Particle in Cell Simulations

Tony Arber

University of Warwick

EPOCH Workshop, Strathcylde, 3-4 Feb 2014

What are PIC Codes?

Aim:
Simulate the full kinetics of a plasma and its interaction with self-consistent EM fields.

Particle in Cell:

- Particles move freely under action of Lorentz force
- Moments of distribution functions (density and velocity)

plus all EM field variables calculated only on a fixed grid.

Macro-particles

In practice too many real particles to solve full problem.

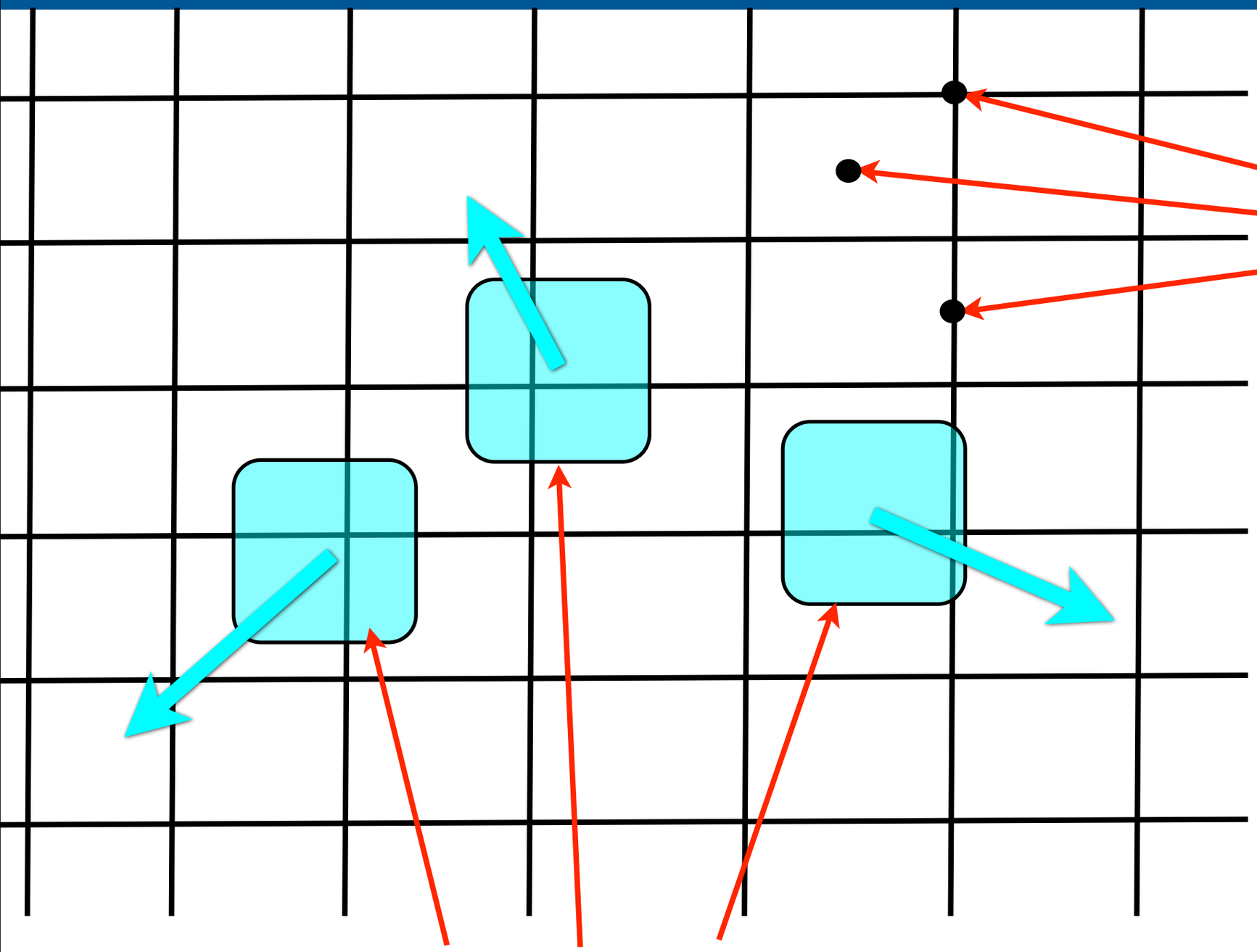
Introduce macro-particles (super-particles) which represent many real particles.

In collisionless limit these macro-particles are free to move through each other.

Since they represent a ‘cloud’ of real particles they occupy a finite volume and have a shape.

They do not change their shape under normal motion, nor do they rotate or have any internal degrees of freedom.

PIC Grids



EM field values & fluid moments stored at fixed locations on a fixed grid

Macro-particles, of finite size, move freely space.

Basic Algorithm

- Update EM fields on fixed grid - needs fluid moments.
- Interpolate EM fields from grid to particle locations.
- Move particles in local EM field.
- Update fluid moments on fixed grid from particles.

This takes values at time-step *n*, e. g. $\mathbf{E}^n$ or $\mathbf{r_i^n}$, and moves them forwards in time by a fixed time-step.

Initialising the macro-particles

For initial conditions we usually know the density, temperature and centre of mass motion for each species, i.e. the fluid moments.

Macro-particles then selected so that they match the probability distribution required, usually Maxwellian.

Macro-particles have the same charge to mass ratio, and hence orbit, as real particles but have much larger actual charge and mass.

Mathematical model

For each particle in the simulation

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}$$

$$\frac{d\mathbf{u}}{dt} = \frac{q}{m}(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

$$\mathbf{u} = \gamma \mathbf{v}$$

$$\gamma = \left(1 + \left(\frac{u}{c}\right)^2\right)^{1/2}$$

Update the EM field through Maxwell's equations

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

$$\frac{\partial \mathbf{E}}{\partial t} = c^2 \nabla \times \mathbf{B} - \frac{\mathbf{j}}{\epsilon_0}$$

Current density found from particle positions and momenta

Maxwell Equations - only 2!

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \longrightarrow \frac{\partial}{\partial t} \nabla . \mathbf{B} = 0$$

Provided div.B=0 initially it will always be zero

$$\frac{\partial \mathbf{E}}{\partial t} = c^2 \nabla \times \mathbf{B} - \frac{\mathbf{j}}{\epsilon_0} \longrightarrow \frac{\partial}{\partial t} \nabla . \mathbf{E} = -\frac{\nabla . \mathbf{j}}{\epsilon_0}$$

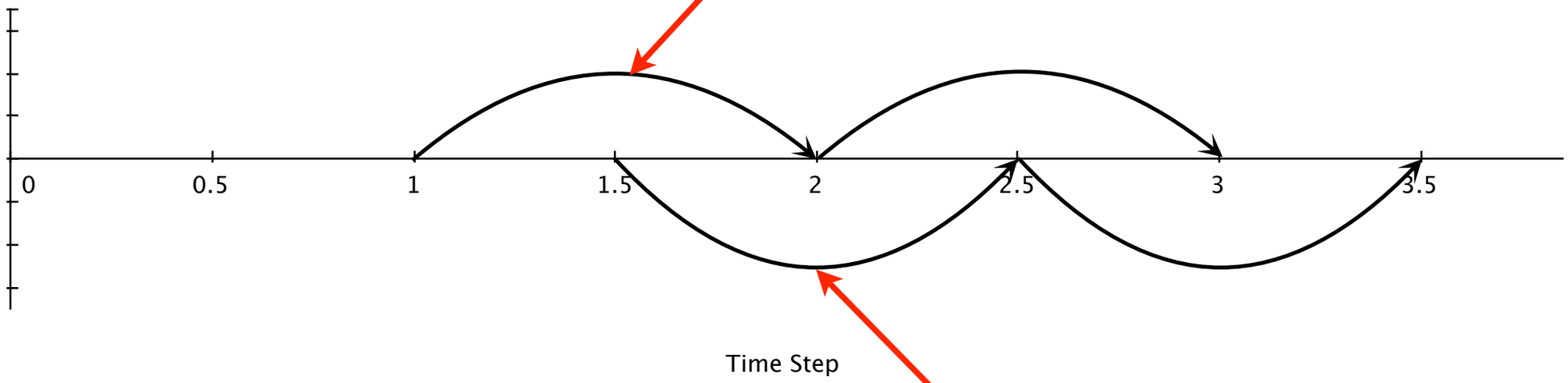
$$\frac{\partial \rho}{\partial t} = -\nabla . \mathbf{j} \longrightarrow \nabla . \mathbf{E} = \frac{\rho}{\epsilon_0}$$

Provided charge is conserved Gauss's law is satisfied

Analytically only need two of Maxwell's equations
Not automatically true for numerical schemes

Field Update - Leapfrog

$$\frac{\mathbf{B}^{n+1} - \mathbf{B}^{n}}{\Delta t} = -\nabla \times \mathbf{E}^{n+1/2}$$



$$\frac{\mathbf{E}^{n+3/2} - \mathbf{E}^{n+1/2}}{\Delta t} = c^2 \nabla \times \mathbf{B}^{n+1} - \frac{\mathbf{j}^{n+1}}{\epsilon_0}$$

RHS time-centred therefore 2nd order accurate, i.e. error $O(\Delta t^2)$

Need E and B at different times

Field Updates - FDTD

Many PIC codes now use Finite Difference Time Domain

$$\frac{\mathbf{E}^{n+1/2} - \mathbf{E}^{n}}{\Delta t/2} = c^2 \nabla \times \mathbf{B}^{n} - \frac{\mathbf{j}^{n}}{\epsilon_0}$$

$$\frac{\mathbf{B}^{n+1/2} - \mathbf{B}^{n}}{\Delta t/2} = -\nabla \times \mathbf{E}^{n+1/2}$$

Update particle position and momentum to find $\mathbf{j}^{n+1}$

$$\frac{\mathbf{B}^{n+1} - \mathbf{B}^{n+1/2}}{\Delta t/2} = -\nabla \times \mathbf{E}^{n+1/2}$$

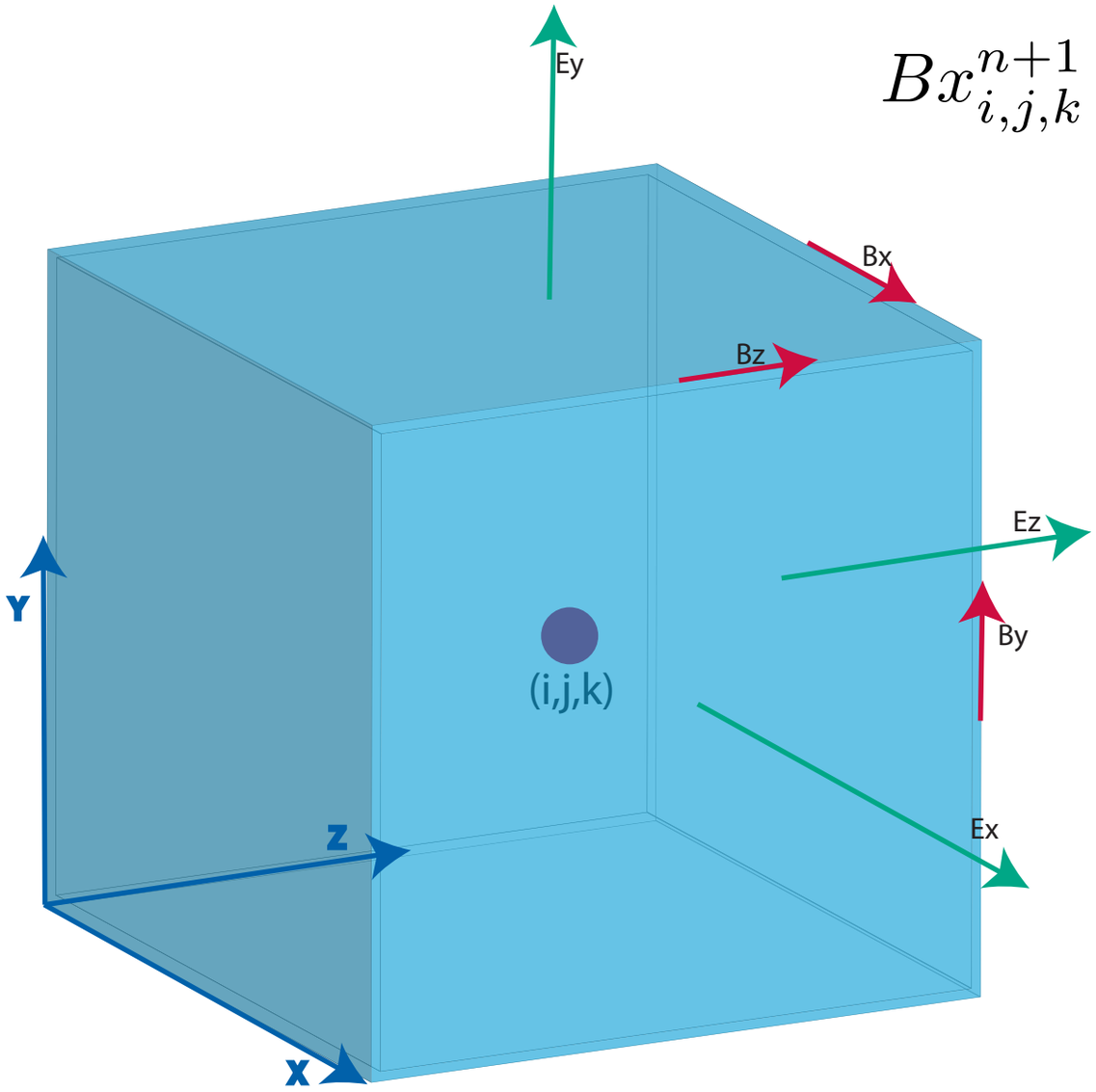
$$\frac{\mathbf{E}^{n+1} - \mathbf{E}^{n+1/2}}{\Delta t/2} = c^2 \nabla \times \mathbf{B}^{n+1} - \frac{\mathbf{j}^{n+1}}{\epsilon_0}$$

Combined scheme a re-ordering of 2nd order Leapfrog - it is the same as Leapfrog!
E and B defined at the same times

Spatial grid - the Yee grid

In 3D the variables are defined on the staggered Yee grid to centre all derivatives

$$\frac{\mathbf{B}^{n+1} - \mathbf{B}^{n}}{\Delta t} = -\nabla \times \mathbf{E}^{n+1/2}$$



$$Bx_{i,j,k}^{n+1} - Bx_{i,j,k}^{n} = -\frac{\Delta t}{\Delta y}\left(Ez_{i,j,k+1}^{n+1/2} - Ez_{i,j,k}^{n+1/2}\right) + \frac{\Delta t}{\Delta z}\left(Ey_{i,j+1,k}^{n+1/2} - Ey_{i,j,k}^{n+1/2}\right)$$

This automatic centering of spatial derivatives is true for all derivatives required

Particle Motion - 1

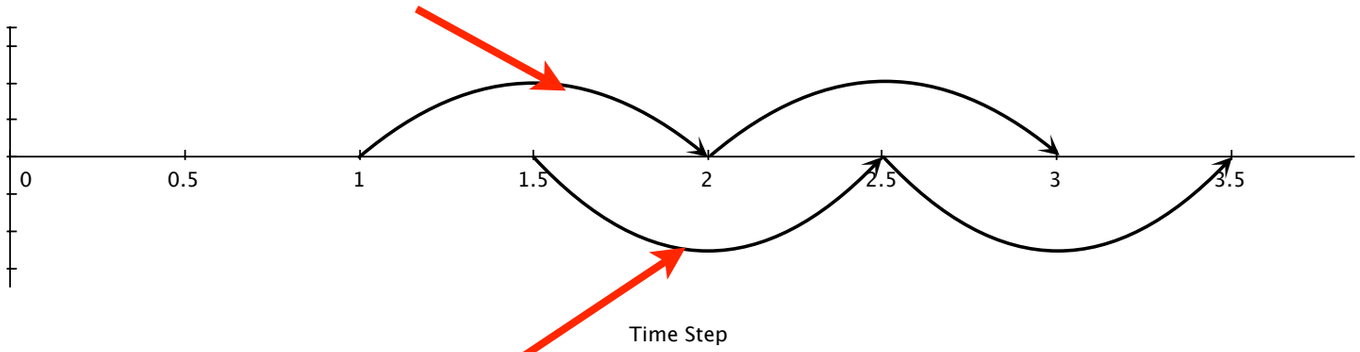
$$\frac{d\mathbf{r}}{dt} = \mathbf{v}$$

$$\frac{d\mathbf{u}}{dt} = \frac{q}{m}(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

Aim is for 2nd order accuracy so need RHS time-centred.

Leapfrog approach

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^{n}}{dt} = \frac{q}{m}(\mathbf{E}^{n+1/2} + \mathbf{v}^{n+1/2} \times \mathbf{B}^{n+1/2})$$



$$\mathbf{B}^{n+1/2} = \mathbf{B}(\mathbf{r}^{n+1/2})$$

$$\mathbf{E}^{n+1/2} = \mathbf{E}(\mathbf{r}^{n+1/2})$$

How to get time-centred velocity?

$$\frac{\mathbf{r}^{n+1/2} - \mathbf{r}^{n+3/2}}{dt} = \mathbf{v}^{n+1}$$

Particle Motion - 2

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{q}{m}(\mathbf{E}^{n+1/2} + \mathbf{v}^{n+1/2} \times \mathbf{B}^{n+1/2})$$

Set time-centred velocity to be an average

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{q}{m}(\mathbf{E}^{n+1/2} + \frac{1}{2}(\mathbf{v}^{n+1} + \mathbf{v}^n) \times \mathbf{B}^{n+1/2})$$

This is implicit as it involves v, and hence u, at the updated time-step on the RHS

Boris algorithm -1

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}=\frac{q}{m}\left(\mathbf{E}^{n+1/2}+\frac{1}{2\gamma^{n+1/2}}(\mathbf{u}^{n+1}+\mathbf{u}^{n})\times\mathbf{B}^{n+1/2}\right)$$

Rearrange as

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

Where the new velocities are defined such that

$$\frac{\mathbf{u}^{+}-\mathbf{u}^{-}}{\Delta t}=\frac{q}{2m\gamma^{n+1/2}}(\mathbf{u}^{n+1}+\mathbf{u}^{n})\times\mathbf{B}^{n+1/2}$$

$$\frac{\mathbf{u}^{+}-\mathbf{u}^{-}}{\Delta t}=\frac{q}{2m\gamma^{n+1/2}}(\mathbf{u}^{+}+\mathbf{u}^{-})\times\mathbf{B}^{n+1/2}$$

vxB rotation

$$\frac{\mathbf{u}^+ - \mathbf{u}^-}{\Delta t} = \frac{q}{2m\gamma^{n+1/2}}(\mathbf{u}^+ + \mathbf{u}^-) \times \mathbf{B}^{n+1/2}$$

Taking the dot product with $\mathbf{u}^+ + \mathbf{u}^-$

$$\left(\mathbf{u}^+\right)^2 - \left(\mathbf{u}^-\right)^2 = 0$$

Hence the magnitude of u is conserved and this is a rotation about B.

As a result gamma is constant through this step and we can take

$$\gamma^{n+1/2} = \gamma^- = \gamma(\mathbf{u}^-)$$

Boris algorithm - 2

Apply half the electric field acceleration

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

Update u-minus to u-plus by rotation

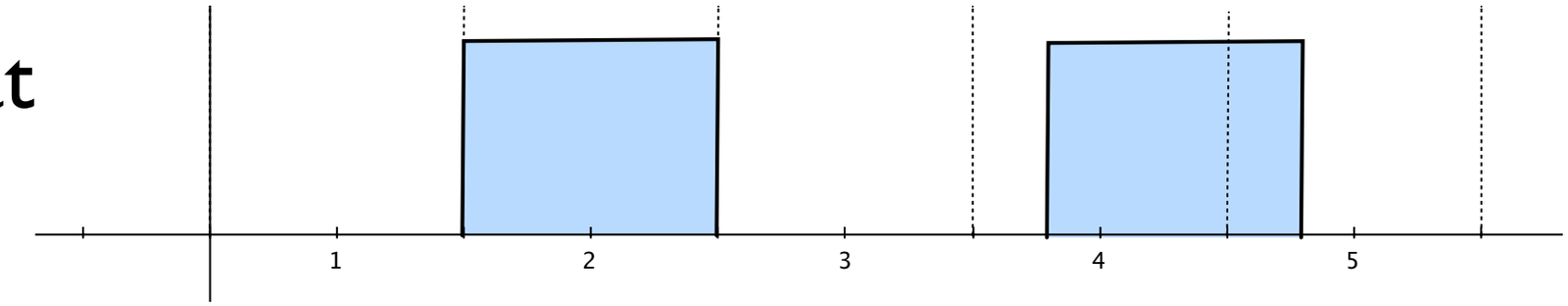
$$\frac{\mathbf{u}^+ - \mathbf{u}^-}{\Delta t} = \frac{q}{2m\gamma^-}(\mathbf{u}^+ + \mathbf{u}^-) \times \mathbf{B}^{n+1/2}$$

Apply half the electric field acceleration

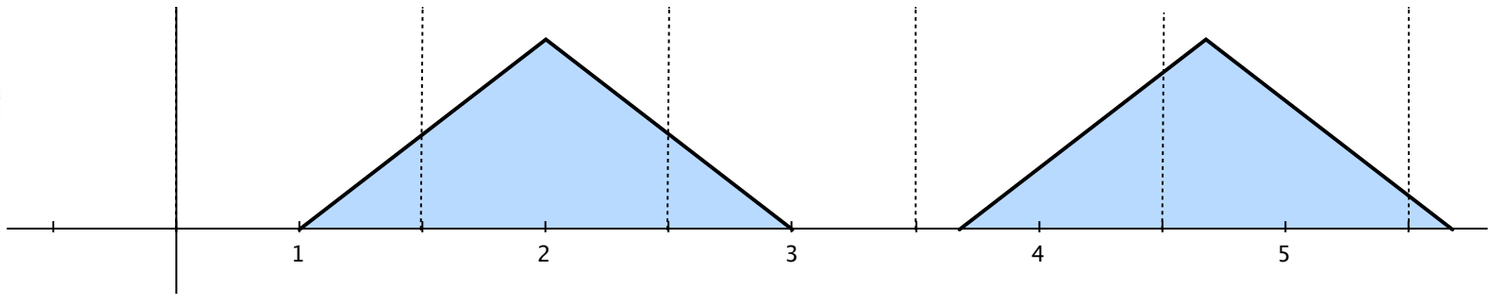
$$\mathbf{u}^{n+1} = \mathbf{u}^+ + \frac{q\Delta t}{2m}\mathbf{E}^{n+1/2}$$

Particle Shape Functions

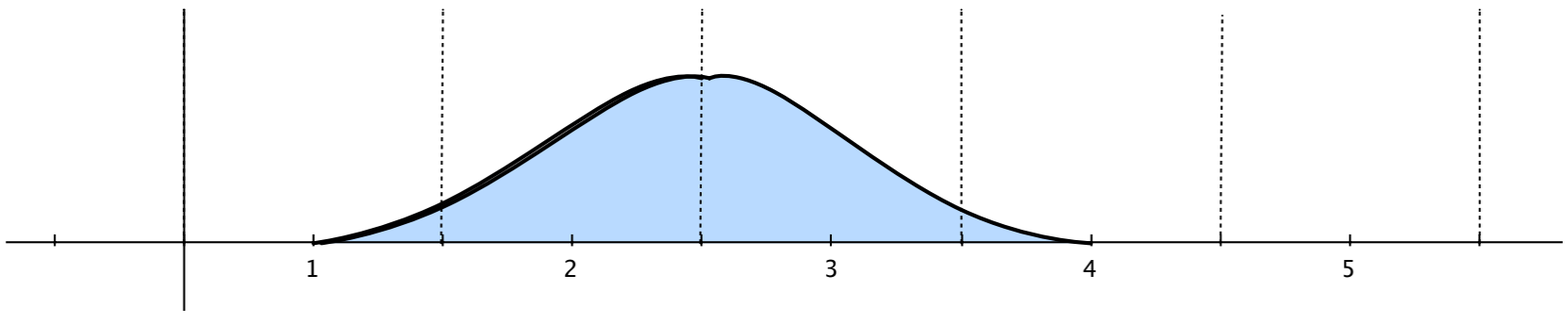
Top hat



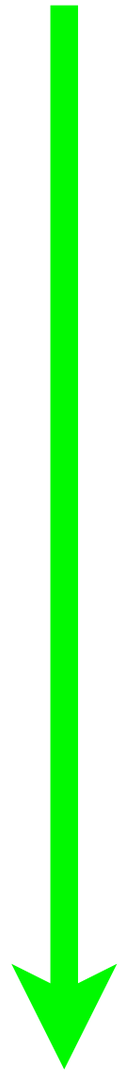
Triangle



Spline

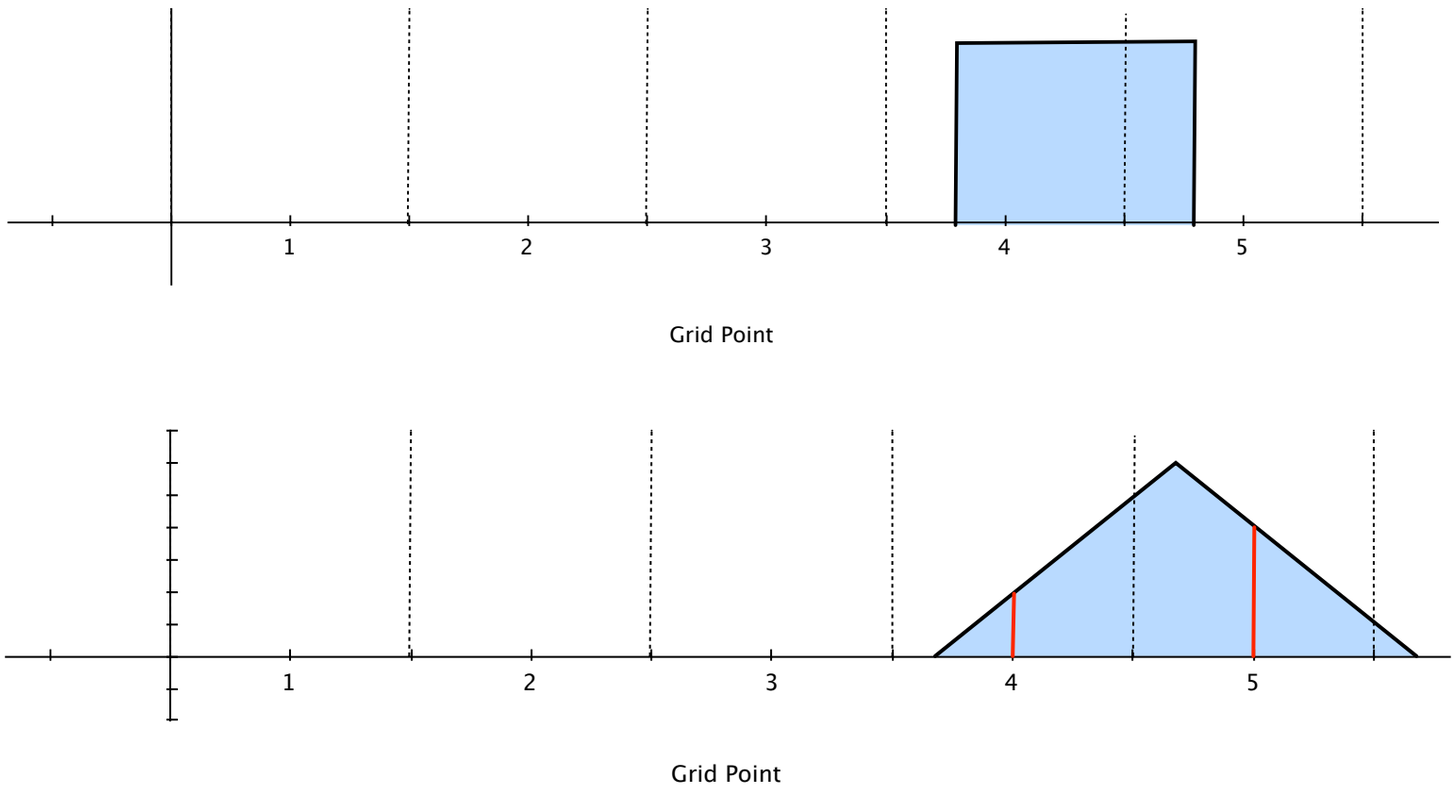


Increasing accuracy



Particle Weight Functions

As a particle moves through the grid its contribution to each grid point is its weight function. This is the integral of the shape function.



Shape

$$\text{Weight} = S(\mathbf{X}_j - \mathbf{r}_i)$$

Weight, i.e. fraction of macro-particle, in cell j at $\mathbf{X}_j$ depends on particle shape and particle position $\mathbf{r}_i$

Birdsall & Langdon call Weight function Shape function

Particle data to grid

EM field solver only needs current density from particles.

$$\mathbf{j}_j^n = \sum_i \mathbf{v}_i^n S(\mathbf{X}_j - \mathbf{r}_i^n)$$

LHS is the current density at time-step *n* at grid point *j*

Could also find the number density through

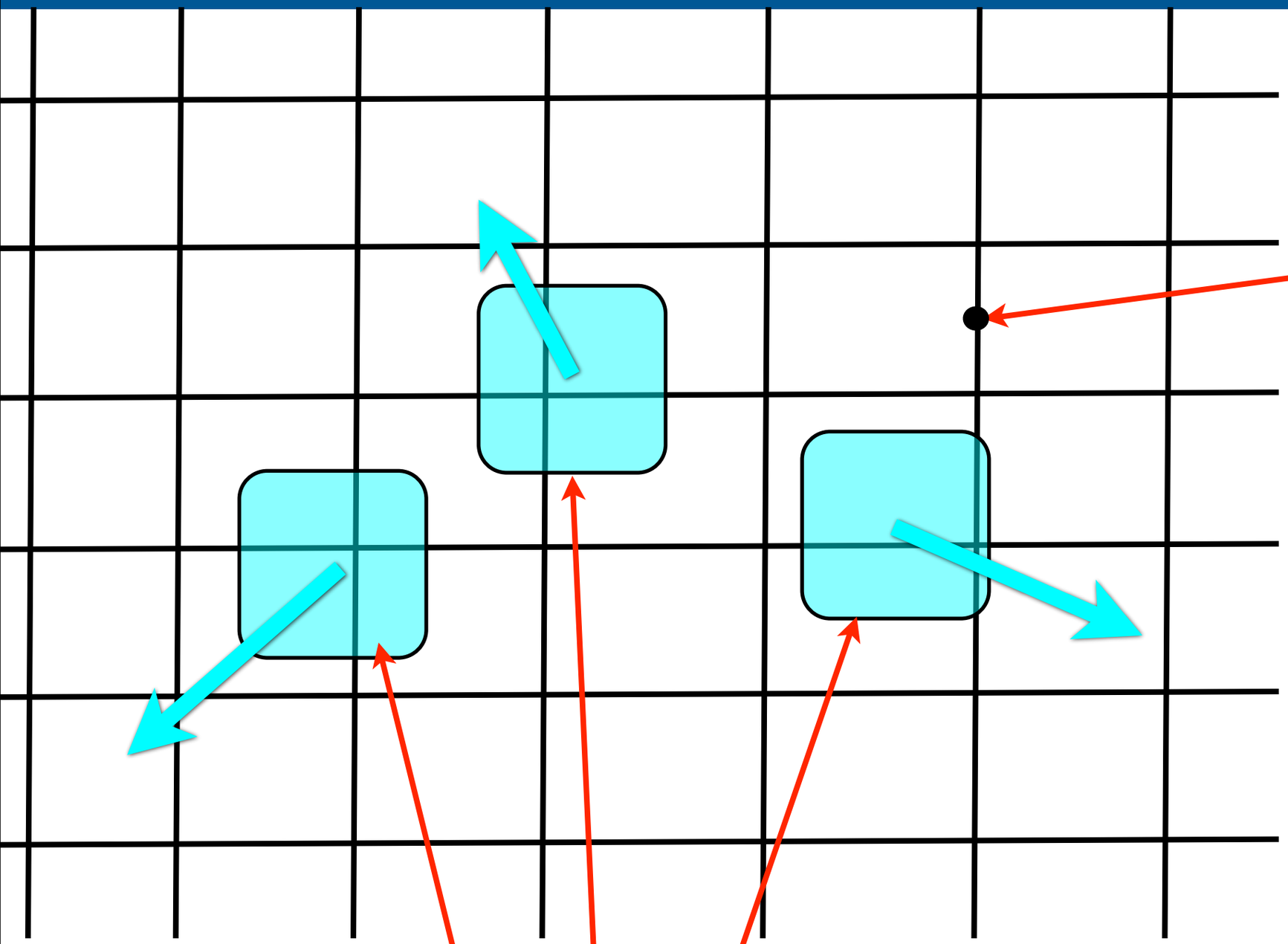
$$n_j^n = \sum_i S(\mathbf{X}_j - \mathbf{r}_i^n)$$

Repeat for each species and then calculate the charge density. Then find that these do not satisfy

$$\frac{\partial \rho}{\partial t} = -\nabla . \mathbf{j}$$

Same as saying that the E-field and charge density do not satisfy Poisson's equation.

Villasenour & Buneman



Current density defined at cell faces

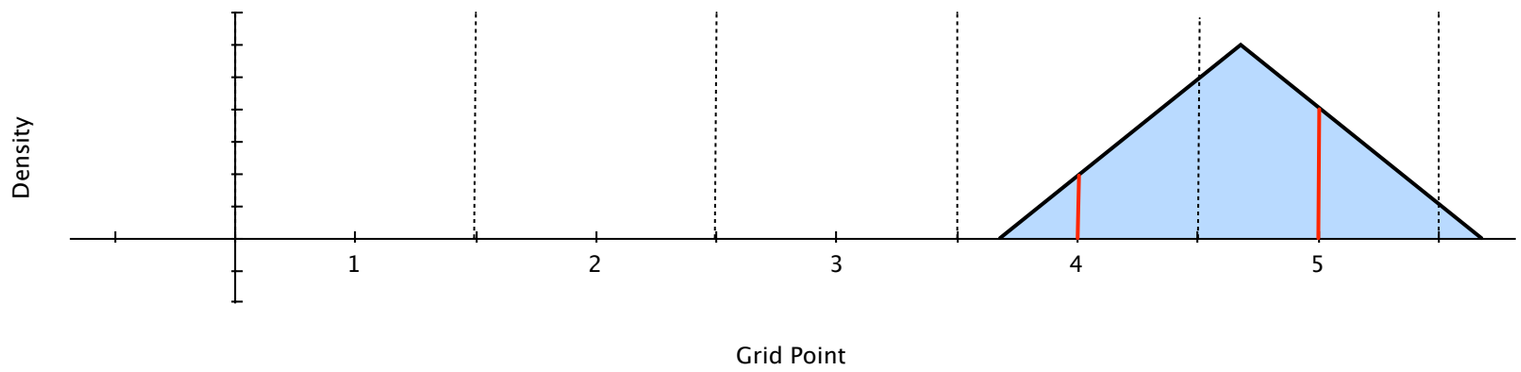
Do not calculate the current density by taking moments of the distribution function. Use particle fluxes instead.

Grid data to particle

To move the particles need to find the E and B fields at the particle position.

Particles not on grid points therefore need to interpolate from the grid to the particle.

Do this using the particle weight functions so that particle to grid and grid to particle are inverses.



$$\mathbf{E}(\mathbf{r}_i^n) = \sum_j \mathbf{E}_j S(\mathbf{X}_j - \mathbf{r}_i^n)$$

Effect of finite number of particles

Finite size, and number, of macro-particles means that the source current density used to update EM fields is noisy.

The real collisionless field should be smooth.

Call the smooth fields $(\mathbf{E}^s, \mathbf{B}^s)$

Fields in PIC code $(\mathbf{E}, \mathbf{B})$

$$\mathbf{E} = \mathbf{E}^s + \delta\mathbf{E}$$

The $\delta\mathbf{E}$ 'noise' field acts to heat the plasma. Usually this is linear in time.

Time-step restrictions - 1

For numerical stability of EM field solver need

$$\Delta t_{em} \leq \frac{\Delta x}{c}$$

Resolving plasma frequency requires

$$\Delta t_p \leq \frac{1}{\omega_{pe}}$$

Stability requires $\Delta t = \min(\Delta t_{em},\ \Delta t_p)$

If the Debye length is resolved then Δt_{em} is the limiter.

If the Debye length is not resolved then Δt_p is the limiter if

$$\Delta x > \frac{c}{\omega_{pe}}$$

Time-step restrictions - 2

If the magnetic field becomes dominant then it is also important to have

$$\Delta t < \frac{1}{\Omega_c}$$

This then resolves the electron gyro-orbits.

In 3D the stability condition for EM waves changes to

$$\Delta t_{em} \leq \frac{1}{c\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)^{1/2}}$$

Real expressions have constants ~1 and factors of π omitted here for simplicity

Summary of PIC code basics

- EM field updated by finite difference scheme.
- Use Yee grid to ensure second order accuracy in space.
- Particles moved by Boris algorithm.
- Only coupling of particles to grid is through current density.
  - Current density found from particle weight function.
- Statistical noise leads to articifial heating.

Practical usage advice

- Test against analytic results where possible
- Increase the number of particles per cell (ppc) to check accuracy
- Increase the number of grid points and ppc to check accuracy
- Try with higher order weight functions to test accuracy
- Run without perturbations to check self-heating
- Basic relativistic EM PIC codes
- Make sure all expected physical lengths and times will be resolved

The *EPOCH* Project

A freely available EM PIC code

Principle Investigators

Prof. A. R. Bell (Oxford)
Prof. R. G. Evans (Imperial)
Prof. T. D. Arber (Warwick)

Developers

Keith Bennett (Warwick)
Chris Brady (Warwick)
Chris Ridgers (York)
Holger Schmidz (RAL)

Based on core algorithm from PSC by Hartmut Ruhl

EPOCH Code and Project

Extendable PIC Open Collaboration (the H is silent!)

EPSRC funded project to develop a UK advanced PIC code.

Core Relativistic EM PIC code is freely available.

Project funds 3 PDRAs to develop the code. These are in Oxford/Imperial/Warwick. Funds for an additional 1.5 years.

Advanced features include:

- Collisions
- Radiation
- Ionisation
- QED effects
- Hybrid schemes

EPOCH Self-Heating

Based on averaging across multiple tests an order of magnitude estimate of the self-heating in EPOCH is

$$\frac{dT_{eV}}{dt_{ps}} = \alpha_H \frac{n_{23}^{3/2} \Delta x_{nm}^2}{n_{ppc}}$$

Temperature in eV, time in ps, density in units of 10^{23} cm^{-3}

Δx_{nm}- typical grid spacing in nm

n_{ppc}- number of particles per cell

α_H - depends on shape function and smoothing

EPOCH Self-Heating - 2

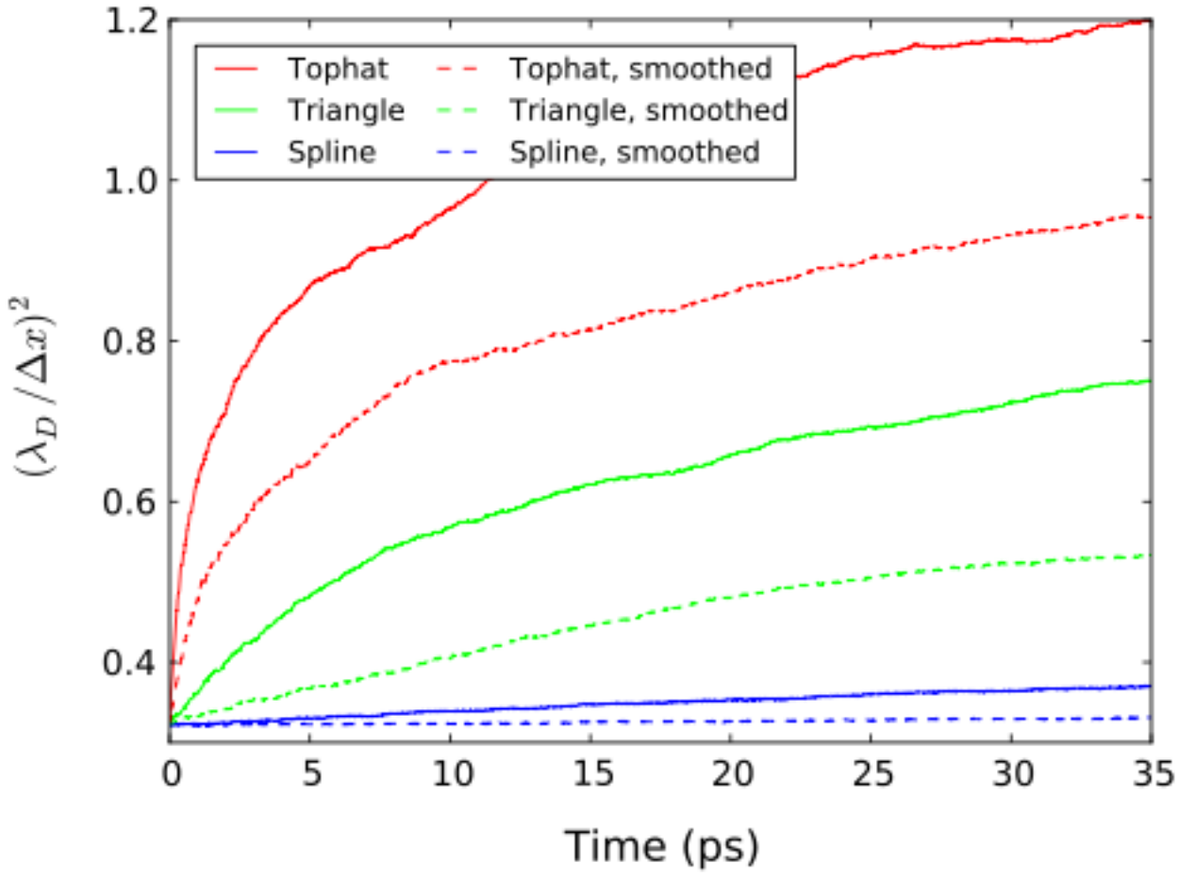


Figure 7: A comparison of self-heating for various particle shape functions.

Shape function	α_H without smoothing	α_H with smoothing
Top-hat	$6,000 \pm 2,000$	$3,000 \pm 300$
Triangle	800 ± 200	200 ± 100
Spline	25 ± 10	2 ± 1

Table 1: Heating coefficients α_H for the different shape functions, with and without additional current smoothing.

Running EPOCH & Visualisation

EPOCH comes with a configurable Makefile.

Runs in MPI - currently tested up to 4096 cores.

Includes readers to input data directly into *IDL*, *VisIt*, *MATLAB* and *gdl*.

Custom *VisIt* reader, and *gdl*, allows users to use freely available visualisation software.

Today's Workshop

Download EPOCH

Run simple test problems

Use visualisation tools

Setup full 2D PIC simulations for laser-plasmas

Run test problems with ionisation, collisions or QED

Understand the Manual