APPLAuSE: Computational Physics

Due on November 20 2015

MATLAB - Luís Silva

Rogério Jorge

Contents

Introduction

Understanding single particle motion dynamics is usually the first step within any plasma physics course. Quoting Francis F. Chen (*Introduction to Plasma Physics and Controlled Fusion*, Springer 1984), plasma has a *schizophrenic* personality, as its densities fall in an intermediate range - collisions may or may not dominate, they depend on the specific process. Hence, charged particle dynamics is important for under-dense plasmas and medium density plasmas. On the other hand, the computational strain provided by particle simulations will depend on the number of particles under consideration.
The objective of this work is to analyse the charged particle motion through the development of a Matlab code. We shall consider a small number of particles as we want a code that runs smoothly in a personal laptop with no parallel computing methods. Typically, we will analyse the behaviour of 10 particles, distributed randomly in a box. There is an imposed external electric and magnetic fields and the interaction between the particles is neglected.
In this case, a charged particle has a circular Larmor gyration plus a drift of the guiding centre. The equation of motion is

$$m\frac{d\vec{v}}{dt} = q(\vec{E} + \vec{v} \times \vec{B}). \tag{1}$$

The cyclotron motion has a frequency ω_c and a radios r_L around the guiding centre of

$$\omega_c = \frac{qB}{m}, \quad r_L = \frac{v_\perp}{\omega_c}. \tag{2}$$

The magnetic field keeps the energy $E = \frac{mv^2}{2}$ constant, so if $\vec{B} = (0, 0, B) \implies v_z =$ constant and $v_\perp = \sqrt{v_x^2 + v_y^2} =$ constant. If an additional electric field is present, particles will drift (which means their guiding centre velocity will have a new finite component) with velocity

$$\vec{v}_E = \frac{\vec{E} \times \vec{B}}{B^2}. \tag{3}$$

We will only mention one more drift that will be analysed bellow - the Grad-B drift. This drift is usually derived by first order Taylor-expanding the magnetic field about the point (x_0, y_0, z_0) and is given by

$$\vec{v}_{\nabla B} = v_\perp r_L \frac{\vec{B} \times \nabla B}{B^2}. \tag{4}$$

We considered electron-like particles, as this work is related to solving the equations of motion, illustrating the trajectories and picture the drifts, the different gyrofrequencies and Larmor radius wouldn't add any physical insight.

Matlab Numerical Code

As Matlab allows the use of different files as single functions, the code was divided into 8 scripts

- Lorentz - serves as "input file" and initialization. Ask the user if he wants a quick test (all the necessary homework plotting) or a customized test.
- Lorentz_Bfield - function to evaluate the magnetic field at each point. It allows a uniform B on z, a sinusoidal and a $1/z$ variation.
- Lorentz_dsolve - differential equation solver, either with 4th order Runge-Kutta or Euler method
- Lorentz_main - initializes position, velocity, energy and error vectors and calls the different functions defined above. At the end of the evaluation it writes to an hdf5 file the results.

- Lorentz_MBDist_histogram - using the initial velocity assigned in Lorentz_main, evaluates if the temperature is being assigned correctly by fitting to a 1-D Maxwell-Boltzmann distribution.
- Lorentz_plot - taking the title, labels and plot name as an input, it renders the graphics in a "publishable" quality.
- Lorentz_video - after solving for the trajectories, creates a loop of plots that saves everything inside a high quality *.mp4* video. It also saves the last plot as a *.pdf* file.
- Lorentz_visualize - plots the trajectory of the particles and defines the axis of the plot.

As input we allow the user after calling the function Lorentz to specify:

- Number of particles
- Particle temperature - in Kelvin
- Magnetic field on z - either uniform, cosine or $1/r$ variation
- Electric field - constant vector on an arbitrary direction
- Solver - Runge-Kutta or Euler
- Plane - 2D plot in the xy, yz, xz plane or in 3D
- Number of steps - total number of time units to numerically simulate.

Afterwords, having defined the electron mass, charge and Boltzmann's constant, we use as timestep and characteristic length

$$dt = 0.01 \times \frac{2\pi}{\omega_c}, \quad L = 10 \times \frac{v_{th}}{\omega_c}, \quad v_{th} = \sqrt{\frac{2k_BT}{m}}. \tag{5}$$

All the resulting movies, figures and hdf5 files are stores in the folder "Results".

Diagnostics

Code Verification

In order to verify the implementation of the Lorentz force solver, we look in Fig. 1 to the trajectories of 10 particles with an uniform magnetic field on the z direction with and without an uniform electric field on the y direction. It is seen on both cases the gyromotion of particles around a B field line in the z axis and in the second case a particle drift velocity in the x axis. We also see the random distribution of particles in the box with all sides equal to L and different particle initial velocity.
To verify that particles are correctly initialized with a specific temperature, we take the initial velocity on a specific direction and perform a non-linear fit to a Gaussian distribution function in Fig. 2. This allows us to analyse the standard deviation σ and mean μ, comparing with the Maxwell-Boltzmann 1-D distribution with $\sigma_{MB} = v_{th}$ and $\mu_{MB} = 0$. The accuracy of this analysis depends on the number of particles used, so we perform a test with a greater set of particles that the rest of the diagnostics.

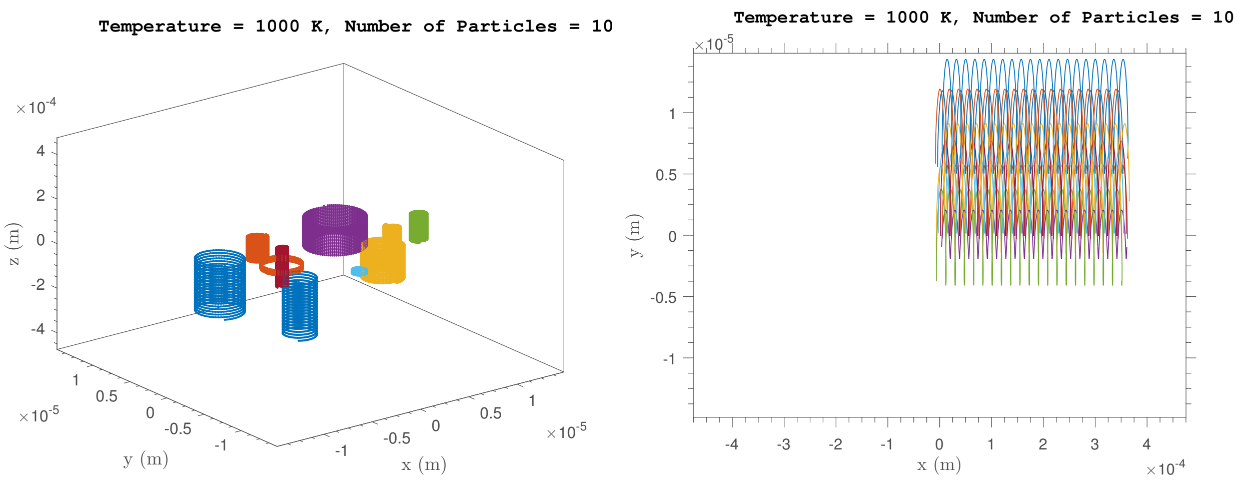

Figure 1: Left: Particle trajectory with uniform B field and no E field. Right: Uniform E field in the y direction causes a drift in the x direction.

Differential Solver Comparison

In the section, we shall use a result discussed in the introduction: if we only have an uniform stationary magnetic field, the energy is conserved. If we had a perfect numerical method, this would be true. Unfortunately, a small error determining the numerical velocity at a specific timestep introduces an error in the energy. A good numerical method should minimize this error, so to compare both methods we look at the relative error

$$\epsilon(t) = \frac{|E(t) - E_0|}{E_0}, \tag{6}$$

where E is the energy and $E_0 = E(t = 0)$. We can clearly see in Fig. 3 that the relative error with Runge-Kutta is several orders of magnitude smaller than with the Euler one.

Particle Drifts

As the code allows the use of an arbitrary (constant) electric field and an arbitrary magnetic field, we can look at the two drifts discussed in the introduction - the $E \times B$ drift and the Grad-B one. The particle trajectories are illustrated in Fig. 4. The uniform electric field that leads to the observed drift is of the form

$$\vec{E} = (0, E_0, 0), \tag{7}$$

where we used $E_0 = 5$ kV/cm. To observe the Grad-B drift, the implemented B field is of the form

$$\vec{B} = (0, B_0 \cos\frac{2\pi y}{L}, 0). \tag{8}$$

This means that particles shall drift in the positive or negative x direction depending on the initial y position of each particle, as we see on Fig. 4.

Conclusion

The code was developed during the week of 16 - 20 of November within the Advanced Computational Physics Course (Matlab module). The two objectives were achieved: start a Matlab code from scratch and understand the specific software operations (matrix multiplication, ODE solver, function and variable definition, ...) and analyse single charged particle motion with the same code. We have also seen how important numerical

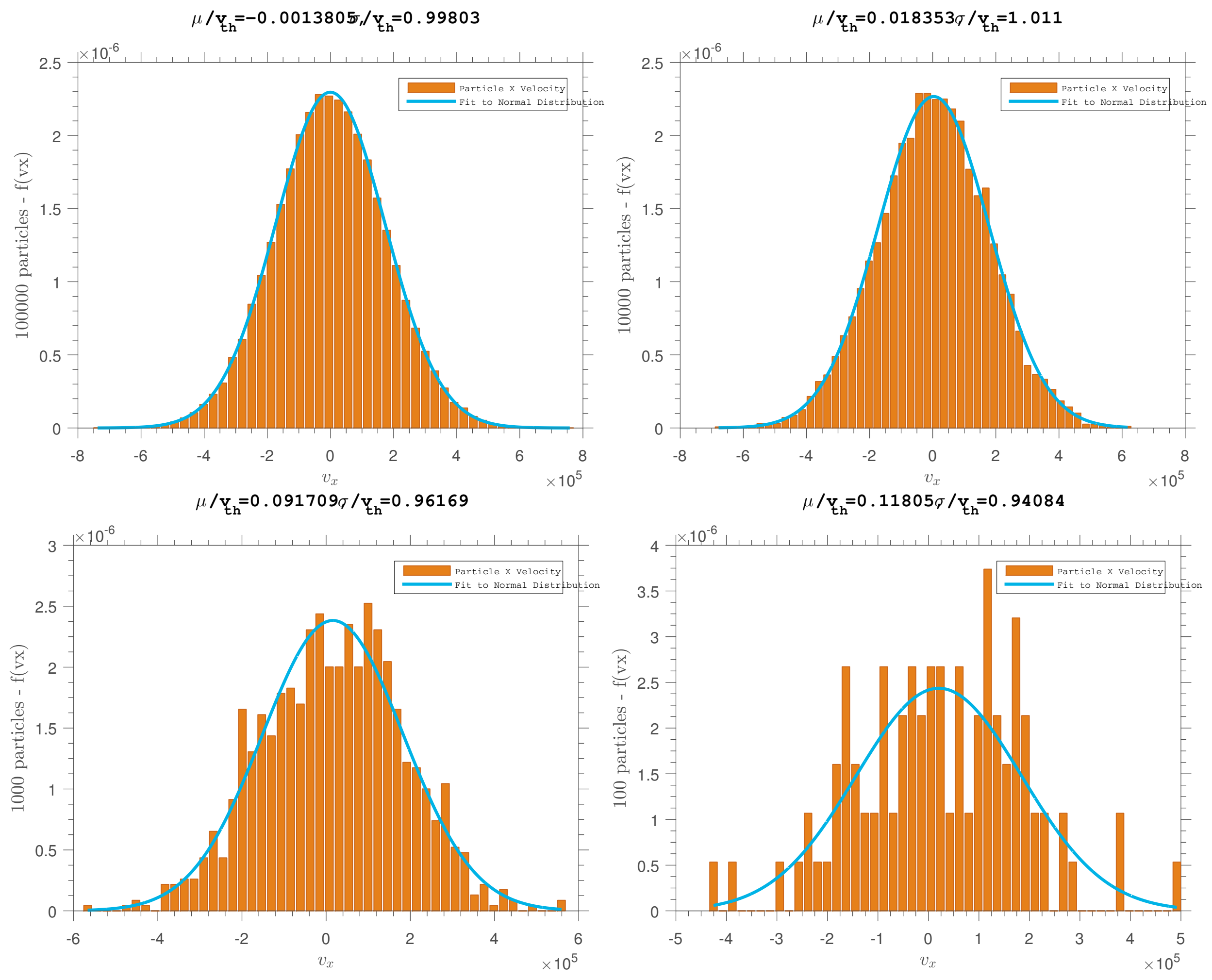

Figure 2: Histogram of the initial particle velocity and resulting non-linear fit to a Gaussian distribution. Fit parameters (mean μ and standard deviation σ) displayed.

methods are in numerical simulations. As seen in Fig. 5, with the Euler method the Larmor radius of the particles increases over time, which is not physical in this scenario.
It allows the user to visualize in 2 our 3 dimensions, specify the number of particles, initial temperature and electromagnetic profiles. Furthermore, the timestep and characteristic lengths are internally defined so to resolve at the gyrofrequency and Larmor radius scales.
As a next step, the inclusion of particle-particle interactions through a simple Coulomb term should be straightforward to implement due to the simplicity of the ODE solver (defined in a specific script). The next challenge would be to implement a Poisson solver, where the electric and magnetic fields would depend on particle dynamics and we would have to resort to Maxwell's equations.

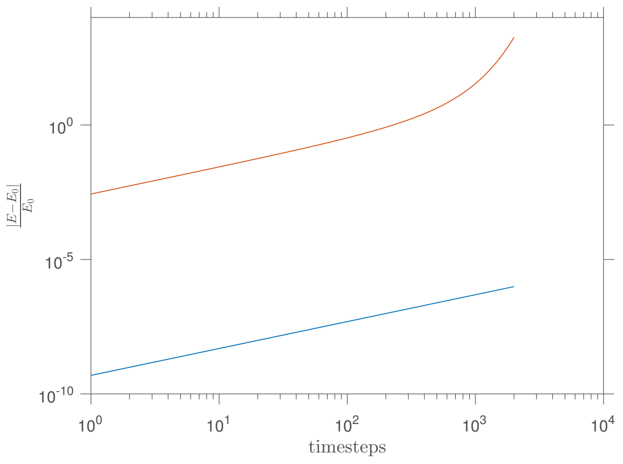

Figure 3: Comparison between differential solvers. The red line is the relative energy error with the Euler method and the blue line with the 4th order Runge-Kutta one.

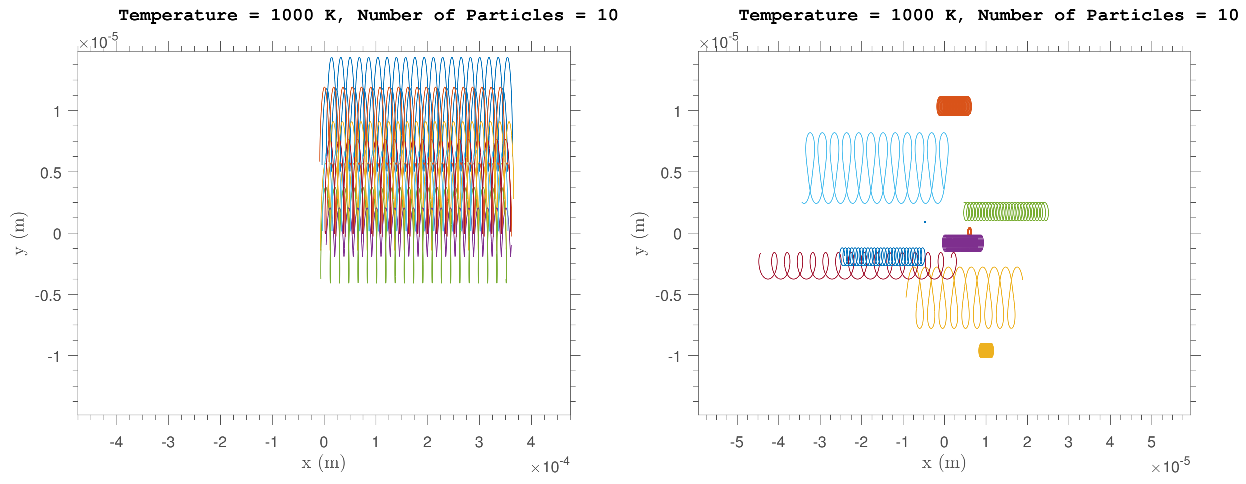

Figure 4: Left: Analysis of $E \times B$ drift to the right. Right: Grad-B drift with direction dependent on the positive or negative y position of the particle.

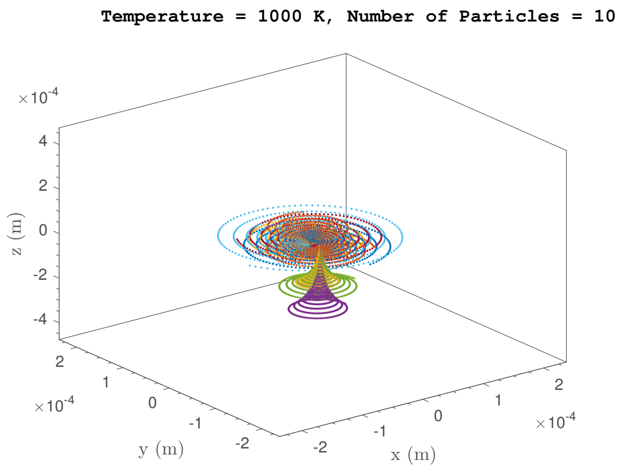

Figure 5: Larmor radius increase due to numerical error with Euler method.