2D Particle-in-Cell (PiC) Simulation

Nathaniel Chen, Yigit Gunsur Elmacioglu,

Kian Orr, Dario Panici

Contributions

Yigit Gunsur Elmacioglu

- ➢ Finite difference implementation with sparse matrix
- ➢ Electric field calculations and interpolations
- ➢ Modularisation of the code with particle, grid and field classes

Dario Panici

- ➢ Energy calculations and comparisons
- ➢ Numerical analysis of different schemes

Kian Orr

- ➢ Boundary conditions for the particle trajectories
- ➢ Comparison of particle trajectories and energies between different integrators
- ➢ Comparison between analytical and numerical methods

Nathaniel Chen

- ➢ Implementation of different integrators and comparison analysis

Problem Statement

- PIC codes are used to model plasma flows in plasma propulsion and plasma physics.
- Outline of PIC:

1. get initial velocity
2. get initial weights with random r_i
3. get initial electric potential ϕ
4. get initial electric field
5. get acceleration from electric field
6. In loop
    (a) update r_i
    (b) update weights
    (c) solve for electric potential ϕ array
    (d) update E_i
    (e) update velocity

For a time-independent system, the Maxwell equations are

$$\nabla \cdot E = \rho/\epsilon_0 \qquad (1a) \qquad \nabla \times E = 0$$
$$\nabla \cdot B = 0 \qquad (1b) \qquad \nabla \times B = 0$$

By combining the E equations,

$$E = -\nabla\phi \implies \nabla^2\phi = -\rho/\epsilon_0$$

$$F = ma = -qE \implies a = -\frac{q}{m}E$$

Problem Statement - No Plasma Limit

- In the limit of low plasma density, the equations for the field simply reduce to the Laplace equation for the electric potential, and no update for the fields is needed during the algorithm

For a time-independent system, the Maxwell equations are

$$\nabla \cdot E = \cancel{\rho/\epsilon_0} \; 0 \qquad (1a) \qquad \nabla \times E = 0$$

$$\nabla \cdot B = 0 \qquad (1b) \qquad \nabla \times B = 0$$

By combining the E equations,

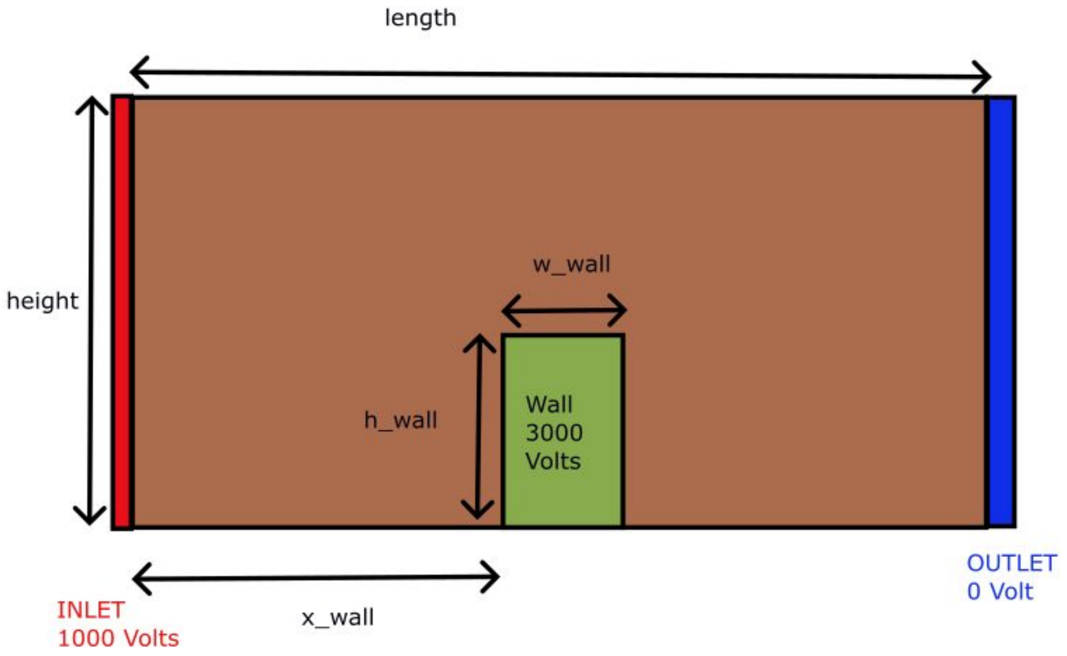
$$E = -\nabla\phi \implies \nabla^2\phi = \overset{0}{\cancel{-\rho/\epsilon_0}}$$

$$F = ma = -qE \implies a = -\frac{q}{m}E$$

1. get initial velocity
2. get initial positions $\mathbf{r}_i = (x_i, y_i)$
3. Solve for electric potential ϕ
4. Get electric field $\mathbf{E} = -\nabla\phi$
5. Particle Pushing Loop:
   (a) Interpolate electric field from grid it was solved on to the position $\mathbf{r}_i$
   (b) get acceleration $\mathbf{a}(x_i, y_i)$
   (c) update velocity $\mathbf{v}_{i+1}$
   (d) update position $\mathbf{r}_{i+1}$
   (e) Check if a domain boundary was crossed, and apply boundary conditions if so

Problem Setup

- Motion of Xenon +1 ions in duct
- The duct has a potential difference between the inlet and the outlet
- The wall in the middle has some specified voltage and cause a disruption in the electric field
- Looking at single particle motion



Problem Setup - Boundary Conditions

Dirichlet at Inlet/Outlet and Wall

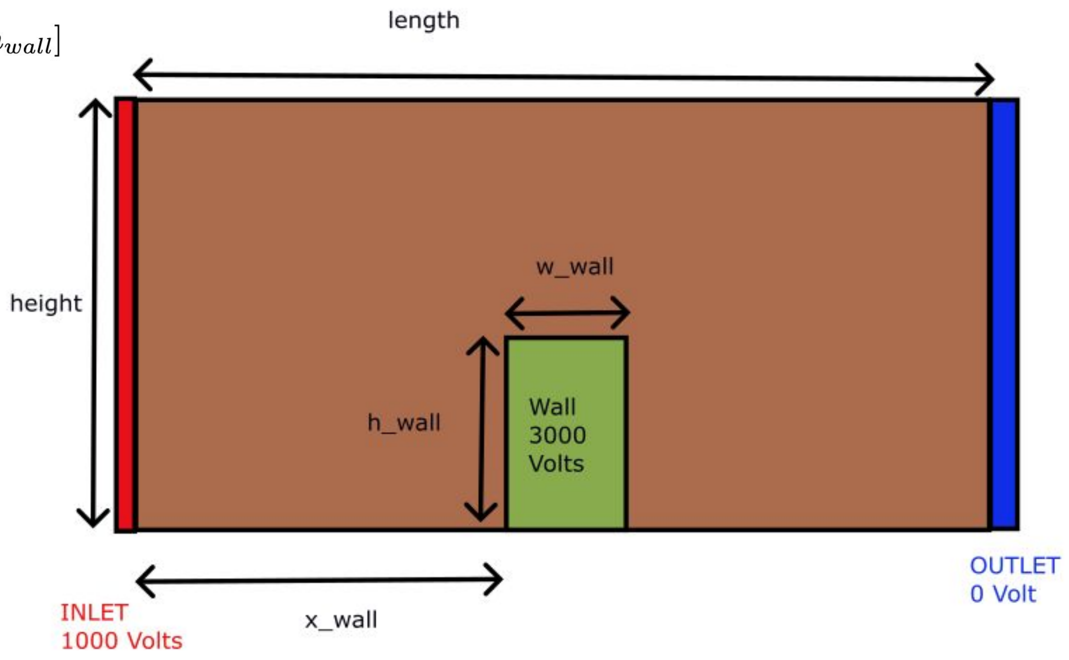
$$\phi(x,y) = \begin{cases} V_{wall} & \text{if } x \in [x_{wall}, x_{wall} + w_{wall}] \text{ and } y \in [0, h_{wall}] \\ V_{in} & \text{if } x = 0 \\ V_{out} & \text{if } x = length \end{cases}$$

Neumann At Top/Bottom

$$\frac{\partial \phi(x, y=0)}{\partial y} = 0$$

$$\frac{\partial \phi(x, y=height)}{\partial y} = 0$$

Note that Dirichlet of the biased wall takes precedence over the Neumann condition on the bottom

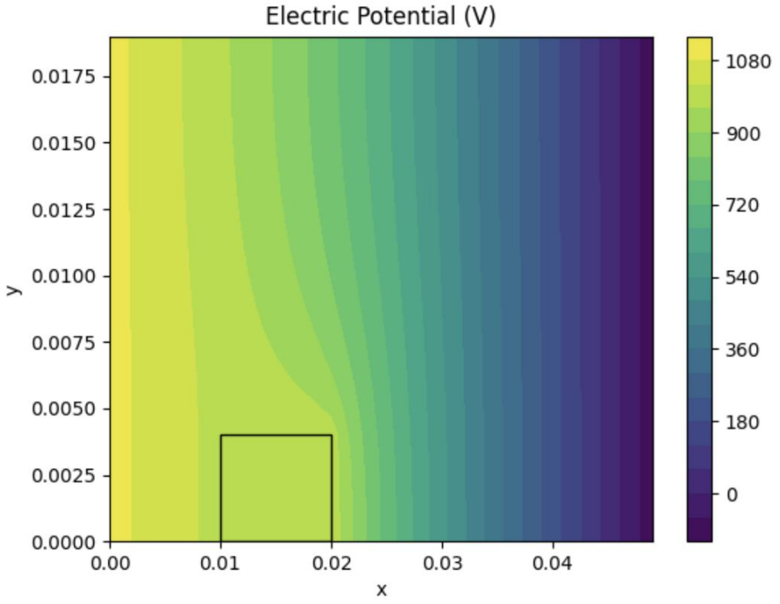


Process of pushing low density plasma

1. get initial velocity
2. get initial positions $\mathbf{r}_i = (x_i, y_i)$
3. Solve for electric potential ϕ
4. Get electric field $\mathbf{E} = -\nabla\phi$
5. Particle Pushing Loop:
   (a) Interpolate electric field from grid it was solved on to the position $\mathbf{r}_i$
   (b) get acceleration $\mathbf{a}(x_i, y_i)$
   (c) update velocity $\mathbf{v}_{i+1}$
   (d) update position $\mathbf{r}_{i+1}$
   (e) Check if a domain boundary was crossed, and apply boundary conditions if so

Simulation Problem setup

- L = 0.05 m
- H = 0.02 m
- Grid spacing = 10^-3 m
- x_wall=0.01m
- w_wall=L/5 m
- h_wall = H/5
- V_in =1100 V
- V_out = -100 Volts
- V_wall = 1000



Integrators

Leapfrog

$$\begin{aligned} v^{n+1/2} &= v^n + \frac{q}{m}E(x^n)\Delta t/2 \\ x^{n+1} &= x^n + v^{n+1/2}\Delta t \\ v^{n+1} &= v^{n+1/2} + \frac{q}{m}E(x^{n+1})\Delta t/2 \end{aligned}$$

Forward Euler

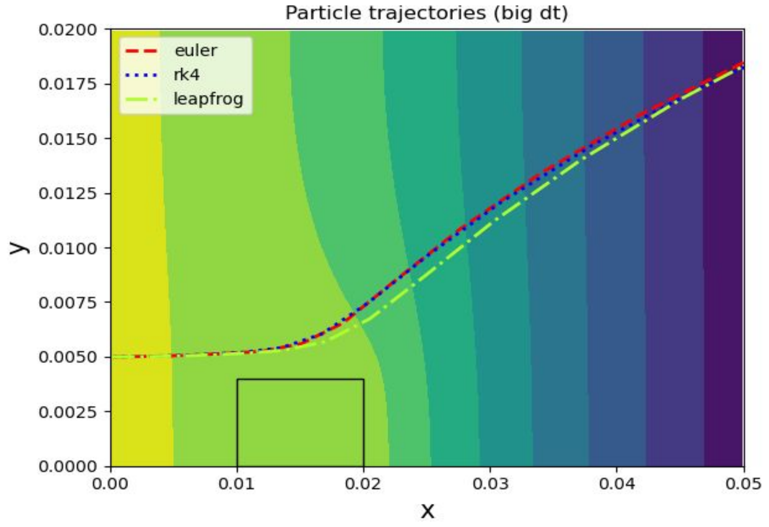
$$\begin{aligned} v^{n+1} &= v^n + \frac{q}{m}E(x^n)\Delta t \\ x^{n+1} &= x^n + v^{n+1}\Delta t \end{aligned}$$

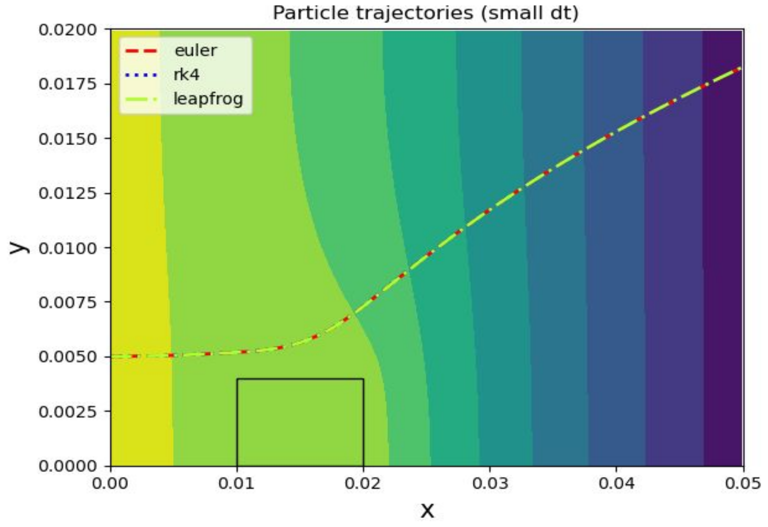
4th order Runge Kutta

$$\begin{aligned} k_1 &= f(\mathbf{x}^n) \\ k_2 &= f(\mathbf{x}^n + k_1\Delta t/2) \\ k_3 &= f(\mathbf{x}^n + k_2\Delta t/2) \\ k_4 &= f(\mathbf{x}^n + k_3\Delta t) \\ \mathbf{x}^{n+1} &= \mathbf{x}^n + (k_1 + k_2 + k_3 + k_4)\Delta t/6 \end{aligned}$$

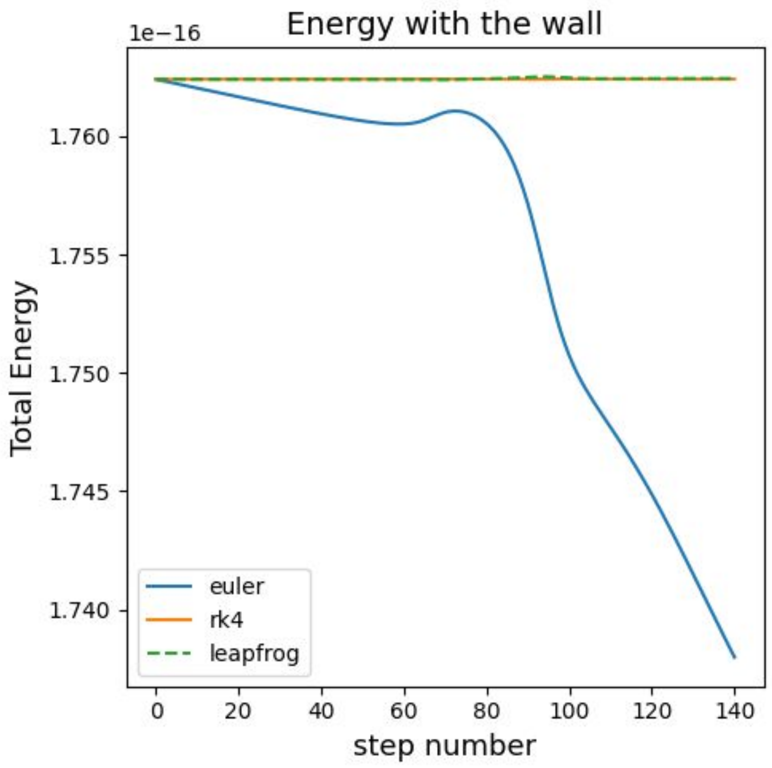
Comparison of path trajectories

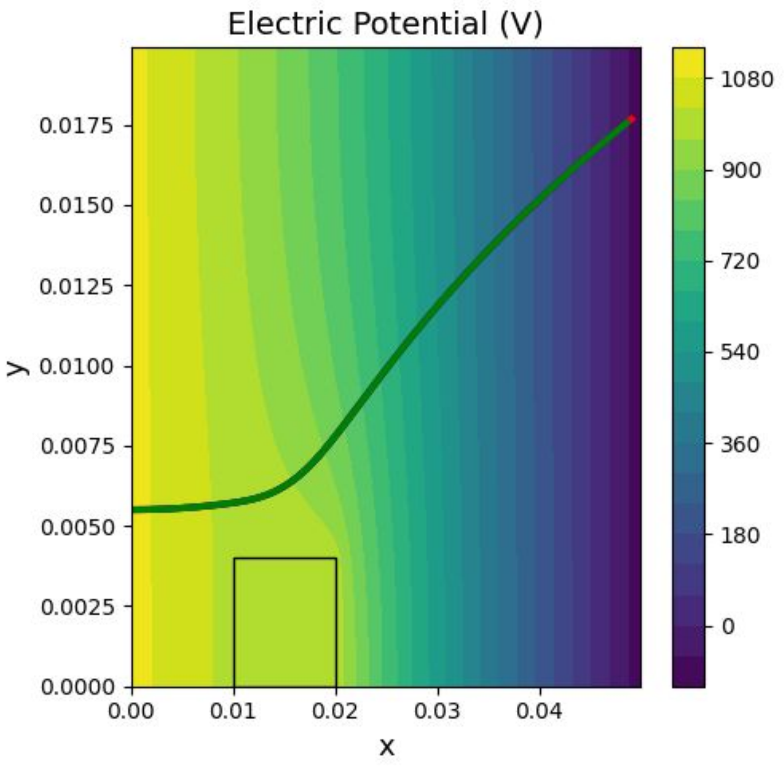
- Differences are apparent between euler, rk4, and leapfrog with big enough time step (~1e-7)





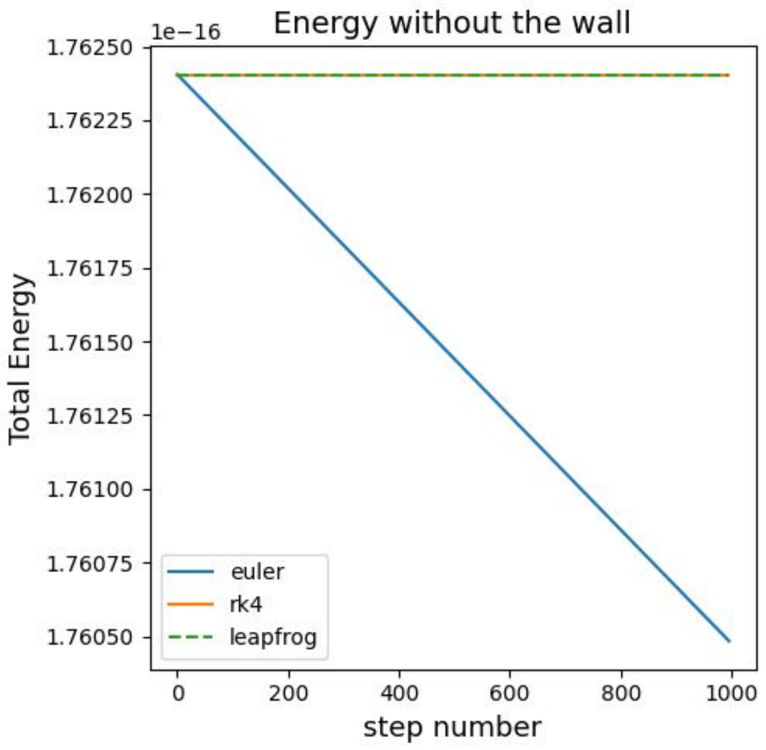
Comparison of energy conservation (with wall)

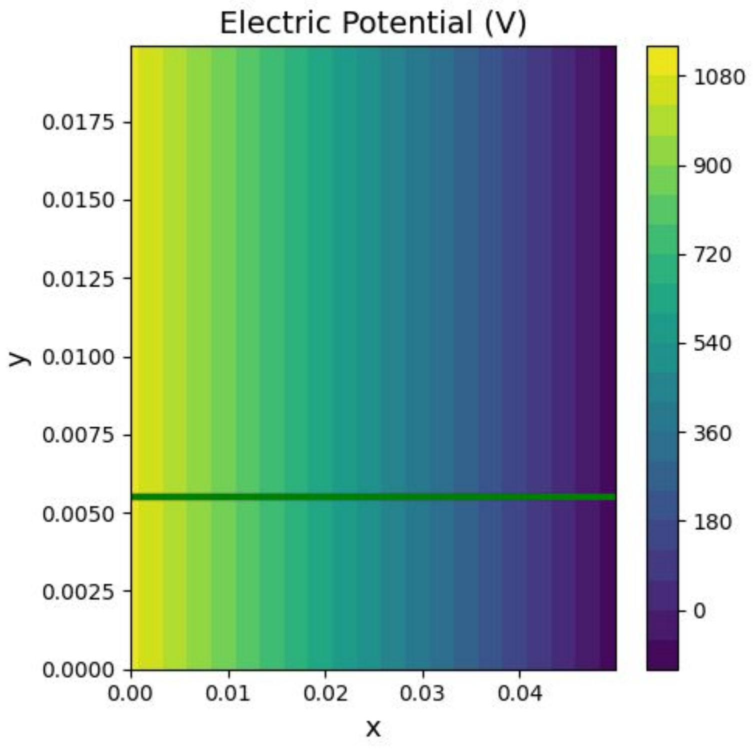
- 



Same Simulation but with no wall, for comparison

Comparison of energy conservation (without wall)





Derivation of Energy Loss for Euler (No wall, so constant Force)

- Euler can be shown to not conserve energy, and we recover the expected scaling of dE with the timestep numerically

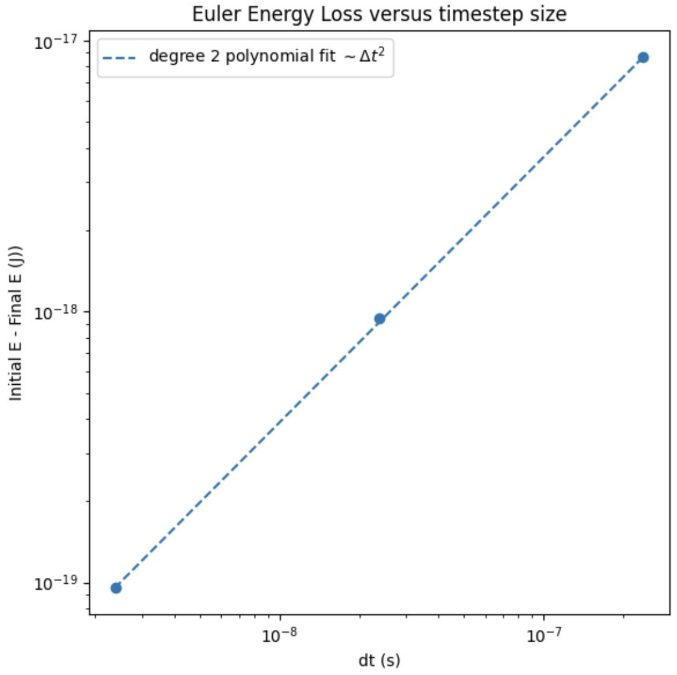
$$v_{i+1} = v_i + a\Delta t$$

$$x_{i+1} = x_i + v_{i+1}\Delta t$$

$$E_{i+1} - E_i = \frac{1}{2}mv_{i+1}^2 - qx_{i+1} - \frac{1}{2}mv_i^2 + qx_i$$

After plugging in and simplifying, we find that

$$= -\frac{q^2}{2m}\Delta t^2$$

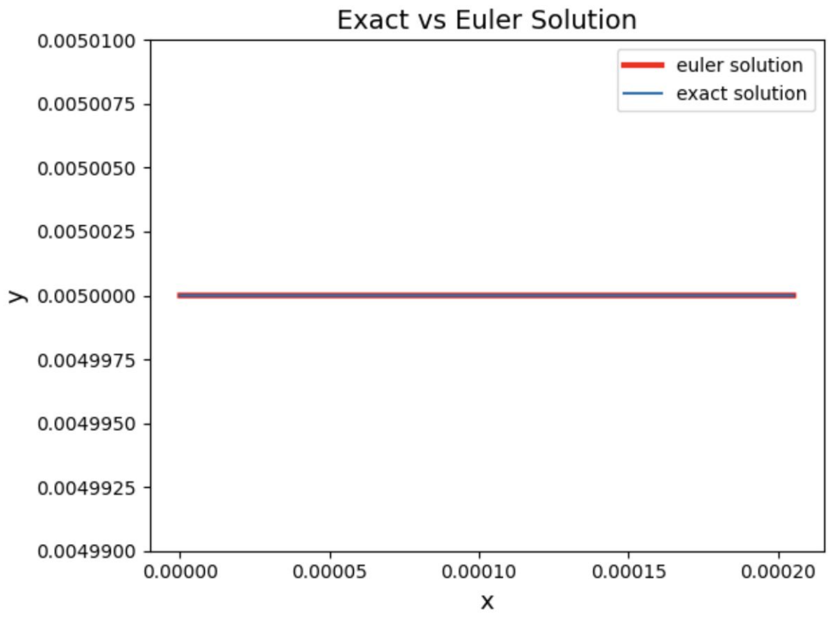


Exact vs. numerical

Analytical solution for a constant field:

$$x = \frac{1}{2}at^2 + v_0 t = \frac{1}{2}\frac{qE}{m}t^2 + v_0 t.$$

This solution is without a wall.


Exact vs Euler Solution
euler solution
exact solution
y
x