Particle-in-Cell (PIC) kinetic simulations
06. One-dimensional electrostatic PIC code

Chun-Sung Jao (饒駿頌)
Assistant Research Scholar,
Institute of Space Science and Engineering,
National Central University, Taiwan

University of São Paulo, 2019.11.25-12.06

www.slido.com code: #P320

What do we have now?

1. Random number generator for particle (velocity) distribution
   06_02_initial.f90

2. Weighting function for putting particle in cell
   06_02_initial.f90

3. Field solver for Poisson's equation
   06_01_Inverse.f90

4. Particle pusher
   06_03_BorisPusher.f90

What do we have now?

1. Random number generator for particle (velocity) distribution
   06_02_initial.f90

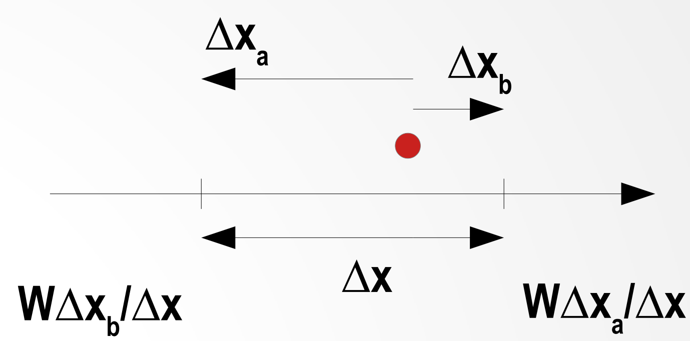
2. Weighting function for putting particle in cell
   06_02_initial.f90

3. Field solver for Poisson's equation
   06_01_Inverse.f90

4. **Field on particles**

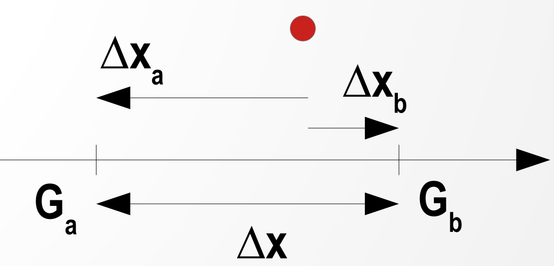
5. Particle pusher
   06_03_BorisPusher.f90

Linear interpolation



(W = 1 or q or m or qv or ...)

$$G_p = G_a \Delta x_b/\Delta x + G_b \Delta x_a/\Delta x$$



(G = E or B)

What do we have now?

1. Random number generator for particle (velocity) distribution
   06_02_initial.f90

2. Weighting function for putting particle in cell
   06_02_initial.f90

3. Field solver for Poisson's equation
   06_01_Inverse.f90

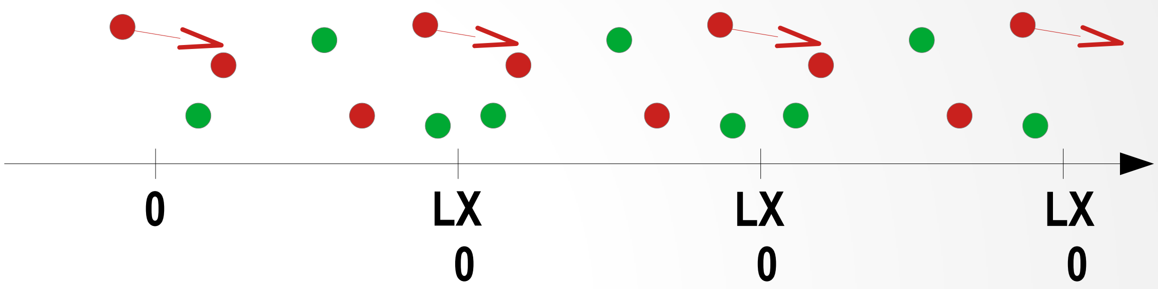
4. Field on particles

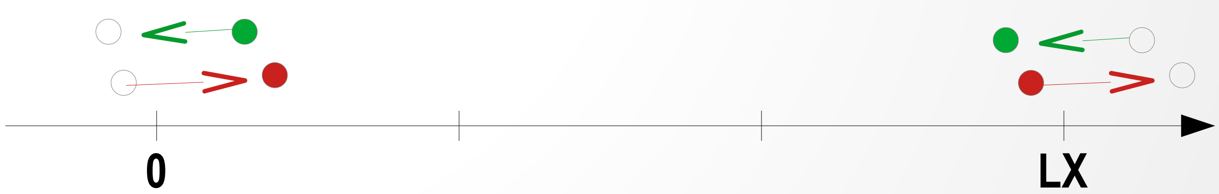
5. Particle pusher
   06_03_BorisPusher.f90

$$\frac{\vec{x}_{t+\Delta t}-\vec{x}_t}{\Delta t}=\vec{v}_{t+\Delta t/2}$$

$$\vec{x}_{t+\Delta t}=\vec{x}_t+\vec{v}_{t+\Delta t/2}\times\Delta t$$

Periodic boundary





What we did yesterday

1. Preset the E field and B field in a (infinite) simulation domain.

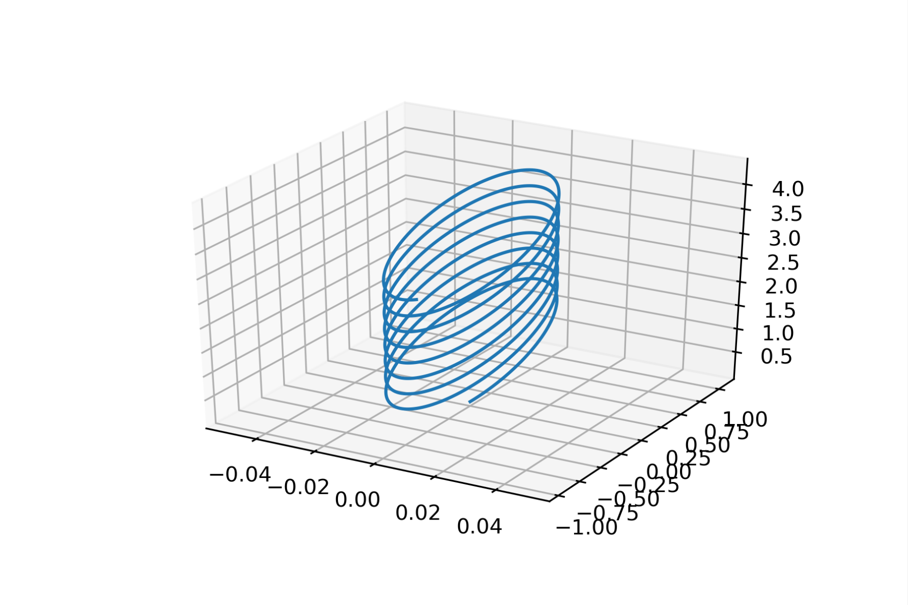
2. Give the position and the velocity to one particle at T = 0.

3. Based on the position of particle, we read the field and get the new particle velocity.

Δt

4. Based on the new particle velocity, we push the particle to the new position.

5. At certain point, we check the status of the particle.



What we are going to do today

1. Preset the E field and B field in a (infinite) simulation domain.
1. Setup a simulation domain (with Lx, Δx and Nx, where Lx = NxΔx).

2. Preset the E field and B field on the grid.
   EFx(0:Nx), EFy(0:Nx), EFz(0:Nx), BFx(0:Nx), BFy(0:Nx), BFz(0:Nx)

3. Preset the arrays for the charge density and electric potential.
   CD(0:Nx), EP(0:Nx)

2. Give the position and the velocity to one particle at T = 0.

3. Based on the position of particle, we read the field and get the new particle velocity.

4. Based on the new particle velocity, we push the particle to the new position.

5. At certain point, we check the status of the particle.

What we are going to do today

1. Setup a simulation domain (with Lx, Δx and NX, where Lx = NXΔx).

2. Preset the E field and B field on the grid.
   EFx(0:NX), EFy(0:NX), EFz(0:NX), BFx(0:NX), BFy(0:NX), BFz(0:NX)

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX), EP(0:NX)

2. Give the position and the velocity to one particle at T = 0.
4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PPy(1:NP), PPz(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)

3. Based on the position of particle, we read the field and get the new particle velocity.

4. Based on the new particle velocity, we push the particle to the new position.

5. At certain point, we check the status of the particle.

What we are going to do today

1. Setup a simulation domain (with Lx, Δx and NX, where Lx = NXΔx).

2. Preset the E field and B field on the grid.
   EFx(0:NX), EFy(0:NX), EFz(0:NX), BFx(0:NX), BFy(0:NX), BFz(0:NX)

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX), EP(0:NX)

4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PPy(1:NP), PPz(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)

   (You may like to output the field and particle data at T = 0)

3. Based on the position of particle, we read the field and get the new particle velocity.
5. Based on the position of particles, we read the field and get the new particle velocity.

4. Based on the new particle velocity, we push the particle to the new position.

5. At certain point, we check the status of the particle.

What we are going to do today

1. Setup a simulation domain (with Lx, Δx and NX, where Lx = NXΔx).

2. Preset the E field and B field on the grid.
   EFx(0:NX), EFy(0:NX), EFz(0:NX), BFx(0:NX), BFy(0:NX), BFz(0:NX)

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX), EP(0:NX)

4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PPy(1:NP), PPz(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)

   (You may like to output the field and particle data at T = 0)

5. Based on the position of particles, we read the field and get the new particle velocity.

4. Based on the new particle velocity, we push the particle to the new position.
6. Based on the new particle velocity, we push the particles to the new position.

5. At certain point, we check the status of the particle.

What we are going to do today

1. Setup a simulation domain (with Lx, Δx and NX, where Lx = NXΔx).

2. Preset the E field and B field on the grid.
   EFx(0:NX), EFy(0:NX), EFz(0:NX), BFx(0:NX), BFy(0:NX), BFz(0:NX)

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX), EP(0:NX)

4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PPy(1:NP), PPz(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)

   (You may like to output the field and particle data at T = 0)

5. Based on the position of particles, we read the field and get the new particle velocity.

6. Based on the new particle velocity, we push the particles to the new position.

7. Base on the new position of particles, we calculate the new E field on the grid.

5. At certain point, we check the status of the particle.

What we are going to do today

1. Setup a simulation domain (with Lx, Δx and NX, where Lx = NXΔx).

2. Preset the E field and B field on the grid.
   EFx(0:NX), EFy(0:NX), EFz(0:NX), BFx(0:NX), BFy(0:NX), BFz(0:NX)

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX), EP(0:NX)

4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PPy(1:NP), PPz(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)

   (You may like to output the field and particle data at T = 0)

Δt

5. Based on the position of particles, we read the field and get the new particle velocity.

6. Based on the new particle velocity, we push the particles to the new position.

7. Base on the new position of particles, we calculate the new E field on the grid.

5. At certain point, we check the status of the particle.

What we are going to do today

1. Setup a simulation domain (with Lx, Δx and NX, where Lx = NXΔx).

2. Preset the E field and B field on the grid.
   EFx(0:NX), EFy(0:NX), EFz(0:NX), BFx(0:NX), BFy(0:NX), BFz(0:NX)

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX), EP(0:NX)

4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PPy(1:NP), PPz(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)

   (You may like to output the field and particle data at T = 0)

Δt

5. Based on the position of particles, we read the field and get the new particle velocity.

6. Based on the new particle velocity, we push the particles to the new position.

7. Base on the new position of particles, we calculate the new E field on the grid.

5. At certain point, we check the status of the particle.
8. After a while, we check the status of the particles and fields.

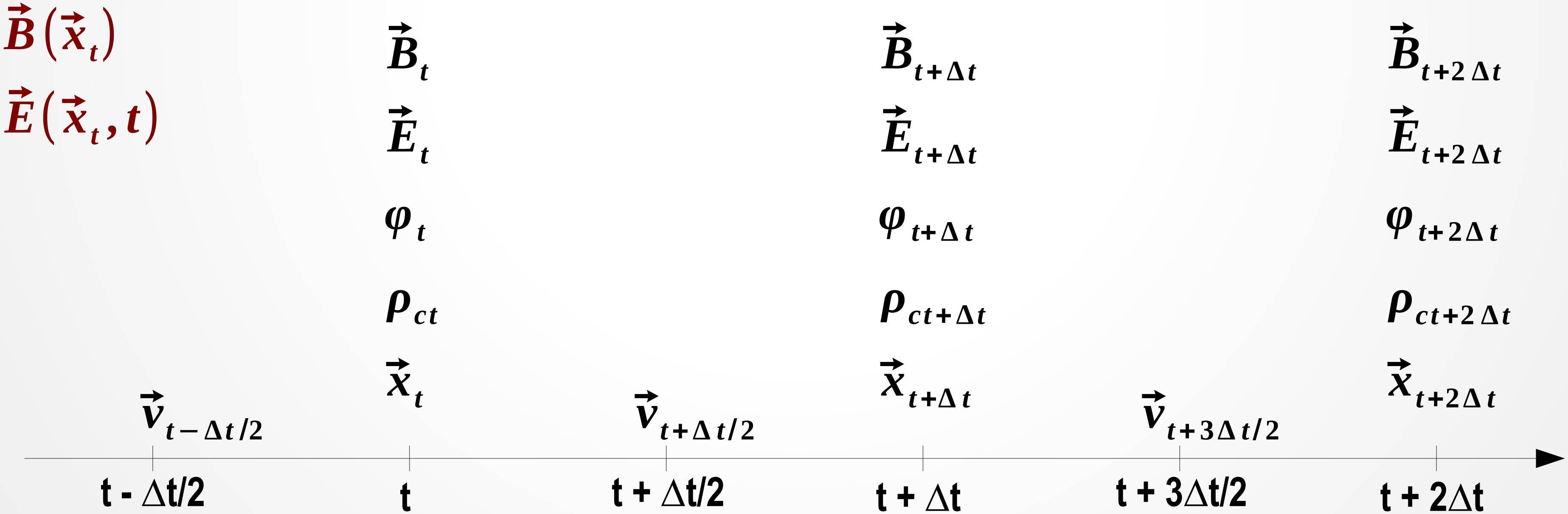
What we are going to do today

1. Setup a simulation domain
2. Preset the E field and B field on the grid.
3. Preset the arrays for the charge density and electric potential.
4. Give the position and the velocity to all particles at T = 0.
   (You may like to output the field and particle data at T = 0)
5. Based on the position of particles, we read the field and get the new particle velocity.
6. Based on the new particle velocity, we push the particles to the new position.
7. Base on the new position of particles, we calculate the new E field on the grid.
8. After a while, we check the status of the particles and fields.

Δt

$$\vec{B}=\vec{B}(\vec{x}_t)$$

$$\vec{E}=\vec{E}(\vec{x}_t,t)$$



One-dimensional electrostatic PIC code

1. Setup a simulation domain (with Lx = 256, $\Delta x = 1$, and NX = 256). ($\Delta t = 0.01$, and NT = 51200)

2. Preset the E field and B field on the grid.
   EFx(0:NX) = 0, BFx(0:NX) = 0, BFy(0:NX) = 0, BFz(0:NX) = 0

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX) = 0, EP(0:NX) = 0

4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PPy(1:NP), PPz(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)

   (You may like to output the field and particle data at T = 0)

Δt

5. Based on the position of particles, we read the field and get the new particle velocity.

6. Based on the new particle velocity, we push the particles to the new position.

7. Base on the new position of particles, we calculate the new E field on the grid.

8. After a while, we check the status of the particles and fields.

One-dimensional electrostatic PIC code

1. Setup a simulation domain (with $Lx = 256$, $\Delta x = 1$, and $NX = 256$). ($\Delta t = 0.01$, and $NT = 25600$)

2. Preset the E field and B field on the grid.
   EFx(0:NX) = 0, BFx(0:NX) = 0, BFy(0:NX) = 0, BFz(0:NX) = 0

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX) = 0, EP(0:NX) = 0

4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)
   06_02_initial.f90
   Ion: m = 1836, e = 1, vd = 0.0, vth = 0.1, NP = 256*256, uniform distribution
   electron1: m = 1 , e = -1, vd = 5.0, vth = 1.0, NP = 128*256, uniform distribution
   electron2: m = 1 , e = -1, vd = -5.0, vth = 1.0, NP = 128*256, uniform distribution

   (You may like to output the field and particle data at T = 0)

5. Based on the position of particles, we read the field and get the new particle velocity.

6. Based on the new particle velocity, we push the particles to the new position.

7. Base on the new position of particles, we calculate the new E field on the grid.

8. After a while, we check the status of the particles and fields.

One-dimensional electrostatic PIC code

1. Setup a simulation domain (with Lx = 256, Δx = 1, and NX = 256). (Δt = 0.01, and NT = 25600)

2. Preset the E field and B field on the grid.
   EFx(0:NX) = 0, BFx(0:NX) = 0, BFy(0:NX) = 0, BFz(0:NX) = 0

3. Preset the arrays for the charge density and electric potential.
   CD(0:NX) = 0, EP(0:NX) = 0

4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)
   06_02_initial.f90
   Ion: m = 1836, e = 1, vd = 0.0, vth = 0.1, NP = 256*256, uniform distribution
   electron1: m = 1 , e = -1, vd = 5.0, vth = 1.0, NP = 128*256, uniform distribution
   electron2: m = 1 , e = -1, vd = -5.0, vth = 1.0, NP = 128*256, uniform distribution

   (You may like to output the field and particle data at T = 0)
   (Output PPx, PVx, EP, EFx. Note the Time of the file name. Make a first plot for PPx-PVx)

5. Based on the position of particles, we read the field and get the new particle velocity.

6. Based on the new particle velocity, we push the particles to the new position.

7. Base on the new position of particles, we calculate the new E field on the grid.

8. After a while, we check the status of the particles and fields.

One-dimensional electrostatic PIC code

1. Setup a simulation domain (with Lx = 256, Δx = 1, and NX = 256). (Δt = 0.01, and NT = 25600)
2. Preset the E field and B field on the grid.
   EFx(0:NX) = 0, BFx(0:NX) = 0, BFy(0:NX) = 0, BFz(0:NX) = 0
3. Preset the arrays for the charge density and electric potential.
   CD(0:NX) = 0, EP(0:NX) = 0
4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)
   06_02_initial.f90
   Ion: m = 1836, e = 1, vd = 0.0, vth = 0.1, NP = 256*256, uniform distribution
   electron1: m = 1 , e = -1, vd = 5.0, vth = 1.0, NP = 128*256, uniform distribution
   electron2: m = 1 , e = -1, vd = -5.0, vth = 1.0, NP = 128*256, uniform distribution

   (You may like to output the field and particle data at T = 0)
   (Output PPx, PVx, EP, EFx. Note the Time of the file name. Make a first plot for PPx-PVx)

Do loop starts.

5. Based on the position of particles, we read the field and get the new particle velocity.
   06_03_BorisPusher.f90
   (Refer to 06_02_initial.f90, do some works for the field on particles)

6. Based on the new particle velocity, we push the particles to the new position.

7. Base on the new position of particles, we calculate the new E field on the grid.

8. After a while, we check the status of the particles and fields.

One-dimensional electrostatic PIC code

1. Setup a simulation domain (with Lx = 256, Δx = 1, and NX = 256). (Δt = 0.01, and NT = 25600)
2. Preset the E field and B field on the grid.
   EFx(0:NX) = 0, BFx(0:NX) = 0, BFy(0:NX) = 0, BFz(0:NX) = 0
3. Preset the arrays for the charge density and electric potential.
   CD(0:NX) = 0, EP(0:NX) = 0
4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)
   06_02_initial.f90
   Ion: m = 1836, e = 1, vd = 0.0, vth = 0.1, NP = 256*256, uniform distribution
   electron1: m = 1 , e = -1, vd = 5.0, vth = 1.0, NP = 128*256, uniform distribution
   electron2: m = 1 , e = -1, vd = -5.0, vth = 1.0, NP = 128*256, uniform distribution

   (You may like to output the field and particle data at T = 0)
   (Output PPx, PVx, EP, EFx. Note the Time of the file name. Make a first plot for PPx-PVx)

Do loop starts.

5. Based on the position of particles, we read the field and get the new particle velocity.
   06_03_BorisPusher.f90 (Refer to 06_02_initial.f90, do some works for the field on particles)

6. Based on the new particle velocity, we push the particles to the new position.
   06_03_BorisPusher.f90 (pay attention for the boundary-crossing particles)

7. Base on the new position of particles, we calculate the new E field on the grid.

8. After a while, we check the status of the particles and fields.

One-dimensional electrostatic PIC code

1. Setup a simulation domain (with Lx = 256, Δx = 1, and NX = 256). (Δt = 0.01, and NT = 25600)
2. Preset the E field and B field on the grid.
   EFx(0:NX) = 0, BFx(0:NX) = 0, BFy(0:NX) = 0, BFz(0:NX) = 0
3. Preset the arrays for the charge density and electric potential.
   CD(0:NX) = 0, EP(0:NX) = 0
4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)
   06_02_initial.f90
   Ion: m = 1836, e = 1, vd = 0.0, vth = 0.1, NP = 256*256, uniform distribution
   electron1: m = 1 , e = -1, vd = 5.0, vth = 1.0, NP = 128*256, uniform distribution
   electron2: m = 1 , e = -1, vd = -5.0, vth = 1.0, NP = 128*256, uniform distribution

   (You may like to output the field and particle data at T = 0)
   (Output PPx, PVx, EP, EFx. Note the Time of the file name. Make a first plot for PPx-PVx)

Do loop starts.

5. Based on the position of particles, we read the field and get the new particle velocity.
   06_03_BorisPusher.f90 (Refer to 06_02_initial.f90, do some works for the field on particles)

6. Based on the new particle velocity, we push the particles to the new position.
   06_03_BorisPusher.f90 (pay attention for the boundary-crossing particles)

7. Base on the new position of particles, we calculate the new E field on the grid.
   06_02_initial.f90 for particle weighting to get charge density, notice the boundary.
   06_01_Inverse.f90 for field solver

8. After a while, we check the status of the particles and fields.

One-dimensional electrostatic PIC code

1. Setup a simulation domain (with Lx = 256, Δx = 1, and NX = 256). (Δt = 0.01, and NT = 25600)
2. Preset the E field and B field on the grid.
   EFx(0:NX) = 0, BFx(0:NX) = 0, BFy(0:NX) = 0, BFz(0:NX) = 0
3. Preset the arrays for the charge density and electric potential.
   CD(0:NX) = 0, EP(0:NX) = 0
4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)
   06_02_initial.f90
   Ion: m = 1836, e = 1, vd = 0.0, vth = 0.1, NP = 256*256, uniform distribution
   electron1: m = 1 , e = -1, vd = 5.0, vth = 1.0, NP = 128*256, uniform distribution
   electron2: m = 1 , e = -1, vd = -5.0, vth = 1.0, NP = 128*256, uniform distribution

   (You may like to output the field and particle data at T = 0)
   (Output PPx, PVx, EP, EFx. Note the Time of the file name. Make a first plot for PPx-PVx)

Do loop starts.

5. Based on the position of particles, we read the field and get the new particle velocity.
   06_03_BorisPusher.f90 (Refer to 06_02_initial.f90, do some works for the field on particles)
6. Based on the new particle velocity, we push the particles to the new position.
   06_03_BorisPusher.f90 (pay attention for the boundary-crossing particles)
7. Base on the new position of particles, we calculate the new E field on the grid.
   06_02_initial.f90 for particle weighting to get charge density, notice the boundary. 06_01_Inverse.f90 for field solver

8. After a while, we check the status of the particles and fields.
   (Output PPx, PVx, EP, EFx at T = 64, 128, 192, 256. Note the Time of the file name.)
   Make the PPx-PVx plots.

One-dimensional electrostatic PIC code

1. Setup a simulation domain (with Lx = 256, Δx = 1, and NX = 256). (Δt = 0.01, and NT = 25600)
2. Preset the E field and B field on the grid.
   EFx(0:NX) = 0, BFx(0:NX) = 0, BFy(0:NX) = 0, BFz(0:NX) = 0
3. Preset the arrays for the charge density and electric potential.
   CD(0:NX) = 0, EP(0:NX) = 0
4. Give the position and the velocity to all particles at T = 0.
   PPx(1:NP), PVx(1:NP), PVy(1:NP), PVz(1:NP)
   06_02_initial.f90
   Ion: m = 1836, e = 1, vdx = 0.0, vdy = vdz = 0.0, vthx = vthy = vthz = 0.1, NP = 256*256, uniform distribution
   electron1: m = 1 , e = -1, vdx = 5.0, vdy = vdz = 0.0, vthx = vthy = vthz = 1.0, NP = 128*256, uniform distribution
   electron2: m = 1 , e = -1, vdx = -5.0, vdy = vdz = 0.0, vthx = vthy = vthz = 1.0, NP = 128*256, uniform distribution

   (You may like to output the field and particle data at T = 0)
   (Output PPx, PVx, EP, EFx. Note the Time of the file name. Make a first plot for PPx-PVx)

Do loop starts.

5. Base on the new position of particles, we calculate the new E field on the grid.
   06_02_initial.f90 for particle weighting to get charge density, notice the boundary. 06_01_Inverse.f90 for field solver

6. Based on the position of particles, we read the field and get the new particle velocity.
   06_03_BorisPusher.f90 (Refer to 06_02_initial.f90, do some works for the field on particles)

7. Based on the new particle velocity, we push the particles to the new position.
   06_03_BorisPusher.f90 (pay attention for the boundary-crossing particles)

8. After a while, we check the status of the particles and fields. Make the PPx-PVx plots.
   (Output PPx, PVx, EP, EFx at T = 64, 128, 192, 256. Note the Time of the file name.)

Do loop ends.

One-dimensional electrostatic PIC code

$\vec{B}=\vec{B}(\vec{x}_t)$

$\vec{E}=\vec{E}(\vec{x}_t,t)$

t - Δt/2	t	t + Δt/2	t + Δt	t + 3Δt/2	t + 2Δt
	$\vec{B}_t$		$\vec{B}_{t+\Delta t}$		$\vec{B}_{t+2\Delta t}$
	$\vec{E}_t$		$\vec{E}_{t+\Delta t}$		$\vec{E}_{t+2\Delta t}$
	φ_t		$\varphi_{t+\Delta t}$		$\varphi_{t+2\Delta t}$
	ρ_{ct}		$\rho_{ct+\Delta t}$		$\rho_{ct+2\Delta t}$
	$\vec{x}_t$		$\vec{x}_{t+\Delta t}$		$\vec{x}_{t+2\Delta t}$
$\vec{v}_{t-\Delta t/2}$		$\vec{v}_{t+\Delta t/2}$		$\vec{v}_{t+3\Delta t/2}$	