Plasmon Particle-in-Cell Solver

Jan Paszkiewicz

14/8/2020

Summary of Features

- Solution of Maxwell’s equations with the Finite-Difference Time-Domain (FDTD) method.
- Uniform grid of rectangular cells.
- Approximates all derivatives as central differences.
- 2D solver with TM polarisation
  - All field components except E_x, B_y, and E_z are zero.
- Particle dynamics including special relativity
- Ability to simulate materials with a Drude-type response.
- Implemented in Python for simplicity.
  - Tested up to 160000 cells and 30000 time steps.
  - Potential for speed improvement by just-in-time compilation with the Numba package.

Maxwell’s Equations for Reference:

Gauss’ law:	$\nabla \cdot \boldsymbol{E} = \frac{\rho}{\varepsilon}$
Gauss’ law for magnetism:	$\nabla \cdot \boldsymbol{B} = 0$
Faraday’s law:	$\nabla \times \boldsymbol{E} = \frac{\partial \boldsymbol{B}}{\partial t}$
Ampere-Maxwell law:	$\nabla \times \boldsymbol{B} = \mu\left(\varepsilon \frac{\partial \boldsymbol{E}}{\partial t} + \boldsymbol{J}\right)$

Central Difference Approximation

- The central difference scheme approximates derivatives as:

$$\frac{\partial f}{\partial x}(x) \approx \frac{f\left(x + {}^{\Delta x}/_{2}\right) - f\left(x - {}^{\Delta x}/_{2}\right)}{\Delta x}$$

- Note the half steps! This results in an error that scales as $O(\Delta x^2)$.
- Compare this with, for example, a forward difference scheme:

$$\frac{\partial f}{\partial x}(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

- This gives errors which scale as $O(\Delta x)$, which means that the central-difference method rapidly becomes better as $\Delta x \rightarrow 0$
- The half-step in the central difference scheme means that certain quantities will be evaluated on different spatial grids and different times from others.

Maxwell’s Equations in FDTD

- We can rearrange Faraday’s law and the Ampere-Maxwell law as follows:

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla \times \boldsymbol{E} \qquad \frac{\partial \boldsymbol{E}}{\partial t} = \frac{1}{\varepsilon}\left(\frac{1}{\mu}\nabla \times \boldsymbol{B} - \boldsymbol{J}\right)$$

- And then express them in discrete time:

$$\boldsymbol{B}\left(t_{n+1/2}\right) = \boldsymbol{B}\left(t_{n-1/2}\right) + \nabla \times \boldsymbol{E}(t_n) \cdot \Delta t$$

$$\boldsymbol{E}(t_{n+1}) = \boldsymbol{E}(t_n) + \frac{1}{\varepsilon}\left(\frac{1}{\mu}\nabla \times \boldsymbol{B}\left(t_{n+1/2}\right) - \boldsymbol{J}\left(t_{n+1/2}\right)\right) \cdot \Delta t$$

Where: $\nabla \times \boldsymbol{E} = \frac{\partial \boldsymbol{E}_z}{\partial x} - \frac{\partial \boldsymbol{E}_x}{\partial z}$, $(\nabla \times \boldsymbol{B})_x = -\frac{\partial \boldsymbol{B}_y}{\partial z}$, $(\nabla \times \boldsymbol{B})_z = \frac{\partial \boldsymbol{B}_y}{\partial x}$

- These are solved alternately to propagate the field values forward in time.
- Note that the $\boldsymbol{E}$ field is sampled at times $t_0, t_1, \dots, t_n$, while the $\boldsymbol{B}$ and $\boldsymbol{J}$ fields are sampled at times $t_{1/2}, t_{1+1/2}, \dots, t_{n+1/2}$.

Gauss' Law

- The FDTD solver does not explicitly ensure Gauss' law for either magnetic or electric fields is valid. However, we can show that:

$$\nabla \cdot \left(\frac{\partial \boldsymbol{B}}{\partial t}\right) = \frac{\partial}{\partial t}(\nabla \cdot \boldsymbol{B}) = \nabla \cdot (\nabla \times \boldsymbol{E}) = 0$$

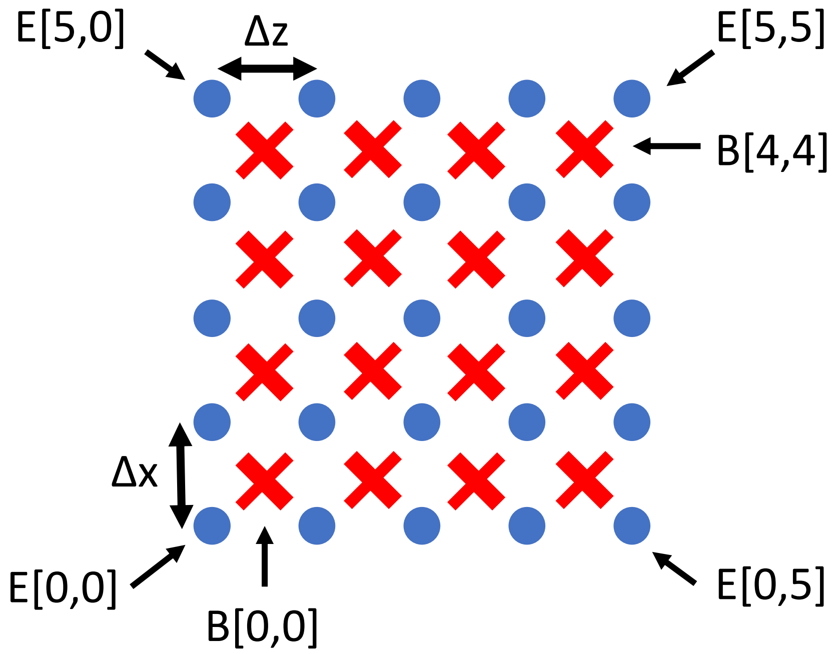
- Due to the vector identity $\nabla \cdot (\nabla \times \boldsymbol{E}) = 0$ for any $\boldsymbol{E}$
- This means that if the initial conditions satisfy $\nabla \cdot \boldsymbol{B} = 0$ then the solution will as well. The same can be applied to the electric field:

$$\nabla \cdot \left(\frac{\partial \boldsymbol{E}}{\partial t}\right) = \frac{\partial}{\partial t}(\nabla \cdot \boldsymbol{E}) = \frac{1}{\mu\varepsilon}\nabla \cdot (\nabla \times \boldsymbol{B}) - \frac{1}{\varepsilon}\nabla \cdot \boldsymbol{J} = -\frac{1}{\varepsilon}\nabla \cdot \boldsymbol{J}$$

- If $\nabla \cdot \boldsymbol{J} = -\frac{\partial \rho}{\partial t}$ and the initial conditions satisfy Gauss' law, then the solution will as well.
- $\nabla \cdot \boldsymbol{J} = -\frac{\partial \rho}{\partial t}$ is a statement of charge continuity, meaning that $\boldsymbol{J}$ must reflect the movement of charges in the domain.
- Care must be taken by the user to ensure that the initial conditions satisfy Gauss' law and that charge continuity is preserved!

Discretisation in Space

- To implement a central-difference approximation for space derivatives, the B and E fields are evaluated at different locations as well as different times, using a Yee grid.
  - Blue dots represent the E-grid and red crosses represent the B-grid.
- The origin of the coordinate system is defined as the bottom-left corner of the E-grid, $E[0{,}0]$.
- To update the E and B fields at each point, information from the four neighbouring points is used. For example, to get the z-component of $\nabla \times \boldsymbol{B}$ at the location of $E[i,j]$ we use:



$$(\nabla \times B)_z = \frac{\partial B_y}{\partial x} \approx \frac{\left(B_y[i,j] + B_y[i,j-1]\right) - \left(B_y[i-1,j] + B_y[i-1,j-1]\right)}{2\Delta x}$$

Boundary Conditions

- Stepping the field at each point forward in time requires information from the neighbouring points.
- This poses a problem for the points on the edge of the grid. Special treatment is required in the form of boundary conditions. Three options are available for each boundary:
  - Perfect magnetic conductor (PMC):
    - Forces $B_y = 0$ on the edge points of the B-grid. The edge points on the E-grid become redundant.
  - Perfect electrical conductor (PEC):
    - Forces the tangential component of E to zero on the edge points. The normal component of E is calculated by assuming that the B field is symmetric about the boundary, i.e for a boundary at $i = 0$, assume $B_y[-1, j] = B_y[0, j]$
  - Periodic boundary:
    - The E field on two opposite edges of the domain is the same, i.e for a boundary at $i = 0$, $\mathbf{E}[N_x - 1, j] = \boldsymbol{E}[0, j]$. The E-field can thus be propagated in time as normal assuming the domain loops round the boundary. Nothing special is done to the B field.

Conducting Materials

- One way to implement an absorbing boundary is to add a region of finite conductivity to act as a termination.
- Tapering the conductivity of the absorber with distance can also help to reduce reflections.
- Conductivity can also be used to represent a material with complex permittivity, using the relation $\mathrm{Im}\{\varepsilon\} = {}^{\sigma}/_{\omega}$
- A conductive material can be implemented by using the relation $\boldsymbol{J} = \sigma\boldsymbol{E}$. However, this requires knowledge of $\boldsymbol{E}$ at a non-integer timestep, ie:

$$\boldsymbol{J}\left(t_{n+{}^{1}/_{2}}\right) = \sigma\boldsymbol{E}\left(t_{n+{}^{1}/_{2}}\right)$$

- This is achieved in this solver by first calculating $\boldsymbol{E}(t_{n+1})$ ignoring conductive media, then using this value to obtain a corrected $\boldsymbol{E}'(t_{n+1})$ which does take conductivity into account:

$$\boldsymbol{E}'(t_{n+1}) = \boldsymbol{E}(t_{n+1}) - \frac{\Delta t}{\varepsilon}\boldsymbol{J}\left(t_{n+{}^{1}/_{2}}\right) \approx \boldsymbol{E}(t_{n+1}) - \frac{\Delta t\sigma}{2\varepsilon}\left(\boldsymbol{E}(t_n) + \boldsymbol{E}(t_{n+1})\right)$$

Drude-Type Materials

- The Drude model treats conductive materials as plasmas, which have a finite response time to applied fields due to the inertia of the electrons. They are characterised by a plasma frequency ω_p and collision frequency f_c. This delay allows phenomena such as plasmons to exist on the interface between a metal and a dielectric.
- The frequency-domain relative permittivity of such a material is given by:

$$\varepsilon_r(\omega) = 1 - \frac{\omega_p^2}{\omega(\omega + i\, f_c)}$$

- In the time domain, this can also be represented by the following relationship between the $\boldsymbol{D}$ and $\boldsymbol{E}$ fields:

$$\boldsymbol{D}(t) = \varepsilon_0(\boldsymbol{E}(t) + \chi(t) * \boldsymbol{E}(t)), \quad \chi(t) = \begin{cases} \frac{\omega_p^2}{f_c}\left(1 - e^{-f_c t}\right), & t \geq 0 \\ 0, & t < 0 \end{cases}$$

Drude-Type Materials

- This response can be represented in discrete time as:

$$\boldsymbol{D}(t_n) = \varepsilon_0 \left(\boldsymbol{E}(t_n) + \frac{\omega_p^2 \Delta t}{f_c} \left(\sum_{i=0}^{n} \boldsymbol{E}(t_i) - \sum_{i=0}^{n} \boldsymbol{E}(t_i) e^{-f_c \Delta t\,(n-i)} \right) \right)$$

- Since we are interested in calculating the time derivative of $\boldsymbol{D}$, we can substitute the above equation into the Maxwell-Ampere law:

$$\boldsymbol{D}(t_{n+1}) = \boldsymbol{D}(t_n) + \left(\frac{1}{\mu} \nabla \times \boldsymbol{B}\left(t_{n+1/2}\right) - \boldsymbol{J}\left(t_{n+1/2}\right) \right) \cdot \Delta t$$

- And get, assuming $f_c \Delta t \ll 1$:

$$\boldsymbol{E}(t_{n+1}) = \boldsymbol{E}(t_n) - \omega_p^2 \Delta t^2 \sum_{i=0}^{n} e^{-f_c \Delta t (n-i)} \boldsymbol{E}(t_i) + \frac{1}{\varepsilon} \left(\frac{1}{\mu} \nabla \times \boldsymbol{B}\left(t_{n+1/2}\right) - \boldsymbol{J}\left(t_{n+1/2}\right) \right) \cdot \Delta t$$

- To avoid having to read the entire history of the E field to calculate the next step, the term $\omega_p^2 \Delta t^2 \sum_{i=0}^{n} e^{-f_c \Delta t (n-i)} \boldsymbol{E}(t_i)$ can be replaced by $\omega_p^2 \Delta t^2 \boldsymbol{S}(t_n)$, where $\boldsymbol{S}$ is a vector field updated at each time step by:

$$\boldsymbol{S}(t_{n+1}) = e^{-f_c \Delta t} \boldsymbol{S}(t_n) + \boldsymbol{E}(t_n)$$

- $\varepsilon \omega_p^2 \Delta t^2 \boldsymbol{S}$ gives the change of polarisation density $\boldsymbol{P}$ in one time step.

Particle Dynamics

- The same idea of central differences as used in calculating the fields is applied to calculating the position and velocity of the particles.
- The particle positions are NOT quantised to the grid.
- A simplified version of the Boris algorithm is used to calculate the electromagnetic acceleration due to the Lorentz force $\boldsymbol{F} = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B})$.
  - The particle is first accelerated by the electric force for half a time step, then rotated by the magnetic force using this value of velocity, then finally accelerated by the electric force for another half time step.
  - Because of this, the particle velocity is calculated twice per time step (every integer and half-integer time)
- The fields acting on the particle are obtained by linear interpolation of the E and B fields at the four grid points surrounding the current location of the particle. (Keeping in mind the different grids used for the E and B fields!)

Particle Dynamics

- The equations used to move the particles are:

$$\boldsymbol{p}(t_n) = \boldsymbol{p}\left(t_{n-1/_2}\right) + \frac{q\Delta t}{2}\boldsymbol{E}(t_n)$$

$$\boldsymbol{p}\left(t_{n+1/_2}\right) = \boldsymbol{p}(t_n) + \frac{q\Delta t}{2}\boldsymbol{p}(t_n) \times \left(\boldsymbol{B}\left(t_{n-1/_2}\right) + \boldsymbol{B}\left(t_{n+1/_2}\right)\right) + \frac{q\Delta t}{2}\boldsymbol{E}(t_n)$$

$$\boldsymbol{x}(t_{n+1}) = \boldsymbol{x}(t_n) + \Delta t\, \boldsymbol{v}\left(t_{n+1/_2}\right)$$

- The velocity and momentum of a particle are related by:

$$\boldsymbol{p} = \frac{\boldsymbol{v}}{|\boldsymbol{v}|} \cdot \frac{m|\boldsymbol{v}|}{\sqrt{1-\left(\frac{|\boldsymbol{v}|}{c}\right)^2}}, \qquad v = \frac{\boldsymbol{p}}{|\boldsymbol{p}|} \cdot \frac{|\boldsymbol{p}|}{m\sqrt{1+\left(\frac{|\boldsymbol{p}|}{mc}\right)^2}}$$

Particle Fields

- In order for the particles to be able to influence the fields, the charge density (for the electrostatic solver) and the current density (for the FDTD solver) of each of the particles is rasterised onto the E-grid.
- The charge density due to each particle is divided amongst a number of grid points around it, weighted by distance to the particle with a truncated Gaussian function.
- Obtaining a value for charge density also requires a value for depth of the domain Δy.
- The current density is obtained by multiplying the charge by the velocity of the particle:

$$\Delta \boldsymbol{J}\left(t_{n+1/2}\right) = \frac{1}{2}\left(\Delta\rho(t_{n+1}) + \Delta\rho(t_n)\right) \boldsymbol{v}\left(t_{n+1/2}\right)$$

Courant Criterion

- The FDTD method becomes unstable if the time step is too large, which is determined by the speed of propagation of information through the domain.
- Each step forward in time causes information to propagate by one cell in the x and z directions, leading to velocities $v_x = \frac{\Delta x}{\Delta t}, v_z = \frac{\Delta z}{\Delta t}$
- For stability, this velocity must exceed the speed of light in all directions including diagonals. In two dimensions, this leads to a maximum time step of:

$$\Delta t < \min(\Delta x, \Delta z)\sqrt{\mu\varepsilon/2}$$

- Where $\min(\Delta x, \Delta z)$ refers to the smaller of the two values $\Delta x, \Delta z$.

Order of Operations

In each time step, operations are performed in the following order:

1. Update the B-field using Faraday’s law
2. Modify the B-field at any PMC boundaries
3. Update particle velocities and positions
4. Obtain the current density from the particle distribution
5. Update the E-field using the Ampere-Maxwell law
6. Calculate the E-field at any periodic boundaries
7. Correct the E-field in conductive media
8. Modify the E-field at any PEC boundaries
9. Update the S vector at any Drude-type media

Electrostatic Solver

- As mentioned earlier, the FDTD method only obeys Gauss' law if the initial conditions do. An electrostatic solver is provided to help with this.
- The Poisson equation $\nabla^2 V = {}^{\rho}/_{\varepsilon}$ is solved iteratively using the Jacobi method:

$$V_{n+1}[i,j] = \frac{1}{4}\left(V_n[i+1,j] + V_n[i-1,j] + V_n[i,j+1] + V_n[i,j-1]\right) - \frac{\Delta x^2 \rho[i,j]}{4\varepsilon[i,j]}$$

- Where $V_n[i,j]$ refers to the potential at location [i,j] at the nth iteration.
- Source potentials (such as ground) are forced to the desired value at each iteration.
- The potentials are iteratively calculated until the potential field is self-consistent (convergence).

Electrostatic Solver

- Modification of Jacobi method for non-square cells:

$$V_{n+1}[i,j] = \frac{\Delta z}{2(\Delta x + \Delta z)}(V_n[i+1,j] + V_n[i-1,j]) + \frac{\Delta x}{2(\Delta x + \Delta z)}(V_n[i,j+1] + V_n[i,j-1]) - \frac{\Delta x \Delta z \rho[i,j]}{4\varepsilon[i,j]}$$

- The E-field is obtained by taking the gradient of the potential. A derivative over two cells is taken to maintain a central difference scheme despite V being rasterised to the E-grid (needed because ρ and ε are also on the E-grid):

$$E_x[i,j] = \frac{V[i+1,j]-V[i-1,j]}{2\Delta x},\ E_z[i,j] = \frac{V[i,j+1]-V[i,j-1]}{2\Delta z}$$

Python Classes

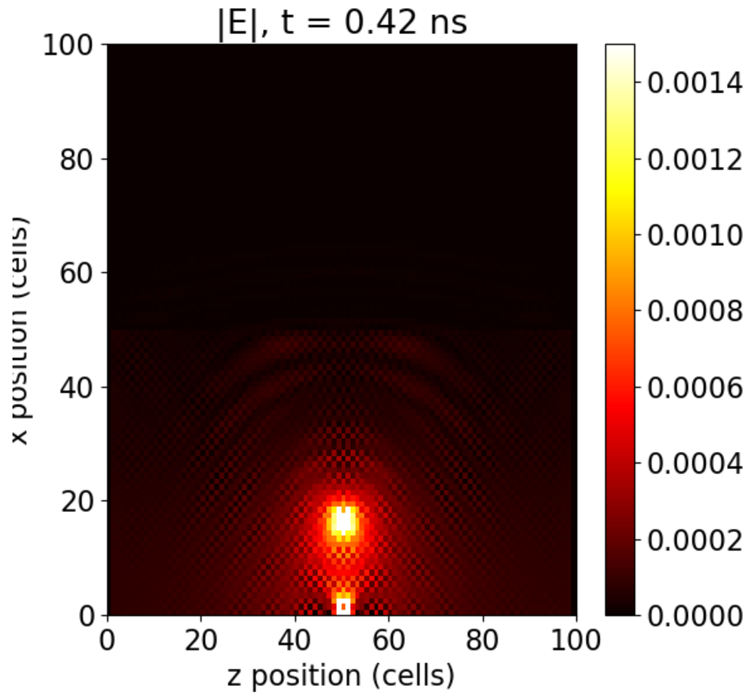
- pic_solver.constants()
  - Contains some physical constants: $\varepsilon_0, \mu_0, q_{electron}, m_{electron}$
- pic_solver.params(Nx, Nz, dx, dy, dz, dt)
  - Contains information about simulation parameters (domain size, step sizes)
- pic_solver.geometry(bc_xmin, bc_xmax, bc_zmin, bc_zmax, eps_r, sigma, w_p, f_c)
  - Contains information about the geometry of the domain
  - The first four parameters define the boundary conditions, and can have the values 'pec', 'pmc', or 'per'.
  - The remaining four are 2D arrays specifying the material properties of each mesh cell: $\varepsilon_r, \sigma, \omega_p, f_c$
  - Setting $\omega_p = 0$ turns off the Drude response
- pic_solver.particle(pos_x, pos_z, vx, vz, q, m)
  - Represents one particle in the simulation.
  - Initisalised with an initial position, velocity, charge, and mass.
  - The variables 'pos_x', 'pos_z', 'vx', 'vz' are lists tracking the particle's position and velocity at each time step.

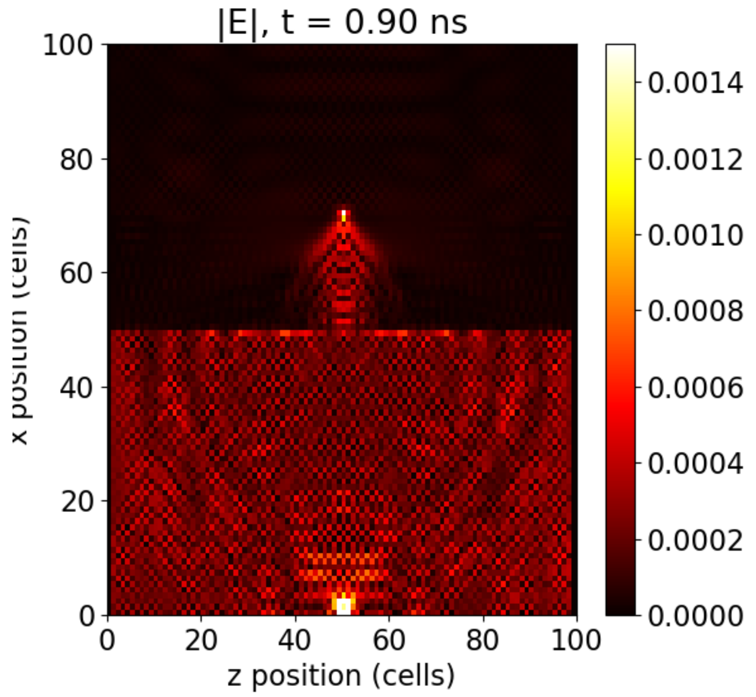
Python Classes

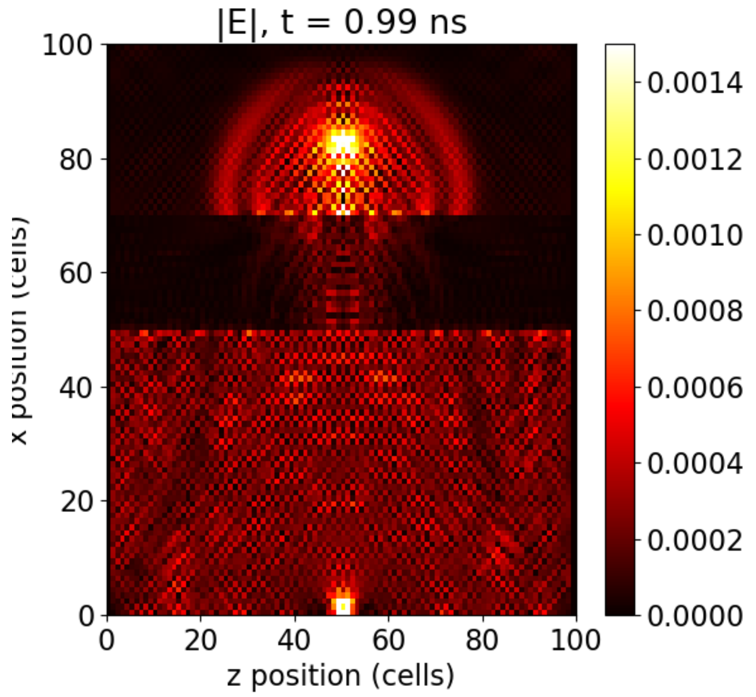
- pic_solver.fields(params, frameno)
  - Class containing an array representing each of the vector fields being calculated: Ex, Ez, Jx, Jz, Sx, Sz, By at a particular timestep ‘frameno’. The S field is labelled Px and Pz internally due to its relation to polarisation density.
- pic_solver.solver(params, constants, geometry)
  - The class performing the actual FDTD solution.
  - Needs to be initialised with a ‘params’, ‘constants’ and ‘geometry’ class to define the simulation
  - The variable ‘frames’ is a list of ‘fields’ classes tracking the evolution of the fields with time. The fields can be edited manually to set up initial conditions.
  - The variable ‘particles’ is a list of ‘particle’ classes defining each particle being tracked. Particle objects should be manually initialised and appended to this list when setting up the simulation.
  - The function ‘propagate_sim()’ adds a new element to the ‘frames’ list representing the next time step, and updates the positions and velocities of each particle.
  - The function ‘calc_energy(frame)’ returns the total electric and magnetic energy in the simulation domain at a given time step.

minimal_example.py

- Shows an example of how to set up and run the solver.
- An electron is accelerated from rest by a 1 MV/m electric field into a dielectric, emitting Cherenkov radiation inside the dielectric followed by transition radiation when it exits the dielectric.
- Note the persistent spot of high electric field at the middle of the bottom edge. This is a result of charge continuity not being preserved. To remedy this, the electrostatic solver should be used to set up the appropriate initial fields (the initial E field is zero in this example despite the presence of a charge in the domain).







Converting to Cylindrical Coordinates

- Cylindrical coordinates could allow an electron emission or impact site to be modelled more realistically, while still keeping the simplicity of a 2D solver.
- This requires only minor changes:
  - Replace the three field components E_x, B_y and E_z with ones in cylindrical coordinates, E_r, B_ϕ and E_z.
  - The definition of $(\nabla \times \boldsymbol{E})_\phi$ and $(\nabla \times \boldsymbol{B})_r$ remain the same.
  - $(\nabla \times \boldsymbol{B})_z = \frac{1}{r}\frac{\partial(rB_\phi)}{\partial r}$, which requires some weighting factors to be added.
  - $B_\phi = 0$ and $E_r = 0$ at the $r = 0$ boundary due to the assumed cylindrical symmetry of the problem.