Introduction to plasma dynamics - GRB project
III. Numerics: a fully electromagnetic PiC implementation

December 12, 2018

We want to extend the electrostatic code you built and include the effect of electromagnetic fields. We will keep the the mono-dimensionality, i.e. we consider spatial variations of all quantities along the x-direction only. This however does not mean that vectors cannot have more than one component; it just implies that no quantities can change along the y and z directions, i.e. there cannot be any dependence on those coordinates.

In the electrostatic code, we only considered one component of the particle velocity, one component of the electric field vector, and one spatial dimension. Such an implementation is called "1D1V". We will now turn to a "1D3V" implementation, i.e. one spatial dimension but three components for vector quantities. In one dimension, the full set of Maxwell's equations (in normalised units) is written, component by component

$$\frac{\partial E_x}{\partial t} = -J_x, \tag{1}$$

$$\frac{\partial E_y}{\partial t} = -\frac{\partial B_z}{\partial x} - J_y, \tag{2}$$

$$\frac{\partial E_z}{\partial t} = \frac{\partial B_y}{\partial x} - J_z, \tag{3}$$

$$\frac{\partial B_x}{\partial t} = 0, \tag{4}$$

$$\frac{\partial B_y}{\partial t} = \frac{\partial E_z}{\partial x}, \tag{5}$$

$$\frac{\partial B_z}{\partial t} = -\frac{\partial E_y}{\partial x}, \tag{6}$$

with the two divergence equations reducing to

$$\frac{\partial E_x}{\partial x} = \rho, \tag{7}$$

$$\frac{\partial B_x}{\partial x} = 0. \tag{8}$$

In our implementations we are ignoring equation (7) for simplicity. Equations (4) and (8) state that in a one-dimensional system, B_x must be uniform in space, *and* it must not change in time. This means that any initial B_x will remain unchanged through the evolution of the system. This also implies that in our simulation, we only have to consider equations (1), (2), (3), (5), and (6). The particle equations of motion reduce to

$$\frac{dx_p}{dt} = v_{x,p}, \tag{9}$$

$$\frac{dv_{x,p}}{dt} = \frac{q_p}{m_p}\left(E_x + v_{y,p}B_z - v_{z,p}B_y\right), \tag{10}$$

$$\frac{dv_{y,p}}{dt} = \frac{q_p}{m_p}\left(E_y - v_{x,p}B_z + v_{z,p}B_x\right), \tag{11}$$

$$\frac{dv_{z,p}}{dt} = \frac{q_p}{m_p}\left(E_z + v_{x,p}B_y - v_{y,p}B_x\right), \tag{12}$$

where only the x-position changes, since there is no motion along y and z.

For implementation purposes, it is clear that you will have to create more vectors than those needed in the electrostatic case: 6 (actually 5, since B_x is a constant) components of the electromagnetic field, 3 components of particle velocity, and 1 component of particle coordinate. Additionally, the particle vectors must be created for each species. However, aside from having to deal with more quantities, the philosophy behind the coding remains unchanged: macro-particles are evolved in time, feeding the evolution of the fields on a discrete grid, and vice versa.

One further change regards the discretisation of the grid quantities. We will now deal with a numerical grid possessing two types of elements: grid nodes (as in the electrostatic case) and cell centres, which is the main change with respect to the previous implementation. Cell centres are grid elements located at the centre of each cell, i.e. displaced of a distance $\Delta x/2$ with respect to the surrounding grid nodes. In such a discrete system made of N_g cells, there are $N_g + 1$ nodes and N_g cell centres. The electric field and current components are located on the grid nodes, while the magnetic field is on the cell centres. This choice of discretisation is called "staggered" (see Figure 1), since electromagnetic fields are staggered in space with respect to each other. The nodes are typically indicated with integer indices i, and centres with half-indices, e.g. $i + 1/2$.

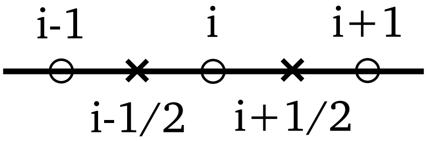

Figure 1: Staggered 1D computational grid. Grid nodes (circles) are labelled with integer indices i. Cell centres (crosses) are labelled with half indices, e.g. $i + 1/2$.

The staggered grid is useful for the approximation of spatial derivatives. If we want to use a centered-difference scheme for approximating $\partial/\partial x$, it follows immediately that spatial derivatives defined on nodes are easily obtained from quantities located on centres, e.g.

$$\left(\frac{\partial B_z}{\partial x}\right)_i \approx \frac{B_{z,i+1/2} - B_{z,i-1/2}}{\Delta x}. \tag{13}$$

By analogy, spatial derivatives defined on centres are obtained from quantities found on nodes, e.g.

$$\left(\frac{\partial E_z}{\partial x}\right)_{i+1/2} \approx \frac{E_{z,i+1} - E_{z,i}}{\Delta x}. \tag{14}$$

This provides an intuitive way to write the discrete equations for the fields. These read now

$$\frac{E^{n+1}_{x,i} - E^n_{x,i}}{\Delta t} = -J^n_{x,i}, \tag{15}$$

$$\frac{E^{n+1}_{y,i} - E^n_{y,i}}{\Delta t} = -\frac{B^n_{z,i+1/2} - B^n_{z,i-1/2}}{\Delta x} - J^n_{y,i}, \tag{16}$$

$$\frac{E^{n+1}_{z,i} - E^n_{z,i}}{\Delta t} = \frac{B^n_{y,i+1/2} - B^n_{y,i-1/2}}{\Delta x} - J^n_{z,i}, \tag{17}$$

$$\frac{B^{n+1}_{y,i+1/2} - B^n_{y,i+1/2}}{\Delta t} = \frac{E^n_{z,i+1} - E^n_{z,i}}{\Delta x}, \tag{18}$$

$$\frac{B^{n+1}_{z,i+1/2} - B^n_{z,i+1/2}}{\Delta t} = -\frac{E^n_{y,i+1} - E^n_{y,i}}{\Delta x}. \tag{19}$$

The discrete equations for the particles are simply

$$\frac{x^{n+1}_p - x^n_p}{\Delta t} = v^n_{x,p}, \tag{20}$$

$$\frac{v^{n+1}_{x,p} - v^n_{x,p}}{\Delta t} = \frac{q_p}{m_p}\left(E^n_{x,p} + v^n_{y,p}B^n_{z,p} - v^n_{z,p}B^n_{y,p}\right), \tag{21}$$

$$\frac{v^{n+1}_{y,p} - v^n_{y,p}}{\Delta t} = \frac{q_p}{m_p}\left(E^n_{y,p} - v^n_{x,p}B^n_{z,p} + v^n_{z,p}B^n_{x,p}\right), \tag{22}$$

$$\frac{v_{z,p}^{n+1} - v_{z,p}^n}{\Delta t} = \frac{q_p}{m_p}\left(E_{z,p}^n + v_{x,p}^n B_{y,p}^n - v_{y,p}^n B_{x,p}^n\right), \tag{23}$$

where the electromagnetic fields at the particle position are again obtained via interpolation. Note that B_x does not require interpolation, since it is a constant everywhere, and can be treated as such.

Implementing all the equations in your code should be straightforward if you start from your Newtonian electrostatic code. Create a copy of that code in a new folder and follow carefully the steps below:

- All essential parameters for the simulation are defined exactly as before: system length, number of cells, number of particles, time step, particle charge, etc. However, now you have to deal with grid quantities defined on nodes and centres. This is best done by creating two arrays of grid positions: $x_N = 0 : \Delta x : L$ for the nodes, and $x_C = \Delta x/2 : \Delta x : L - \Delta x/2$ for the centres (the last one will not be used, but it is good to have it). Remember: there are $N_g + 1$ nodes and N_g centres in your simulation. Create corresponding arrays for the grid-defined components of the electric field and current (on the nodes) and for the magnetic field (on the centres). Additionally, you must create arrays for the magnetic field *defined on the nodes*. Why this is necessary will become clear later.

- Now define all arrays for the particle position and components of velocity, exactly as before, for each species in your simulation. For now, initialise a Maxwellian velocity distribution by selecting normally-distributed random numbers for each component of the velocity.

- Our four-step PiC cycle requires some modifications. First, we need to calculate all fields (electric and magnetic) at the particle position. However, **B** is defined on the cell centres, while our interpolation formula works for interpolation from the nodes. Hence, we first need to calculate the magnetic field on the nodes. This is easy to achieve with a first interpolation step, from centres to nodes: simply,

  $$B_i = \frac{B_{i+1/2} + B_{i-1/2}}{2}, \tag{24}$$

  by simple algebraic averaging. Note that you need to apply boundary conditions for the first and last node. For our choice of periodic boundaries,

  $$B_0 = \frac{B_{1/2} + B_{N_g-1/2}}{2}, \tag{25}$$

  $$B_{N_g} = B_0. \tag{26}$$

  There's no need to interpolate B_x, which is a constant. Now that we have both **E** and **B** on the nodes, we can apply linear interpolation at the particle position:

  $$E_p = E_i\left(1 - \frac{x_p - x_i}{\Delta x}\right) + E_{i+1}\frac{x_p - x_i}{\Delta x}, \tag{27}$$

  $$B_p = B_i\left(1 - \frac{x_p - x_i}{\Delta x}\right) + B_{i+1}\frac{x_p - x_i}{\Delta x}, \tag{28}$$

  for each component of the electric and magnetic field.

- We can now evolve the particle position and velocity. From the discrete equations of motion,

  $$x_p^{n+1} = x_p^n + \Delta t v_{x,p}^n, \tag{29}$$

  $$v_{x,p}^{n+1} = v_{x,p}^n + \frac{q_p \Delta t}{m_p}\left(E_{x,p}^n + v_{y,p}^n B_{z,p}^n - v_{z,p}^n B_{y,p}^n\right), \tag{30}$$

  $$v_{y,p}^{n+1} = v_{y,p}^n + \frac{q_p \Delta t}{m_p}\left(E_{y,p}^n - v_{x,p}^n B_{z,p}^n + v_{z,p}^n B_{x,p}^n\right), \tag{31}$$

  $$v_{z,p}^{n+1} = v_{z,p}^n + \frac{q_p \Delta t}{m_p}\left(E_{z,p}^n + v_{x,p}^n B_{y,p}^n - v_{y,p}^n B_{x,p}^n\right). \tag{32}$$

  Clearly, this step can be combined with the interpolation step above, after the magnetic field has been redefined on the nodes. Do not forget to apply boundary conditions to the particle position.

- We now need to compute the current that enters Maxwell's equations. This is done exactly, as before, except that three components are now present. After initialising all components to 0, with a `for` loop over all particles,

$$J_{x,i} = J_{x,i} + \frac{q_p v_{x,p}}{\Delta x}\left(1 - \frac{x_p - x_i}{\Delta x}\right), \tag{33}$$

$$J_{x,i+1} = J_{x,i+1} + \frac{q_p v_{x,p}}{\Delta x}\frac{x_p - x_i}{\Delta x}, \tag{34}$$

$$J_{y,i} = J_{y,i} + \frac{q_p v_{y,p}}{\Delta x}\left(1 - \frac{x_p - x_i}{\Delta x}\right), \tag{35}$$

$$J_{y,i+1} = J_{y,i+1} + \frac{q_p v_{y,p}}{\Delta x}\frac{x_p - x_i}{\Delta x}, \tag{36}$$

$$J_{z,i} = J_{z,i} + \frac{q_p v_{z,p}}{\Delta x}\left(1 - \frac{x_p - x_i}{\Delta x}\right), \tag{37}$$

$$J_{z,i+1} = J_{z,i+1} + \frac{q_p v_{z,p}}{\Delta x}\frac{x_p - x_i}{\Delta x}, \tag{38}$$

  and then apply boundary conditions, for each component, on the leftmost and rightmost nodes, as done in the 1D1V case.

- Finally, we can evolve the fields. Looking at the discrete equations, for the electric field this can be easily done with a `for` loop over $N_g + 1$ nodes:

$$E^{n+1}_{x,i} = E^n_{x,i} - \Delta t J^n_{x,i}, \tag{39}$$

$$E^{n+1}_{y,i} = E^n_{y,i} - \Delta t \frac{B^n_{z,i+1/2} - B^n_{z,i-1/2}}{\Delta x} - \Delta t J^n_{y,i}, \tag{40}$$

$$E^{n+1}_{z,i} = E^n_{z,i} + \Delta t \frac{B^n_{y,i+1/2} - B^n_{y,i-1/2}}{\Delta x} - \Delta t J^n_{z,i}, \tag{41}$$

  which for E_y and E_z require to apply boundary conditions for the leftmost and rightmost nodes, e.g.

$$E^{n+1}_{y,0} = E^n_{y,0} - \Delta t \frac{B^n_{z,1/2} - B^n_{z,N_g-1/2}}{\Delta x} - \Delta t J^n_{y,0}, \tag{42}$$

$$E^{n+1}_{y,N_g} = E^{n+1}_{y,0}, \tag{43}$$

  and the same for E_z. We must complete the field update step by calculating the new magnetic field with a `for` loop over N_g cell centres:

$$B^{n+1}_{y,i+1/2} = B^n_{y,i+1/2} + \Delta t \frac{E^n_{z,i+1} - E^n_{z,i}}{\Delta x}, \tag{44}$$

$$B^{n+1}_{z,i+1/2} = B^n_{z,i+1/2} - \Delta t \frac{E^n_{y,i+1} - E^n_{y,i}}{\Delta x}, \tag{45}$$

  where there is no need to apply boundary conditions to **B**. This completes the four-step PiC cycle for an electromagnetic 1D3V implementation.

Starting from the skeleton PiC code you wrote for the electrostatic case, introducing these modifications should be fairly easy. You can verify your implementation by simulating again a Maxwellian plasma: if you initialise the thermal velocity in all directions, you can plot the $(x_p, v_{x,p})$, $(x_p, v_{y,p})$, $(x_p, v_{z,p})$ phase spaces and check that the velocity distribution remains random for the entire simulation.

It is also useful to already modify the calculation of monitoring quantities we will need later, in order to account for the extra velocity and electromagnetic field components. The kinetic energy at each time step is now given by

$$e_K = \sum_p m_p (v^2_{x,p} + v^2_{y,p} + v^2_{z,p})/2, \tag{46}$$

and the electric and magnetic energies are

$$e_E = \sum_{i=0}^{N_g-1} (E^2_{x,i} + E^2_{y,i} + E^2_{z,i})\Delta x/2, \tag{47}$$

$$e_B = \sum_{i=0}^{N_g-1} (B_{x,i}^2 + B_{y,i}^2 + B_{z,i}^2)\Delta x/2, \tag{48}$$

and you can also store the maximum velocity in each direction.