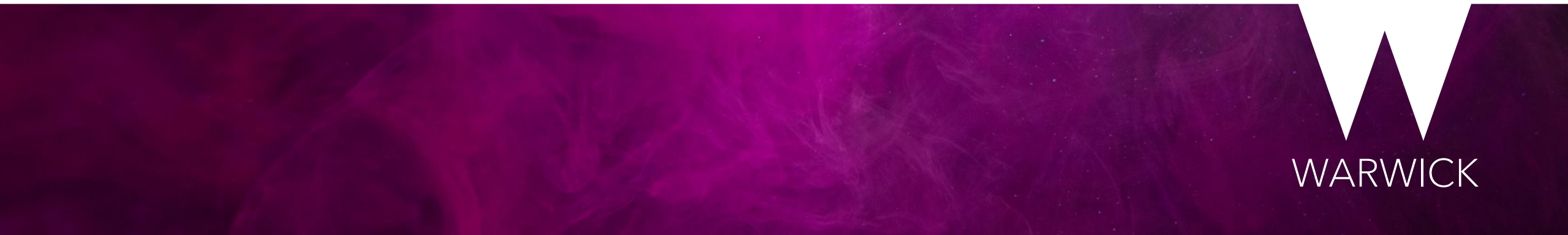
EPOCH workshop

Stuart Morris
28/Mar/2025

Stuart.Morris@warwick.ac.uk
University of Warwick



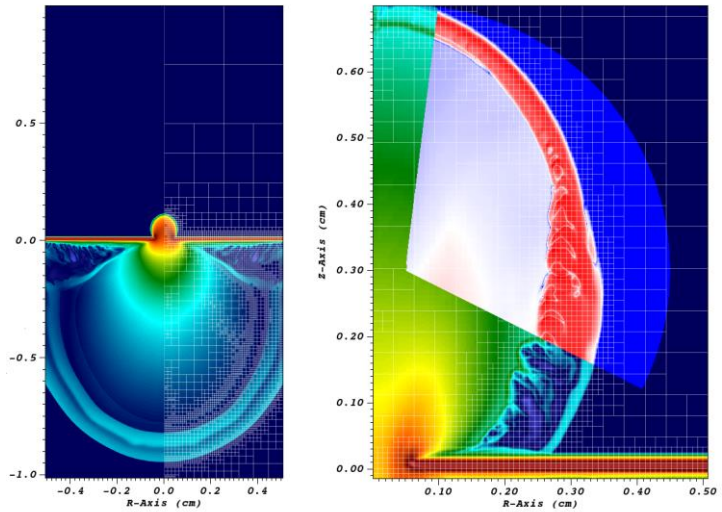
Section 1

Modelling Plasma Physics

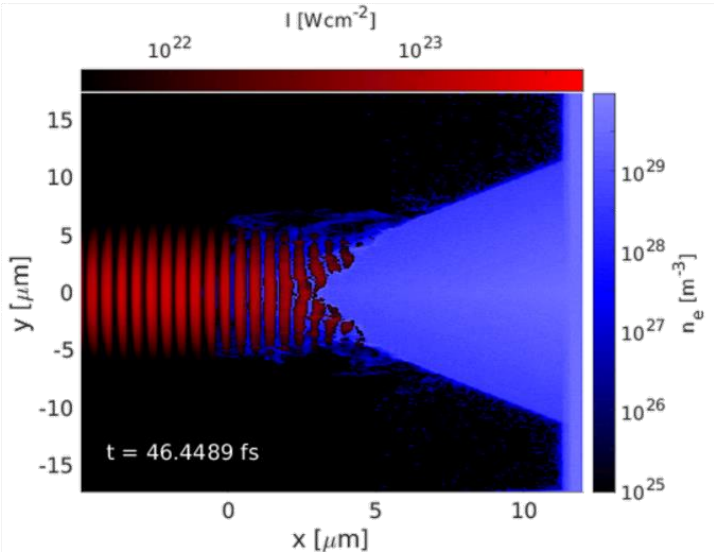
Plasma modelling

- Hydro-dynamic
  - Model fluid properties on grid (n_e, T_e)
  - Update grid with fluid equations
  - Approximate plasma within cell
- Domains:
  - Nanosecond
  - Millimetre

- Kinetic
  - Model particle motion in fields
  - Maxwell's eq., Lorentz force
  - Allows modelling non-equilibrium states
- Domains:
  - Femtosecond-picosecond
  - Micrometre



FLASH output from documentation

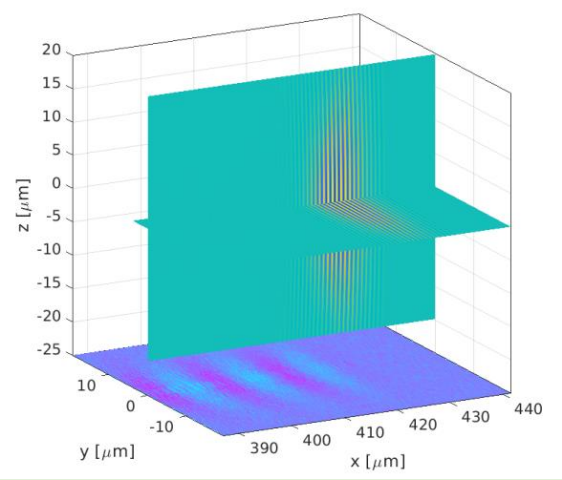


EPOCH laser-solid interaction

Laser-matter applications

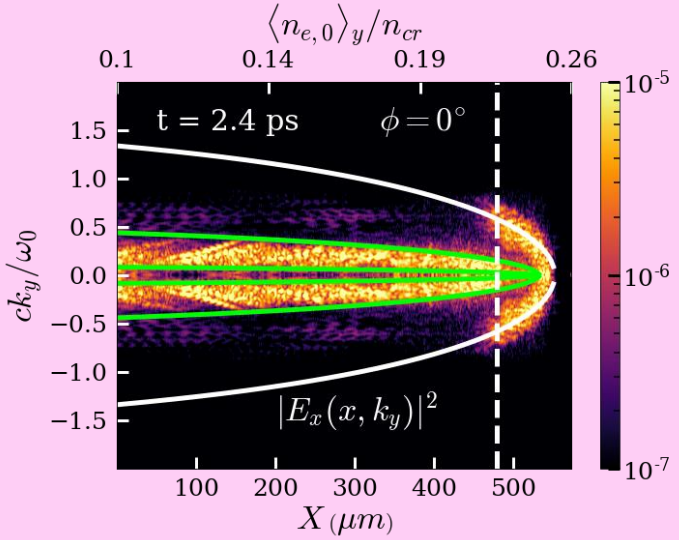
Gas target

- Laser wakefield acceleration
- Betatron radiation source



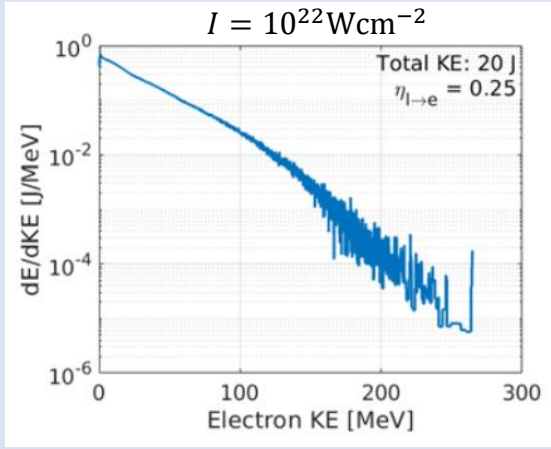
Plasma target

- Laser-plasma instabilities
- ICF research

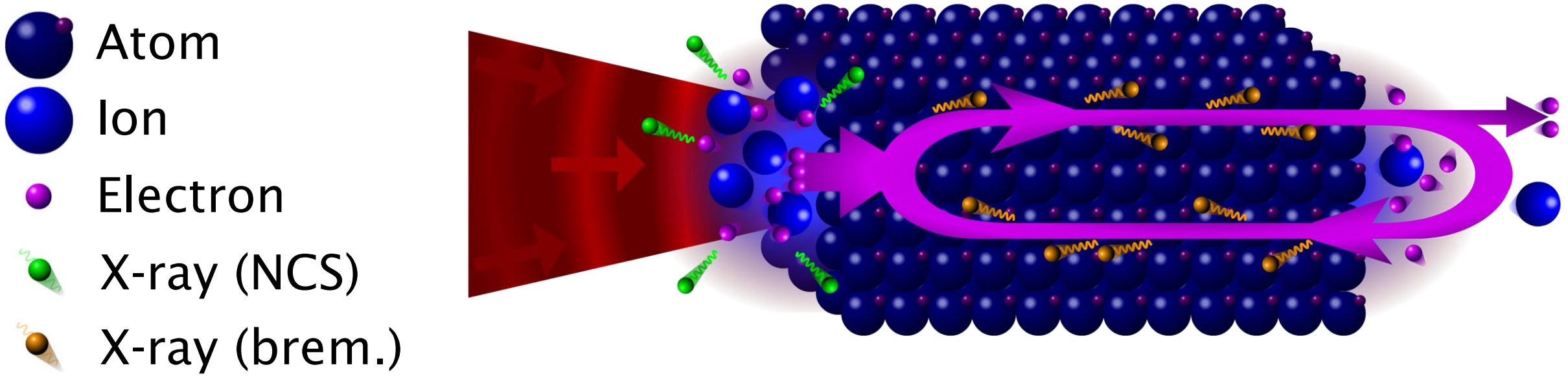


Solid target

- Ion beam source
- Radiation source
- Pair-plasma research



High Power Lasers and targets



High intensity lasers ionise matter upon contact

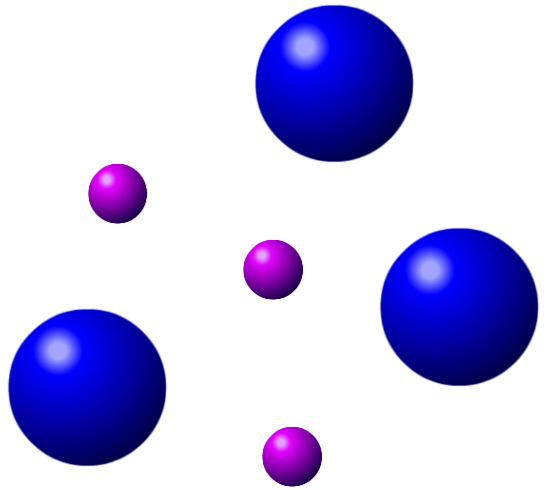
The laser then passes through a plasma

High energy particles created, these may radiate

Most e^- trapped by sheath fields, continuously traversing the target

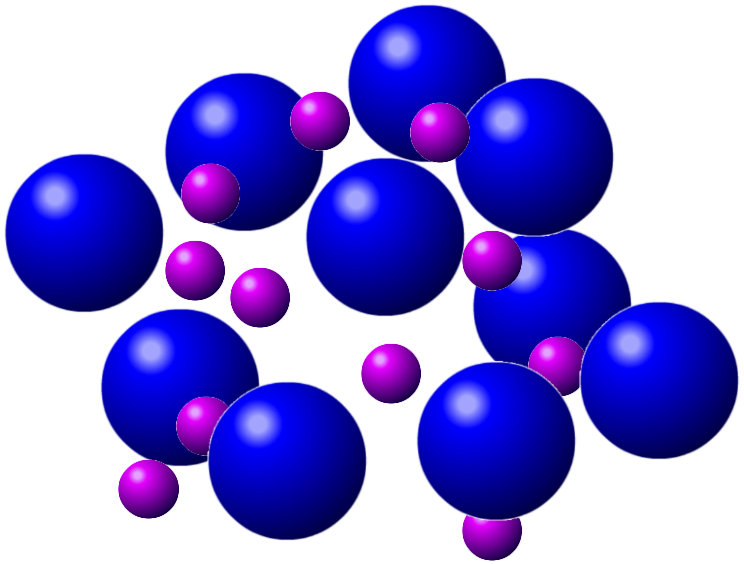
Plasma physics is easy!

- Neglecting ionisation and radiation…
- 5 equations of plasma physics:
  - $\frac{\partial \boldsymbol{E}}{\partial t} = c^2 \nabla \times \boldsymbol{B} - \frac{1}{\epsilon_0} \boldsymbol{J}$
  - $\frac{\partial \boldsymbol{B}}{\partial t} = -\nabla \times \boldsymbol{E}$
  - $\nabla \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0}$
  - $\nabla \cdot \boldsymbol{B} = 0$
  - $\boldsymbol{F} = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B})$



Plasma physics is hard!

- Huge number of particles in the system
- In fully ionised Al
  - $n_i \sim 6 \times 10^{28}\ m^{-3}$
  - $n_e \sim 8 \times 10^{29}\ m^{-3}$
- In a cubic micron:
  - 6×10^{10} ions
  - 8×10^{11} electrons
- Calculating all fields and particle motion is infeasible

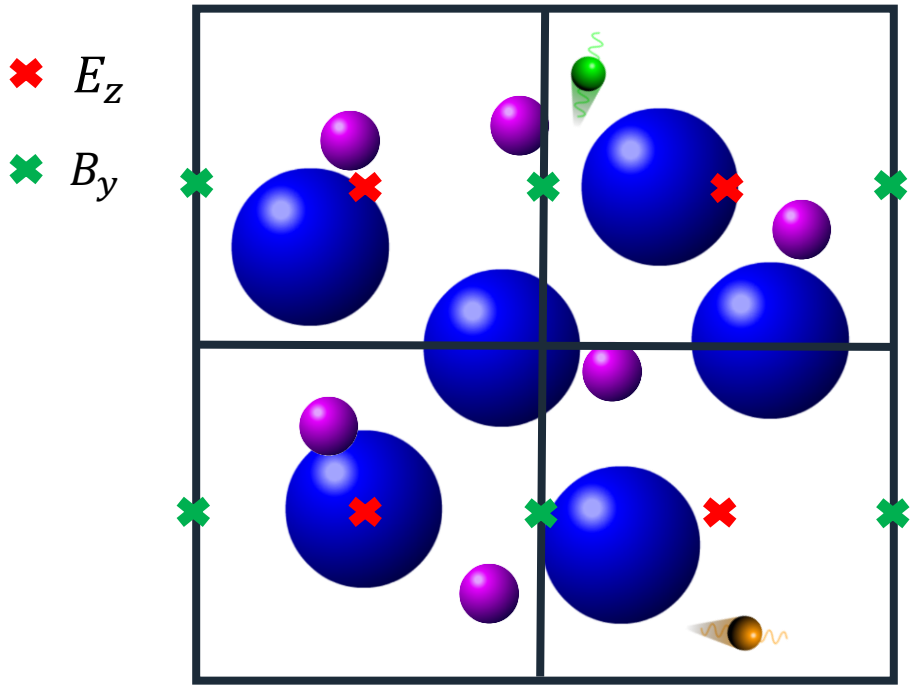


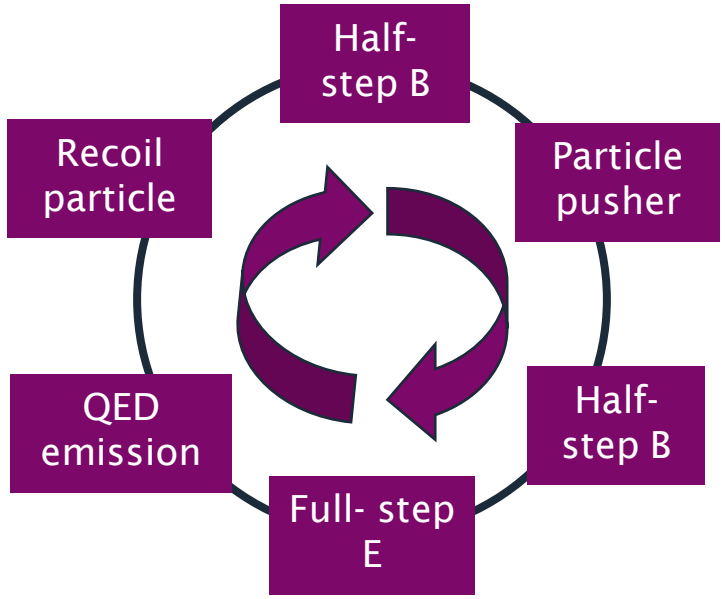
EPOCH

- **E**xtendible
- **P**article-in-cell
- **O**pen source
- **C**ollaborative
- (**H**)

- MPI – parallelisation
- QED physics packages
- Particle collisions
- Bremsstrahlung radiation
- Ionisation routines

PIC loop



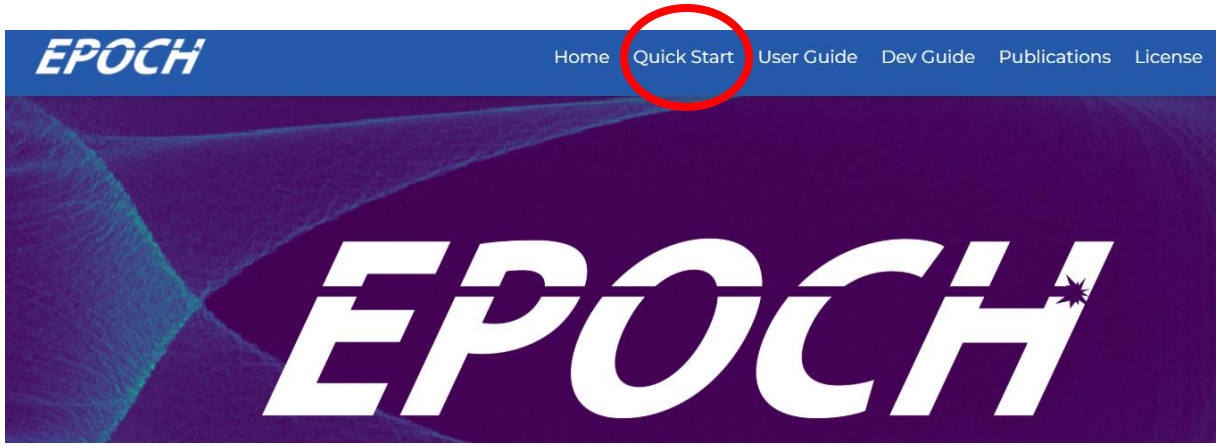


Particle and field properties updated using the particle-in-cell loop

Secondary physics processes calculated from rates using Monte Carlo

Demo: Install EPOCH!

- EPOCH source-code: *https://github.com/Warwick-Plasma/epoch*
- EPOCH documentation: *https://epochpic.github.io/*
  - Installation instructions in Quick Start



Install dependencies (if not done before)

```
sudo apt-get install gfortran
sudo apt-get install openmpi-bin openmpi-common libopenmpi-dev libgtk2.0-dev
```

Clone source-code from GitHub

```
git clone --recursive https://github.com/Warwick-Plasma/epoch.git
```

Build the code

```
make COMPILER=gfortran -j4
```

Input deck

- EPOCH looks for a set-up input.deck file
  - Full format available in User Documentation

- Split into blocks. Basic blocks include:
  - control: setup domain properties and runtime
  - boundaries: determine boundary behaviour
  - species: describe a particle species to simulate
  - output: which properties do we write to file?

- Additional properties:
  - Maths parsing
  - Define constants/functions
  - Comments, marked with a #

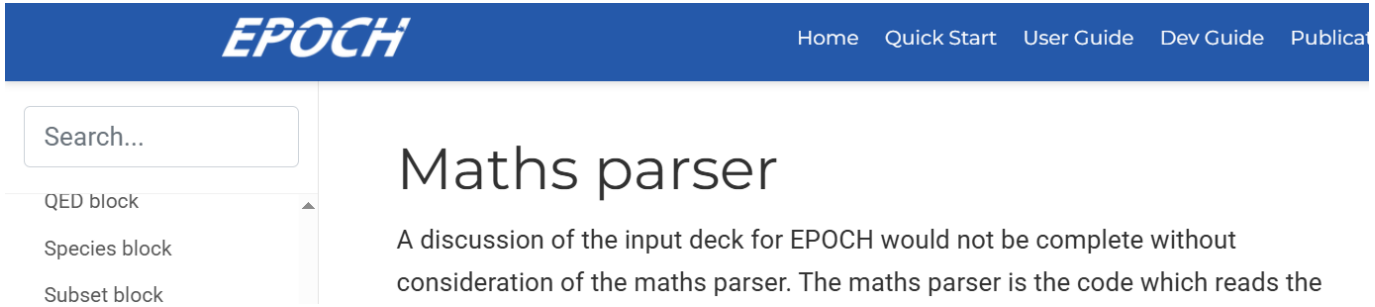
```
begin:control
    # Define 1D domain
    nx = 500       # Cell count
    x_min = 0      # [m]
    x_max = 25e-6  # [m]

    stdout_frequency = 10  # How often to we print status to terminal?

    t_end = 10.0e-15  # How long does ths simulation run for? [s]

    npart = 50 * nx  # Number of macro-particles to load
end:control

begin:boundaries
    bc_x_min = open
    bc_x_max = open
end:boundaries

begin:constant
    # Create a new variable: electron kinetic energy [J]
    electron_ke = 1.0e6 * ev

    # Calculate corresponding drift momentum
    electron_energy = electron_ke + me * c^2
    electron_px = sqrt((electron_energy/c)^2 - (me*c)^2)
end:constant

begin:species
    # Identify species
    name = Electron
    mass = 1.0
    charge = -1.0

    frac = 1.0  # Fraction of macro-particles going to this species

    # Initialise solid block between 10um and 11um
    drift_x = electron_px
    density = if (x gt 10.0e-6, 1e28, 0)
    density = if (x gt 11.0e-6, 0, density(Electron))
end:species

begin:output
    dt_snapshot = 1.0e-15
    number_density = always
end:output
```

Constant block

- constant block
  - Anything defined here can be used after
  - E.g. define “electron_ke” parameter, 1 MeV
  - Use this to derive an electron drift momentum
  - This is used with “drift_x” in the species block

- Available maths parsing tools are given in the documentation
  - Go to user-guide
  - Select maths parser in the left-hand menu

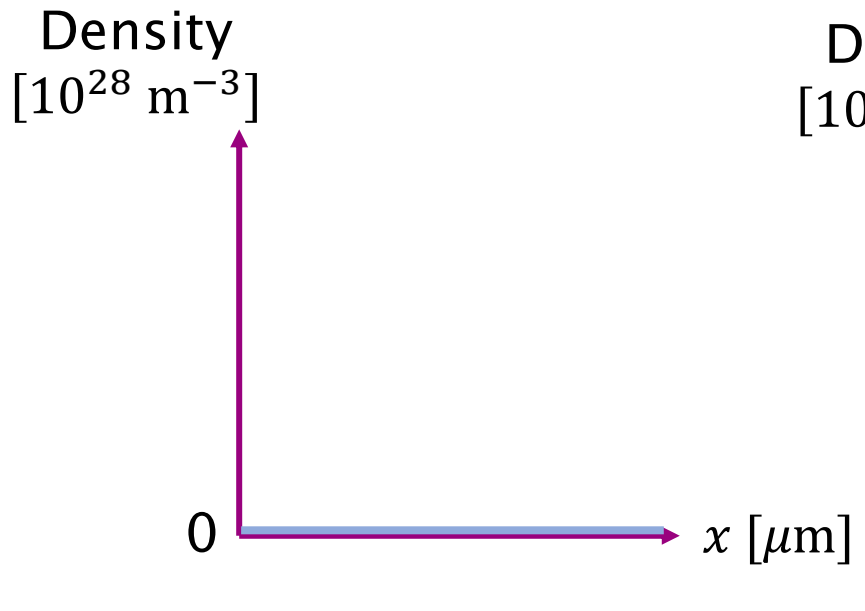
```
begin:constant
    # Create a new variable: electron kinetic energy [J]
    electron_ke = 1.0e6 * ev

    # Calculate corresponding drift momentum
    electron_energy = electron_ke + me * c^2
    electron_px = sqrt((electron_energy/c)^2 - (me*c)^2)
end:constant

begin:species
    # Identify species
    name = Electron
    mass = 1.0
    charge = -1.0

    frac = 1.0  # Fraction of macro-particles going to this species

    # Initialise solid block between 10um and 11um
    drift_x = electron_px
    density = if (x gt 10.0e-6, 1e28, 0)
    density = if (x gt 11.0e-6, 0, density(Electron))
end:species
```

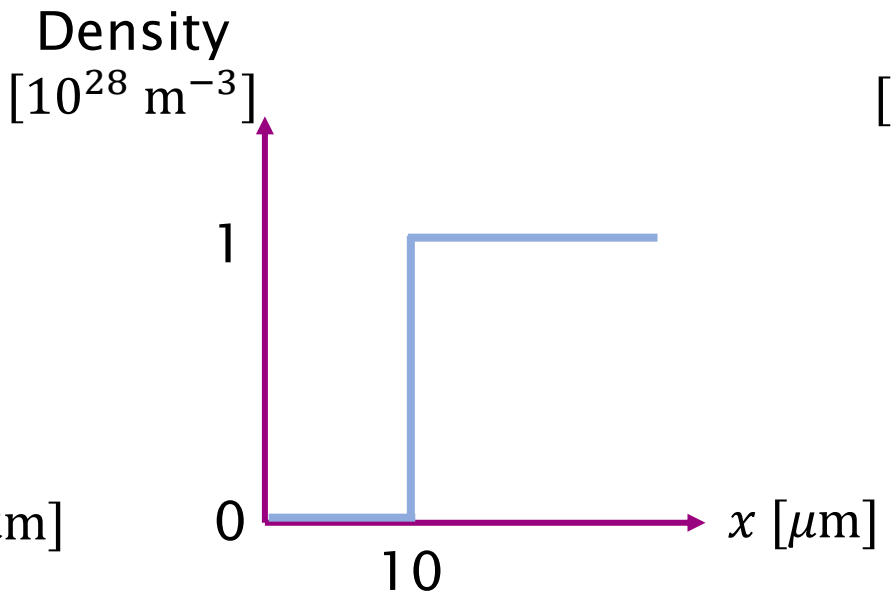


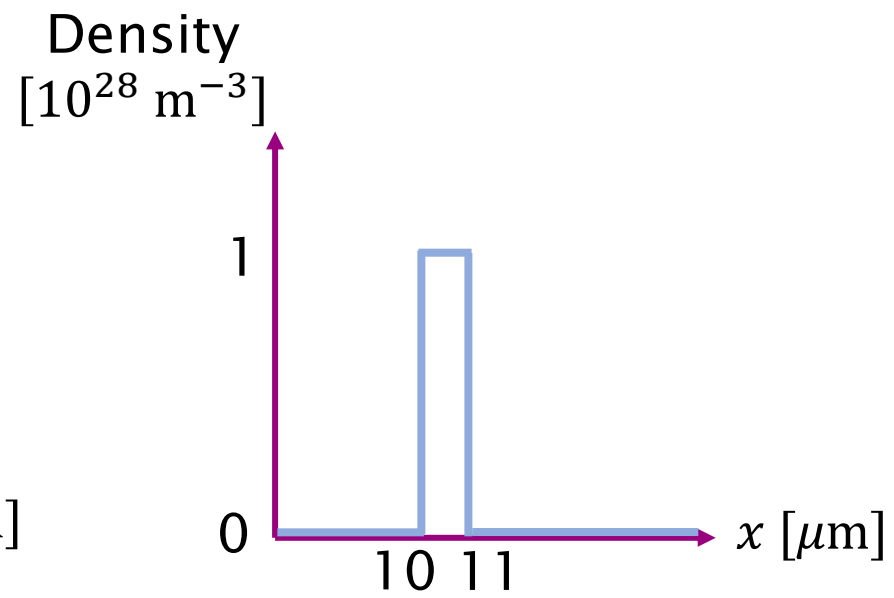
Building on functions

- Can initialise complex geometries by building lists of functions
  - EPOCH if statement has condition-if-else structure
  - Initially, the density of a species is 0 in every cell

```
density = if (x gt 10.0e-6, 1e28, 0)
density = if (x gt 11.0e-6, 0, density(Electron))
```







Run the input.deck

- Running instructions are on the Quick-Start page, EPOCH documentation website:

> To run any EPOCH code, you must first navigate to the sub-directory of the code in a Linux terminal, such that the `ls` command shows the `bin` sub-directory. EPOCH expects you to be here when looking up physics tables. For example, the 2d code can be run by entering `epoch2d` and running:
>
> ```
> mpirun -np 4 ./bin/epoch2d
> ```
>
> You will then be prompted to enter the path to the directory containing your `input.deck` file. This step may be skipped by piping in the path directly into the run command, using:
>
> ```
> mpirun -np 4 ./bin/epoch2d <<< /path/to/input/
> ```

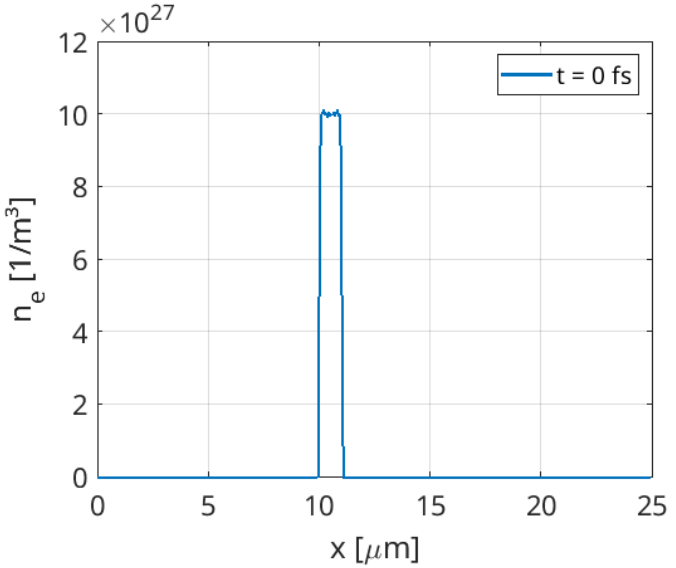
- Initialising a 1 MeV electron bunch, 1 micron size, $10^{28}\ \mathrm{m}^{-3}$ density.
  - What do we think it will do?

Reading the data

- EPOCH has created some special “Self-Describing-Format” SDF files.
  - These are written in binary for memory efficiency
  - Can’t be read directly, need an SDF reading package

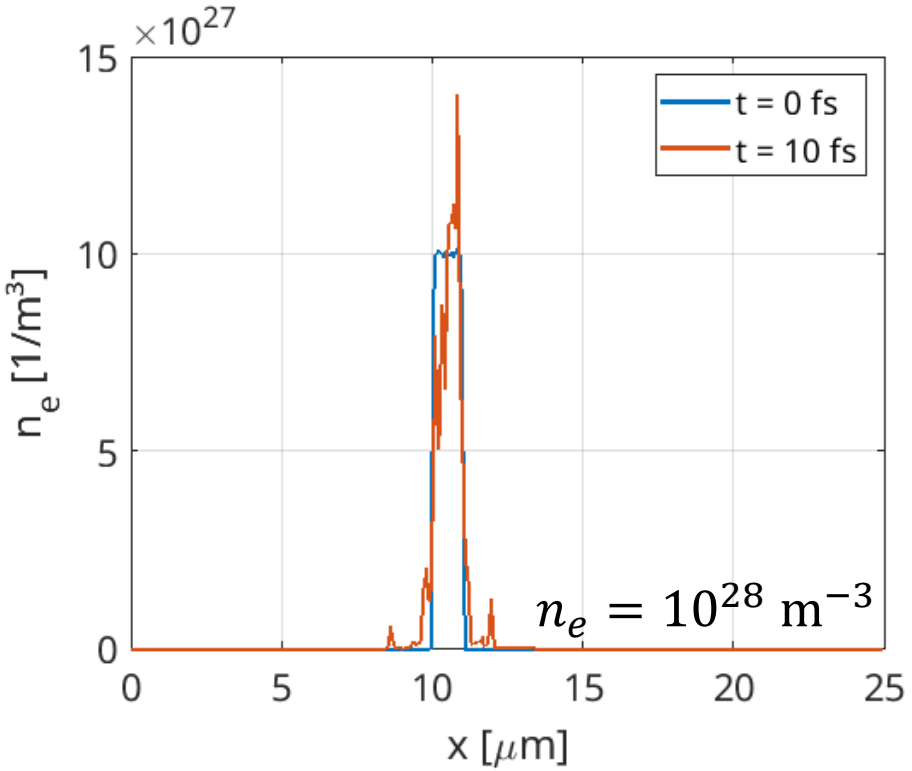
- A few options: Visit, MATLAB, Python…
  - This workshop will use sdf-xarray, a York-developed SDF reader for Python
  - https://sdf-xarray.readthedocs.io/en/latest/key_functionality.html

Now presenting: Joel Adams!



Weird result

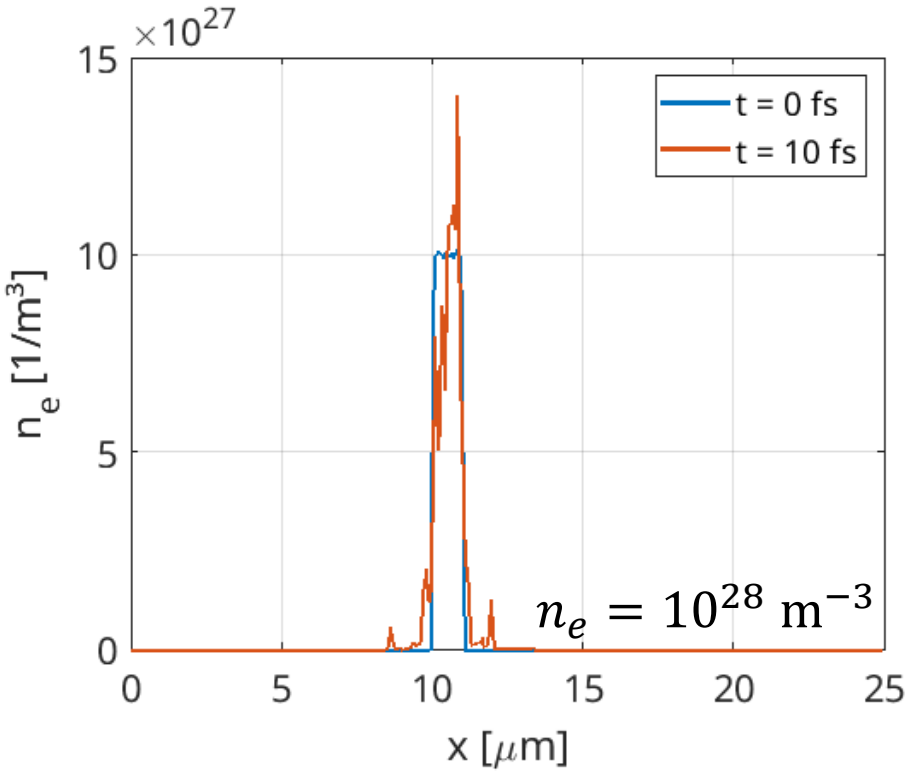
- Our 1 MeV electron bunch hasn't moved – why?

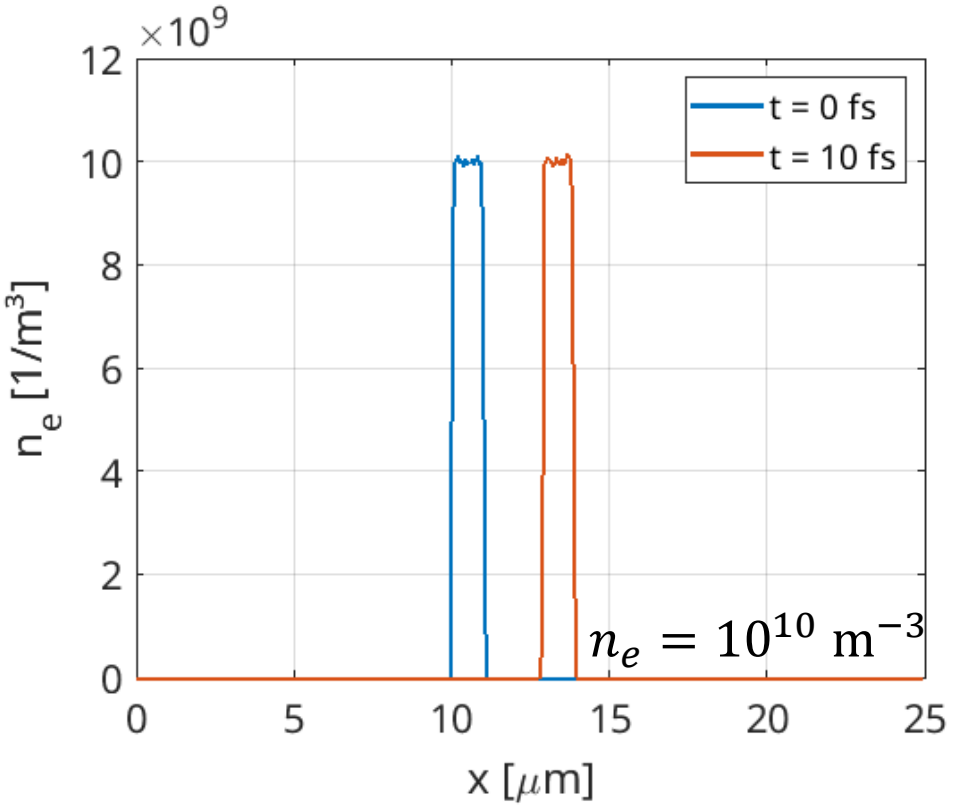


- Try changing the bunch density in the input.deck file to $10^{10}\ \mathrm{m}^{-3}$

Weird result

- Our 1 MeV electron bunch hasn't moved – why?
  - A lower density bunch actually does move?





Section 2

Field solver

Discretisation

- Maxwell's equations:
  - $\frac{\partial \boldsymbol{E}}{\partial t} = c^2 \nabla \times \boldsymbol{B} - \frac{1}{\epsilon_0}\boldsymbol{J}$
  - $\frac{\partial \boldsymbol{B}}{\partial t} = -\nabla \times \boldsymbol{E}$

  } Time evolution

  - $\nabla \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0}$
  - $\nabla \cdot \boldsymbol{B} = 0$

  } Boundary conditions

First order method:

$$\frac{\partial f}{\partial t} \approx \frac{\Delta f}{\Delta t}$$

$$\Delta \mathbf{E} = \left(c^2 \nabla \times \boldsymbol{B} - \frac{1}{\epsilon_0}\boldsymbol{J} \right) \Delta t$$

$$\Delta \boldsymbol{B} = -(\nabla \times \boldsymbol{E}) \Delta t$$

- For $\boldsymbol{E}$ update:

$$E_x^{n+1} = E_x^n + \Delta t \left(c^2 \left(\frac{\Delta B_z^n}{\Delta y} - \frac{\Delta B_y^n}{\Delta z} \right) - \frac{1}{\epsilon_0} J_x^n \right)$$

$$E_y^{n+1} = E_y^n + \Delta t \left(c^2 \left(\frac{\Delta B_x^n}{\Delta z} - \frac{\Delta B_z^n}{\Delta x} \right) - \frac{1}{\epsilon_0} J_y^n \right)$$

$$E_z^{n+1} = E_z^n + \Delta t \left(c^2 \left(\frac{\Delta B_y^n}{\Delta x} - \frac{\Delta B_x^n}{\Delta y} \right) - \frac{1}{\epsilon_0} J_z^n \right)$$

- For $\boldsymbol{B}$ update:

$$B_x^{n+1} = B_x^n - \Delta t \left(\frac{\Delta E_z^n}{\Delta y} - \frac{\Delta E_y^n}{\Delta z} \right)$$

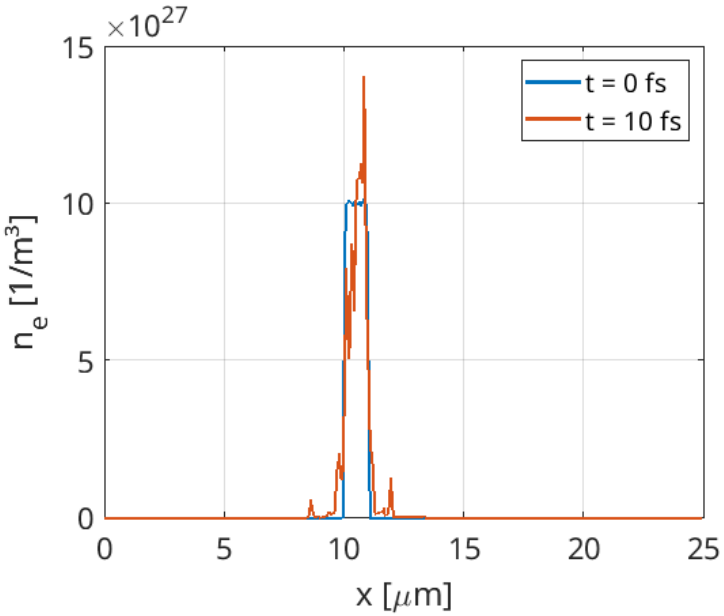
$$B_y^{n+1} = B_y^n - \Delta t \left(\frac{\Delta E_x^n}{\Delta z} - \frac{\Delta E_z^n}{\Delta x} \right)$$

$$B_z^{n+1} = B_z^n - \Delta t \left(\frac{\Delta E_y^n}{\Delta x} - \frac{\Delta E_x^n}{\Delta y} \right)$$

Why didn't the 1 MeV bunch move?

- Default initial conditions in EPOCH:
  - $\boldsymbol{E} = 0$
  - $\boldsymbol{B} = 0$

- Assumes every cell starts neutral, field solver only sees $\boldsymbol{J}$
  - In example: EPOCH assumes dense, neutralising ion block present
  - This prevents e^- bunch from escape



- For $\boldsymbol{E}$ update:

$$E_x^{n+1} = E_x^n + \Delta t \left(c^2 \left(\frac{\Delta B_z^n}{\Delta y} - \frac{\Delta B_y^n}{\Delta z} \right) - \frac{1}{\epsilon_0} J_x^n \right)$$

$$E_y^{n+1} = E_y^n + \Delta t \left(c^2 \left(\frac{\Delta B_x^n}{\Delta z} - \frac{\Delta B_z^n}{\Delta x} \right) - \frac{1}{\epsilon_0} J_y^n \right)$$

$$E_z^{n+1} = E_z^n + \Delta t \left(c^2 \left(\frac{\Delta B_y^n}{\Delta x} - \frac{\Delta B_x^n}{\Delta y} \right) - \frac{1}{\epsilon_0} J_z^n \right)$$

- For $\boldsymbol{B}$ update:

$$B_x^{n+1} = B_x^n - \Delta t \left(\frac{\Delta E_z^n}{\Delta y} - \frac{\Delta E_y^n}{\Delta z} \right)$$

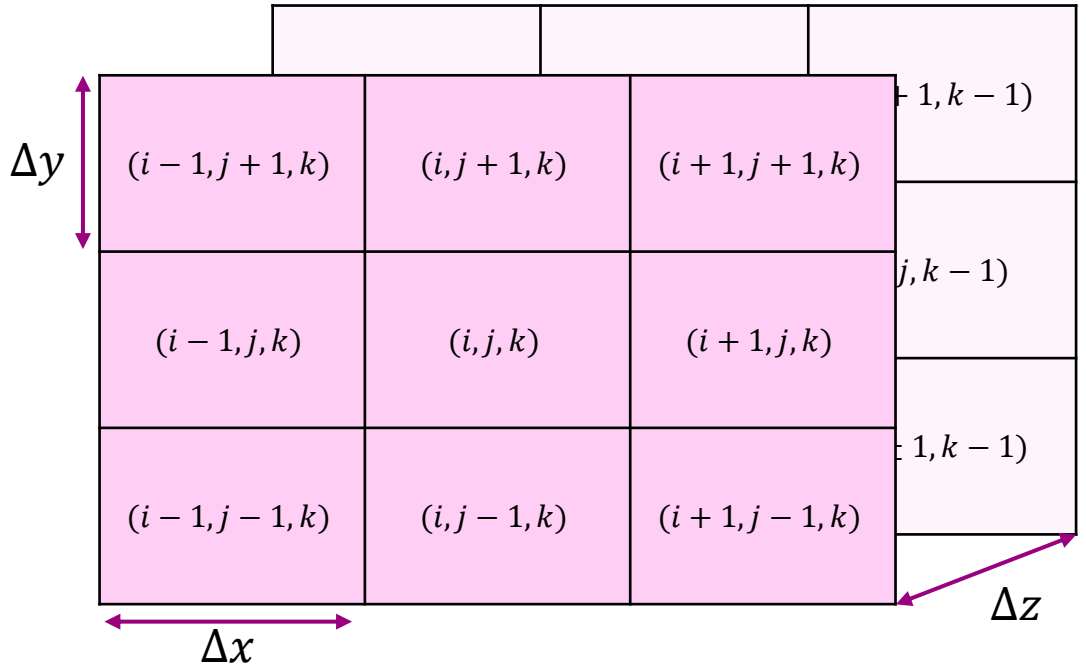
$$B_y^{n+1} = B_y^n - \Delta t \left(\frac{\Delta E_x^n}{\Delta z} - \frac{\Delta E_z^n}{\Delta x} \right)$$

$$B_z^{n+1} = B_z^n - \Delta t \left(\frac{\Delta E_y^n}{\Delta x} - \frac{\Delta E_x^n}{\Delta y} \right)$$

Basic implementation

- For spatial gradients, you could consider neighbouring cells:

$$\left(\frac{\Delta f}{\Delta x}\right)_{i,j,k} \approx \frac{f(i-1,j,k) - f(i+1,j,k)}{2\Delta y}$$
$$\left(\frac{\Delta f}{\Delta y}\right)_{i,j,k} \approx \frac{f(i,j+1,k) - f(i,j-1,k)}{2\Delta y}$$
$$\left(\frac{\Delta f}{\Delta z}\right)_{i,j,k} \approx \frac{f(i,j+1,k) - f(i,j-1,k)}{2\Delta z}$$



$$E_x^{n+1}(i,j,k) = E_x^n(i,j,k) + \Delta t\left(c^2\left(\frac{B_z^n(i,j+1,k) - B_z^n(i,j-1,k)}{2\Delta y} - \frac{B_y^n(i,j,k+1) - B_y^n(i,j,k-1)}{2\Delta z}\right) - \frac{1}{\epsilon_0}J_x^n(i,j,k)\right)$$

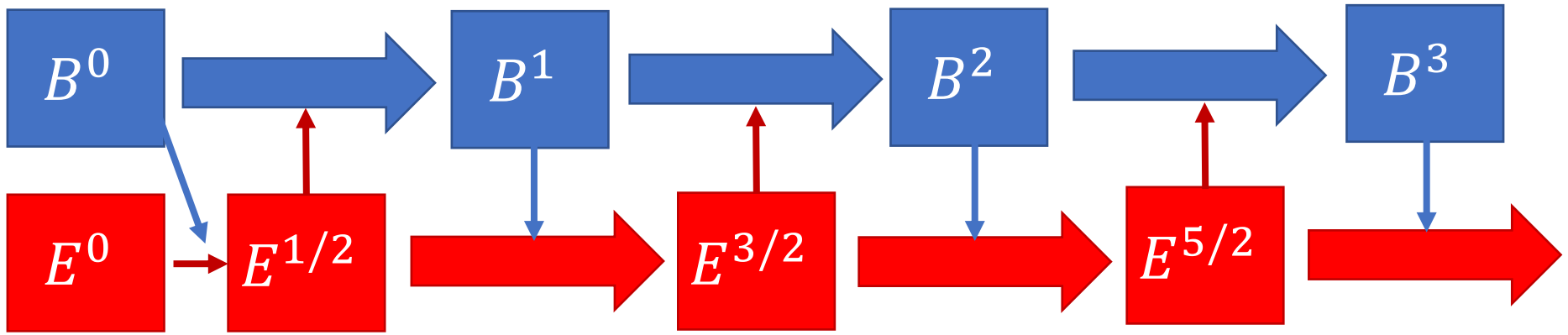
Leap-frog solver

- Second-order solver, use time-centred fields:

$$E_x^{n+1/2} = E_x^{n-1/2} + \Delta t \left(c^2 \left(\frac{\Delta B_z^n}{\Delta y} - \frac{\Delta B_y^n}{\Delta z} \right) - \frac{1}{\epsilon_0} J_x^n \right)$$

$$B_x^{n+1} = B_x^n - \Delta t \left(\frac{\Delta E_z^{n+1/2}}{\Delta y} - \frac{\Delta E_y^{n+1/2}}{\Delta z} \right)$$

- Time-step Δt restricted by CFL condition: $c\Delta t < \min(\Delta x, \Delta y, \Delta z)$
- Create half-time-step displacement using first-order update for the first step



Yee-stagger

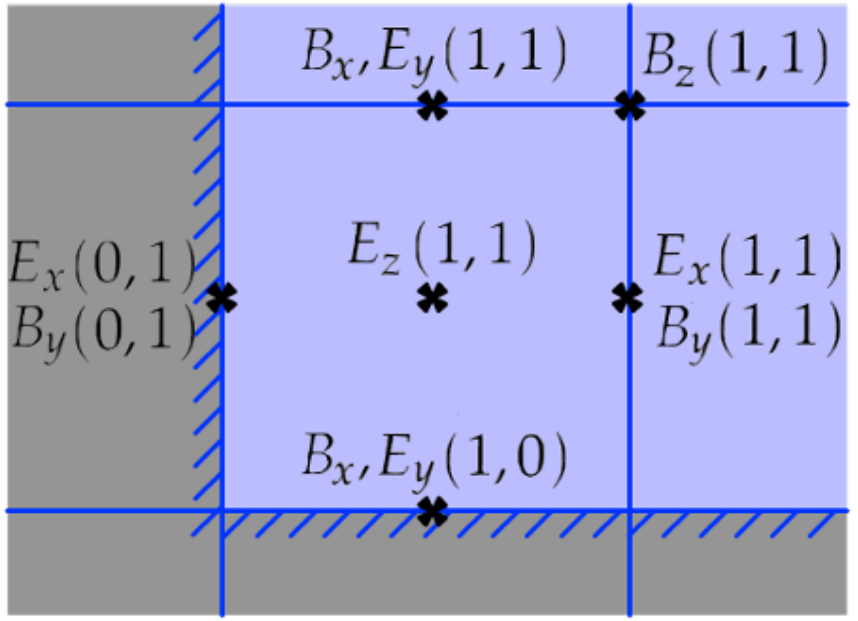
- Do we really need both neighbours for the curl? No!
  - Yee stagger: evaluate fields at different points in each cell

- Off-set rules:
  - E: shift up a half-cell in the component direction
  - B: shift up a half-cell in every direction except component direction
  - 1D/2D: ignore shift in omitted dimensions
  - J: Evaluate current at the same point as E

- New gradients:

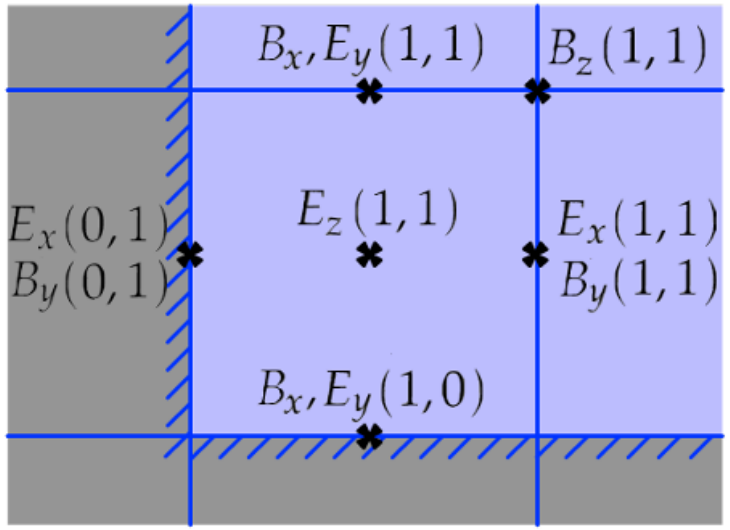
$$\left(\frac{\Delta B_y}{\Delta x}\right)_{i,j,k} = \frac{B_y(i,j,k) - B_y(i-1,j,k)}{\Delta x}$$

$$\left(\frac{\Delta B_x}{\Delta y}\right)_{i,j,k} = \frac{B_x(i,j,k) - B_x(i,j-1,k)}{\Delta y}$$



Boundary conditions

- EPOCH simulation domain is surrounded by ghost-cells
  - Used to average E, B fields on near-edge particles
- Two basic field boundaries:

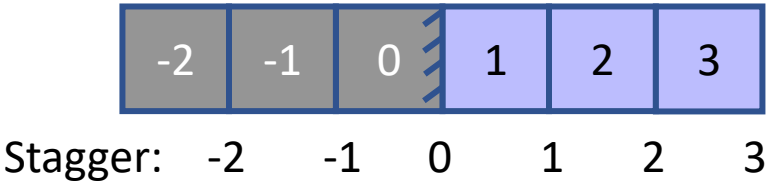


Reflect

- Staggered fields:
  - $f(0) = 0$
  - $f(-1) = f(1)$
  - $f(-2) = f(2)$
- Cell-centre fields:
  - $f(0) = f(1)$
  - $f(-1) = f(2)$
  - $f(-2) = f(3)$

Periodic

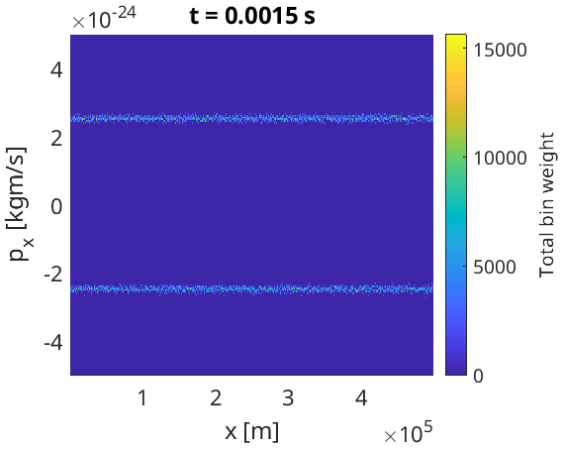
- Let n_x be the last cell calculated normally
- All fields:
  - $f(0) = f(n_x)$
  - $f(-1) = f(n_x - 1)$
  - $f(-2) = f(n_x - 2)$
  - $f(n_x + 1) = f(1)$

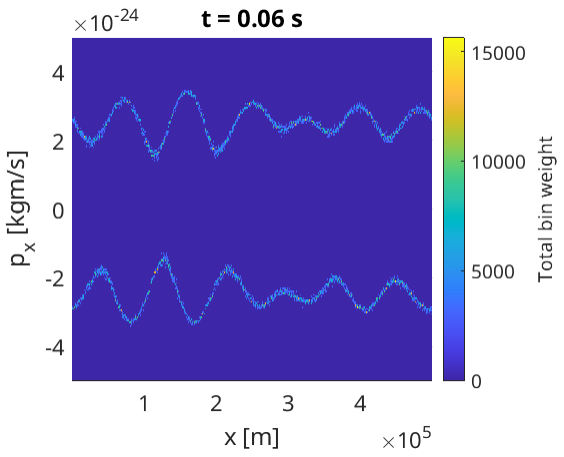


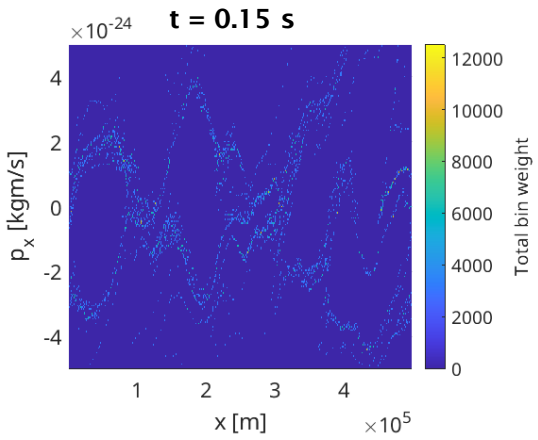
Demo: two-stream instability

- Two plasma beams pass through each other in a periodic domain
- Noise in currents generates fields in field solver
- Phase space of particles start to “swirl”
- EPOCH can directly output the phase-space of plasma particles, give it a try!

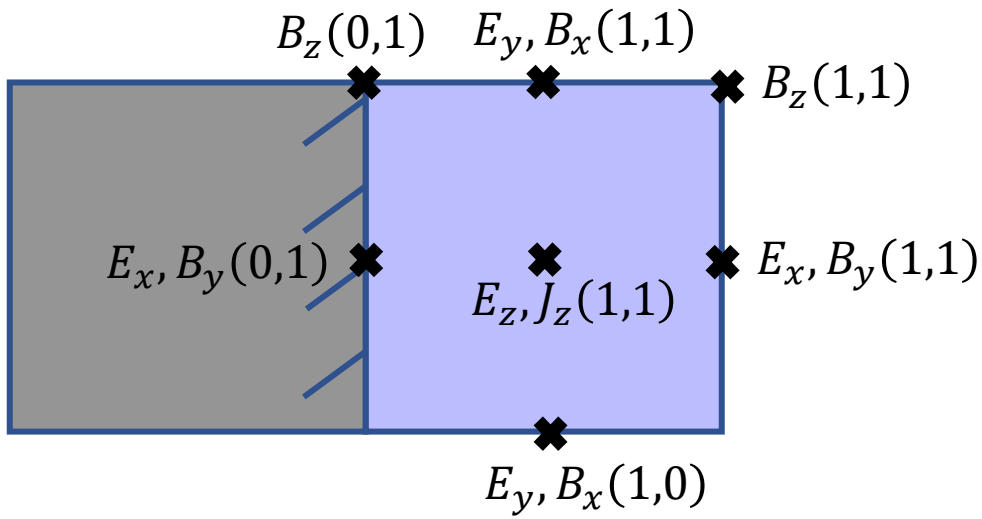
```
begin:dist_fn
  name = x_px
  ndims = 2

  direction1 = dir_x
  direction2 = dir_px

  # Range is ignored for spatial coordinates
  range1 = (1, 1)
  range2 = (-5e-24, 5e-24)

  # Resolution is ignored for spatial coordinates
  resolution1 = 1
  resolution2 = 200

  include_species:Left
  include_species:Right
end:dist_fn
```







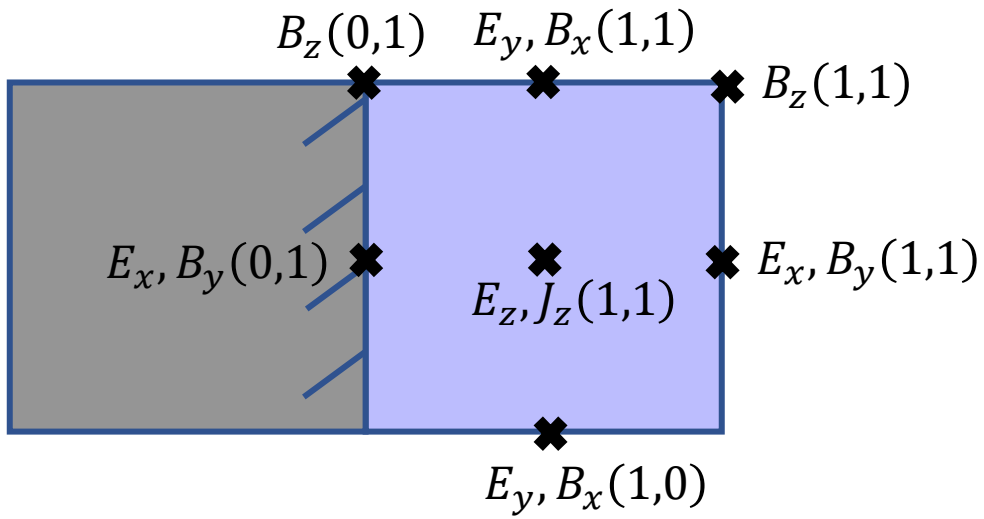
Laser boundary



- Special boundaries are used for lasers
  - Can do two-sided gradient for $E_x(0,1)$, but not $B_y(0,1)$ or $B_z(0,1)$
  - Consider $B_y(0,1)$ condition
- Firstly, consider the time-gradient:
  - $\left(\frac{\partial E_z}{\partial t}\right)^n \approx \frac{1}{\Delta t}\left(E_z^{n+1/2} - E_z^{n-1/2}\right)$
- Time averaging condition:
  - $\boldsymbol{E}^n \approx \frac{1}{2}\left(\boldsymbol{E}^{n+1/2} + \boldsymbol{E}^{n-1/2}\right)$
  - $\boldsymbol{E}^{n+1/2} = 2\boldsymbol{E}^n - \boldsymbol{E}^{n-1/2}$
- Hence:
  - $\left(\frac{\partial E_z}{\partial t}\right)^n \approx \frac{1}{\Delta t}\left(2E_z^n - 2E_z^{n-1/2}\right)$
- Field solver doesn't have E_z^n though!
  - But we do have $\boldsymbol{B}^n$ …
- Assume inward plane waves, so $E_z = -cB_y$
- Let $S = E_z = \frac{1}{2}\left(E_z - cB_y\right)$, so:
  - $\left(\frac{\partial E_z}{\partial t}\right)^n = \frac{1}{\Delta t}\left(4S^n + 2B_y^n - 2E_z^{n-1/2}\right)$

Laser boundary – part 2

- So we have:
  - $\left(\frac{\partial E_z}{\partial t}\right)^n = \frac{1}{\Delta t}\left(4S^n + 2B_y^n - 2E_z^{n-1/2}\right)$
- Ampere-Maxwell:
  - $\left(\frac{\partial E_z}{\partial t}\right)^n = c^2 \frac{(B_y)_{1,1}^n - (B_y)_{0,1}^n}{\Delta x} - \frac{1}{\epsilon_0}(J_z)_{1,1}^n$



Hint: $\frac{\partial E_z}{\partial t}$ is evaluated at the $J_z(1,1)$ position.
There's no B_y here, so average:

$$2B_y^n \approx (B_y)_{1,1}^n + (B_y)_{0,1}^n$$

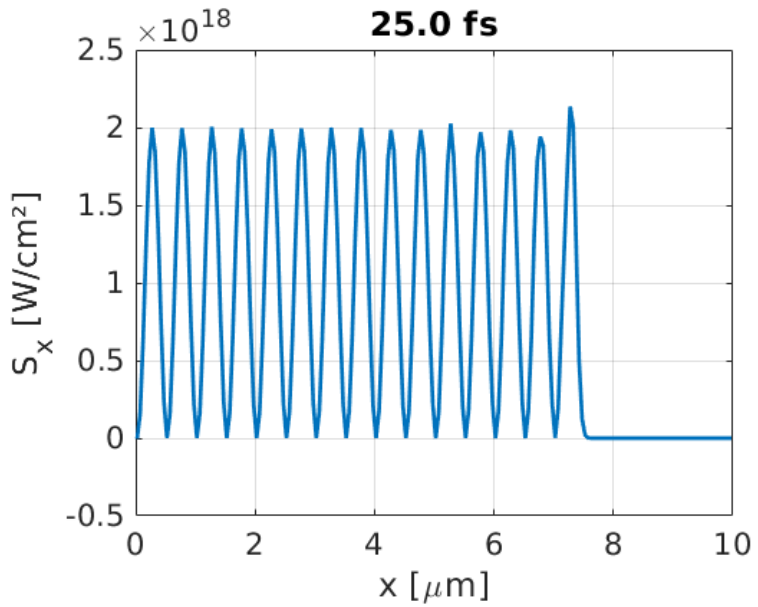
- Combining the two yields:
  - $(B_y)_{0,1}^n = \left(\frac{c^2}{\Delta x} + \frac{1}{\Delta t}\right)^{-1}\left(-\frac{4}{\Delta t}S^n + \left(\frac{c^2}{\Delta x} - \frac{1}{\Delta t}\right)(B_y)_{1,1}^n + \frac{2}{\Delta t}(E_z)_{1,1}^{n-1/2} - \frac{1}{\epsilon_0}(J_z)^n\right)$

Example: Plane wave 1D

- Inject a simple plane wave into a 1D EPOCH simulation
  - Plot the Poynting Flux:

$$\boldsymbol{S} = \frac{1}{\mu_0}(\boldsymbol{E} \times \boldsymbol{B})$$

- Front bulge:
  - Incomplete oscillation on front
- Twice the injected wavelength?
  - Peak and trough of E_y are both peaks




```
begin:control
    nx = 200
    t_end = 25e-15
    x_min = 0
    x_max = 10e-6
    stdout_frequency = 100
end:control

begin:boundaries
    bc_x_min = simple_laser
    bc_x_max = open
end:boundaries

begin:laser
    boundary = x_min
    intensity_w_cm2 = 1.0e18
    lambda = 1.0e-6
end:laser

begin:output
    dt_snapshot = t_end
    poynt_flux = always
end:output
```

Example: Plane wave 1D – big cells

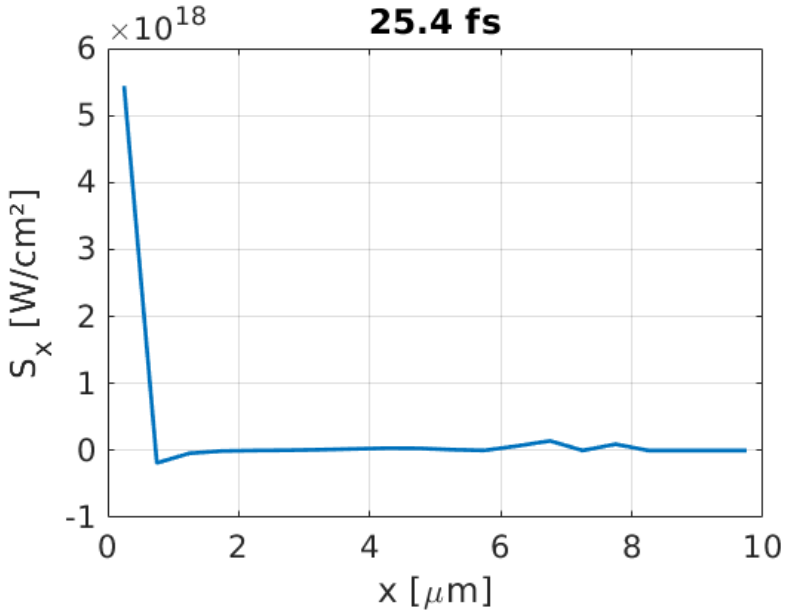
- What resolution can we get away with?
  - Last example: 200 cells
  - Now try 20 cells
  - What will happen?

```
begin:control
    nx = 20
    t_end = 25e-15
    x_min = 0
    x_max = 10e-6
    stdout_frequency = 100
end:control
```

Example: Plane wave 1D – big cells

- What resolution can we get away with?
  - Last example: 200 cells
  - Now try 20 cells

- Field solver relies on gradients
  - Too few evaluation points on the wave prevents this



Rule of thumb:
Need about 20 cells
per wavelength

```
begin:control
    nx = 20
    t_end = 25e-15
    x_min = 0
    x_max = 10e-6
    stdout_frequency = 100
end:control
```

Example: Gaussian pulse1D

- Add a temporal profile to the laser
- EPOCH `gauss` function (not a true Gaussian):

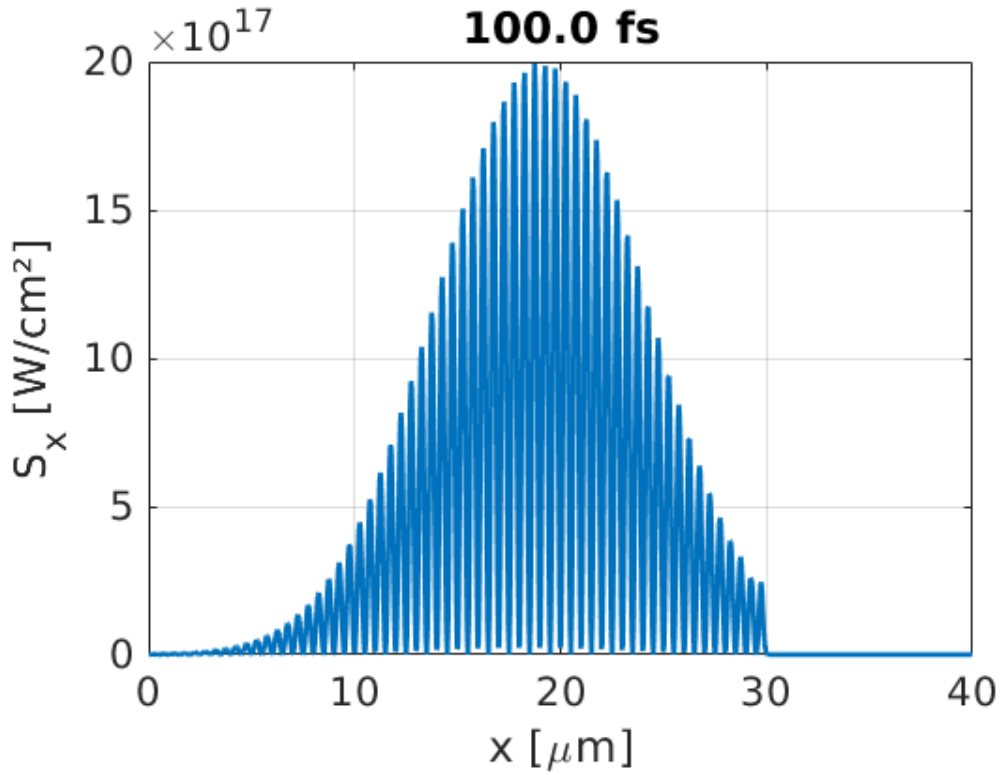
$$\text{gauss}(x, x_0, w) = e^{-\frac{(x-x_0)^2}{w^2}}$$

- The profile is applied to the E field amplitude
  - $E \propto I^{1/2}$
  - This function, applied to E, gives a real Gaussian in I (intensity)

```
begin:control
    nx = 1000
    t_end = 100e-15
    x_min = 0
    x_max = 40e-6
    stdout_frequency = 100
end:control

begin:boundaries
    bc_x_min = simple_laser
    bc_x_max = open
end:boundaries

begin:constant
    t_fwhm = 40.0e-15
    w_t = t_fwhm / sqrt(2*loge(2))
    t_hw01m = 0.5 * t_fwhm * sqrt(loge(10)/loge(2))
end:constant

begin:laser
    boundary = x_min
    intensity_w_cm2 = 1.0e18
    lambda = 1.0e-6
    t_profile = gauss(time,t_hw01m,w_t)
end:laser

begin:output
    dt_snapshot = t_end
    poynt_flux = always
end:output
```

Example: Gaussian pulse1D

- Pulse starts at 10% maximum intensity
- Half-max between around 13 to 25 μm
  - At speed c, around 40 fs, as expected

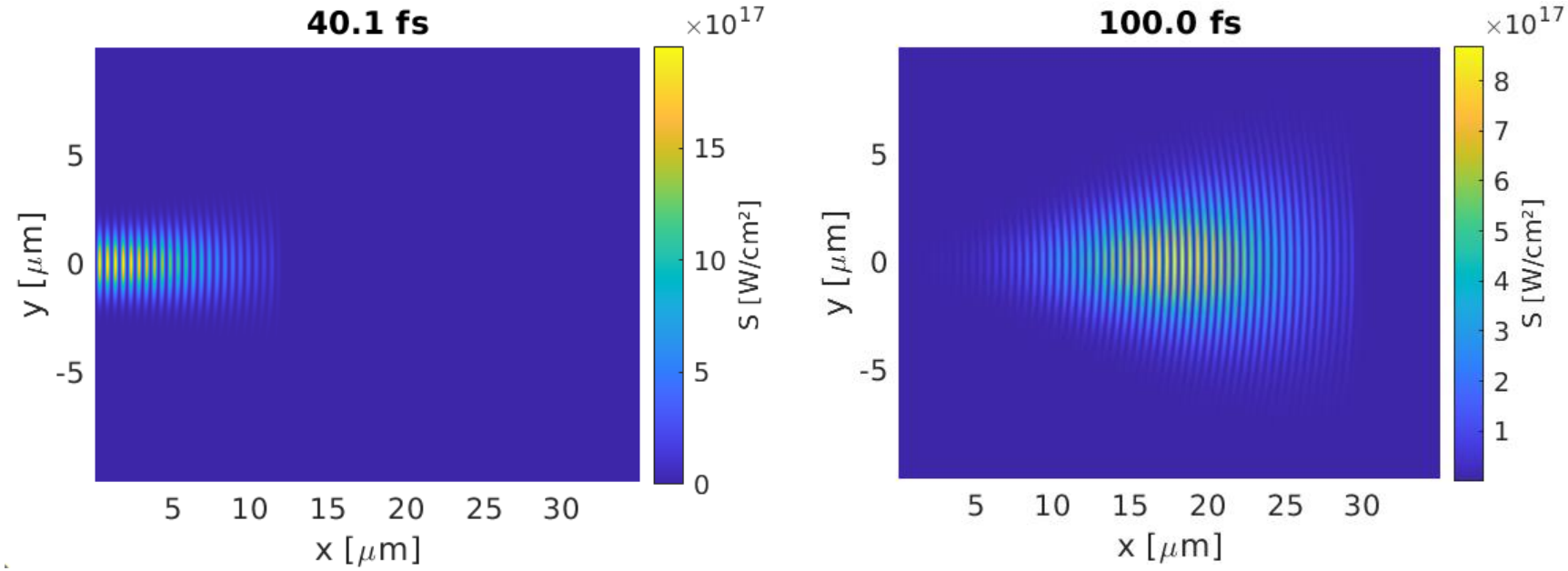



```
begin:control
    nx = 1000
    t_end = 100e-15
    x_min = 0
    x_max = 40e-6
    stdout_frequency = 100
end:control

begin:boundaries
    bc_x_min = simple_laser
    bc_x_max = open
end:boundaries

begin:constant
    t_fwhm = 40.0e-15
    w_t = t_fwhm / sqrt(2*loge(2))
    t_hw01m = 0.5 * t_fwhm * sqrt(loge(10)/loge(2))
end:constant

begin:laser
    boundary = x_min
    intensity_w_cm2 = 1.0e18
    lambda = 1.0e-6
    t_profile = gauss(time,t_hw01m,w_t)
end:laser

begin:output
    dt_snapshot = t_end
    poynt_flux = always
end:output
```

Example: Gaussian pulse 2D

- Model a Gaussian pulse in 2D
  - Gaussian temporal profile
  - Gaussian spatial profile

- See how it spreads out and loses intensity – why does it do this?




```
begin:control
    nx = 700
    ny = 400
    t_end = 100e-15
    x_min = 0
    x_max = 35e-6
    y_min = -10e-6
    y_max = 10e-6
    stdout_frequency = 100
end:control

begin:boundaries
    bc_x_min = simple_laser
    bc_x_max = open
    bc_y_min = open
    bc_y_max = open
end:boundaries

begin:constant
    t_fwhm = 40.0e-15
    y_fwhm = 2.0e-6
    w_t = t_fwhm / sqrt(2*loge(2))
    w_y = y_fwhm / sqrt(2*loge(2))
    t_hw01m = 0.5 * t_fwhm * sqrt(loge(10)/loge(2))
end:constant

begin:laser
    boundary = x_min
    intensity_w_cm2 = 1.0e18
    lambda = 1.0e-6
    profile = gauss(y,0,w_y)
    t_profile = gauss(time,t_hw01m,w_t)
end:laser

begin:output
    dt_snapshot = 10 * femto
    poynt_flux = always
end:output
```

Example: Gaussian beam 2D

- Paraxial approximation:
  - Laser comes in and out of focus
  - Valid if minimum waist is much smaller than wavelength
- Previous example:
  - Simulated tight focus on axis
  - Divergence from this is expected

$$\boldsymbol{E}(r,x) = E_0 \frac{w_0}{w(x)} e^{-r^2/w(x)^2} e^{-i\left(kx + k\frac{r^2}{2R_c(x)} - \psi(x)\right)} \hat{\boldsymbol{y}}$$

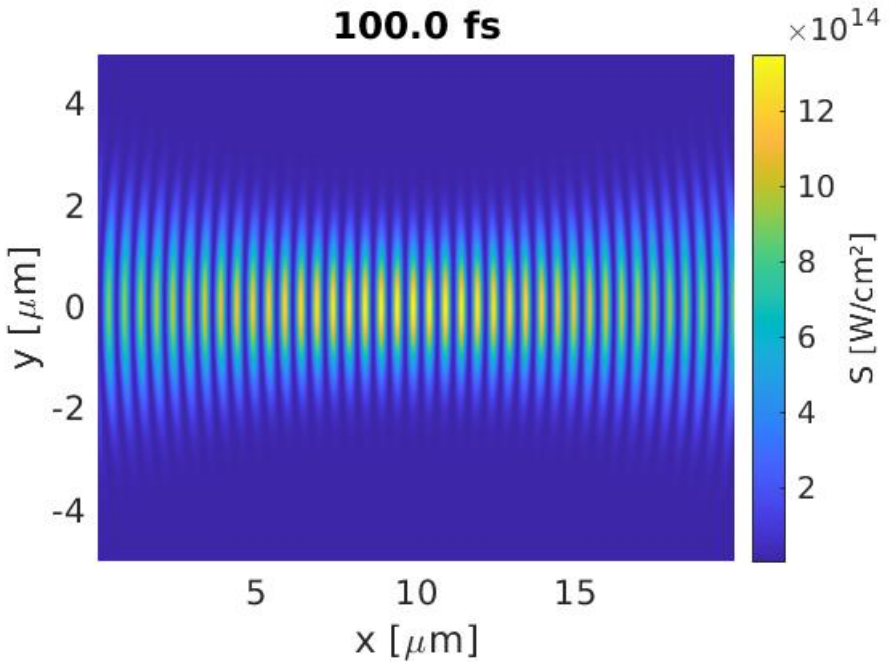
where

- r is the radial distance from the laser propagation axis
- x is axial distance along the wave, with $x = 0$ at the focus
- E_0 is the peak electric field amplitude at the focus
- $w(x)$ is the beam-waist at x (radial distance where field strength drops by e^{-1})
- w_0 is $w(x = 0)$
- k is the laser wave-vector
- $R_c(x)$ is the radius of curvature at x
- $\psi(x)$ is the Gouy phase correction

Example: Gaussian beam 2D

- Setup beam on axis to focus after 10 μm
- Peak expected intensity still not reached
  - Focal spot too tight c.f. wavelength
  - Paraxial approximation at limit

$$\boldsymbol{E}(r,x) = E_0 \frac{w_0}{w(x)} e^{-r^2/w(x)^2} e^{-i(kx + k\frac{r^2}{2R_c(x)} - \psi(x))} \hat{\boldsymbol{y}}$$




```
begin:constant
    I_fwhm = 2.0e-6          # FWHM of laser intensity
    I_peak_Wcm2 = 1.0e15     # 0.5 * eps0 * c * E_peak^2
    las_lambda = 1.0e-6      # Laser wavelength
    foc_dist = 10.0e-6        # Boundary to focal point distance
end:constant

begin:constant
    las_k = 2.0 * pi / las_lambda
    w0 = I_fwhm / sqrt(2.0 * loge(2.0))                    # Beam Waist
    ray_rang = pi * w0^2 / las_lambda                      # Rayleigh range
    w_boundary = w0 * sqrt(1.0 + (foc_dist/ray_rang)^2)    # Waist on boundary
    I_boundary = I_peak_Wcm2 * (w0 / w_boundary)^2         # Intens. on boundary
    rad_curve = foc_dist * (1.0 + (ray_rang/foc_dist)^2)   # Boundary curv. rad.
    gouy = atan(-foc_dist/rad_curve)                       # Boundary Gouy shift
end:constant

begin:laser
    boundary = x_min
    intensity_w_cm2 = I_boundary
    lambda = las_lambda
    phase = las_k * y^2 / (2.0 * rad_curve) - gouy
    profile = gauss(y, 0, w_boundary)
end:laser
```

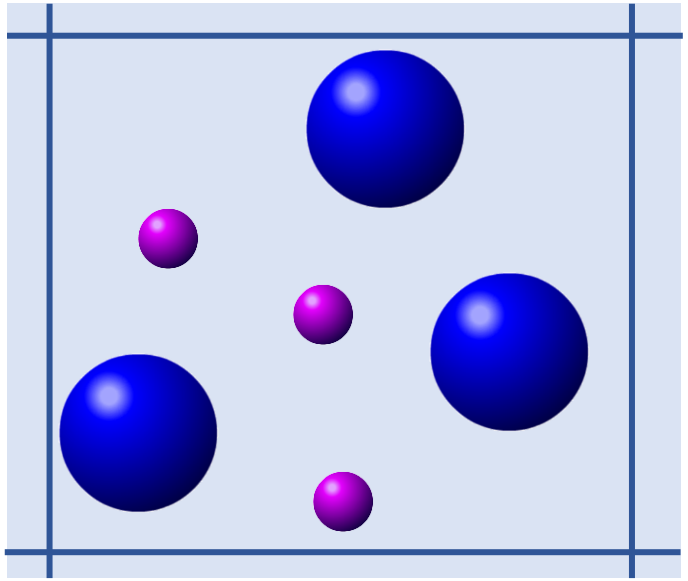
Section 3

Particles

Macro-particles

- Load macro-particles from species block
- Macro-particles loaded at random positions in a cell
  - Number of marco-particles, N_{ppc}: npart or frac
- Weight: $W_e = n_e V / N_{ppc}$, for density, n_e, cell volume, V
  - In 1D/2D, missing dimensions of cell-size 1m
  - This is fine, EPOCH equations only sensitive to density
- Momentum distribution:
  - Default: Maxwellian at cell temperature
  - User-given drift momentum applied
  - Custom distributions possible

```
begin:constant
  drift_p = 2.5e-24
  temp = 273
  dens = 10
end:constant


begin:species
  # Rightwards travelling electrons
  name = Right
  charge = -1
  mass = 1.0
  temp = temp
  drift_x = drift_p
  number_density = dens
  npart = 4 * nx
end:species
```



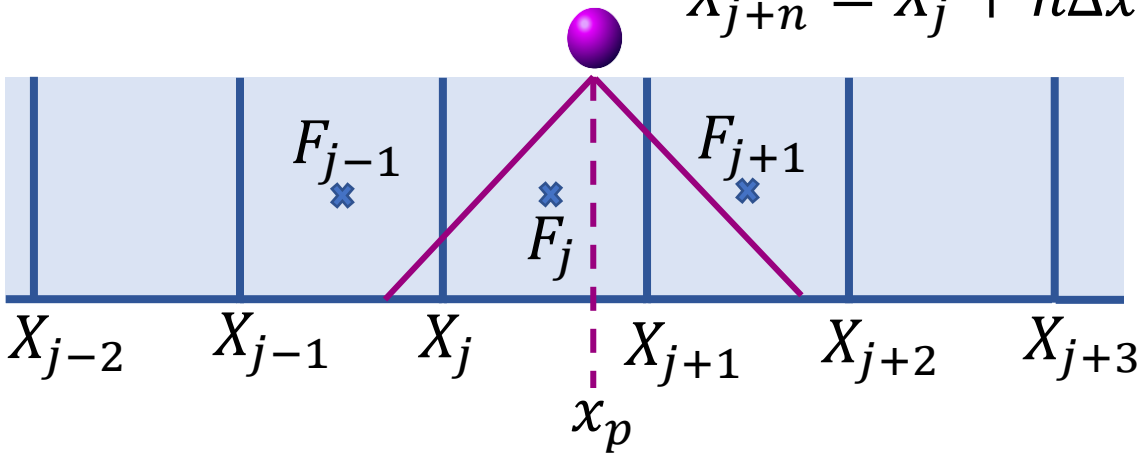
1D shape functions

- Macro-particles have "shape"
  - Default is triangular:

$$S(x) = \begin{cases} 1 - \dfrac{|x - x_p|}{\Delta x} & |x - x_p| < \Delta x \\ 0 & \text{otherwise} \end{cases}$$

$$X_j = \text{floor}\left(\frac{x_p}{\Delta x}\right)\Delta x$$

$$X_{j+n} = X_j + n\Delta x$$



- Cell-centred field parameter on particle:
  - Let low-x simulation edge have $x = 0$
  - Cell centre of first cell at $x = \Delta x/2$

$$f = \frac{X_j}{\Delta x} - \left(\frac{x_p}{\Delta x} - \frac{1}{2}\right)$$

$$F = \frac{\left(\frac{1}{4} + f^2 + f\right)F_{j-1} + \left(\frac{3}{2} - 2f^2\right)F_j + \left(\frac{1}{4} + f^2 - f\right)F_{j+1}}{2}$$

- Sanity check: $x_p = X_j$, then $F = \frac{1}{2}(F_{j-1} + F_j)$
- Particle weight, W_e, in each cell:

$$W_{j-1} = \frac{1}{2}W_e\left(\frac{1}{4} + f^2 + f\right)$$

$$W_{j-1} = \frac{1}{2}W_e\left(\frac{3}{2} - 2f^2\right)$$

$$W_{j+1} = \frac{1}{2}W_e\left(\frac{1}{4} + f^2 - f\right)$$

Alternative shape functions

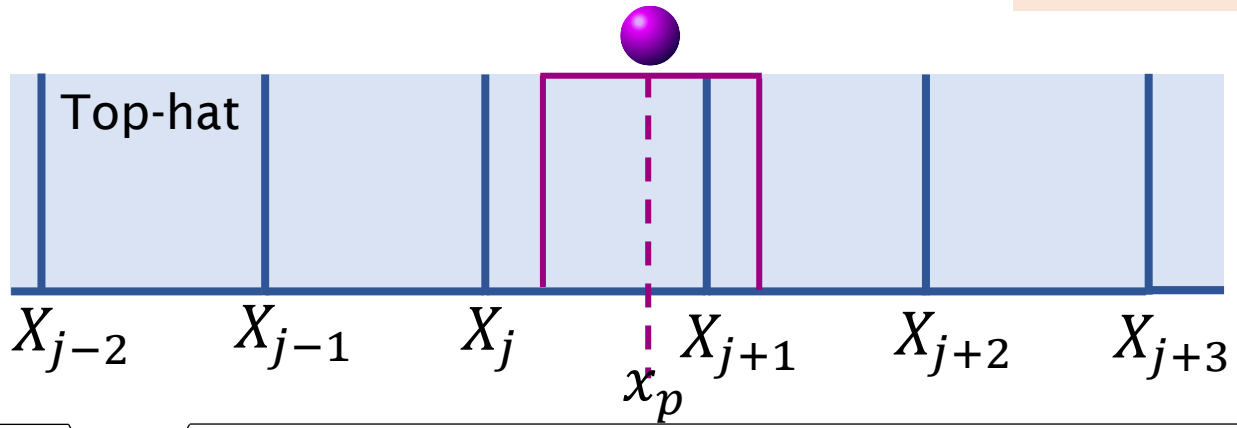
- Macro-particles can have one of three shapes:
  - Top-hat (shape size: Δx)
  - Default (shape size: $2\Delta x$)
  - B-Spline (shape size: $4\Delta x$)

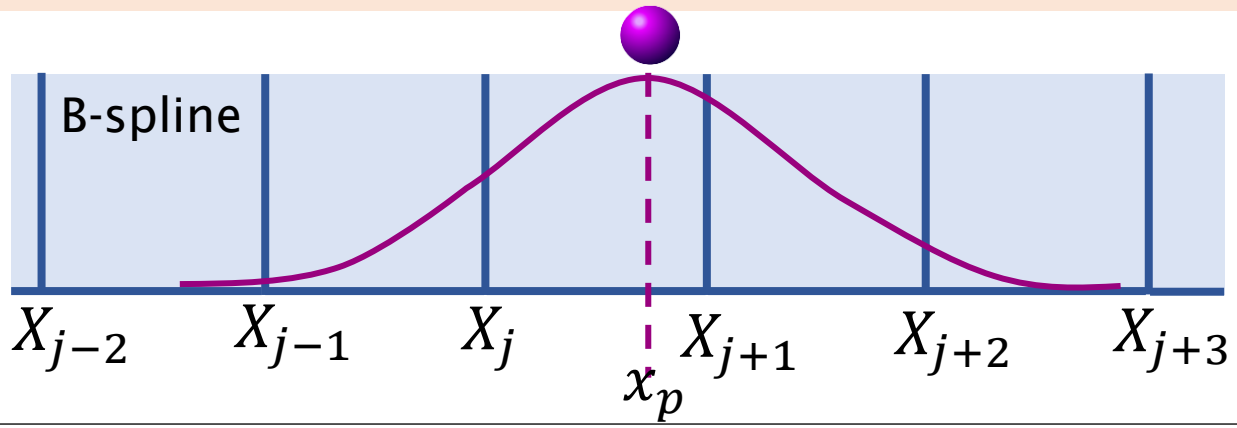
- Large particle shapes have smoother currents
  - Larger shapes also take longer to compute

$$S(x) = \begin{cases} 1/\Delta x & |x - x_p| < \Delta x/2 \\ 0 & \text{otherwise} \end{cases}$$

$$S(x) = \begin{cases} 1 - \dfrac{|x - x_p|}{\Delta x} & |x - x_p| < \Delta x \\ 0 & \text{otherwise} \end{cases}$$

$$S(x) = \frac{1}{6\Delta x^4} \begin{cases} \left(-|x - x_p| + 2\Delta x\right)^3 & \Delta x \le |x - x_p| \le 2\Delta x \\ 3|x - x_p|^3 - 6\Delta x|x - x_p|^2 + 4\Delta x^3 & |x - x_p| < \Delta x \\ 0 & \text{otherwise} \end{cases}$$

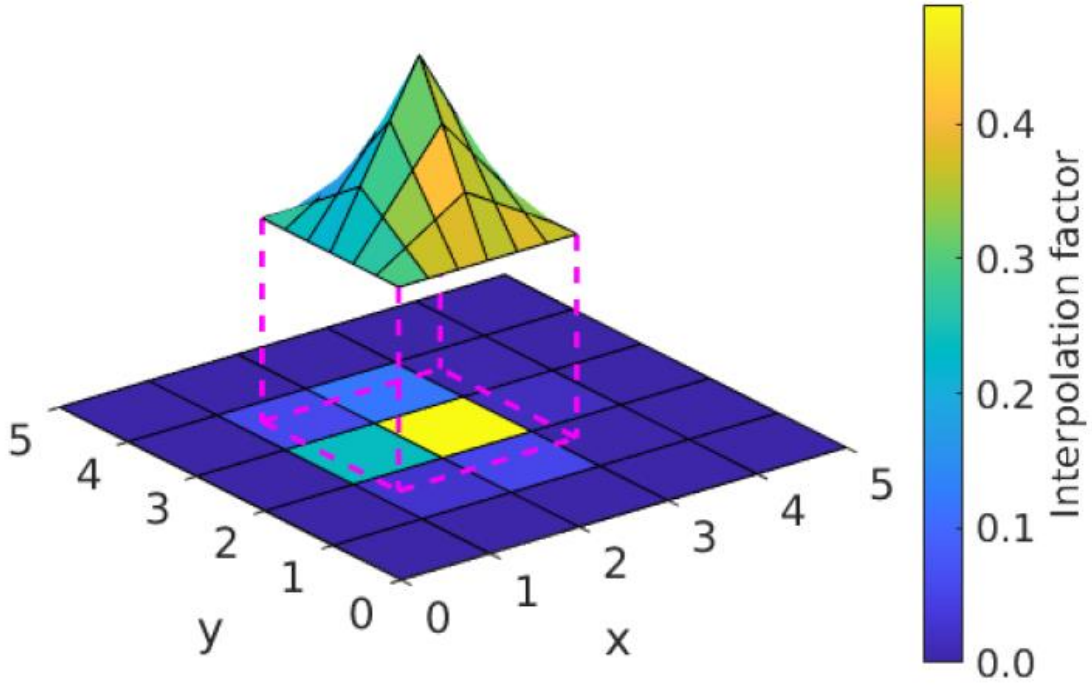




General shape functions

- Macro-particle shapes in 2D/3D
  - Calculate shape function in each dimension
  - Multiply 1D weighting factors for each dimension in each cell

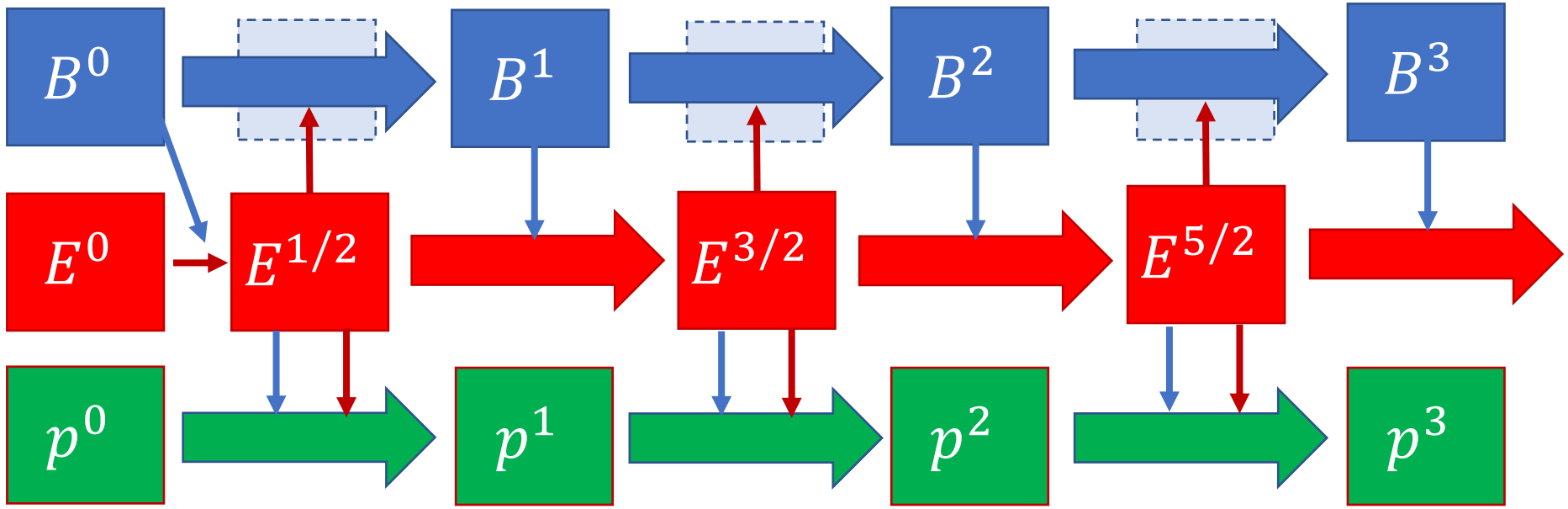
- 2D triangular weighting example:



Boris pusher

- Lorentz force:
  - $\frac{d}{dt}\boldsymbol{p} = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B})$
- Discretised Lorentz force:
  - $\boldsymbol{p}^{n+1} = \boldsymbol{p}^{n} + q\Delta t\left(\boldsymbol{E}^{n+\frac{1}{2}} + \frac{1}{\gamma^{n+\frac{1}{2}}m}\boldsymbol{p}^{n+\frac{1}{2}} \times \boldsymbol{B}^{n+\frac{1}{2}}\right)$

- $\boldsymbol{E}^{n+\frac{1}{2}}$, known from field solve
- $\boldsymbol{B}^{n+\frac{1}{2}}$, split B update into two half-steps
- $\boldsymbol{p}^{n+\frac{1}{2}}$, approximately $\frac{1}{2}(\boldsymbol{p}^{n+1} + \boldsymbol{p}^{n})$
- $\gamma^{n+\frac{1}{2}}$, see next slide



Boris pusher

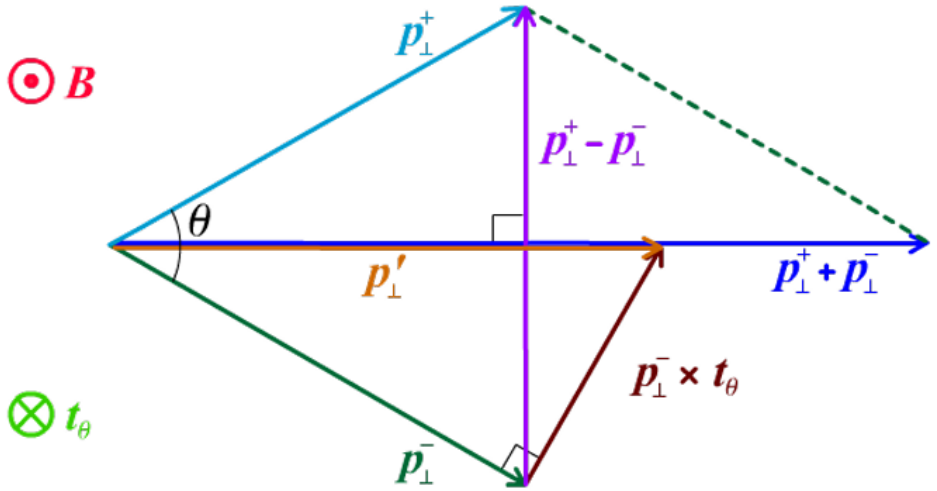
- $\boldsymbol{p}^{n+1} = \boldsymbol{p}^{n} + q\Delta t\left(\boldsymbol{E}^{n+\frac{1}{2}} + \frac{1}{2\gamma^{n+\frac{1}{2}}m}(\boldsymbol{p}^{n+1}+\boldsymbol{p}^{n}) \times \boldsymbol{B}^{n+\frac{1}{2}}\right)$

- Introduce $\boldsymbol{p}^{-}$, $\boldsymbol{p}^{+}$, and split update into acceleration and rotation
  - Two half-step accelerations:
  - $\boldsymbol{p}^{-} = \boldsymbol{p}^{n} + \frac{\Delta t}{2} q\boldsymbol{E}^{n+\frac{1}{2}}$
  - $\boldsymbol{p}^{n+1} = \boldsymbol{p}^{+} + \frac{\Delta t}{2} q\boldsymbol{E}^{n+\frac{1}{2}}$

- $\gamma^{n+\frac{1}{2}}$ is the γ factor associated with $\boldsymbol{p}^{-}, \boldsymbol{p}^{+}$:
  - $\gamma^{n+\frac{1}{2}} = \sqrt{1 + \frac{\boldsymbol{p}^{-}\cdot\boldsymbol{p}^{-}}{m^2c^2}}$

- Substitute p^{+} and p^{-} into above to reveal rotation:

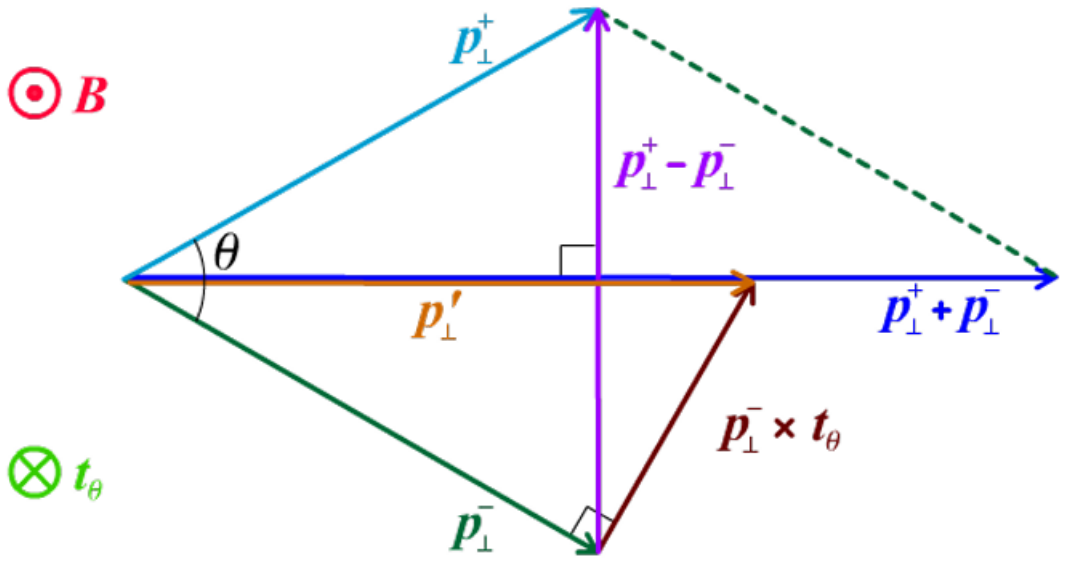
$$\boldsymbol{p}^{+} = \boldsymbol{p}^{-} + \frac{q\Delta t}{2\gamma^{n+\frac{1}{2}}m}(\boldsymbol{p}^{+}+\boldsymbol{p}^{-}) \times \boldsymbol{B}^{n+\frac{1}{2}}$$



Boris pusher

- $\boldsymbol{p}^{+} = \boldsymbol{p}^{-} + \frac{q\Delta t}{2\gamma^{n+\frac{1}{2}}m}(\boldsymbol{p}^{+} + \boldsymbol{p}^{-}) \times \boldsymbol{B}^{n+\frac{1}{2}}$
- From cross-product, we see component parallel to $\boldsymbol{B}$: $p_{\parallel}^{+} = p_{\parallel}^{-}$
- Define:
  - $\boldsymbol{t}_{\theta} = \frac{q\Delta t}{2\gamma^{n+\frac{1}{2}}m}\boldsymbol{B}^{n+\frac{1}{2}}$
  - $\boldsymbol{p}' = \boldsymbol{p}^{-} + \boldsymbol{p}^{-} \times \boldsymbol{t}_{\theta}$
- From geometry, $\boldsymbol{p}' \times \boldsymbol{t}_{\theta}$ is parallel to $\boldsymbol{p}^{+} - \boldsymbol{p}^{-}$
  - Magnitude of $\boldsymbol{p}$ cannot change in $\boldsymbol{B}$ rotation
  - Hence $|\boldsymbol{p}^{-}| = |\boldsymbol{p}^{+}|$, and:

$$\boldsymbol{p}^{+} = \frac{2}{1 + t_{\theta}^{2}}(\boldsymbol{p}' \times \boldsymbol{t}_{\theta}) + \boldsymbol{p}^{-}$$



- This relationship links $\boldsymbol{p}^{n+1}$ to $\boldsymbol{p}^{n}$

Macro-particle current (grid direction)

- Continuity Equation:

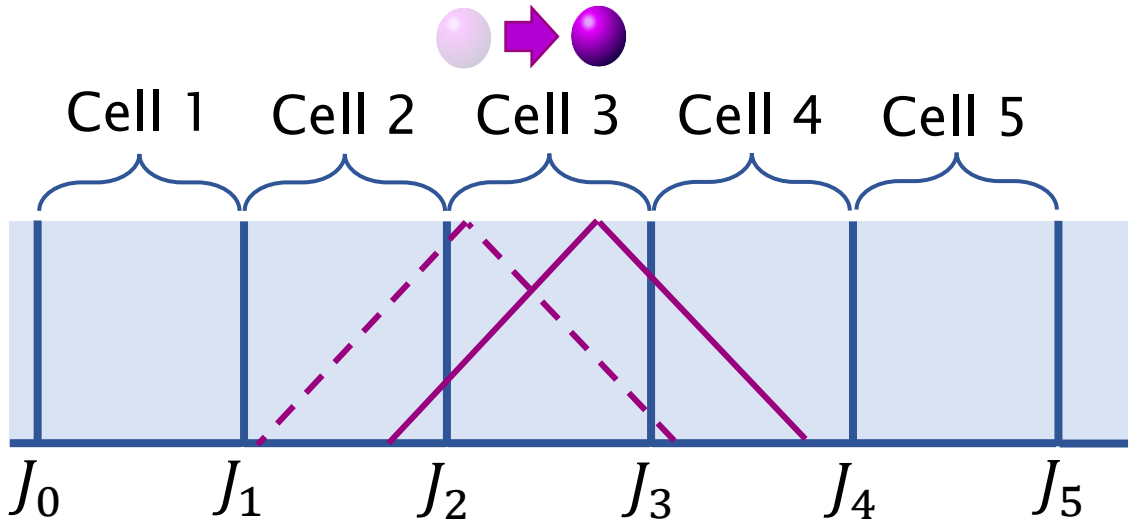
$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \boldsymbol{J}$$

- Particle shapes independent in each dimension:

$$\frac{\Delta \rho_x}{\Delta t} = -\frac{\Delta J_x}{\Delta x}$$

- Cell 2 weight change: -0.3
  - $J_2 - J_1 = \frac{\Delta x}{\Delta t}\left(-0.3\frac{W_e q_e}{V}\right)$
  - We see no charge passes J_1 line, so $J_1 = 0$
  - Macro-weight, W_e, particle charge, q_e
  - Cell volume, V, cell-size Δx, time-step Δt
- Next, $J_3 = J_2 + \frac{\Delta x}{\Delta t}\left(+0.15\frac{W_e q_e}{V}\right)$
  - Repeat until all boundaries considered

Cell	Weight before	Weight after
1	0	0
2	0.40	0.10
3	0.55	0.70
4	0.05	0.20
5	0	0



Macro-particle current (missing direction)

- Lower dimension simulation (1D/2D), missing dimensions have:
  - Current evaluation point: cell-centre
  - In 1D:

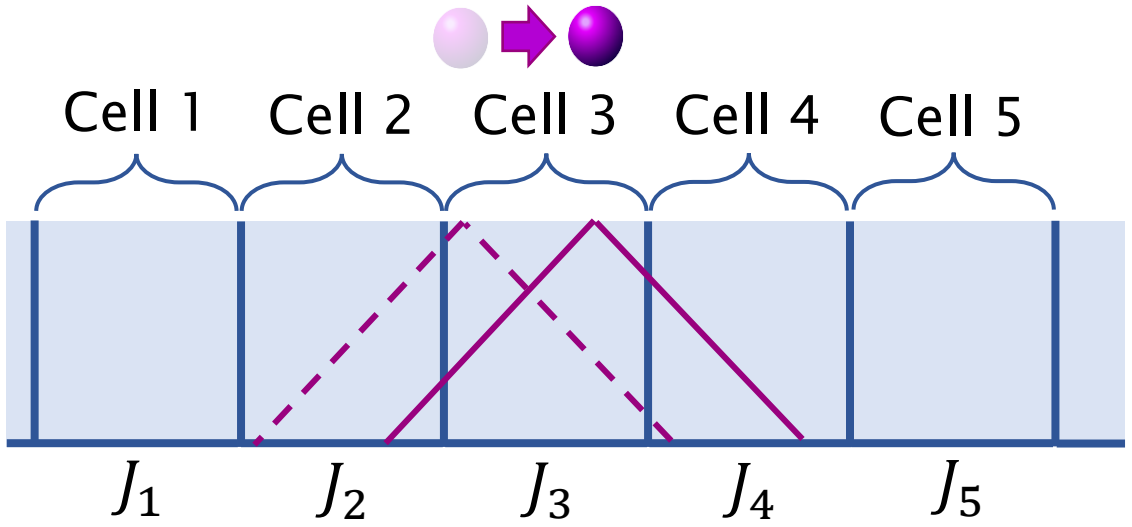
$$J_y = \frac{W_e q_e}{V} v_y \langle g \rangle$$

  - Average weight fraction in cell over Δt, $\langle g \rangle$

- $\langle g \rangle$ in cells:
  - 2: 0.25
  - 3: 0.625
  - 4: 0.125

- Similar in 2D for J_z
  - $\langle g \rangle$ involves weight changes in x and y
  - $\langle g \rangle = g_x g_y + \frac{1}{2} g_x \Delta g_y + \frac{1}{2} g_y \Delta gx + \frac{1}{3} \Delta g_x \Delta g_y$
  - g_i is initial i-weight, Δg_i is weight change

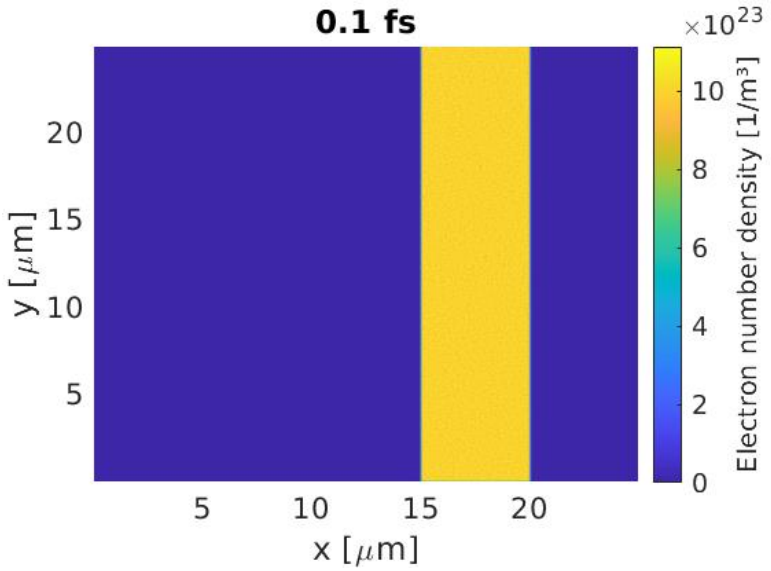
Cell	Weight before	Weight after
1	0	0
2	0.40	0.10
3	0.55	0.70
4	0.05	0.20
5	0	0

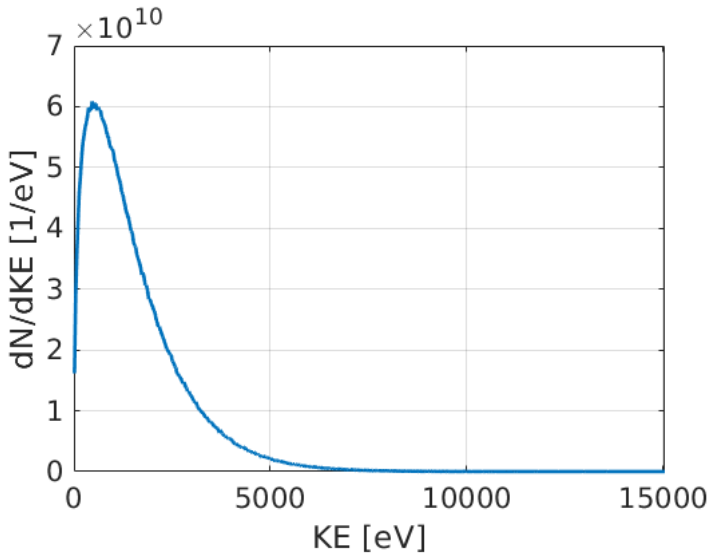


Example: Particle Loading (2D)

- Load a basic ionised carbon target
  - Maxwellian momentum distribution
  - Thin foil target shape

- Can plot either:
  - Grid-averaged quantities (density, temperature)
  - Particle properties (to get energy spectra)
  - Can also ignore some species for particle dumps

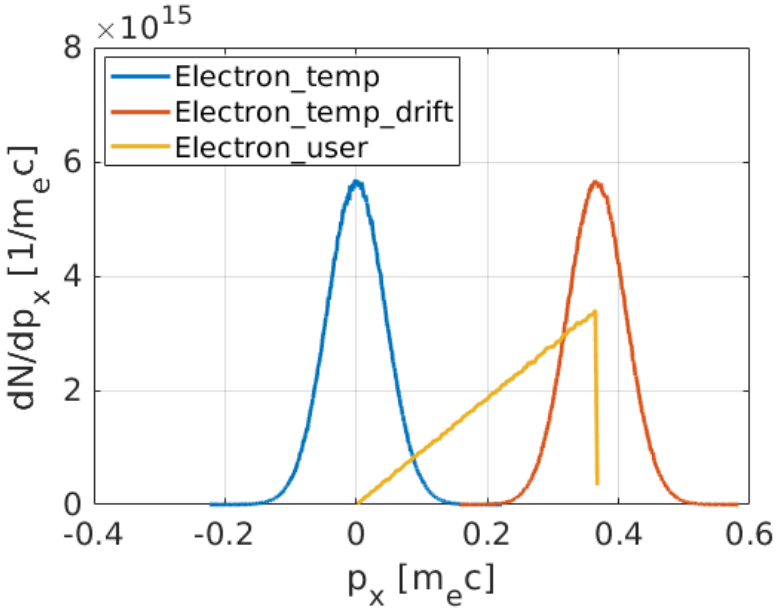





```
begin:control
    nx = 500
    ny = 500
    t_end = 1.0e-15
    x_min = 0
    x_max = 25e-6
    y_min = 0
    y_max = 25e-6
    stdout_frequency = 100
    npart = 50 * (5.0e-6/(x_max-x_min)) * nx * ny
end:control

begin:boundaries
    bc_x_min = open
    bc_x_max = open
    bc_y_min = open
    bc_y_max = open
end:boundaries

begin:species
    name = Electron
    mass = 1.0
    charge = -1.0
    frac = 0.8
    density = if (x gt 15.0e-6, 1.0e24, 0)
    density = if (x gt 20.0e-6, 0, density(Electron))
    temp_ev = 1000
end:species

begin:species
    name = Carbon
    mass = 22033.0
    charge = 6.0
    frac = 0.2
    density = density(Electron) / 6
    temp_ev = 1000
    dump = F       # Ignore this species in particle-dump
end:species

begin:output
    dt_snapshot = t_end
    number_density = always + species
    px = always
    py = always
    pz = always
    weight = always
end:output
```

Example: Momentum Distributions

- EPOCH can load:
  - Maxwellian (default)
  - Maxwell-Juttner (relativistic Maxwellian)
  - User-defined distribution
  - Any of the above with a drift momentum




```
begin:species
    name = Electron_temp
    mass = 1.0
    charge = -1.0
    npart = 5 * nx * ny
    density = 1.0e24
    temp_ev = 1000
end:species

begin:species
    name = Electron_temp_drift
    mass = 1.0
    charge = -1.0
    npart = 5 * nx * ny
    density = 1.0e24
    temp_ev = 1000
    drift_x = 1.0e-22
end:species

begin:species
    name = Electron_user
    mass = 1.0
    charge = -1.0
    npart = 5 * nx * ny
    density = 1.0e24
    dist_fn = px / 1.0e-22
    dist_fn_px_range = (0, 1.0e-22)
end:species
```

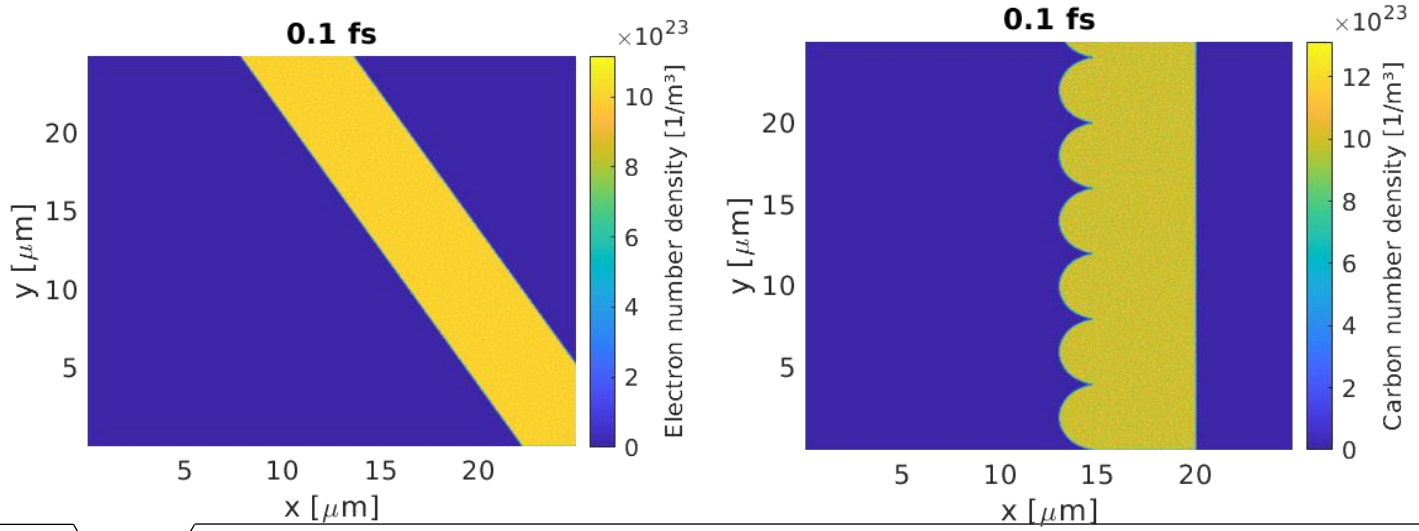
- User-defined method:
  - Provide a function $f(p_x, p_y, p_z)$ and momentum range
  - EPOCH uniformly samples $p_x,\ p_y,\ p_z$
  - Assumes function has maximum value of 1, calculate f
  - Rejection sampling: keep $p_x,\ p_y,\ p_z$ with probability f

Example: Funky Targets

- Use the constant block to make functions
  - Makes it easier to make complex targets
  - Define rotated system for rotated targets
  - Define periodically repeating functions too

- Electron: `x_rot`, `y_rot` are rotated x and y

- Carbon: `y_r` is repeat distance, `y_f` is repeating fraction




```
begin:constant
    x0 = 15.0e-6
    y0 = 12.5e-6
    theta = 30.0 / 180.0 * pi
    x_rot =  (x-x0)*cos(theta) + (y-y0)*sin(theta)
    y_rot = -(x-x0)*sin(theta) + (y-y0)*cos(theta)
end:constant

begin:species
    name = Electron
    mass = 1.0
    charge = -1.0
    frac = 0.8
    density = if (x_rot gt 0.0, 1.0e24, 0)
    density = if (x_rot gt 5.0e-6, 0, density(Electron))
    temp_ev = 1000
end:species

begin:constant
    x_front = 15.0e-6
    circle_radius = 2.0e-6
    y_r = 2.0 * circle_radius
    y_f = 2.0*((y / y_r) - floor(y / y_r))-1.0
end:constant

begin:species
    name = Carbon
    mass = 22033.0
    charge = 6.0
    frac = 0.2
    density = if ((x-x_front)^2 + (y_f * circle_radius)^2 lt circle_radius^2,\
                  1.0e24, 0)
    density = if (x gt x_front and x lt (x_front + 5.0e-6), \
                  1.0e24, density(Carbon))
    temp_ev = 1000
end:species
```

Example: Particle probes

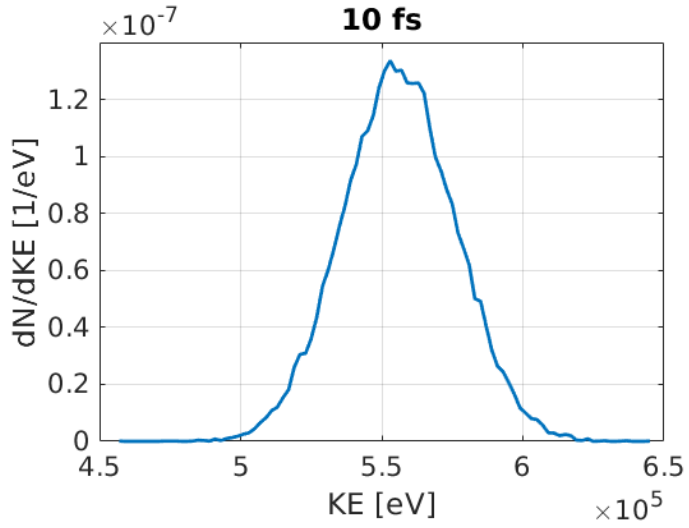
- Special type of output
  - Collects properties of particles which pass a “probe”
  - Defined by a point/line/plane in 1D/2D/3D simulations
  - Only saves particles passing in one direction (the normal)
  - Can filter by species type and particle energy
  - `ek_max = -1.0` means no upper limit for kinetic energy

- Probes dump position, momentum and weight by default, when particles pass the probe
  - Optionally: particle ID, and probe-crossing-time too

- Probes only written to file when particles have passed the probe
  - Only writes particles which have passed since the last dump

- Useful if there are too many particles to dump otherwise

```
begin:output
    dt_snapshot = t_end
    particle_probes = always
end:output

begin:probe
   name = Electron_probe
   point = (4.0e-6, 0.0)
   normal = (1.0, 0.0)
   ek_min = 0.0
   ek_max = -1.0
   include_species:Electron
   dumpmask = always
end:probe
```



Example: Self Heating

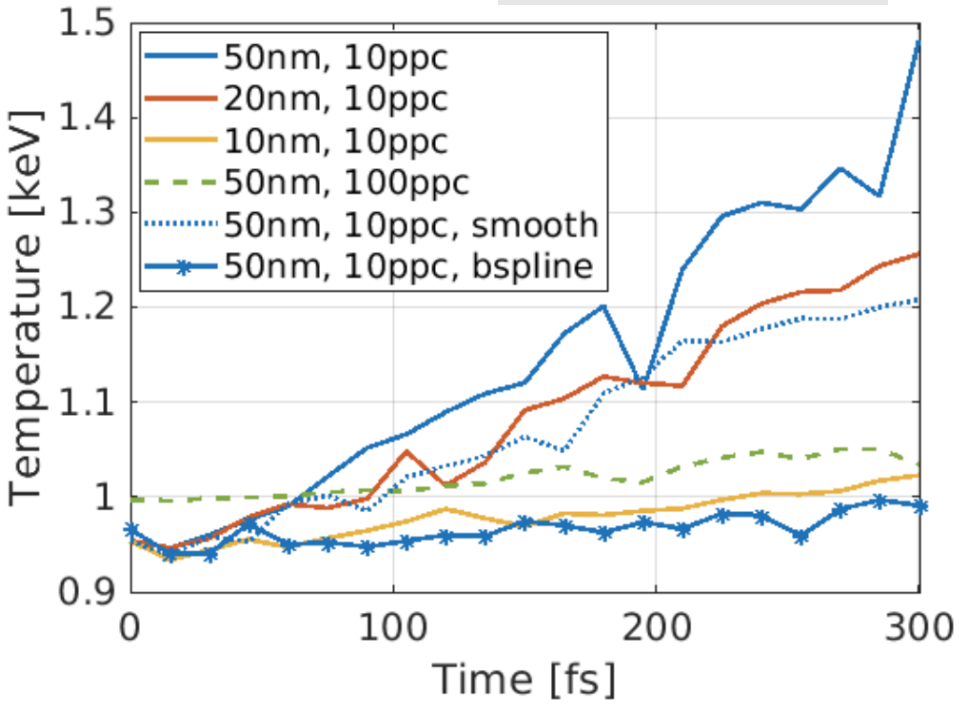
- At low resolution, macro-particle motion causes large currents
  - Non-physical, we should have $> 10^{20}$ particles giving smooth distribution

- Noisy currents give noisy E field
  - Large noise-driven gradients generate large B
  - Noisy fields cause non-physical particle heating
  - Worse in colder, denser plasmas

- Currents can be smoothed by using:
  - Larger particle shapes
  - Averaging currents across cells (smooth_currents)
  - More particles per cell
  - Smaller cells

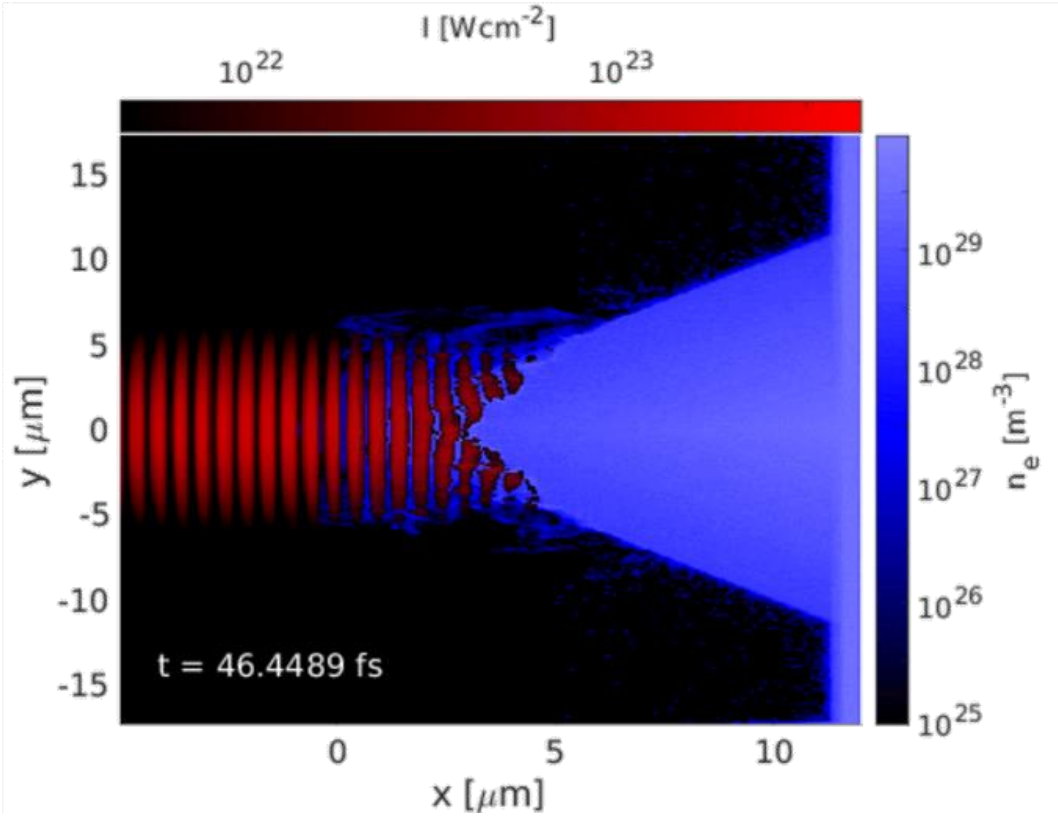
- Test with small, periodic domain

```
begin:constant
    cell_size = 50.0e-9
    parts_per_cell = 10
end:constant

begin:control
    nx = 10
    ny = 10
    t_end = 300.0e-15
    x_min = 0
    x_max = nx * cell_size
    y_min = 0
    y_max = ny * cell_size
    stdout_frequency = 100
    smooth_currents = T
end:control
```



Typical laser-solid simulation

- Inject a laser into a simulation with a target
  - Ensure cell-size/particle number minimises self-heating
- Pre-plasma:
  - Add exponentially decaying plasma to target front
  - Mimics effects of laser pre-heating
- Convergence testing:
  - Once you have a final result, try doubling or halving the resolution
  - If the result doesn't change by much, result has converged
  - Physical results don't depend on numerical parameters!



Section 4

Physics Packages

Current EPOCH physics packages

- Bremsstrahlung (e^-)
- Bremsstrahlung (e^+)
- Bethe-Heitler pair production (e^-e^+)
- Bethe-Heitler pair production ($\mu^-\mu^+$)*
- Background elastic-scatter collisions
- Nuclear-trident pair production*

- Non-linear Compton scatter
- Breit-Wheeler pair production
- QED-trident pair production
- Field ionisation

- Dielectronic recombination*
- Radiative recombination*
- Binary elastic-scatter collisions (1 species)
- Binary elastic-scatter collisions (2 species)
- Electron-impact collisional ionisation
- 3-body recombination*

Key:

- Particle interacts with species background
- Particle interacts with field background
- Particle interacts with local particles

* Not in main release, can be accessed on development branches

Sampling theory: constant rate

- $P(dt) = \lambda dt$
  - $P(dt)$: Probability of event over dt
  - dt: Small time interval
  - λ: Event rate

- $\bar{P}(\Delta t) = \left(1 - P(dt)\right)^n = \left(1 - \frac{\lambda \Delta t}{n}\right)^n$
  - $\bar{P}(\Delta t)$: Probability of no event over Δt
  - Δt: Long time interval
  - n: Number of small time-steps in Δt ($\Delta t = ndt$)

- $\lim_{n \to \infty} \left(\bar{P}(\Delta t)\right) = e^{-\lambda \Delta t}$

- $P_{k>0}(\Delta t) = 1 - e^{-\lambda \Delta t}$
  - $P_{k>0}(\Delta t)$: Prob. of at least 1 event over Δt
  - k: Number of events over Δt (assume $k = 1$)

Key equations:

$$P(\Delta t) = 1 - e^{-\lambda \Delta t}$$

Sampling theory: changing rate

- Replace $\lambda\Delta t$ with $\tau = \int_0^{\Delta t} \lambda(t)dt$
- $\bar{P}(\tau) = e^{-\tau}$
  - This is a PDF, normalised from $\tau = 0 \rightarrow \infty$
- $\bar{F}(\tau) = \int_0^{\tau} e^{-\tau'} d\tau' = -e^{-\tau}$
  - CDF of survival to optical depth τ
- To sample an optical-depth for event-trigger τ_e, replace $\bar{F}(\tau)$ with x_r (random number from 0 to 1) and find τ
- $x_r = -e^{-\tau_e}$
- $\tau_e = \ln(-x_r)$

Key equations:

$$\tau_e = \ln(-x_r)$$

Optical depth method:

1. Sample τ_e for each particle
2. Each step, subtract τ from τ_e
3. When $\tau_e < 0$, trigger event
4. Sample new τ_e if particle survives

Compilation flags

- Several EPOCH features are locked behind compilation flags
  - These slow the code, only use if needed

- To activate:
  - Uncomment the DEFINES line in the Makefile
  - Run terminal command: make clean
  - Recompile the code: make COMPILER=gfortran

- Extra features:
  - Different macro-particle shapes
  - Physics packages
  - Extended IO (e.g. particle ID, probe-time)

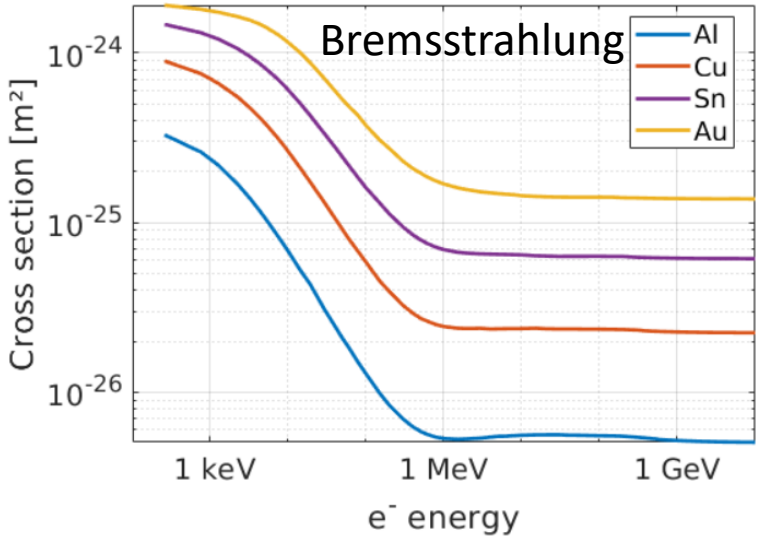
```
206     # Use second order particle weighting (default is third order).
207     #DEFINES += $(D)PARTICLE_SHAPE_TOPHAT
208
209     # Use fifth order particle weighting (default is third order).
210     #DEFINES += $(D)PARTICLE_SHAPE_BSPLINE3
211
212     # Include a unique global particle ID. The first flag defines the ID using
213     # an 8-byte integer, the second uses 4-bytes.
214     #DEFINES += $(D)PARTICLE_ID
215     #DEFINES += $(D)PARTICLE_ID4
216
217     # Include QED routines
218     #DEFINES += $(D)PHOTONS
219
220     # Use the Trident process for pair production
221     #DEFINES += $(D)TRIDENT_PHOTONS
222
223     # Include bremsstrahlung routines
224     #DEFINES += $(D)BREMSSTRAHLUNG
```

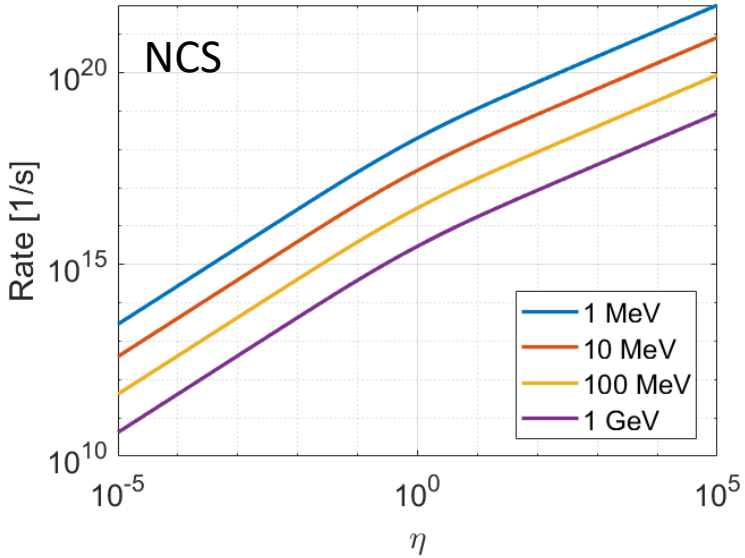
Radiation processes

- Two types of radiation:
  - Bremsstrahlung radiation
  - Non-linear Compton scatter (synchrotron radiation)

- Bremsstrahlung:
  - Trigger rate: $\lambda = n_i v_e \sigma$
  - σ: Cross-section (Seltzer-Berger tables)
  - n_i: Background ion density
  - v_e: Incident electron speed

- NCS (Ridgers 2014):
  - Trigger rate: $\lambda = \frac{\sqrt{3}\alpha_f c}{\lambda_c}\frac{\eta}{\gamma}h(\eta)$
  - η scales with field-strength and electron energy





Morris *et al* (2021). *Physics of Plasmas*, *28*(10).

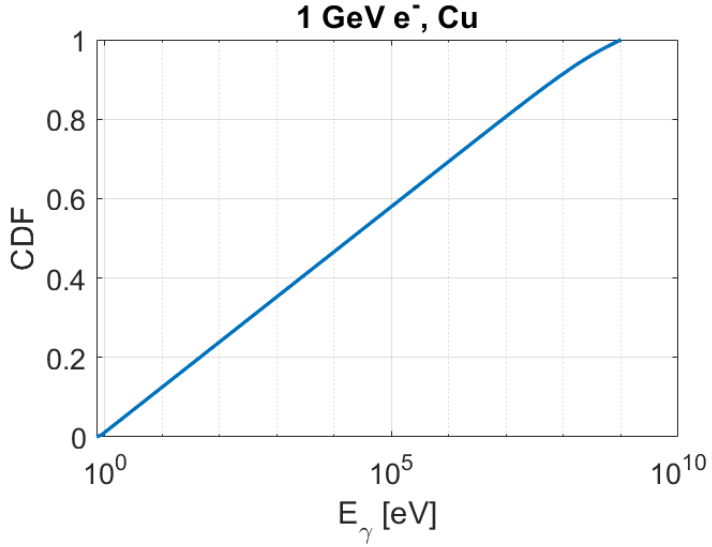
Ridgers *et al* (2014). *Journal of computational physics*, *260*, 273-285.

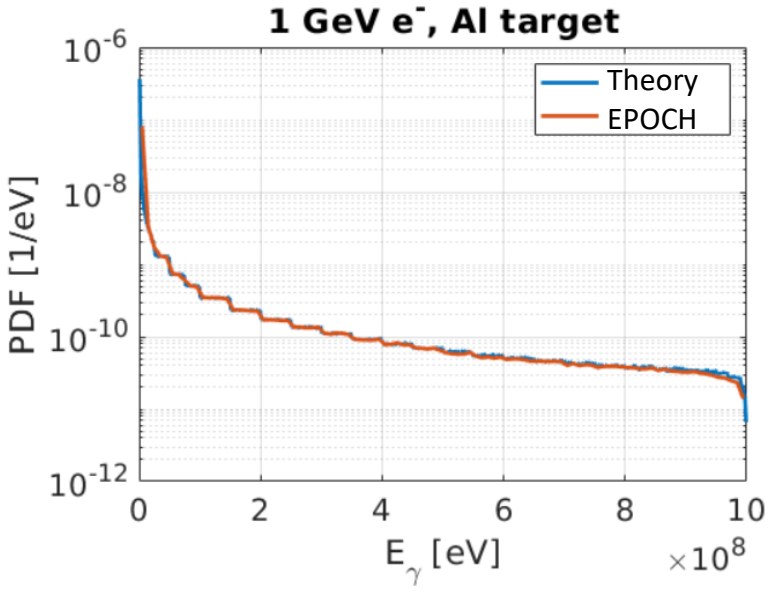
Macro-photon emission

- In photon emission:
  - Incident e^- has momentum reduced
  - Macro-photon created at e^- position

- Macro-photon energy:
  - Normalise $\frac{d\sigma}{dE_\gamma}$ to get photon emission energy distribution, PDF
  - Integrate PDF to get cumulative distribution function CDF
  - Replace CDF with a random number, return corresponding E_γ

- Photon angular distribution obtained from model fits
  - Usually parallel to incident e^- at high energy



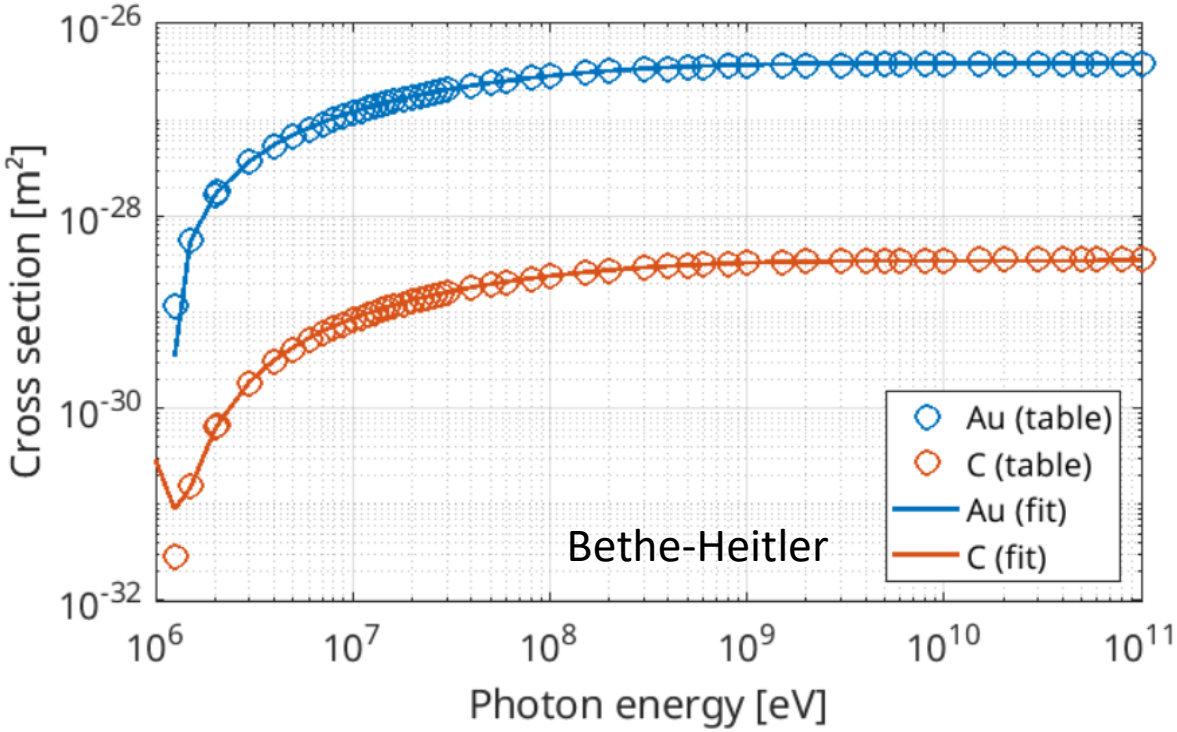


Pair production processes

- Main branch pair-production in EPOCH:
  - Breit-Wheeler (photon decay in laser fields)
  - QED-trident (e^- radiating an e^-e^+ pair in laser field)
  - Bethe-Heitler (photon decay in nucleus Coulomb field)

- Incident particle:
  - Destroyed (photon)
  - Recoiled (electron)

- Pair energies sampled using CDF method
  - CDF values read from tables
  - Create macro-electron and macro-positron



Martinez, *et al* (2019). *Physics of Plasmas*, *26*(10).
Ridgers *et al* (2014). *Journal of computational physics*, *260*, 273-285.

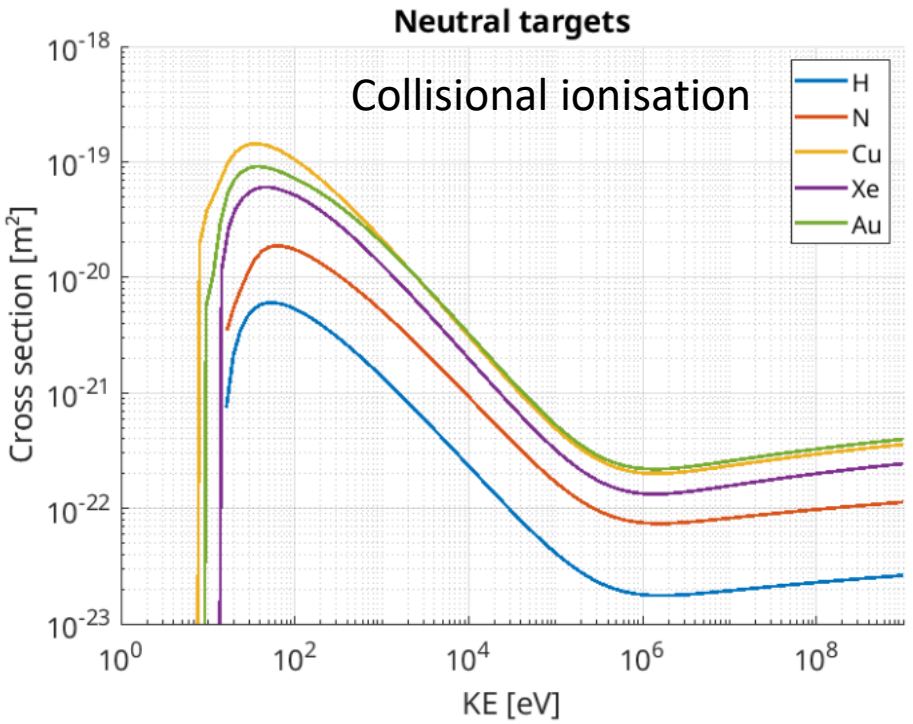
Ionisation processes

- Main branch:
  - Field ionisation
  - Collisional ionisation

- Species identified by charge-state and atomic number
  - Ionisation rates calculated
  - Macro-ions move to different species once ionised

- Field ionisation:
  - Ion releases electron in strong external field

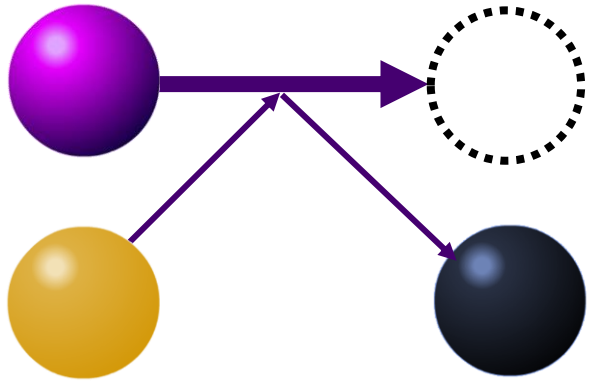
- Collisional ionisation:
  - Incident electron collides with bound electron
  - Energy transfer in collision ejects bound electron



Morris *et al* (2022) *Physics of Plasmas*, *29*(12).
Lawrence-Douglas (2013). *Ionisation effects for laser-plasma interactions by particle-in-cell code* (PhD thesis, University of Warwick).

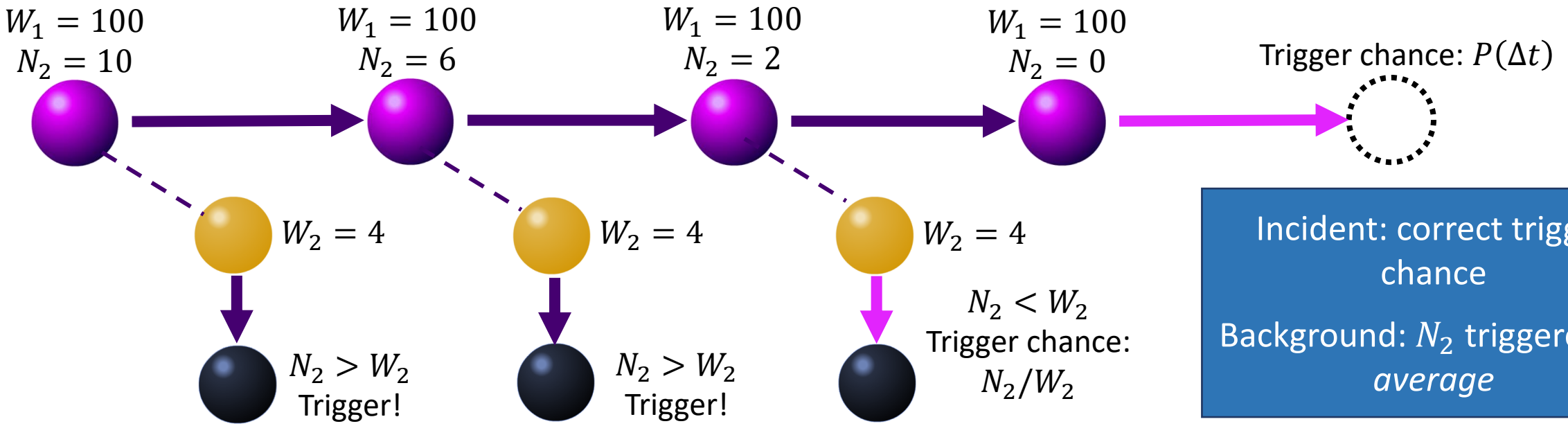
Collisional ionisation sampling

- Incident macro-particle represents W_1 real particles



- Each incident has interaction chance $P(\Delta t)$
  - Number of affected background particles: $N_2 = W_1 P(\Delta t)$
  - Trigger background macro-particles until N_2 are affected

- Trigger incident with chance $P(\Delta t)$ - e.g. 10%:



Incident: correct trigger chance

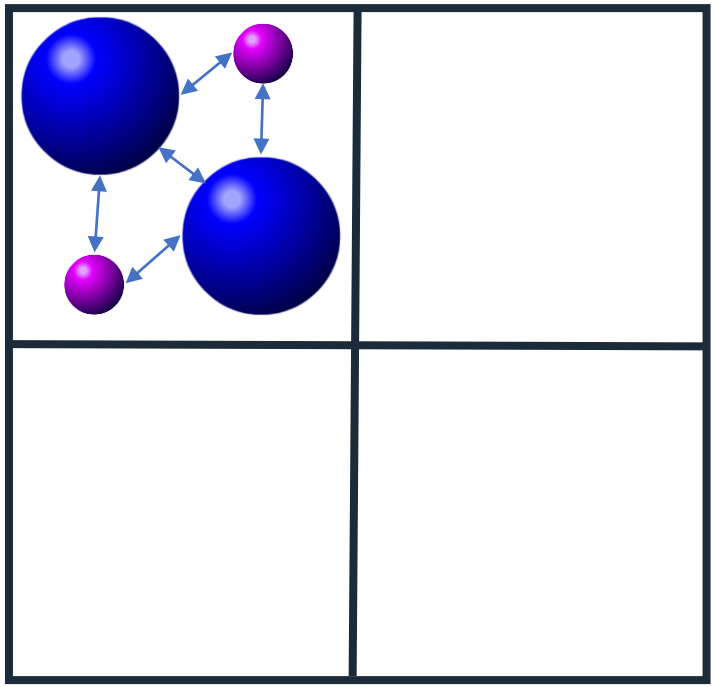
Background: N_2 triggered *on average*

Collisions

- Fields can only be resolved between cells
  - What about local interactions?

- Separate package models collisions within the same cell

- Binary-collision algorithm:
  - Particles paired together
  - Scattering estimated and applied



Nanbu (1997). *Physical Review E*, *55*(4), 4642.
Pérez *et al* (2012). *Physics of Plasmas*, *19*(8).

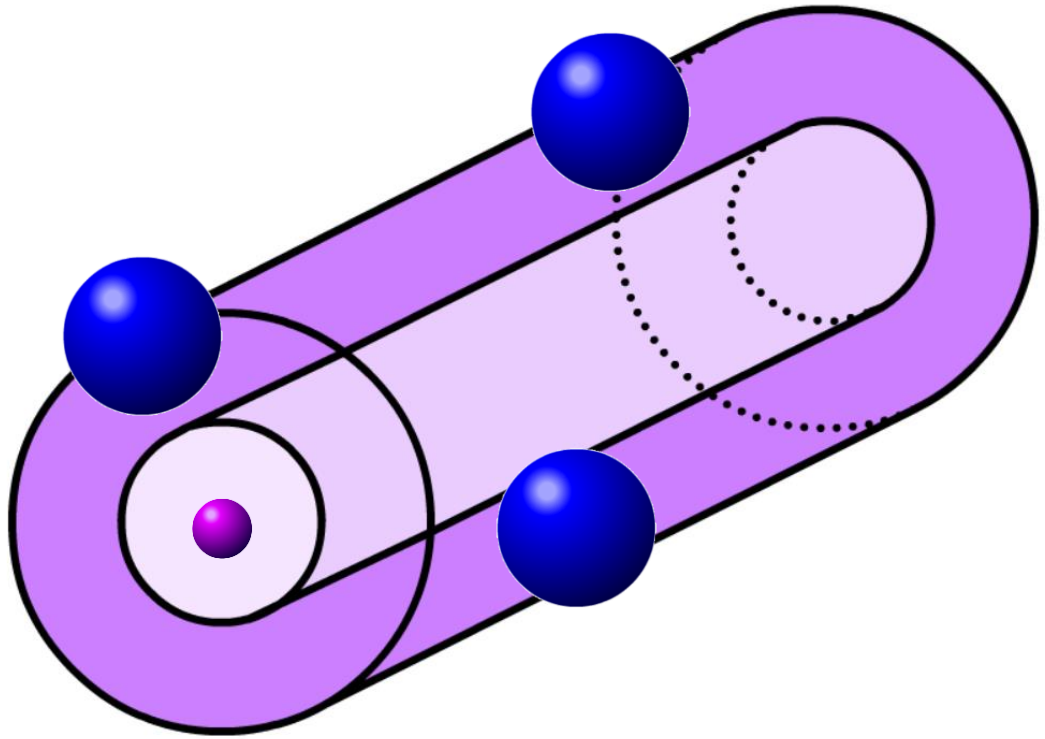
Nanbu-Pérez scatter

- Numerical experiment to determine scatter

- Randomly assign impact parameters inside cylinder
  - $b_{\max}$: Debye length λ_D
  - $b_{\min}$: Large angle scatter limit b_0

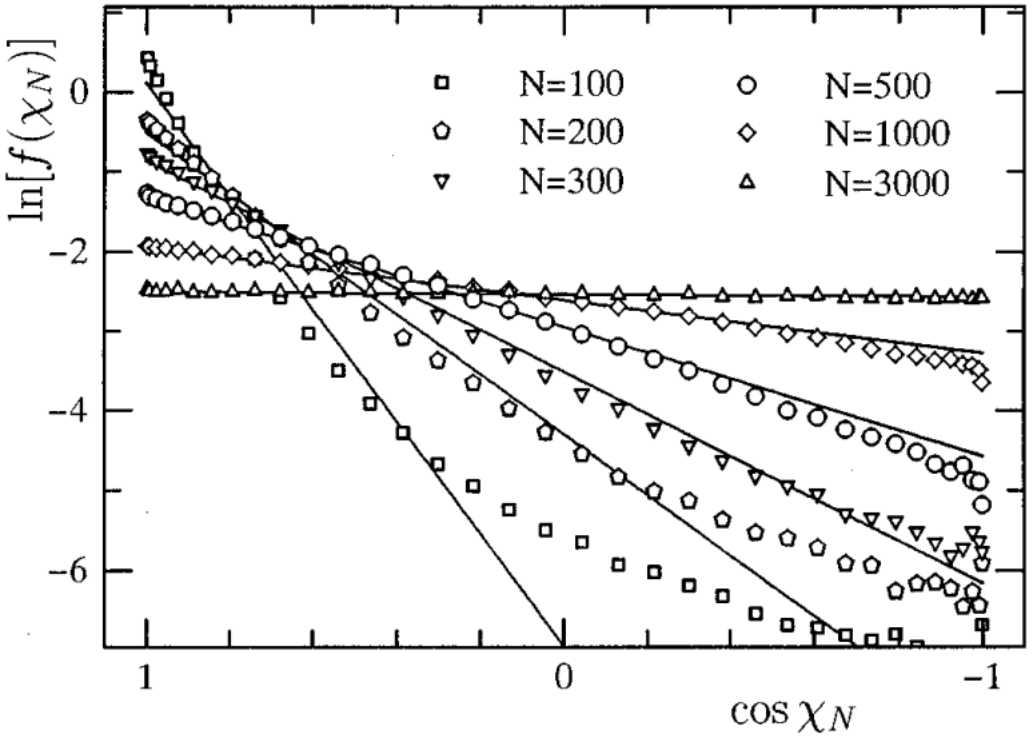
- Calculate cumulative scatter angles using single particle scatter:

$$\tan\frac{\theta_k}{2} = \frac{|q_\alpha q_\beta|}{4\pi\epsilon_0\mu v_r^2 b}$$



Nanbu-Pérez scatter

- $\cos\chi_N$: Cosine of cumulative scatter angle after N scatter events
- $\ln[f(\chi_N)]$: Probability distribution of scatter angle after N scatter events
  - Low N, peaked near small angle
  - High N, flat distribution: isotropic
- Apply linear fit to distributions
- In practice:
  - Weighted step: $s = n_\beta g \pi b_0^2 (\ln\Lambda)\Delta t$
  - Sampling term: $\coth A - A^{-1} = e^{-s}$
  - Angle: $\cos\chi = \frac{1}{A}\ln(e^{-A} + 2x_r \sinh A)$



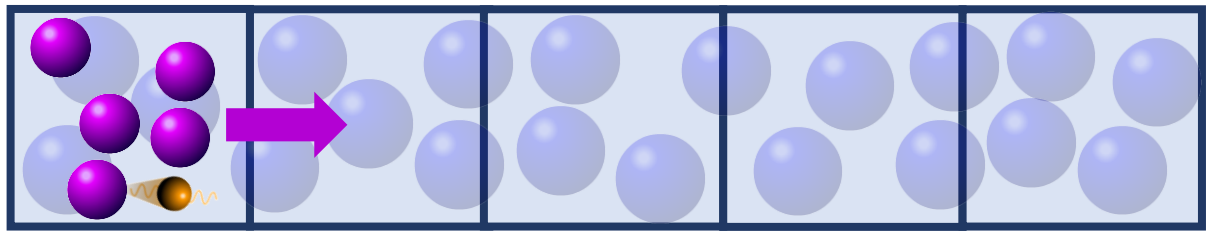
Example: Bremsstrahlung radiation

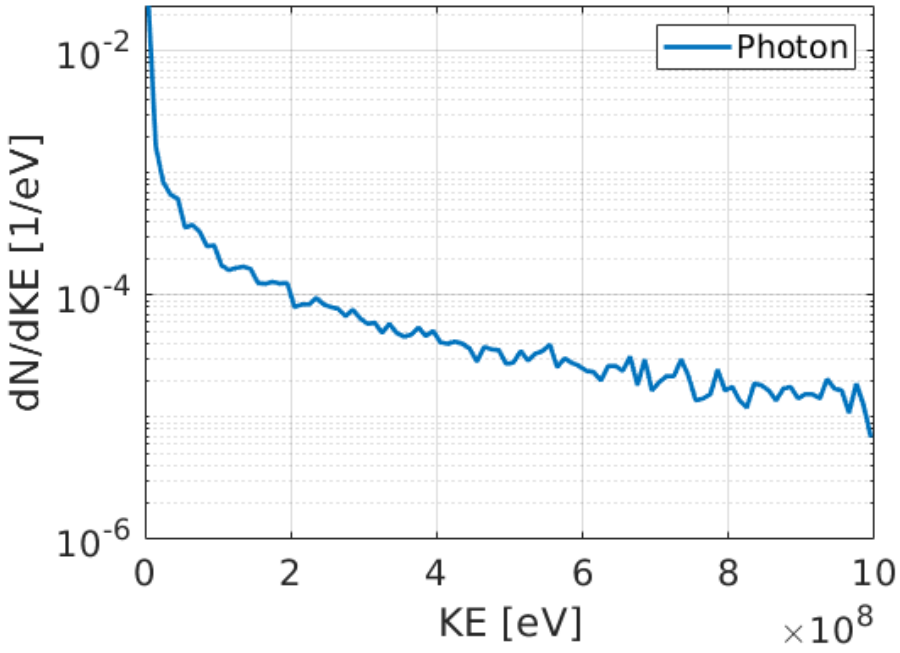
- 1 GeV e^- passing through 5.5 mm of Al, over 20 ps

- Bremsstrahlung block sets physics properties:
  - Always recoil electrons when they radiate
  - If emitted photon energy > 1 keV, create a macro-photon for it
  - Option to increase emission number, reducing macro-weight (not used)
  - Do not move photons, let them sit at creation point
  - Ignore theoretical σ increase for heated targets (plasma-screening)

- Plot KE distribution of radiated photons

```
begin:bremsstrahlung
  use_bremsstrahlung = T
  start_time = 0
  produce_photons = T
  photon_energy_min = 1 * kev
  photon_weight = 1.0
  photon_dynamics = F
  use_bremsstrahlung_recoil = T
  use_plasma_screening = F
end:bremsstrahlung
```



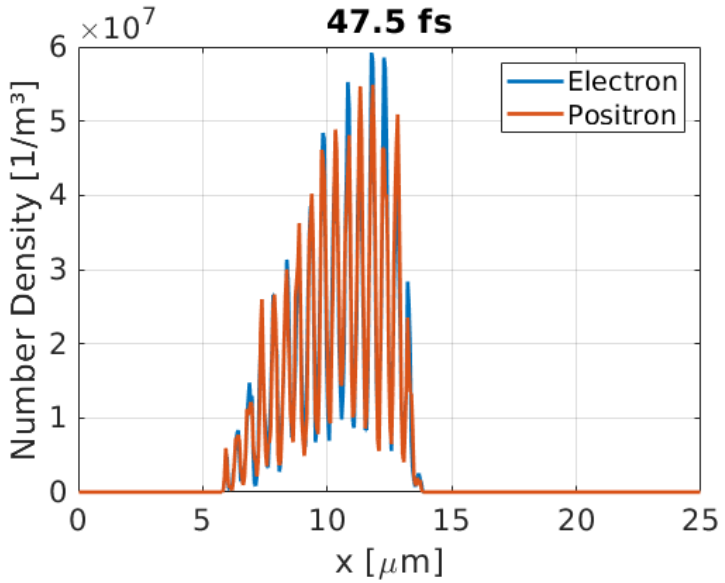


Example: Breit-Wheeler pair production

- 100 MeV photon bunch incident on a 2×10^{23} Wcm^{-2} laser

- qed block sets process properties:
  - Activate pair production
  - Allow photons to move
  - Also provides instructions for e^- NCS radiation, similar to brem.

- Plot number density of pair-particles near simulation end

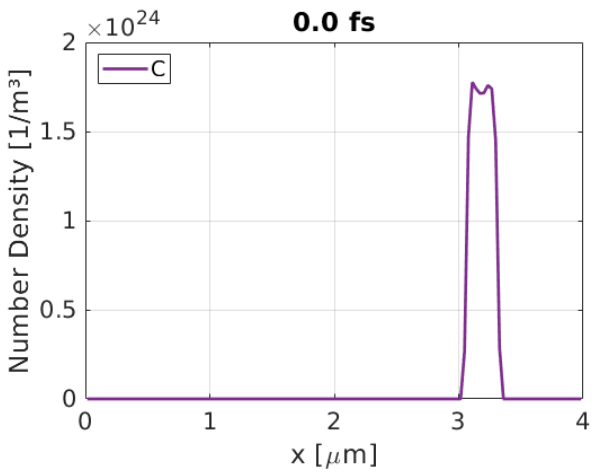


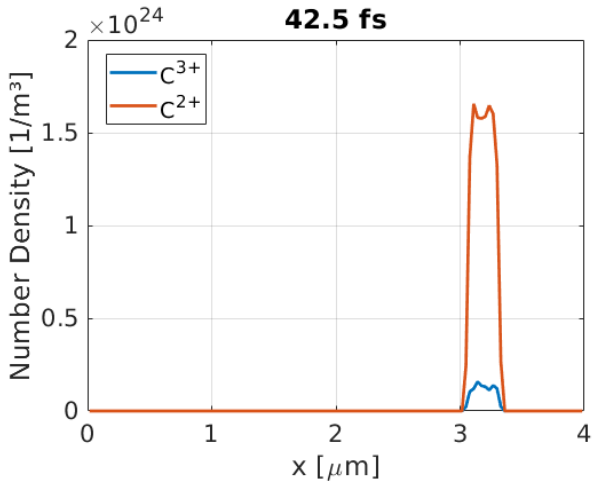

```
begin:qed
  use_qed = T
  produce_photons = T
  photon_energy_min = 50 * kev
  produce_pairs = T
  photon_dynamics = T
end:qed

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = open
end:boundaries

begin:laser
  boundary = x_min
  intensity_w_cm2 = 2.0e23
  lambda = 1.0e-6
end:laser

begin:species
  name = Photon_Beam
  frac = 1
  rho = if (x gt 20.0e-6, 1.0e10, 0)

  drift_x = -100.0e6 * ev / c

  identify:photon
end:species
```

Example: Field ionisation

- Shoot a 3×10^{15} Wcm^2 laser at a block of neutral C
  - Without field ionisation, the laser would pass through the target
  - (neutral targets have no currents for the field solver)
  - Switch on field ionisation in control block

- Set up Carbon species for ionisation
  - ionise: Mark Carbon as an ionisation base-species
  - ionise_limit: Number of ionised species to create (up to C^{3+} here)
  - unique_electron_species: Make a separate e^- species for each ionised charge-state





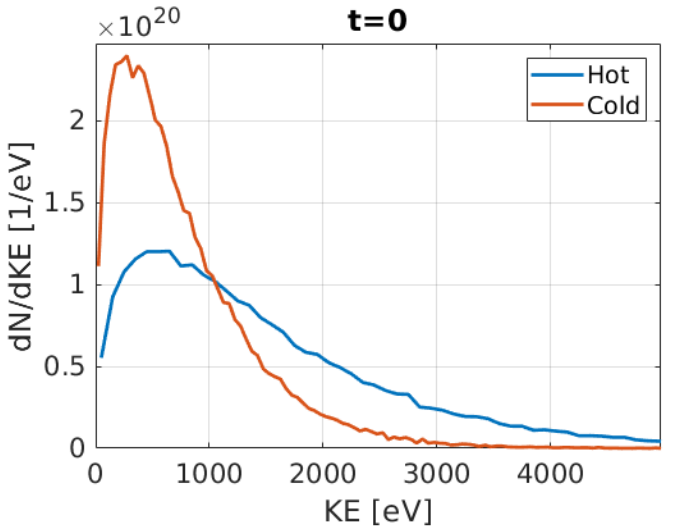

```
begin:control
  nx = 128
  npart = 20 * nx
  t_end = 42.4e-15
  x_min = 0
  x_max = 4.0e-6
  field_ionisation = T
end:control

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = open
end:boundaries

begin:laser
  boundary = x_min
  intensity = 3.0e15 * 1.0e4
  lambda = 800.0e-9
  t_profile = 1
  t_end = 10.7e-15
end:laser

begin:species
  name = Carbon
  charge = 0
  atomic_no = 6
  mass = 1836.2 * 16.0
  frac = 1
  rho = if ((x lt 3.30e-6) and (x gt 3.05e-6), 1.74e24, 0)
  ionise = T
  ionise_limit = 3
  unique_electron_species = T
end:species

begin:output
  dt_snapshot = t_end/100
  number_density = always + species
end:output
```

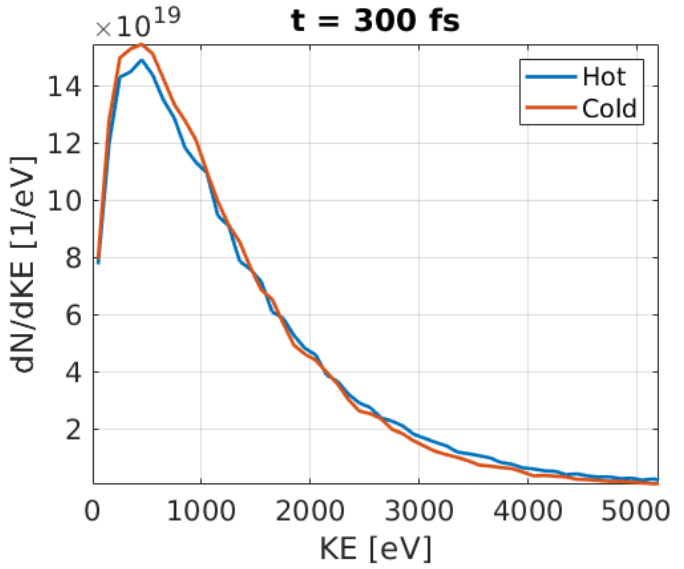
Example: Collisional equilibration

- Start with two overlapping e^- species
  - Initial temperatures: 1 keV, 500 eV
  - Switch on collisions, watch thermal equilibration

- Collisions:
  - collide: Identify which species pairs can collide
  - coulomb_log: Set to calculate $\ln(\Lambda)$ in each cell, each step. Very slow – much better to give an approximate value (e.g. 5)
  - coll_n_step: Run collisions once every n steps, for timestep $n\Delta t$. Speed-up technique for boosting collisions number

```
begin:collisions
   use_collisions = T
   use_nanbu = T
   coulomb_log = auto
   collide = all
   coll_n_step = 10
end:collisions
```





Section 5

Viking Simulations

Running EPOCH on clusters

- EPOCH has been run on different scales:
  - Small work-stations (1-8 cores)
  - Archer2 (up to 65k cores)

- On clusters:
  - Load FORTRAN and MPI modules
  - Download and compile as on the EPOCH Quick Start page
  - Write a Slurm jobscript, loading the same modules again

- Try to test code locally first:
  - Fewer particles per cell, or over 1 time-step
  - Make sure you've got the right simulation!

Data filtering techniques

- Big simulations have a lot of data! Keep it manageable
  - Probes take less memory than particle dumps
  - Distribution functions also better than particle dumps
  - Use subsets to minimise particles in particle dumps
  - Use multiple output blocks with different dt_snapshot

- Some vars support single-precision output, useful if binning data

- Some vars let you output values averaged over a few steps, to reduce noise

- See: https://epochpic.github.io/documentation/input_deck/input_deck_output_block.html

```
begin:output
   file_prefix = grid_vars
   name = regular_dump
   dt_snapshot = 10.0e-15
   ex = always + single
   ey = always + single
   ez = always + single
   bx = always + single
   by = always + single
   bz = always + single
   temperature = always + species + single
   poynt_flux = always + single
   number_density = always + species + single
end:output

begin:output
   file_prefix = particle_data
   name = particle_dump
   dt_snapshot = 100e-15
   px = always + single
   py = always + single
   particle_weight = always + single
end:output
```

```
begin:subset
   name = background
   random_fraction = 0.1
   include_species:electron
   include_species:proton
end:subset

begin:subset
   name = high_gamma
   gamma_min = 1.3
   include_species:electron
end:subset

begin:output
   particles = background + high_gamma + always
   px = background + high_gamma
   py = background
   pz = always
```

Restarting simulations

- In a SLURM job-script, you need to say how much time you need. What if you don't request enough?

- When a file called STOP appears in the input.deck directory:
  - EPOCH prints a restart dump file
  - Simulation is then stopped, STOP file deleted

- If you add this line to your jobscript:

```
(sleep 16200; touch STOP) &
```

  - STOP file created after simulation runs for 16200 seconds
  - Make this occur about an hour before your allocated cluster time ends

- This forces EPOCH to make a restart dump before you run out of compute time!
  - Simulation can be restarted in another submission

```
begin:control
  nx = 500
  ny = 500
  t_end = 100 * femto
  x_min = 0.0
  x_max = 10.0e-6
  y_min = -5.0e-6
  y_max = 5.0e-6
  stdout_frequency = 10

  restart_snapshot = 0003.sdf
end:control
```

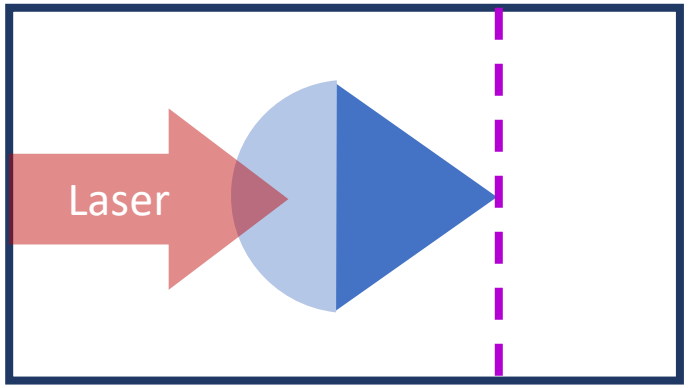
Viking setup

- We're going to run a simulation on Viking!
  - Need to log in to Viking
  - Clone over EPOCH following Quick Start instructions
  - Load correct modules
  - Obtain a job-script

- We'll need someone familiar with running EPOCH on Viking…

Challenge simulation

- Domain:
  - x: 0 to 20 μm
  - y: -10 to 10 μm
  - Cell size: $20\ \mathrm{nm} \times 20\ \mathrm{nm}$
- Laser:
  - Focal spot FWHM: $5\ \mu m$
  - Focus position: $(x, y) = (10, 0)\ \mu m$
  - Temporal FWHM: 40 fs
  - Wavelength $1\ \mu m$
  - Peak cycle-averaged intensity: $10^{22}\ \mathrm{Wcm^{-2}}$
- Target:
  - Fully ionised carbon, $n_{i0} = 6 \times 10^{28}\ \mathrm{m^{-3}}$, with e^-
  - Two parts:
    1. Triangle, n_{i0} density, vertices: $(10, \pm 2)$, (12,0) μm
    2. Semi-circle, radius $2\mu m$, meeting triangle
  - Circle density: $n_{i0} e^{(x-10\mu m)/1\mu m}$
  - Temperature: 1 keV for e^- and C
- Physics:
  - NCS, min photon energy 500 keV
  - Allow photon dynamics
- Output:
  - Probes at $12\ \mu m$ for e^- over 100 keV, and all C^{6+}, γ
  - Number density of each species
  - Poynting flux
- Running:
  - End time: 150 fs
  - This should take under 10 minutes on 48 cores



Challenge implementation

WARWICK

```
begin:control
    nx = 1000
    ny = 1000
    t_end = 150e-15
    x_min = 0
    x_max = 20e-6
    y_min = -10e-6
    y_max = 10e-6
    stdout_frequency = 10
    smooth_currents = T
end:control

begin:boundaries
    bc_x_min = simple_laser
    bc_x_max = open
    bc_y_min = open
    bc_y_max = open
end:boundaries

begin:qed
    use_qed = T
    produce_photons = T
    photon_energy_min = 500 * kev
    produce_pairs = F
    photon_dynamics = T
end:qed

begin:constant
    I_fwhm = 5.0e-6          # FWHM of laser intensity
    I_peak_Wcm2 = 1.0e22     # 0.5 * eps0 * c * E_peak^2
    las_lambda = 1.0e-6      # Laser wavelength
    foc_dist = 10.0e-6       # Boundary to focal point distance
    t_fwhm = 40.0e-15        # Temporal FWHM
end:constant

begin:constant
    las_k = 2.0 * pi / las_lambda
    w0 = I_fwhm / sqrt(2.0 * loge(2.0))                  # Beam Waist
    ray_rang = pi * w0^2 / las_lambda                    # Rayleigh range
    w_boundary = w0 * sqrt(1.0 + (foc_dist/ray_rang)^2)  # Waist on boundary
    I_boundary = I_peak_Wcm2 * (w0 / w_boundary)^2       # Intens. on boundary
    rad_curve = foc_dist * (1.0 + (ray_rang/foc_dist)^2) # Boundary curv. rad.
    gouy = atan(-foc_dist/rad_curve)                     # Boundary Gouy shift

    w_t = t_fwhm / sqrt(2*loge(2))
    t_hw01m = 0.5 * t_fwhm * sqrt(loge(10)/loge(2))
end:constant
```

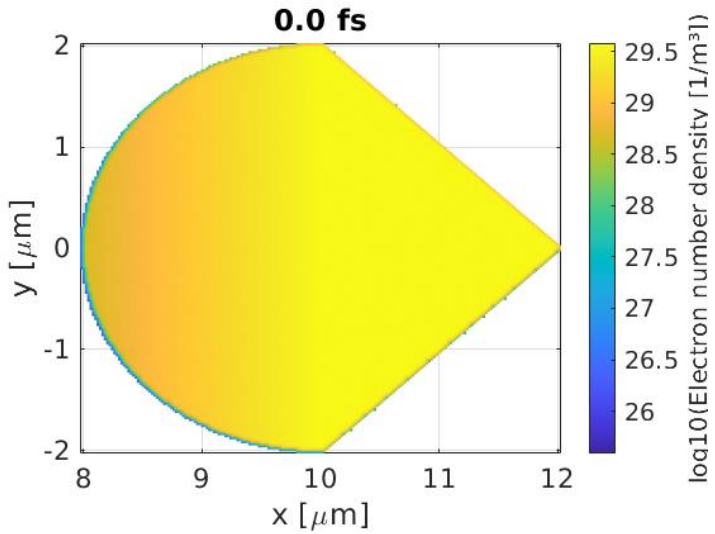
```
begin:laser
    boundary = x_min
    intensity_w_cm2 = I_boundary
    lambda = las_lambda
    phase = las_k * y^2 / (2.0 * rad_curve) - gouy
    profile = gauss(y, 0, w_boundary)
    t_profile = gauss(time,t_hw01m,w_t)
end:laser

begin:constant
    circ_rad = 2.0e-6
    ni0 = 6.0e28
    ne0 = ni0 * 6
    plas_scale = 1.0e-6
end:constant

begin:species
    name = Electron

    # Front circle
    density = if ((x-10.0e-6)^2 + y^2 lt circ_rad^2, \
                  ne0 * exp((x-10e-6)/plas_scale), \
                  0)
    density = if(x gt 10.0e-6, 0, density(Electron))

    # Rear triangle
    density = if ((x gt 10.0e-6) and abs(y) lt -(x-12.0e-6), \
                 ne0, \
                 density(Electron))

    temp_ev = 1.0e3

    npart = 2.5e6

    identify:electron
end:species

begin:species
    name = C
    mass = 22033
    charge = 6

    density = density(Electron) / 6
    temp_ev = 1.0e3

    npart = 1e6
end:species
```

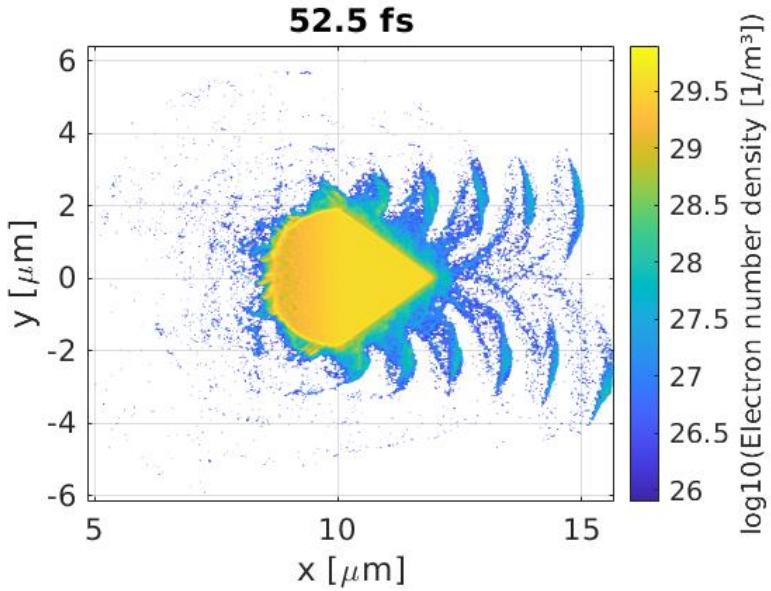
```
begin:species
    name = Photon
    npart = 0
    identify:photon
end:species

begin:output
    name = o1
    dt_snapshot = t_end/20
    poynt_flux = always
    ex = always
    ey = always
    ez = always
    number_density = always + species
    temperature = always + species
    particle_probes = always
end:output

begin:probe
   name = Electron_probe
   point = (12.0e-6, 0.0)
   normal = (1.0, 0.0)
   ek_min = 100 * kev
   ek_max = -1.0
   include_species:Electron
   dumpmask = always
end:probe

begin:probe
   name = Carbon_probe
   point = (12.0e-6, 0.0)
   normal = (1.0, 0.0)
   ek_min = 0.0
   ek_max = -1.0
   include_species:C
   dumpmask = always
end:probe

begin:probe
   name = Photon_probe
   point = (12.0e-6, 0.0)
   normal = (1.0, 0.0)
   ek_min = 0.0
   ek_max = -1.0
   include_species:Photon
   dumpmask = always
end:probe
```

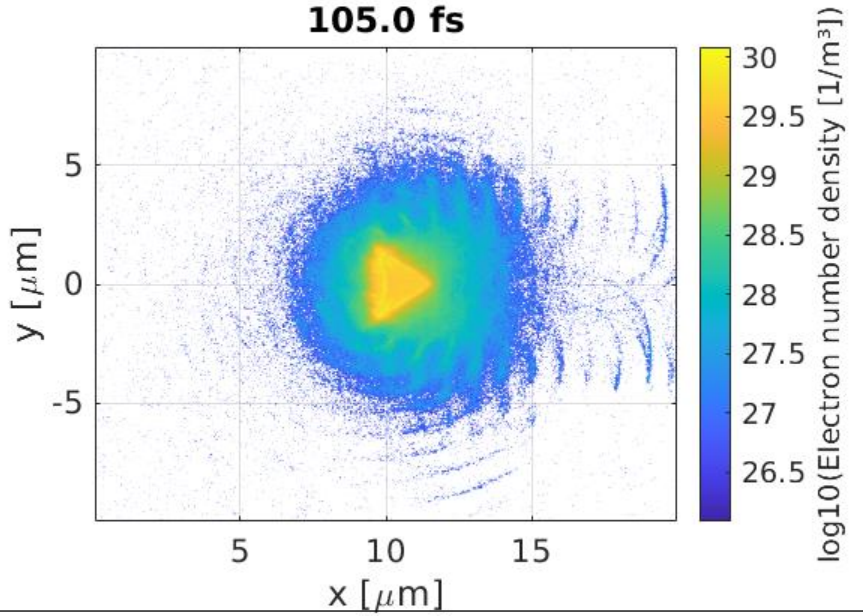
Electron density (log scale)

- e^- pre-plasma is compressed by laser
  - Hot e^- ejected in bunches along laser direction
  - De-compression around centre at later times

- e^- are light enough to be carried by the laser, do we see these beams with ions?



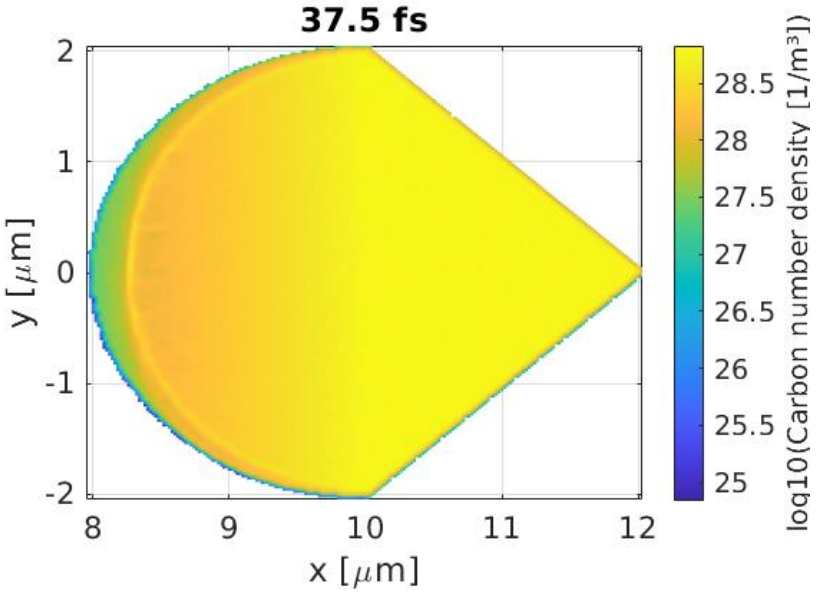


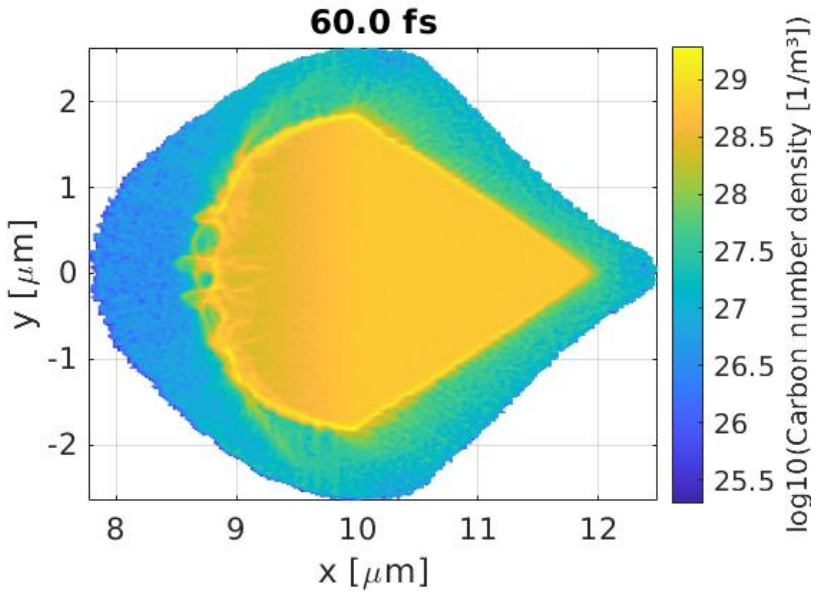


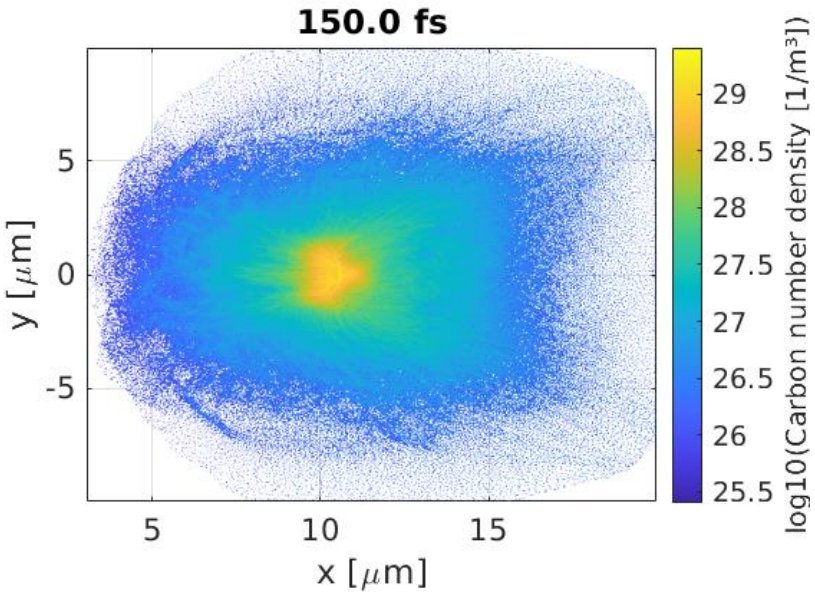
Ion density (log scale)

- Ions start undergoing compression
  - Instability forms on compression front at later times
  - Ions begin ejection from the target

- Simulation end:
  - Full target decompression occurs, like for e^- density

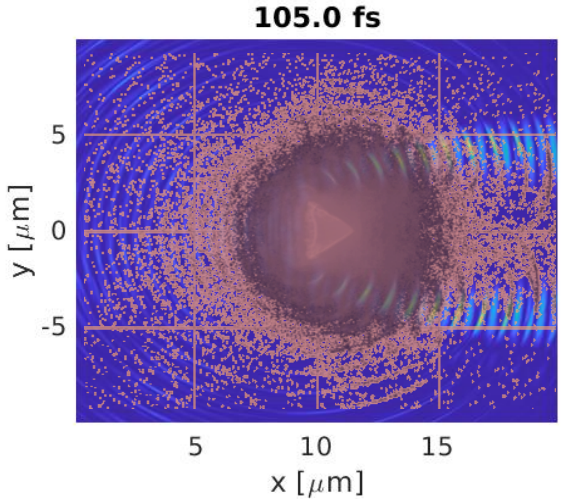


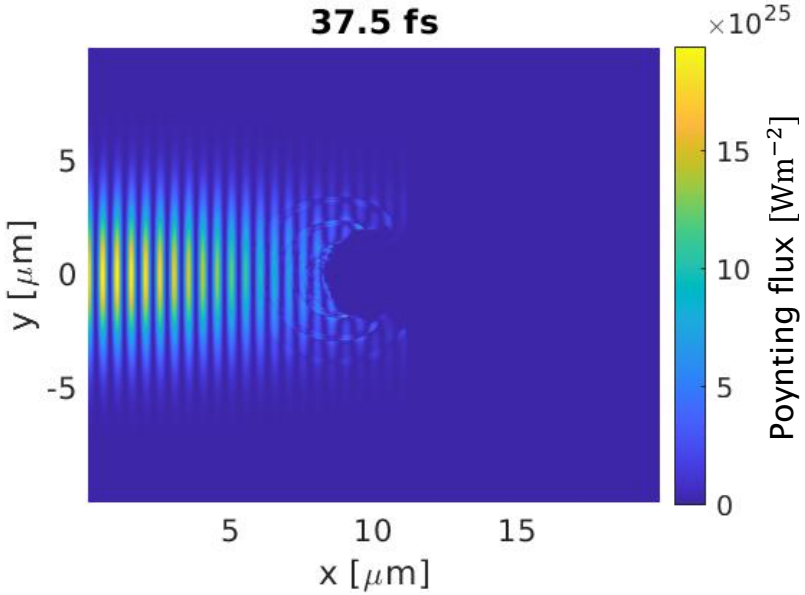


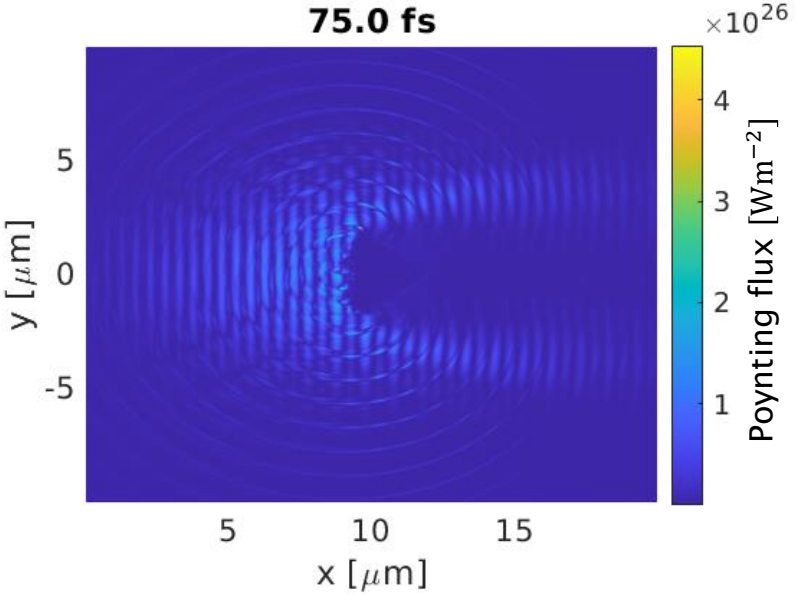


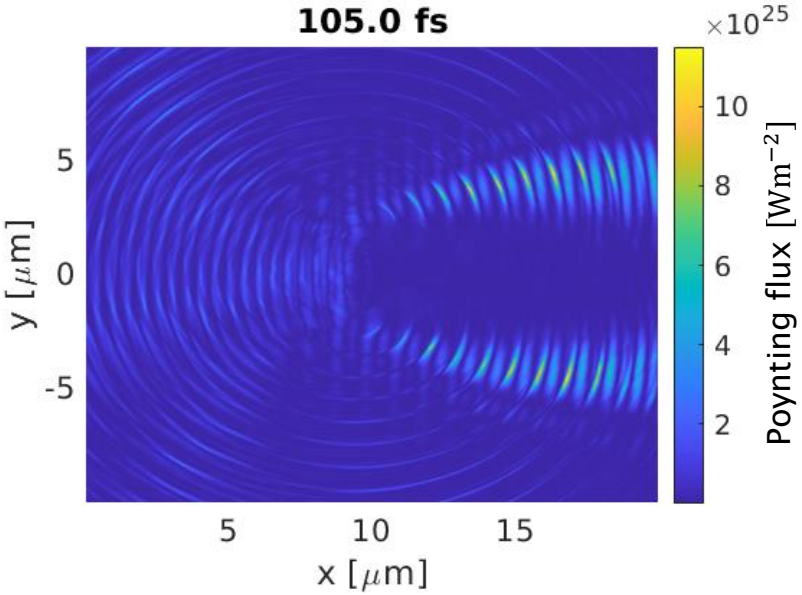
Poynting flux

- Laser propagation blocked by target over critical density
  - Some wave-fronts reflected radially outwards
  - Focal spot larger than target, some laser energy passes round
- Compound figure shows e^- density and Poynting flux
  - Laser passes through lower density e^- on target edge





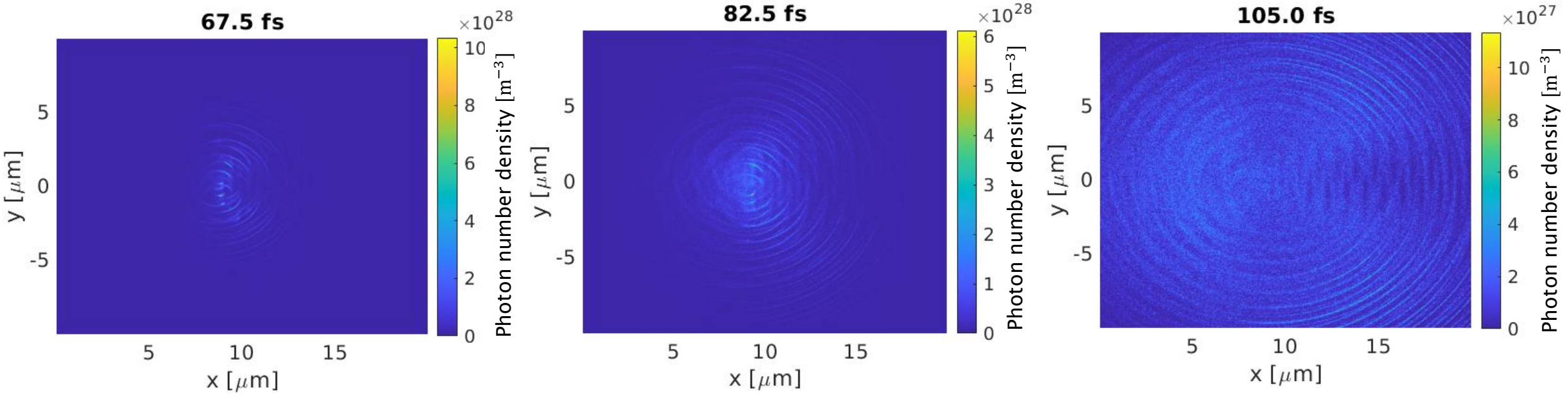




Photon emission

- Photons originate on the front of the circular pre-plasma
  - These propagate radially outwards
  - Highest photon densities in directions about $\pm 50°$ from laser axis
  - Distribution more isotropic at later times

- Ripple effects likely coincide with peak laser field strength, over the laser wavelength



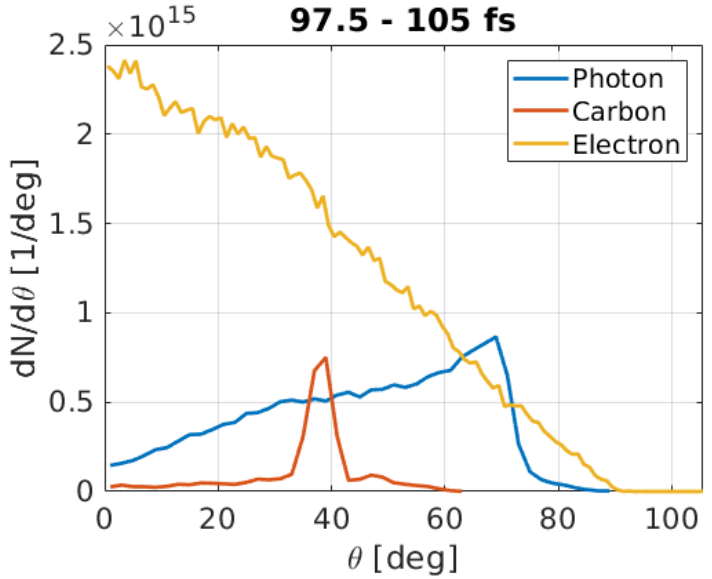
Probe snapshot (beam spread)

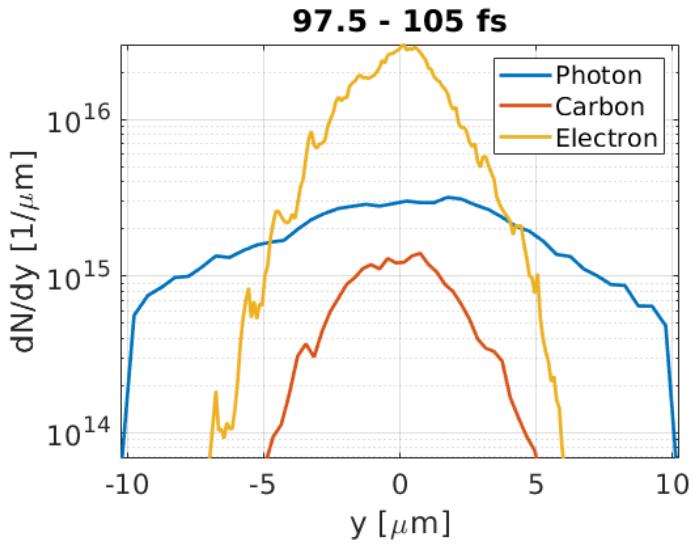
- Properties of particles passing probe between 97.5 and 105 fs:

- Angular:
  - e^- distribution peaks in laser direction
  - Ions emerge $40°$, normal to rear surfaces
  - Photons mostly have high angle

- Spatial:
  - Some e^- spread
  - High γ spread
  - Little spread for C

- High emission angle and low spread suggests C^{6+} originate close to probe





Probe snapshot (energies)

- Different species have different energies
  - Electrons reach up to 10-100 MeV
  - Photons on the order of 1-10 MeV
  - Carbon ions exceeding 100 MeV

- Different snapshots may have different distributions
  - Ion beams: target-normal sheath-acceleration
  - Ejected e^- set up strong fields on target-surfaces