Computational Modeling Project 1

Pablo Santos Guerrero

October 2 2025

Abstract

A Python Method that can switch between the Euler and the second-order Euler-Richardson integrator is presented. The code is validated against analytic solutions for uniform electric and magnetic fields, for crossed **E** and **B** configurations, and for two mutually charged particles interacting via the Coulomb force. For each benchmark the relevant diagnostics (position vs. time, kinetic energy, inter-particle distance, total mechanical energy) are produced and discussed. The manuscript only references the implementation (contained in the file `charged_particle.py`) rather than reproducing the source code.

Contents

1 Theory

1.1 Lorentz force

A particle of charge q moving with velocity $\boldsymbol{v}$ in static fields $\boldsymbol{E}$ and $\boldsymbol{B}$ obeys the Lorentz force law

$$\boldsymbol{F} = q\big(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}\big).$$

Combined with Newton's second law one obtains the equation of motion

$$m\,\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = q\big[\boldsymbol{E}(\boldsymbol{r}) + \boldsymbol{v} \times \boldsymbol{B}(\boldsymbol{r})\big], \qquad \frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{v}. \tag{1}$$

In the present work the fields are taken to be spatially uniform and time-independent, so analytic solutions are available for validation.

1.2 Analytic reference solutions

(a) **Uniform electric field** $\boldsymbol{E} = E_0\,\hat{\mathbf{x}}$. The magnetic term vanishes, the particle experiences a constant acceleration $\boldsymbol{a} = (qE_0/m)\,\hat{\mathbf{x}}$. The exact trajectory is

$$\boldsymbol{v}(t) = \boldsymbol{v}_0 + \frac{qE_0}{m}\,t\,\hat{\mathbf{x}}, \tag{2a}$$

$$\boldsymbol{r}(t) = \boldsymbol{r}_0 + \boldsymbol{v}_0 t + \frac{1}{2}\frac{qE_0}{m}\,t^2\,\hat{\mathbf{x}}. \tag{2b}$$

(b) **Uniform magnetic field** $\boldsymbol{B} = B_0\,\hat{\mathbf{z}}$. Speed is conserved; the motion is a helix with cyclotron frequency $\omega_c = |q|B_0/m$ and Larmor radius $r_L = mv_\perp/(|q|B_0)$. The perpendicular components evolve as

$$\boldsymbol{v}_\perp(t) = v_\perp\big[-\sin(\omega_c t + \phi)\,\hat{\mathbf{x}} + \cos(\omega_c t + \phi)\,\hat{\mathbf{y}}\big],$$

and the parallel component remains constant.

(c) **Crossed fields** ($\boldsymbol{E} \perp \boldsymbol{B}$). For $\boldsymbol{E} = E_0\,\hat{\mathbf{x}}$ and $\boldsymbol{B} = B_0\,\hat{\mathbf{z}}$ the drift velocity is the classic $\boldsymbol{v}_d = \dfrac{\boldsymbol{E} \times \boldsymbol{B}}{B^2} = \dfrac{E_0}{B_0}\,\hat{\mathbf{y}}$. If the initial velocity equals $\boldsymbol{v}_d$ the particle travels on a straight line; otherwise a cycloid results.

2 Numerical integration schemes

All explicit schemes advance the state $(\boldsymbol{r}^n, \boldsymbol{v}^n)$ from $t_n = n\Delta t$ to $t_{n+1} = t_n + \Delta t$.

2.1 Forward Euler

$$\boldsymbol{a}^n = \frac{q}{m}\Big[\boldsymbol{E}(\boldsymbol{r}^n) + \boldsymbol{v}^n \times \boldsymbol{B}(\boldsymbol{r}^n)\Big], \tag{3a}$$

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \Delta t\,\boldsymbol{a}^n, \tag{3b}$$

$$\boldsymbol{r}^{n+1} = \boldsymbol{r}^n + \Delta t\,\boldsymbol{v}^n. \tag{3c}$$

The global truncation error is $\mathcal{O}(\Delta t)$.

2.2 Euler-Richardson (mid-point) scheme

$$\boldsymbol{a}^n = \frac{q}{m}\Big[\boldsymbol{E}(\boldsymbol{r}^n) + \boldsymbol{v}^n \times \boldsymbol{B}(\boldsymbol{r}^n)\Big], \tag{4a}$$

$$\boldsymbol{v}^{n+1/2} = \boldsymbol{v}^n + \tfrac{\Delta t}{2}\,\boldsymbol{a}^n, \qquad \boldsymbol{r}^{n+1/2} = \boldsymbol{r}^n + \tfrac{\Delta t}{2}\,\boldsymbol{v}^n, \tag{4b}$$

$$\boldsymbol{a}^{n+1/2} = \frac{q}{m}\Big[\boldsymbol{E}(\boldsymbol{r}^{n+1/2}) + \boldsymbol{v}^{n+1/2} \times \boldsymbol{B}(\boldsymbol{r}^{n+1/2})\Big], \tag{4c}$$

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \Delta t\,\boldsymbol{a}^{n+1/2}, \tag{4d}$$

$$\boldsymbol{r}^{n+1} = \boldsymbol{r}^n + \Delta t\,\boldsymbol{v}^{n+1/2}. \tag{4e}$$

This algorithm is second order ($\mathcal{O}(\Delta t^2)$) and exhibits excellent energy conservation for the problems considered here.

3 Software architecture

All numerical work is carried out by the Python module `charged_particle.py`. The most important objects are:

`Method` - class that implements a single particle pusher. Its method `run()` executes either the Euler step (option "euler") or the Euler-Richardson step (option "euler-richardson") and returns the time array together with the position and velocity arrays.

`TwoParticleMethod` - extension of `Method` that adds the mutual Coulomb acceleration $\boldsymbol{a}_{ij} = \frac{q_i q_j}{4\pi\varepsilon_0}\frac{\boldsymbol{r}_i - \boldsymbol{r}_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|^3}$. The same two integration choices are available.

Utility functions - the module contains small helper routines that evaluate the analytic solutions (`analytic_electric`, `analytic_magnetic`), the $\mathbf{E}\times\mathbf{B}$ drift (`drift_velocity`), and several Matplotlib wrappers used to generate the figures.

The **driver functions** listed in the next section orchestrate the runs and call the plotting utilities. Each driver produces one or more figures; the mapping between driver and figure is given explicitly.

4 Numerical experiments and associated figures

All runs were performed with the same source file; the only things that change are the argument values passed to the driver functions.

4.1 Pure electric field

The routine `run_electric_test()` (section 5 of the source file) loops over three field/velocity combinations and over a set of time steps $\Delta t \in \{0.2, 0.1, 0.05, 0.02, 0.01\}$. For each case it computes the maximum absolute position error with respect to the analytic solution (2) and finally calls the plotting routine `plot_convergence`.

- **Figure 1** – "Convergence of the position error versus Δt for the Euler (solid circles) and Euler-Richardson (solid squares) schemes." The slope of the Euler curve is ≈ 1; the Richardson curve has slope ≈ 2, confirming the theoretical orders.
- **Figure 2** – "Position component $x(t)$ for the first electric-field configuration, together with the analytic solution (black line)." Both integrators are shown; the Richardson curve matches the analytic solution to the plotted resolution.

4.2 Pure magnetic field

The driver `run_magnetic_test()` uses three initial velocities (perpendicular, parallel, mixed) and the same set of time steps. It evaluates the radius of the projected circular motion and the kinetic energy as functions of time, and calls the wrappers `plot_convergence` (radius error) and `plot_energy`.

- **Figure 3** – "Radius error versus Δt." The Euler scheme shows a clear drift of the Larmor radius, whereas the Richardson scheme keeps the radius constant within $\sim 10^{-4}$ relative error.
- **Figure 4** – "Kinetic energy versus time for the perpendicular-velocity case." The Euler curve drifts upward/downward, while the Richardson curve remains flat, illustrating the superior energy behaviour.

4.3 Crossed electric and magnetic fields

The function `run_crossed_test()` runs two orientations:

(a) $\boldsymbol{E} \perp \boldsymbol{B}$ (standard drift configuration);

(b) $\boldsymbol{E} \parallel \boldsymbol{B}$.

For each orientation three initial velocities are tried (rest, exact drift, arbitrary). The driver calls `plot_trajectory` (3-D view) and `plot_1d` to display a selected component.

- **Figure 5** – "Trajectory in the perpendicular configuration for three initial velocities (Euler: blue, Richardson: red)." The cycloid (starting from rest) and the straight drift (starting with $\boldsymbol{v}_d$) are both clearly visible.
- **Figure 6** – "Trajectory in the parallel configuration (again three initial velocities)." When $\boldsymbol{v}_0 = 0$ the particle accelerates along $\boldsymbol{B}$; when $\boldsymbol{v}_0 \parallel \boldsymbol{B}$ the motion is a simple translation; the mixed case shows a superposition of both effects.

4.4 Two interacting particles (Coulomb force)

The driver `run_two_body_test()` creates two scenarios:

(i) Repulsive case – both particles carry the electron charge;

(ii) Attractive case – one particle carries $+e$ (proton-like), the other $-e$.

Both particles start at rest at a distance $d_0 = 10^{-9}\,\mathrm{m}$ and no external fields are applied. The driver invokes `plot_distance` and `plot_total_energy`.

- **Figure 7** – "Inter-particle distance versus time (repulsive case)." The curve rises rapidly (the particles fly apart); the axis is automatically scaled to the millimetre range so that the motion is visible.
- **Figure 8** – "Total mechanical energy versus time (repulsive case)." The Richardson scheme conserves energy to better than 10^{-5} relative error, as expected from a second-order explicit method.
- **Figure 9** – "Inter-particle distance versus time (attractive case)." The particles accelerate towards each other, cross, and then separate again; the distance curve exhibits a pronounced minimum.

- **Figure 10** – "Total mechanical energy versus time (attractive case)." Kinetic energy grows while the potential energy becomes more negative, leaving the sum essentially constant.

All the figures above are automatically written to the current working directory as PNG files when the corresponding driver function is called. The manuscript therefore remains pure text while the source code lives in the separate module.

5 Discussion

- **Order of accuracy.** The convergence study (Figure 1) demonstrates that the Euler-Richardson algorithm achieves the expected $\mathcal{O}(\Delta t^2)$ error, allowing an order of magnitude larger time step for a given tolerance compared with the forward Euler method.

- **Energy conservation.** In the pure-magnetic case the magnetic force does no work; only the second-order scheme (Figure 4) preserves the kinetic energy. The two-particle simulations (Figures 8 and 10) show that the total mechanical energy (kinetic + Coulomb) remains constant to machine precision with the Richardson step, confirming that the implementation respects the underlying Hamiltonian structure.

- **Physical insight from crossed fields.** The trajectories produced by `run_crossed_test()` illustrate the classic $\mathbf{E} \times \mathbf{B}$ drift: when the initial velocity equals $\boldsymbol{v}_d$ the particle follows a straight line, otherwise a cycloid is observed. The parallel-field case shows pure acceleration along $\boldsymbol{B}$, as expected from Eq. (1).

- **Scalability and future extensions.** The current implementation computes the Coulomb interaction with a direct $O(N^2)$ sum, which is acceptable for the two-particle test but would become a bottleneck for many-body simulations. Replacing the pairwise loop by a Barnes- Hut tree or a particle-mesh scheme would permit scaling to thousands of particles without changing the overall structure presented here.

6 Conclusion

A lightweight, openly readable Python module (`charged_particle.py`) has been presented that implements both the forward Euler and the second-order Euler-Richardson particle pushers. By reproducing analytic benchmarks for uniform electric and magnetic fields, for crossed fields, and for two interacting charges, the code demonstrates the expected convergence rates and the superior energy behaviour of the higher-order scheme. The manuscript references the relevant functions (`Method.run`, `TwoParticleMethod.run`, `run_electric_test`, ...) instead of reproducing the source, making it compact while still providing a full audit trail for the results.

References

1. J. D. Jackson, *Classical Electrodynamics*, 3rd ed., Wiley (1998).

2. C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation*, IOP (2004).

3. W. H. Press *et al.*, *Numerical Recipes*, 3rd ed., Cambridge University Press (2007).