Simulation of plasma with `cpic` using OmpSs

Rodrigo Arias Mallo

June 2, 2019

Contents

Chapter 1

Introduction

It may be surprising to find out that the most common state of matter is plasma when we look at the universe. A plasma is an ionized gas consisting of ions and free electrons distributed over a region in space. in which at least one electron of the atom is separated, so it remains positively charged (ionized) [1]. Usually this happens in the vacuum

1.1 Motivation

1.2 Objectives

1.3 Context

1.4 Structure

The structure of the document follows the diagram shown in the figure 1.1. In chapter 1, plasma is described as a physical phenomenon and we focus on the relevant properties that we want to study, from which we derive a mathematical model. In chapter 4 and **??**. The discretization of the mathematical model allows the computer simulation by using numerical algorithms.

Physical phenomenon

Mathematical model

Discretization

Numerical algorithms

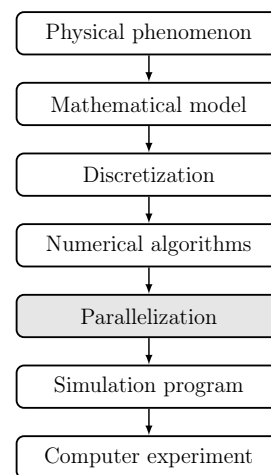Parallelization

Simulation program

Computer experiment

Figure 1.1: Principal steps in computer experiment

Chapter 2

Related work

[Briefly talk about other simulation software and techniques. Also we may want to add some historical references.]

Chapter 3

Introduction to plasma physics

[Introduce the reader in context. Some terminology to cover the scope of the simulation]

Chapter 4

Plasma simulation

4.1 The particle-in-cell method

Solving the Vaslov equation requires a large amount of numerical resources. The particle in cell method, approximates the solution by discretization of the fields.

The method is divided in four main phases:

- Particle motion.

- Charge accumulation.

- Solve field equation.

- Interpolation of fields in particle position.

4.2 Particle mover

In order to move the particles, the equations of motion need to be solved:

$$m\frac{d\boldsymbol{v}}{dt} = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \tag{4.1}$$

$$\frac{d\boldsymbol{v}}{dt} = \boldsymbol{v} \tag{4.2}$$

Several methods are available, but we will focus on the Boris integrator.

4.2.1 Boris integrator

Consists of three steps:

1. Add half of the electric impulse

2. Rotate

3. Add the remaining half electric impulse

The Boris integrator computes the velocity of a particle in a constant electric field $\boldsymbol{E}$ and a constant magnetic field $\boldsymbol{B}$. We have the velocity $\boldsymbol{v}_{t-\Delta t/2}$ of the particle at $t - \Delta t/2$ as we use the leapfrog integrator.
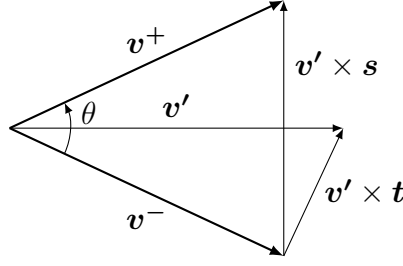
Figure 4.1: Velocity space rotation from $\boldsymbol{v}-$ to $\boldsymbol{v}+$

Add half electric impulse   We define $\boldsymbol{v}^-$ as the velocity after half a electric impulse:

$$\boldsymbol{v}^- = \boldsymbol{v}_{t-\Delta t/2} + \frac{q\boldsymbol{E}}{m}\frac{\Delta t}{2}$$

Rotate for the magnetic field   The rotation is done in two steps, first the half rotation is computed, with an angle of $\theta/2$:

$$\boldsymbol{v}' = \boldsymbol{v}^- + \boldsymbol{v}^- \times \boldsymbol{t}$$

Then the rotation is completed by symmetry, using the $\boldsymbol{s}$ vector

$$\boldsymbol{s} = \frac{2\boldsymbol{t}}{1 + \boldsymbol{t}^2}$$

as

$$\boldsymbol{v}^+ = \boldsymbol{v}^- + \boldsymbol{v}' \times \boldsymbol{s}$$

4.3   Charge accumulation

The charge density ρ is a scalar field

4.4   Field equations

Once we have the charge density ρ we can compute the electric field $\boldsymbol{E}$ by the integration of the field equations

$$\boldsymbol{E} = -\nabla\phi \tag{4.3}$$

$$\nabla \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0} \tag{4.4}$$

Which can be combined into the Poisson equation

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0} \tag{4.5}$$

Different methods can be used to obtain the electric field, but we will focus on finite difference equations and spectral methods.

4.4.1   Finite differences

$$\boldsymbol{E}_x(x, y) = \frac{\phi(x - 1, y) - \phi(x + 1, y)}{2\,\Delta x} \tag{4.6}$$

$$\boldsymbol{E}_y(x, y) = \frac{\phi(x, y - 1) - \phi(x, y + 1)}{2\,\Delta y} \tag{4.7}$$

4.4.2   Spectral methods

Chapter 5

Simulator design

[After determining the mathematical model and the discretization, we want to begin the discussion on how to build the simulator.]

5.1   Field solver

Talk about MFT and the data layout.

Chapter 6

The simulator

[This chapter is focused on the simulator implementation]

6.1   Data structures

The simulator is designed for space domains of one or two dimensions. In order to parallelize the computation of each step, the space domain is distributed in blocks. First the space domain is split in one specific dimension into MPI blocks, which will be distributed among each compute node. Communications will be needed to share information between MPI blocks.

The second hierarchy splits MPI blocks into task blocks, which can run in parallel inside a compute node. Communications are not needed, as we can use shared memory in the same compute node.

Inside each task block, we have a small portion of the space domain: the grid points of the fields and the particles inside the physical space of the block. Additionally, ghost points are placed at the boundaries of the positive neighbours in each dimension of the problem.

A summary of the data layout can be seen in the figure 6.1, where the physical placement of each block corresponds to the physical position of the grid points. The 1D domain has 2 MPI blocks with 3 task blocks each, and 3 grid points per block with 1 ghost point. In total, the space is discretized in 18 grid points, which require 24 with the ghost points.

Similarly, for 2D the number of ghost points increase, as the frontier now has 2 dimensions, leading to blocks with 6 grid points and 6 ghost points. The whole domain is discretized in 120 grid points, a total of 240 with ghost points.

6.2   Simulation flow

The simulation follows a very precise set of steps to ensure the correct behavior of the physical simulation. Three main stages can be easily identified: Initialization, loop and finish.

6.2.1   Initialization

The iteration counter is initially set to -2, as we are going to do two previous phases before the simulation begins:

Allocation phase   After P processes were created (as in MPI processes), now we create the different structures to hold the simulation data. First the fields are distributed into task blocks, grouped in each process. For each specie, we distribute the particles based on the particle index
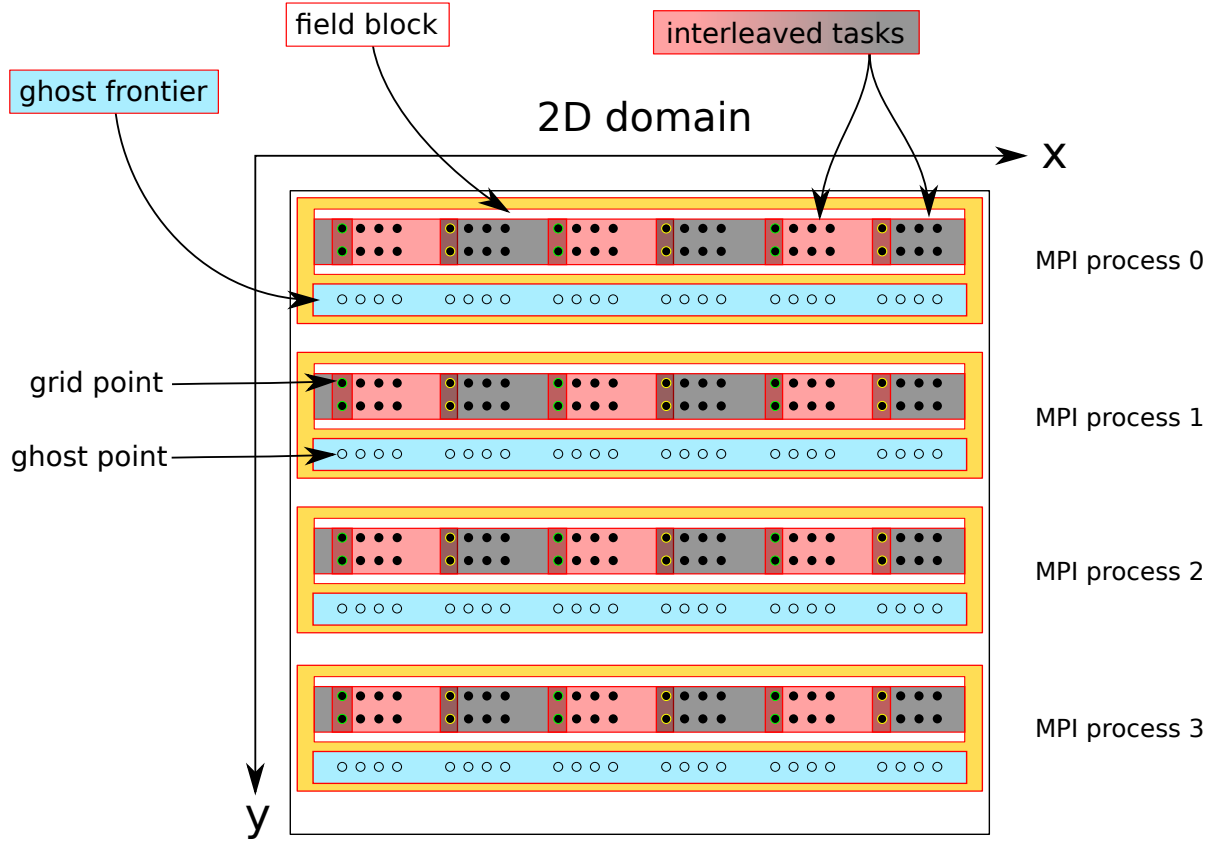
Figure 6.1: Domain blocks

p_i, minimizing the difference of the number of particles between blocks. The position, velocity and other parameters are set on each particle, independently of the actual block they reside.

Once all particles are initialized, we begin to move them to the correct block, based on the particle position, and increment the iteration counter. Finally, the charge density field ρ is initially computed, as we want the begin the simulation with the computation of the electric field E from ρ

Rewind phase   The simulation time is not advanced equally for the speed and position of the particles. At time t the velocity computed at time $t + \Delta t/2$ whereas the position is computed at time t. In order to begin the simulation, the velocity of the particles is advanced half time-step backwards in time. This extra step is computed at the iteration $i = -1$, as we need the iteration counter to be always increasing (as is used in the message exchange as unique identifier).

6.2.2   Loop

The loop of the simulation perform four main phases:

- Solve field equation to get E from ρ.

- Interpolate E at particle positions E_p.

- Particle motion based on E_p and B_0.

- Accumulate charge density ρ at the new position of particles.

Solver

We use the MFT method to solve the equation:

$$\nabla^2 E = -\frac{\rho}{\epsilon_0} \tag{6.1}$$

6.2.3 Finish

Here we hopefully save some information of the simulation to disk...

Chapter 7

Analysis

In order to reduce the amount of CPU time involved in each step of the simulation, the best strategy is to reduce the time spent in the most time consuming part.

The CPU time involved in each part of the simulation may depend on various factors, such as the number of grid points, the number of particles or the boundary conditions. As an example, consider a simulation with a large number of grid points, with few particles—the computation of the electric field (`field_E`) will dominate the simulation time, as shown in the figure 7.1a. In a case of a large number of particles and a smaller grid, the particle interpolation (`particle_E`) dominates the whole execution as seen in the figure 7.1b.
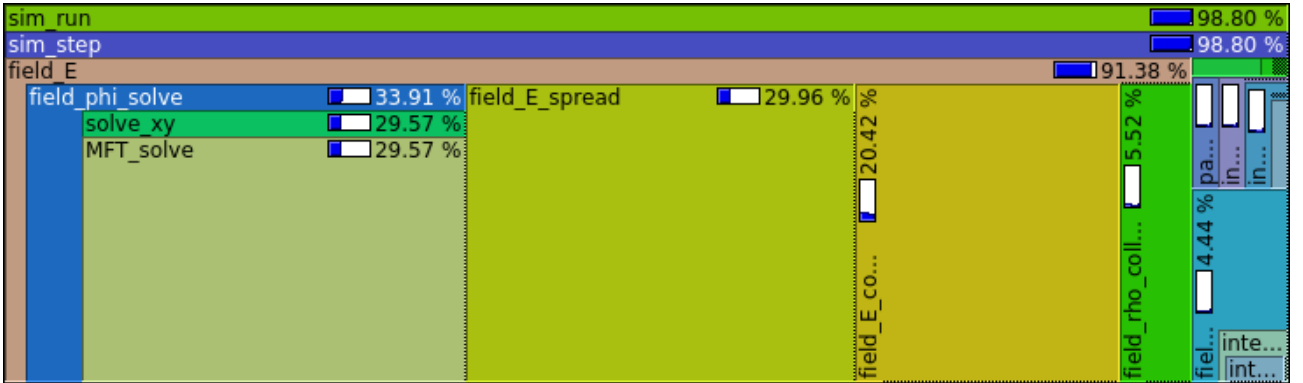
(a) 1024 particles, 512x512 grid points
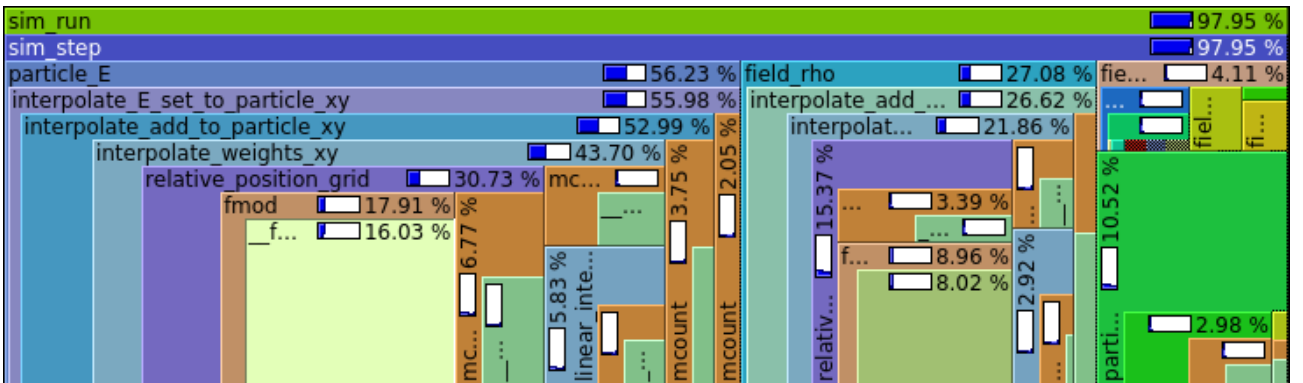
(b) 10240 particles, 64x64 grid points

Figure 7.1: Comparison of the time spent in each function at two different simulations.

In order to optimize the general use case, different inputs will be tested and the main simulation steps will be characterized. Furthermore, different algorithms or methods may be used to improve the speed. As an example, the LU algorithm is compared with the spectral method MFT.

7.1   Analysis with varying inputs

Chapter 8

Caveats and limitations

Here is a list of things that are not implemented in the simulator, but may be added in a future work.

- More than 2 dimensions

- Fully electromagnetic simulation

- Relativistic particle movement

- Heterogeneous architecture (GPU+CPU...)

- Energy conserving codes

- Visualization of big simulations (paraview)

Caveats that need to be fixed

- Allow to specify plasma frequency

- Validation with other simulation codes

Bibliography

[1] F. F. CHEN, *Introduction to Plasma Physics and Controlled Fusion: Volume 1: Plasma Physics*, Plenum Press.