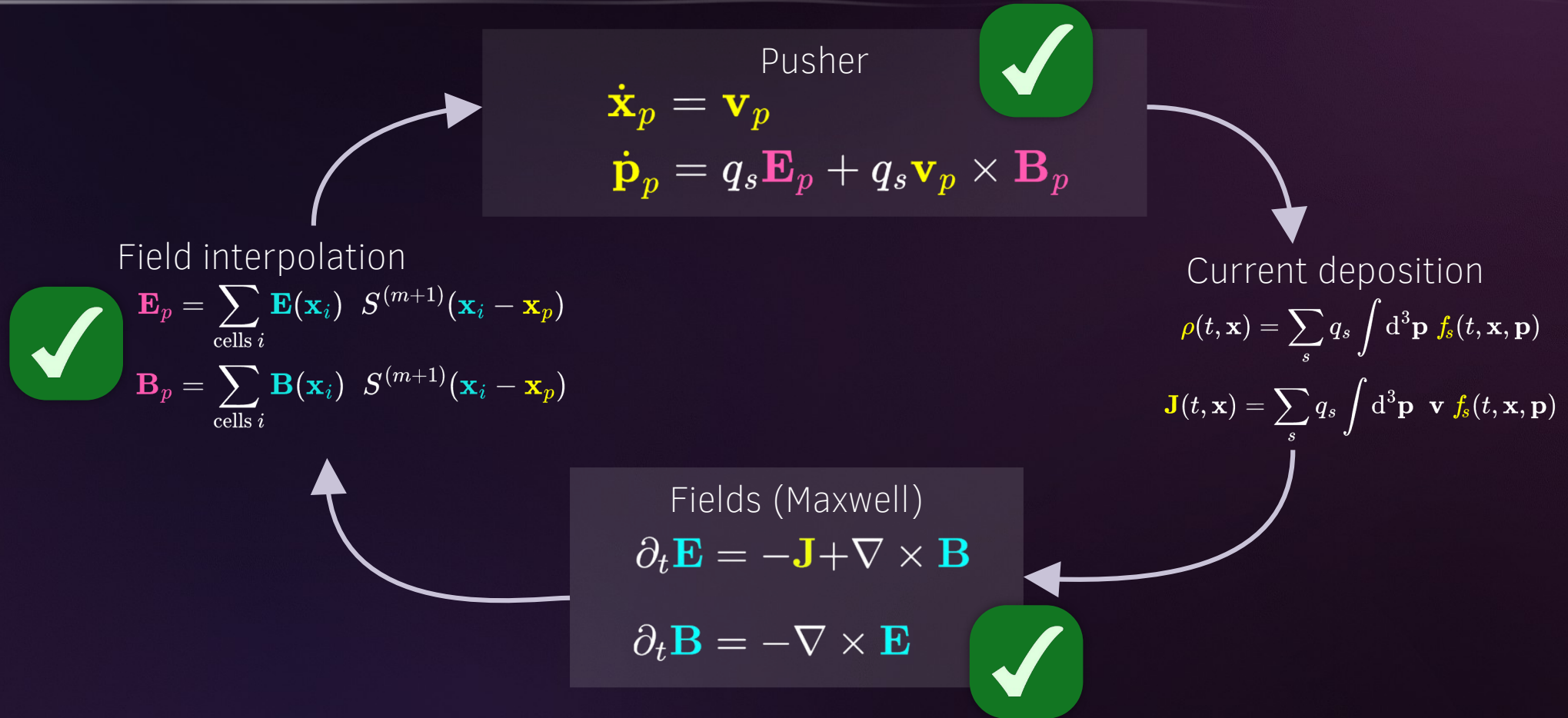
$$\mathbf{B}_p = \sum_{\text{cells } i} \mathbf{B}(\mathbf{x}_i) S^{(m+1)}(\mathbf{x}_i - \mathbf{x}_p)$$

It is possible to choose the **interpolation order m**

A higher order requires more computation, but reduces the noise

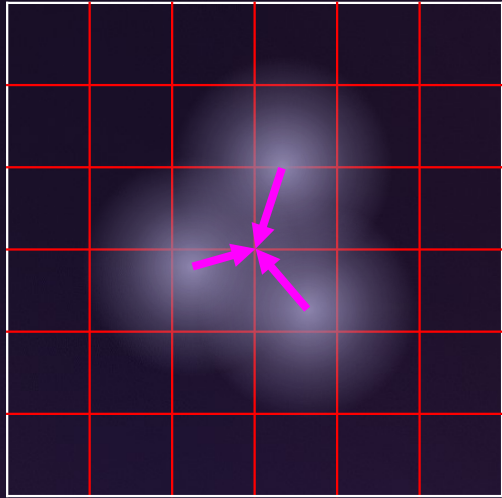


The Maxwell-Vlasov system



Calculate the current from macro-particles

$$\left\{ \begin{array}{l} f_s(t, \mathbf{x}, \mathbf{p}) = \sum_{p=1}^N w_p S(\mathbf{x} - \mathbf{x}_p(t)) \delta(\mathbf{p} - \mathbf{p}_p(t)) \\ \mathbf{J}(t, \mathbf{x}) = q_s \int d^3\mathbf{p} \, \mathbf{v} f_s(t, \mathbf{x}, \mathbf{p}) \end{array} \right. \longrightarrow \mathbf{J}(\mathbf{x}_i) = \sum_{\text{particles}} q_s w_p \mathbf{v}_p S(\mathbf{x}_i - \mathbf{x}_p)$$



macro-particles surrounding a grid point “deposit” their current

Unfortunately, this does not satisfy the charge-conservation equation

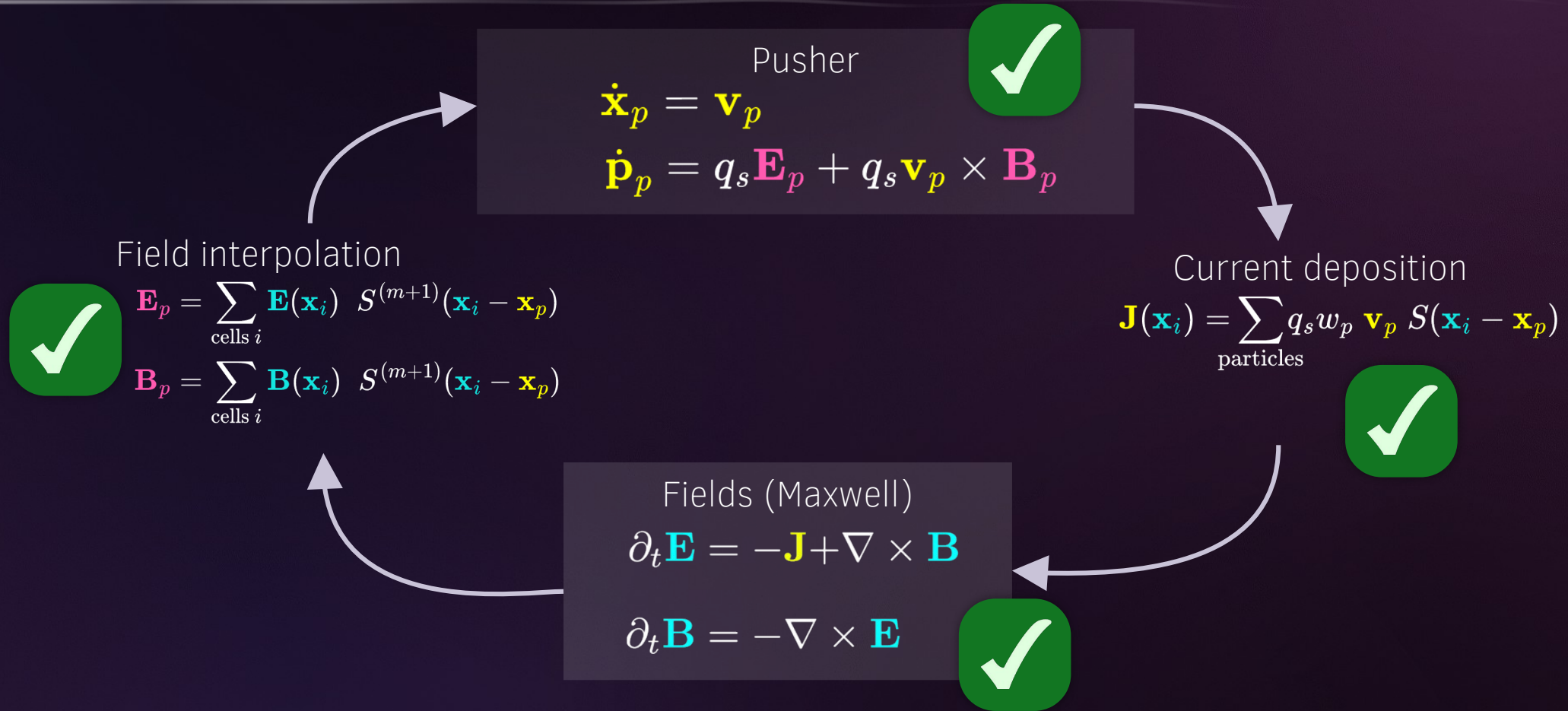
Forcing charge conservation ...

$$\left\{ \begin{array}{l} f_s(t, \mathbf{x}, \mathbf{p}) = \sum_{p=1}^N w_p S(\mathbf{x} - \mathbf{x}_p(t)) \delta(\mathbf{p} - \mathbf{p}_p(t)) \\ \rho(t, \mathbf{x}) = \sum_s q_s \int d^3\mathbf{p} f_s(t, \mathbf{x}, \mathbf{p}) \\ \partial_t \rho + \nabla \cdot \mathbf{J} = 0 \end{array} \right.$$

$$\Rightarrow \mathbf{J}(\mathbf{x}_i + d\mathbf{x}/2) - \mathbf{J}(\mathbf{x}_i - d\mathbf{x}/2) = \dots$$

Esirkepov's method

The Maxwell-Vlasov system



Many more aspects must be considered

- Is the simulation stable and physical?
- How to initialize a simulation (plasma, laser, external fields, ...)
- Boundary conditions?
- Other geometries (cylindrical, spherical, ...)
- Add missing physics (collisions, QED, reactions, ...)
- How to make a parallel simulation (on many computers)

Advertisement

Smilei) workshop



indico.math.cnrs.fr/e/smilei6

25–27 Jan 2027 in Milan (Italy)



POLITECNICO
MILANO 1863

DIPARTIMENTO
DI ENERGIA



- Project review
- Contributed oral and poster presentations
- Training on Smilei for newcomers

Stability of PIC codes



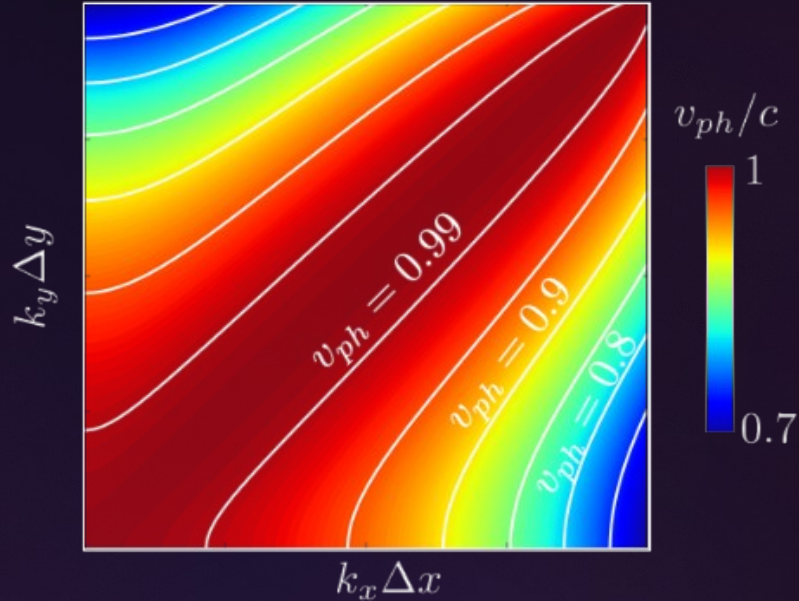
Often, a PIC simulation won't crash when the results are meaningless.

Users must understand the limitations and test.

The numerical vacuum is dispersive and anisotropic !

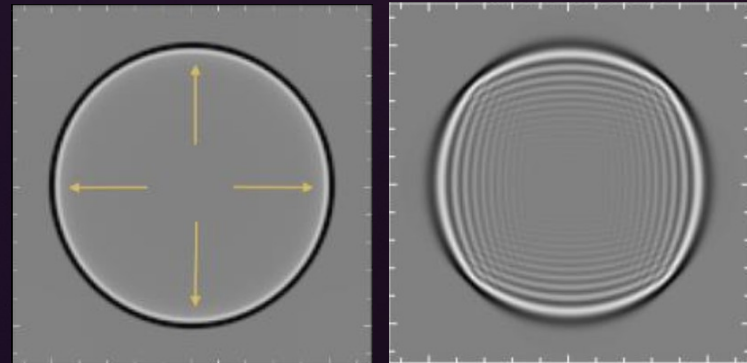
FDTD equations + search for wave-like solutions

➔ **Dispersion relation** $\Delta t^{-2} \sin^2(\omega \Delta t / 2) = \sum_{a=x,y,z} \Delta a^{-2} \sin^2(k_a \Delta a / 2)$



Dispersive vacuum !

Example: wave expanding from point source
physical numerical



The timestep cannot be too large

From the dispersion relation, one can show that **stability requires:**

$$\Delta t < \left(\sum_{a=x,y,z} \Delta a^{-2} \right)^{-1/2}$$

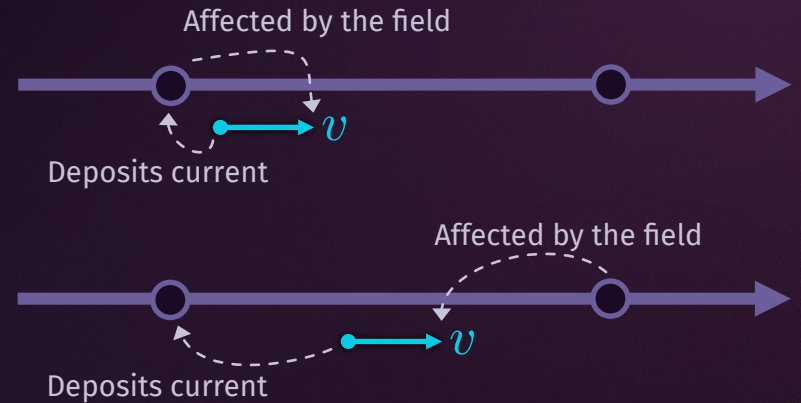
Courant-Friedrich-Levy
(CFL) condition

(otherwise ω becomes imaginary)

Note: The dispersion of plasma waves also requires $\Delta t < 2 / \omega_p$

Numerical plasma heating caused limited resolution

- Imagine a macro-particle moving through the grid
- The feedback from the grid depends on the particle position



- Correlation between particle position and grid feedback (not random)
- Fluctuations of the particle are coupled to grid modes
- Excites unphysical modes

The feedback does not average to 0 → **numerical heating**

How to remove numerical heating?

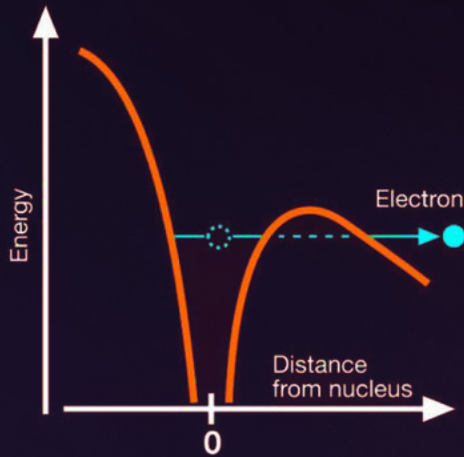
- Use $\Delta x \lesssim \lambda_D$ → physical Debye shielding (unstable feedback drops quickly)
→ But very costly, especially in 3D
- Use higher-order shape functions → less “jumps” from cell to cell → less feedback
→ Significant performance loss
- Use current filtering → suppresses high-frequency unstable modes

Some missing physics in PIC codes

The PIC model is missing some physics

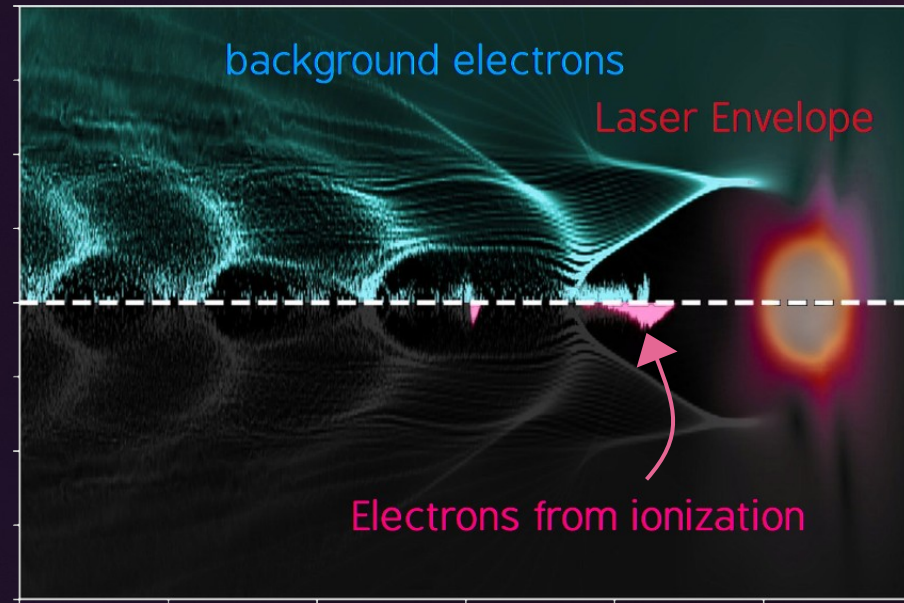
- Vlasov → No particle-particle interactions
- Maxwell → No quantum effects
- Grid → No high-frequency photons
- Fixed charge → No ionization / atomic physics
- Fixed mass → No nuclear physics

Tunnel ionisation often dominates in high-intensity interaction



- Atomic potential is deformed by the intense laser field
- probability for tunnelling electron
- Ionization can be added using a Monte-Carlo process
(probability given by laser intensity and tabulated ionization potential)

Example:
laser wakefield acceleration



Collisions are missing in Vlasov

$$\partial_t f_s + \mathbf{p} \cdot \nabla f_s + q_s (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_p f_s = -q_s \langle (\delta \mathbf{E} + \mathbf{v} \times \delta \mathbf{B}) \delta f_s \rangle .$$

Collisions neglected

(PIC code does not treat short range interactions)

- No collisional stopping power
- No collisional absorption
- No collisional thermalisation
- No electrical/thermal conductivity
- ...

Collisions are 2-particle interactions → requires N-body treatment?

Calculating all collisions between all particles is too expensive

- Restrict calculation to macro-particles in 1 cell (remains as local as possible)
- Macro-particles $\neq$ real particles $\rightarrow$ only need to represent the change of $f_s(t, \mathbf{x}, \mathbf{p})$
- Simplest approach:
 - Calculate averages like $\rho, T_e, T_i, \dots$
 - For each particle: pick Δp randomly from theoretical model
- Other approach: binary collisions
 - Pair particles 2 by 2  $\rightarrow$ samples the distribution after many iterations
 - For each pair: modify the Δp using a Monte-Carlo model

Pairing, like in binary collisions, can be used for other physics

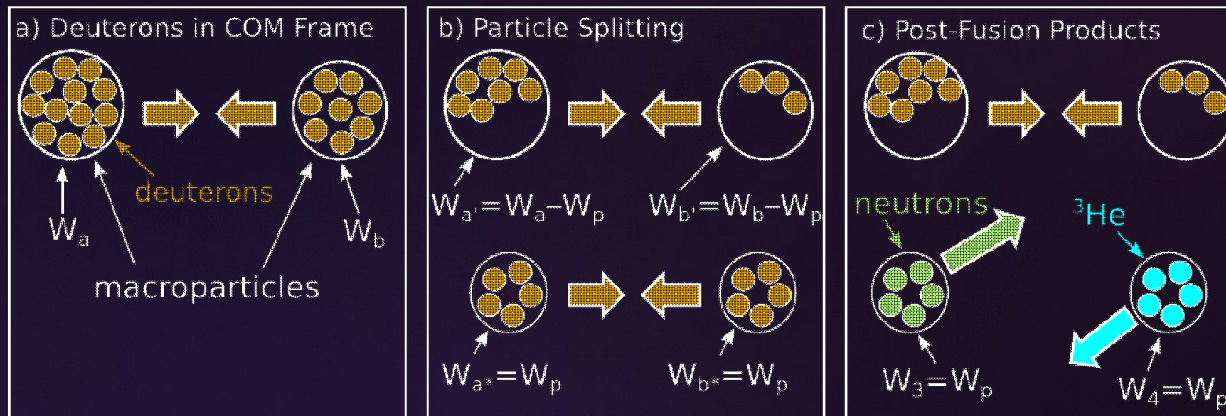
- Collisional ionisation ($e^- + \text{ion} \rightarrow \text{new } e^-$)
- Photoionisation ($\gamma + \text{ion} \rightarrow \text{new } e^-$)
- Bremsstrahlung ($e^- + \text{nucleus} \rightarrow \gamma$)
- Bethe-Heitler process ($\gamma + \text{nucleus} \rightarrow e^- + e^+$)
- Nuclear reactions
- ...

All these processes are generally treated using Monte-Carlo.
(Random number compared to theoretical probability of event)

They require tabulated cross-section data

Nuclear reactions require special treatment

- Imagine you have 100 macro-particles in 1 cell, but probability of reaction is 10^{-6}
→ no reactions happen
- A “multiplier” is introduced:
 - multiplies the reaction rate
 - divides the statistical weight of macro-particles that are created
 - Macro-particles are “split” to conserve the total number of real particles



Higginson et al., JCP **388**, 439 (2019)

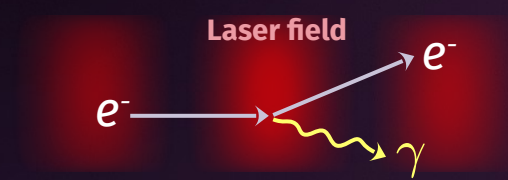
Many QED processes may be added

$$E_s \sim 10^{18} \text{ V/m}$$

- Inverse Compton Scattering

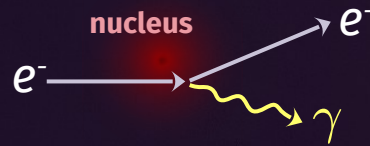
High-energy photon cannot be represented on the grid $\rightarrow$ macro-photon

QED parameter $\chi = \frac{\gamma}{E_s} \sqrt{(\vec{E} + \vec{v} \times \vec{B})^2 - (\vec{v} \cdot \vec{E})^2 / c^2} \sim \gamma \frac{E}{E_s} \rightarrow$ quantum if ~ 1 or above



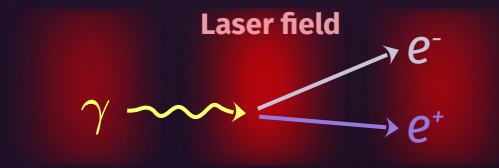
1 macro-particle

- Bremsstrahlung



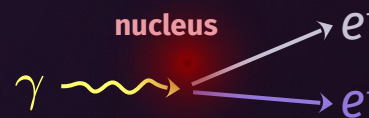
2 macro-particles

- Breit-Wheeler pair creation



1 macro-particle

- Bethe-Heitler pair creation



2 macro-particles

Challenges in high-performance computing

How much resources does a PIC simulation need?

Example:

3D simulation, 1000^3 cells, 10 particles per cell, 10^6 iterations

➔ 10^{16} macro-particle computations

In a ideal case, 1 macro-particle takes 25 ns to compute

➔ 3000 days on a single processor !

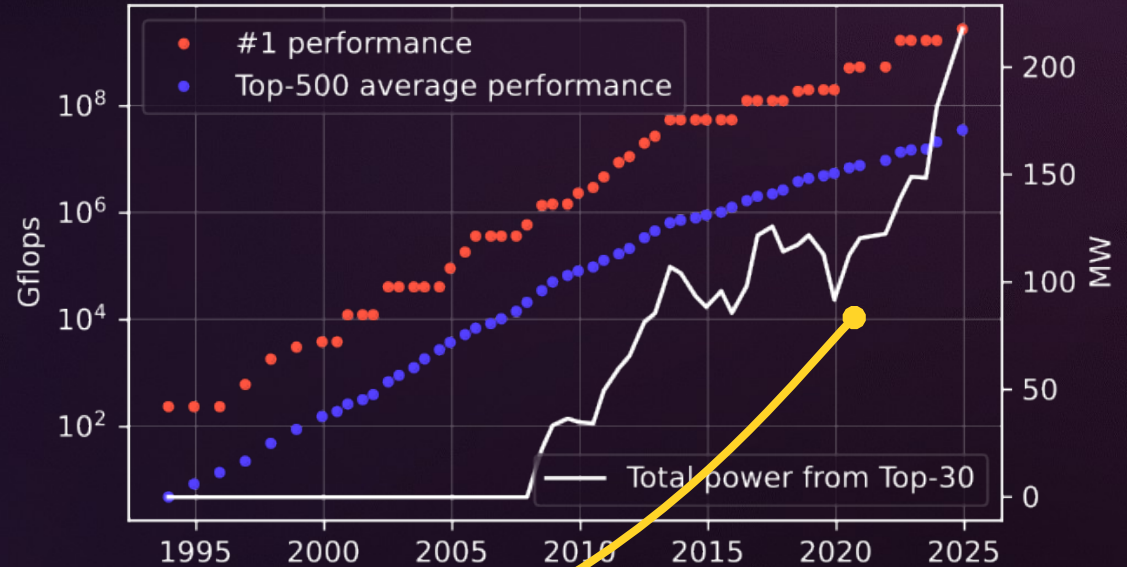
10^{10} macro-particles occupy a lot of memory

➔ ~ 1 TB of RAM

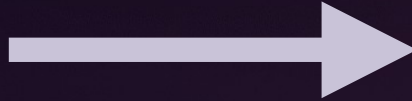


Supercomputers

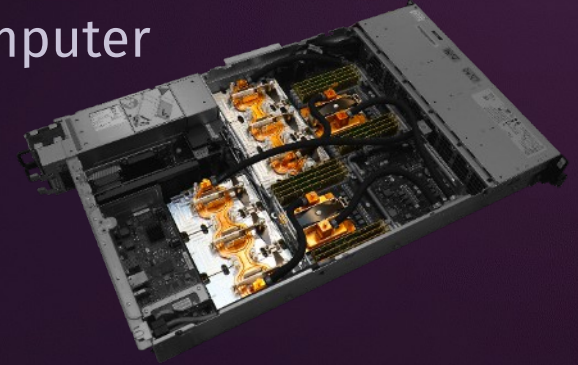
Thousands of nodes, each containing tens or hundreds of cores



Components of a computer

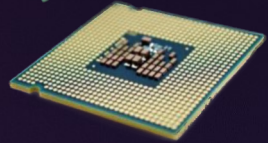


Supercomputer
node



Volatile memory (RAM)

> 1 TB



Processor (CPU)

Contains
several
cores

32 to 256 cores
“intra-node” parallelism

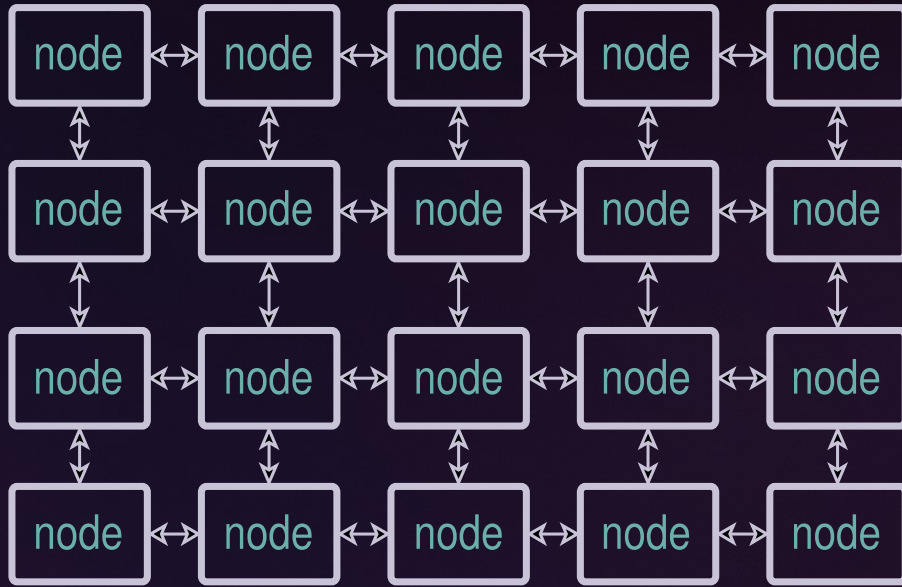


Accelerator (GPU)

Up to 8 GPUs
(> 10 Tflops)

GPU software

A supercomputer contains many nodes



Nodes connected
through a fast network

“inter-node” parallelism



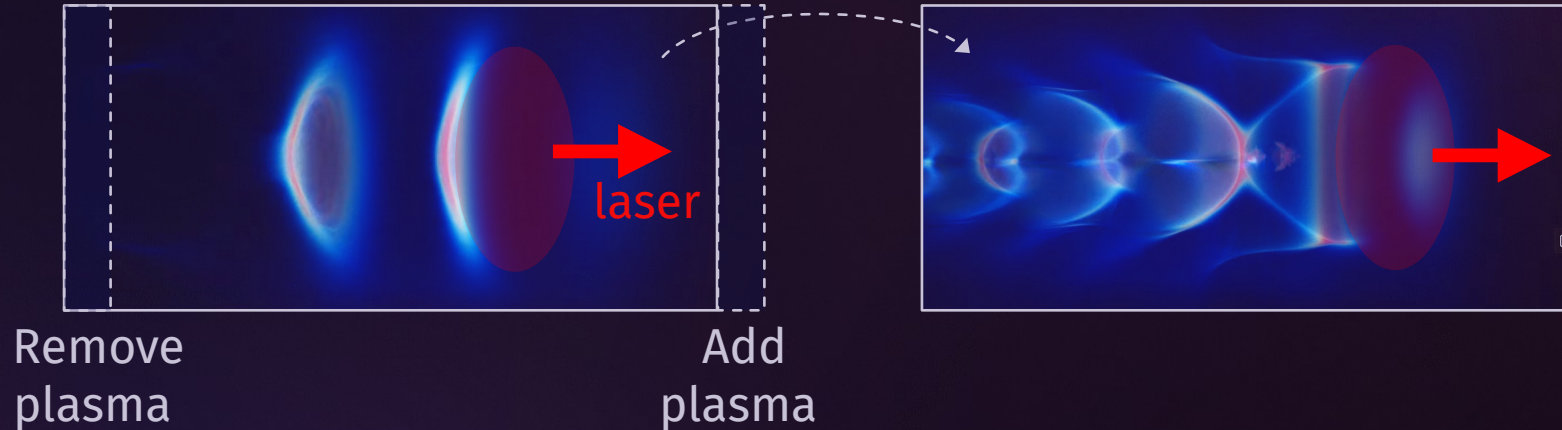
Software challenges

- CPU-specific optimizations
- Multi-level parallelism (intra-node and inter-node)
- GPU-compatible code
- Memory usage
- Writing large files
- Splitting the plasma in “tiles” to distribute the computation

Advanced models to speed up the calculation

Moving window

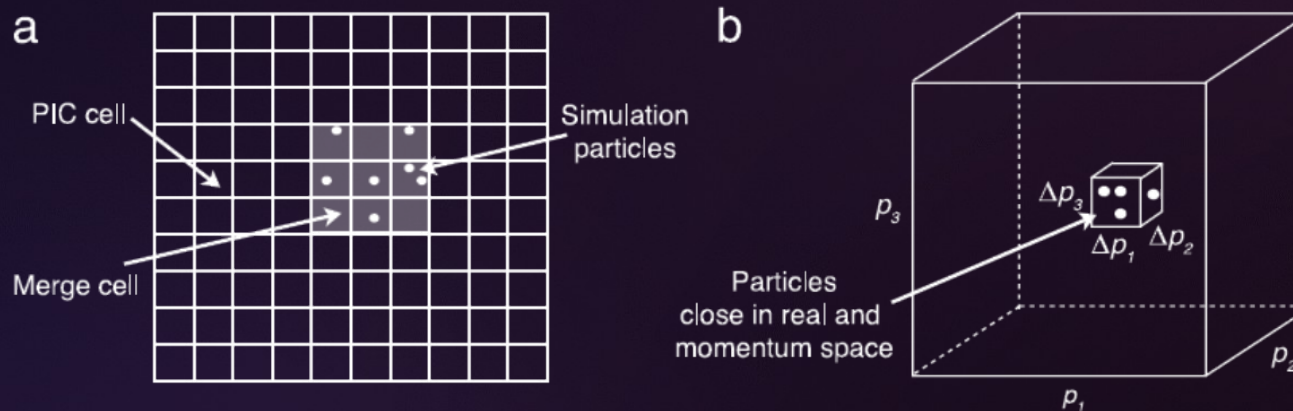
- Wakefield acceleration: cannot simulate centimeters
- The PIC box is “moving” together with the laser



Particle-merging

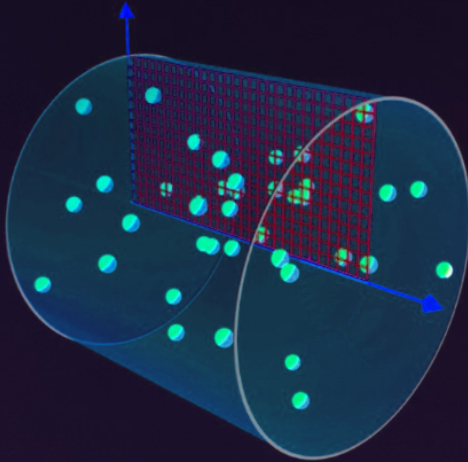
- Some physics creates a lot of particles (ionisation, pair production, ...)
- Too many particles won't fit in memory
- Merge macro-particles close in phase-space

M. Vranic et al. / Computer Physics Communications 191 (2015) 65–73



Different representations for the fields

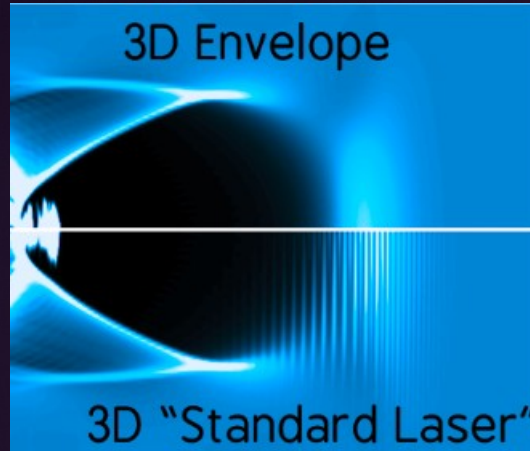
Cylindrical geometry with azimuthal modes



Particles are 3D
Fields are 2D
+ Fourier modes in θ

➔ strongly reduces computational cost

Envelope solvers



Envelope equation
instead of full Maxwell

➔ less resolution needed

Pseudo-spectral solvers

Fields are represented in
Fourier space
➔ can help reduce artifacts

Many other solvers, filters, ...

And much more ...

- Multi-level meshes (with different resolutions)
- Fluid species (not represented as macro-particles)
- Boosted simulation frame
- Advanced boundary conditions (absorbing, thermalising, ...)
- ...

Modern PIC codes

EPOCH

Smilei)

PICon**GPU** 

WarpX

OSIRIS

CALDER

...

OSIRIS continuous improvements since 1999 !

- IST (Portugal) + UCLA (USA)
- First version ~ 1999, fortran
- Only a small portion is open-source (since 2023)
Otherwise requires a MOU to obtain the code.
- Cutting-edge code with continuous developments
- QED, collisions, ionisation, many advanced models, ...
- Text input
- MPI + openMP parallelisation, load balancing, ...
- GPU support

EPOCH an early open-source code

- University of Warwick
- First version ~ 2011, fortran, based on old code
- Most-cited by far, as it was ~ the only open-source code
- New c++ version, renewed activity since ~ 2021
- QED and other physics
- Text input
- Own output file format (SDF)
- MPI parallelisation, load balancing, ...

- Several laboratories in France
- First release ~ 2018, c++ and python
- Completely open-source
- Cutting-edge code with continuous developments
- QED, collisions, ionisation, many advanced models, ...
- Python script as input
- MPI + openMP parallelisation, load balancing, ...
- GPU support

- Various laboratories in the US (mainly LBNL) + a few people in Europe
- First version ~ 2018, python and c++
- Completely open-source
- Cutting-edge code with continuous developments
- Many additional model, but specifically: experimental *AMR*
- Python input
- MPI + openMP parallelisation, load balancing, ...
- GPU support

Advertisement

Smilei) workshop



indico.math.cnrs.fr/e/smilei6

25–27 Jan 2027 in Milan (Italy)



POLITECNICO
MILANO 1863

DIPARTIMENTO
DI ENERGIA



- Project review
- Contributed oral and poster presentations
- Training on Smilei for newcomers