Pontificia Universidad Católica de Chile
Facultad de Física
Instituto de Física

DEVELOPMENT OF PARTICLE-IN-CELL CODE USING THE IMPLICIT MOMENT METHOD AND MONTE-CARLO COLLISIONS FOR LABORATORY PLASMA SIMULATION

by
Francisco Hernández Vargas

Supervisor: Prof. Mario Favre

Reviewers: Prof. Mario Riquelme

Prof. Felipe Veloso

This Thesis is Submitted in Partial Fulfilment of the Requirements
for the Degree of Master of Science in Physics

Santiago - 2020

Acknowledgments

Agradezco enormemente a mi familia y amigos que hicieron esto posible. Sin su apoyo y cariño habría sido muy difícil lograr este trabajo.

Gracias al grupo de plasma, a Milenko, Pavel, Vicente, Gonzalo, Francisca, Fabián, Miguel por acompañar las jornadas en el laboratorio. A Carlos por recibirme en la oficina-cowork. Al Ari por sus consejos y apoyo.

Agradezco también a los profesores, especialmente a Mario por supervisar el trabajo y estar atento a responder todas mis dudas. Gracias Edmund por los consejos, no solo de plasma sino que de la vida en general.

I am very grateful for funding granted by FONDECYT cod. 1180100, and the Instituto de Física de la Pontificia Universidad Católica de Chile.

Abstract

Plasma simulations have been used for decades as a link between theoretical and experimental physics, helping to understand the dynamics and phenomena that occur in plasma. Among the existing methods there are fluid codes (magnetohydrodynamics), kinetic codes (particle-in-cell) and hybrids codes (which combine the previously mentioned methods), each with its pros and cons.

In this thesis, a C++ simulation code was developed using the particle-in-cell implicit moment method (PIC-IMM), which has the advantage of having fewer constrains than other methods (such as the explicit PIC), maintaining the kinetic and non-linear effects. In addition, a collision module was implemented using Monte Carlo Collisions (MCC).

The code was tested with verification and validation tests. The correct movement and stability of the particles was verified, as well as the dispersion of the electrostatic fields.

Among the validation tests, the case of two-stream instability, magnetic reconnection using a Harris-like sheet, Xenon charge-exchange collisions, and Argon collisions in a rf plasma jet was analyzed.

Each of the tests was analyzed and the results are discussed in the respective chapters. Although the results obtained are satisfactory, it remains to simulate new laboratory experiments, and compare with experimental data.

Contents

List of Figures

List of Abbreviations

PIC	.....	Particle-in-Cell
MCC	.....	Monte-Carlo Colissions
IMM	.....	Implicit Moment Method
MHD	.....	Magnetohydrodynamics
DSMC	.....	Direct Simulation Monte Carlo
BC	.....	Boundary Conditions
GMRES	.....	Generalized Minimal Residual
CPU	.....	Central Processing Unit
MPI	.....	Message Passing Interface
IDE	.....	Integrated Development Environment
FDTD	.....	Finite-difference time-domain
IVDF	.....	Ion Velocity Distribution Function

Chapter 1

Introduction

1.1 Plasma comes in different scales

A plasma can be described as a globally neutral ionized gas, composed of electrons and ions. Plasmas are characterized by their collective behavior, and their properties have been studied for decades given their importance and applications in science and technology. More than 99% of the observable universe is made up of plasma [4], and there are plenty of phenomena determined by plasma physics, such as solar corona, solar wind, our planet's ionosphere and fusion plasmas. Understanding these phenomena implies understanding the properties and dynamics of plasma.

An important parameter in plasma is the Debye length, which measures the distance at which particles are electrically shielded from one another. This scale, as well as other parameters, can vary by several orders of magnitude between plasmas. Some of the characteristics found in plasmas in the universe are summarized in Table 1.1

Plasma	Density (m^{-3})	Temperature (keV)	Magnetic Field (T)	Debye length (m)
Interstellar	10^{6}	10^{-5}	10^{-9}	0.7
Solar wind	10^{7}	10^{-2}	10^{-8}	7
Ionosphere	10^{12}	10^{-4}	10^{-5}	$2 \cdot 10^{3}$
Solar corona	10^{12}	10^{-1}	10^{-3}	0.07
Ion thruster	10^{15}	10^{-3}	–	$4 \cdot 10^{-4}$
Arc discharge	10^{20}	10^{-3}	10^{-1}	$7 \cdot 10^{-7}$
Tokamak	10^{20}	1	10	$7 \cdot 10^{-5}$
Inertial Confinement Fusion	10^{28}	10	–	$7 \cdot 10^{-9}$

Table 1.1: Table with some of the plasma parameters for different plasmas [5]

1.2 The age of supercomputing

Plasmas are inherently complex systems, where particles interact collectively with each other through electromagnetic fields in addition to experiencing one-to-one collisions. In some simple enough cases, analytical solutions exist to describe the evolution of the system. However, in many cases the solution is so complex that such analytical treatment is not possible in practice.

For these cases there are two alternatives: to simplify the physical model or to look for a numerical solution. Given that the computational power has increased considerably in the last decades (see Fig. 1.1), the second alternative has become an increasingly used option in plasma physics, making it possible to study the evolution of realistic physical systems, and allowing to compare the most complex models with experimental results.

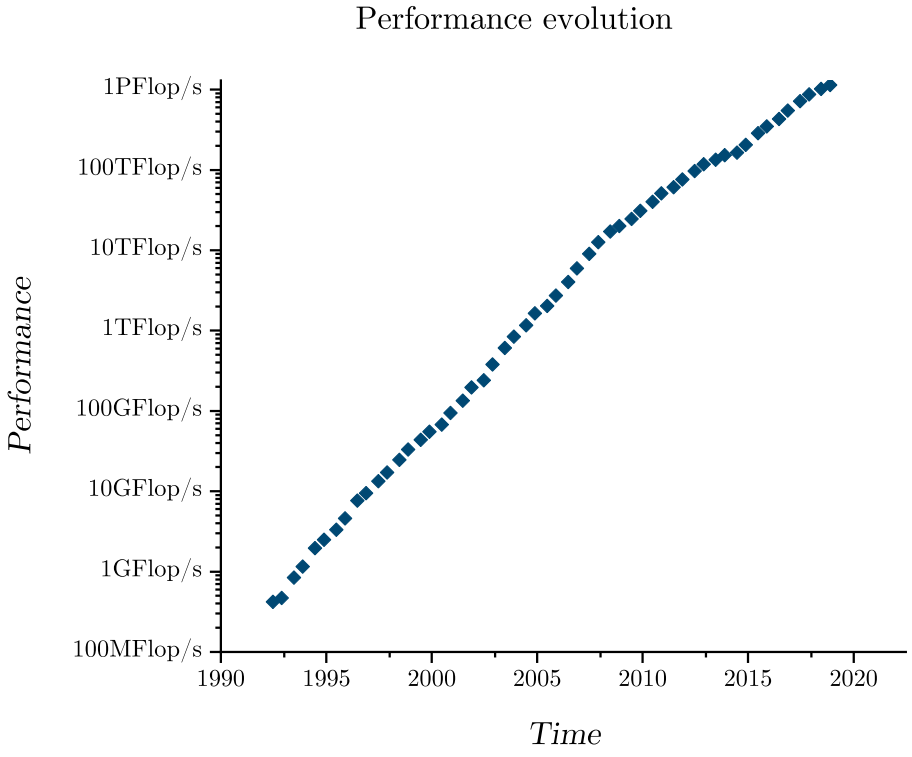


Figure 1.1: Growth of supercomputing power over time. Data represent the average performance of 500 most powerful supercomputers at the time. Data from https://www.top500.org, retrieved on May 15,2020.

This is how computational physics has positioned itself as a key area between theoretical and experimental physics. One of the great challenges is to be able to use models that are feasible to simulate in human time, while rescuing as much of their physics as possible.

In plasma physics, one of those challenges has to do with the multiple scales involved in plasmas. As mentioned above, there are plasma parameters that can present differences of

several orders of magnitude. For example, in a Tokamak, the electron plasma oscillation period $1/\omega_{pe}$ is of the order of 10^{-12}s, while the energy confinement time τ_E is of the order of 10^0s [6]. Thus, capture simultaneously both MHD and kinetic effects, we would have to increase the simulation time by many orders of magnitude, compared to simulations that only capture the MHD time scales. Including the kinetic effects , however, is important, since various phenomena depend on the dynamics of the electrons, such as magnetic reconnection.

A large number of plasma simulation codes have been previously developed, with different methods and approaches. Some of them are GORGON (Imperial College)[7], HYDRA[8] (National Ignition Facility)), DISCO[9] (University of California) using the MHD method, and VORPAL[10] (University of Colorado), EPOCH[11] (Warwick), VPIC[12] (Los Alamos National Laboratory), PIConGPU[13] (Helmholtz-Zentrum Dresden-Rossendorf), ALaDyn[14] (Univ. Di Bologna), Smilei[15] (collaborative) among many others that use particle-in-cell.

1.3 About this thesis

The objective of this thesis is to develop a simulation code based on the particle-in-cell implicit-moment method (PIC-IMM) combined with Monte-Carlo collisions (MCC). In this way, we expect to be able to kinetically simulate dense plasmas, while preserving the information of non-linear and collective effects. This work is partially inspired by the work of Schmidt [16], where they simulated a focus plasma device using a implicit PIC code. In their results, they managed to obtain neutral energies of the order of magnitude shown by the experiments, unlike the MHD and hybrid methods, which do not capture the physics of the eletrons.

Chapter 2 will summarize the simulation methods currently used to simulate plasmas, reviewing the advantages and disadvantages of each one. At the end of the chapter, a complete description of the methods used in the code will be made, including PIC-IMM and Monte-Carlo collisions.

Chapter 3 will present the results of the developed code, testing it against various verification and validation tests, including the propagation of oscillations, two-stream instability and magnetic reconnection on a Harris current sheet.

Finally, in Chapter 4, the conclusions of this work will be detailed, as well as the proposed future work.

Chapter 2

Plasma simulations

There are several models to describe plasmas. Usually we can categorize them into three types: kinetic, hybrid and fluid (or MHD). Fluid models make use of the MHD equations to describe the evolution of plasmas. One of the downside of MHD is that they do not contain information about the kinetic effects, which can be relevant in the dynamics of magnetic reconnection, and the evolution of the system[17].

In the kinetic model, all plasma species (electrons, ions and neutrals) are trated as particles, so the full kinetic and non-linear effects are included in the simulation. One of the most used kinetic methods is called particle-in-cell (PIC). Of course, as we include more physics, the computational cost will be higher, and for this reason PIC simulations are often limited to small simulation domains and/or a reduced number of dimensions[18].

Hybrid method is an intermediate solution, where neutrals and ions are simulated as particles and electrons are simulated as a background fluid ensuring global plasma neutrality [19]. This approach relax the PIC stability constrains and allows larger timesteps as well as larger grid sizes.

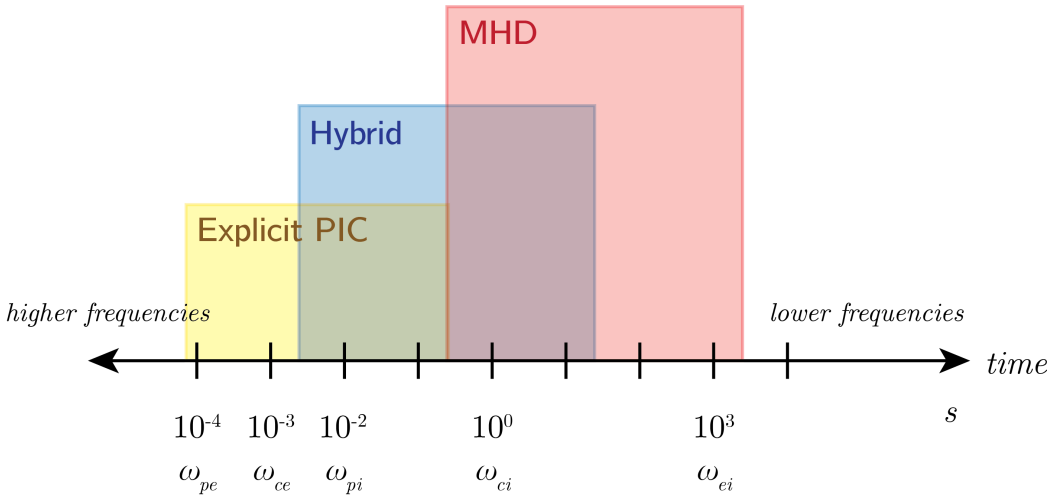


Figure 2.1: Time scales that different simulation methods can achieve. In this example, Earth's magnetotail parameters [1] was considered, with $T_i = 1keV$, $T_e = 0.1keV$, $n = 0.3cm^{-3}$ and $B = 30nT$.

Some of the time scales that each methods seeks to cover are shown in Fig. 2.1 for a given plasma. We can notice that MHD can cover lower frequencies, whereas kinetic simulations can cover higher frequencies. MHD simulations are prefered when kinetic effects are not so relevant and when temporal and spatial scales are large.

In this chapter we will describe briefly the MHD method, and then, we will introduce the PIC method and its variant Implicit Moment Method (IMM), which was used in this thesis. After that, the Monte-Carlo Collisions method will be introduced, as well as the specific models of Xenon and Argon implemented in the developed software.

2.1 MHD

Magnetohydrodynamics (MHD) studies the behavior of electrically conducting fluids. The term was introduced by Hannes Alfvén in 1942[20], and the equations that describe this model are a combination of the Navier–Stokes equations and Maxwell's equations. The simplest MHD model is the ideal MHD, which is described by the following equations (in CGS units) [21]

1. Mass continuity equation
$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{2.1}$$

2. Energy equation
$$\frac{d}{dt}\left(\frac{p}{p^{\gamma}}\right) = 0 \tag{2.2}$$

3. Momentum equation
$$\rho\left(\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla\right)\mathbf{v} - \mathbf{J} \times \mathbf{B} + \nabla p = 0 \tag{2.3}$$

4. Ampere's law
$$\nabla \times \mathbf{B} = \frac{4\pi}{c}\mathbf{J} \tag{2.4}$$

5. Faraday's law
$$\frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0 \tag{2.5}$$

6. Ideal Ohm's law
$$\mathbf{E} + \frac{1}{c}\mathbf{v} \times \mathbf{B} = 0 \tag{2.6}$$

7. And the magnetic divergence constraint
$$\nabla \cdot \mathbf{B} = 0 \tag{2.7}$$

where ρ is the mass density, J is the current density, $\mathbf{v}$ is the plasma velocity, $\mathbf{B}$ is the magnetic field, $\mathbf{E}$ is the electric field, p is the plasma pressure, γ is the adiabatic index (the heat capacity ratio, usually 5/3), t is time and c is the speed of light.

The MHD model is only valid when high-frequencies and short scales are not relevant. In other words, the MHD is a low-frequency and long-wavelength approximation, and the following restrictions must be met to keep the model valid:

1. The collisional frequency is high, so, Maxwellian distribution is assumed and plasma is close to thermal equilibium ($T_i = T_e$).
2. Spacial-scales are larger than the Debye length and ion/electron cyclotron radius.
3. Timescales are longer than ω_p^{-1} (inverse of plasma frequency) and $\omega_{ci,ce}^{-}1$ (inverse of ion / electron cyclotron frequencies).
4. Relativistic and quantum mechanics effects are not important.

Despite these restrictions, this model is widely used since it allows simulating large dimensions with larger timesteps, especially in space plasmas [22]. There are extensions of the ideal MHD model, such as resistive and hall MHD, that help to improve the model and are even capable of simulating magnetic reconnections [23]. In many cases, however, the kinetic effects are important[24] like, for example, when the particle distribution are not Maxwellian (such as cosmic rays) or when the plasma is weakly ionized. In those cases it is necessary to use codes that can include kinetic and non-linear effects, such as the PIC method.

2.2 Particle-in-cell method

The particle-in-cell (PIC) method is characterized by the use of superparticles to simulate the evolution of the system's distribution function, allowing information to be maintained from a kinetic point of view and thus including non-linear and collective effects.

Simulating a system with the real number of particles is computationally prohibitive. Hence, the use of computational superparticles which represents a cluster of real particles, which in turn represents a bounded domain in the space-phase. Hence, this approach will be valid for the case of weakly coupled systems.

The electric and magnetic fields are solved in a discrete mesh, while the particles live in a continuum space domain. This interaction between mesh and particles gives the method its name. Another important property of this method is that it is not necessary to assume any particular initial distribution function in the simulation.

Next, the derivation of the PIC method will be described. This section and the next one are strongly based on the descriptions made by Lapenta [23], Markidis [22] and Vu[2].

Let us consider a non-collisional plasma. The temporal evolution of the particles distribution $f_s(\mathbf{x}, \mathbf{v}, t)$ of a plasma in phase-space is given by the known Vlasov-Maxwell equation [25].

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \frac{\partial f_s}{\partial \mathbf{x}} + \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_s}{\partial \mathbf{v}} = 0 \tag{2.8}$$

where q_s and m_s are the charge and mass of each species respectively. Although we are not considering collisions at the moment, they can be easily added afterward.

In PIC, the distribution function of particles of a species $f_s(\mathbf{x}, \mathbf{v}, t)$ can be calculated as the sum of the distribution function $f_p(\mathbf{x}, \mathbf{v}, t)$ of each computational particle:

$$f_s(\mathbf{x}, \mathbf{v}, t) = \sum_{n=1}^{N_s} f_p(\mathbf{x}, \mathbf{v}, t) \tag{2.9}$$

where N_s is the total number of particles of species s. Since each computational particle represents a small area in the phase-space, we can assign a fixed shape function S for the particle position and velocity.

$$f_p(\mathbf{x}, \mathbf{v}, t) = N_p S_{\mathbf{x}} \left(\mathbf{x} - \mathbf{x}_p\right) S_{\mathbf{v}} \left(\mathbf{v} - \mathbf{v}_p\right) \tag{2.10}$$

where $S_{\mathbf{x}}$ and $S_{\mathbf{v}}$ represent the shape functions and N_p the number of real particles for each superparticles. For the spatial shape function, the most common in PIC methods is to use the b-spline functions[26], and a delta function for the velocity shape function [27]

The b-spline function are defined recursively as

$$b_0(\xi) = \begin{cases} 1 & \text{if } |\xi| < 1/2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

And the following functions as

$$b_l = \int_{-\infty}^{\infty} b_0(\xi - \xi') b_{l-1}(\xi') d\xi' \tag{2.12}$$

where b_0 is the first spline function and b_l the subsequent. An important property is $\sum_i^{\infty} b_l(\xi + i) = 1$ regardless of the central point and $\int_{\infty}^{\infty} b_l(\xi) d\xi = 1$. Figure 2.2 shows the three firsts b-spline functions.

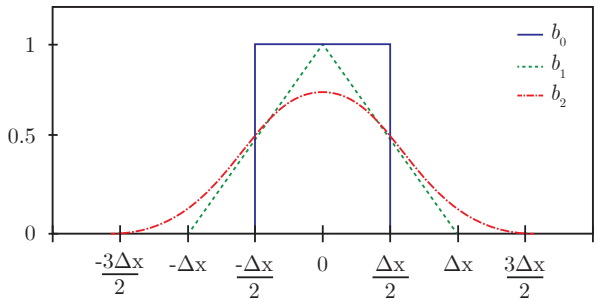


Figure 2.2: First three b-spline functions $b_l(\frac{x}{\Delta x})$

In order to obtain the equations of motion, we have to get the first moments of the Vlasov equation. Substituting Eq. 2.10 in 2.8, multipliying by $\mathbf{x}$ and $\mathbf{v}$ and integrating, one can derive[17]:

$$\begin{aligned} \frac{dN_p}{dt} &= 0 \\ \frac{d\mathbf{x}_p}{dt} &= \mathbf{v}_p \\ \frac{d\mathbf{v}_p}{dt} &= \frac{q_s}{m_s}(\mathbf{E}_p + \mathbf{v}_p \times \mathbf{B}_p) \end{aligned} \tag{2.13}$$

We notice that Eq. 2.13 resemble the Newton equations of motion. The electric and magnetic field acting on the particle can be calculated by integrating the shape function and the electromagnetic field over the computational domain V

$$\mathbf{E}_p = \int_V S_{\mathbf{x}}\left(\mathbf{x} - \mathbf{x}_p\right) \mathbf{E}(\mathbf{x}) d\mathbf{x} \tag{2.14}$$

$$\mathbf{B}_p = \int_V S_{\mathbf{x}}\left(\mathbf{x} - \mathbf{x}_p\right) \mathbf{B}(\mathbf{x}) d\mathbf{x} \tag{2.15}$$

In PIC method, the Maxwell equations are solved in a grid, hence, we need to introduce an interpolation function W

$$W(\mathbf{x}_g - \mathbf{x}_p) = \int_V S_{\mathbf{x}}\left(\mathbf{x} - \mathbf{x}_p\right) b_0\left(\frac{\mathbf{x} - \mathbf{x}_g}{\Delta \mathbf{x}}\right) d\mathbf{x} = \frac{b_1(\mathbf{x} - \mathbf{x}_p)}{\delta \mathbf{x}} \tag{2.16}$$

where the g and p subscripts refers to grid and particle values respectively.

So, now, we can represent the electric and magnetic fields acting on a particle as

$$\mathbf{E}_p = \sum_g \mathbf{E}_g W(\mathbf{x}_g - \mathbf{x}_p) \tag{2.17}$$

$$\mathbf{B}_p = \sum_g \mathbf{B}_g W(\mathbf{x}_g - \mathbf{x}_p) \tag{2.18}$$

Maxwell's equations need to be solved in order to calculate the electric and magnetic fields.

$$\nabla \cdot \mathbf{E} = 4\pi\rho \tag{2.19}$$

$$\nabla \times \mathbf{E} = -\frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} \tag{2.20}$$

$$\nabla \times \mathbf{B} = \frac{1}{c}\left(4\pi\mathbf{J} + \frac{\partial \mathbf{E}}{\partial t}\right) \tag{2.21}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.22}$$

where ρ and $\mathbf{J}$ are the charge and current density respectively. In PIC we can calculate those quantities as

$$\rho = \sum_s q_s \int f_s d\mathbf{v} \tag{2.23}$$

$$\mathbf{J} = \sum_s q_s \int \mathbf{v} f_s d\mathbf{v} \tag{2.24}$$

Using the interpolation function, we can write 2.23 and 2.24 as

$$\{\rho, \mathbf{J}\}_g = \frac{1}{\Delta\mathbf{x}} \sum_s \sum_p q_s \{1, \mathbf{v}_p\} W(\mathbf{x}_g - \mathbf{x}_p) \tag{2.25}$$

In explicit PIC, the electric and magnetic field are usually solved using the finite-difference time-domain (FDTD) method and its variations[28]. Those methods allows an easily calculation of discretized curl and divergence operators for the calculation of Maxwell's equations.

2.2.1 Temporal discretization

To move the particles, we need to integrate over time the equation of motions. where q_s is the charge of the particle of species s. The simplest numerically stable option is the *leapfrog algorithm*, in which the position is calculated at times n and the velocity at times $n + 1/2$. This algorithm has the advantage that is as simple as Euler algorithm, but the small difference of calculating the position and velocity at different times makes it stable at second order.

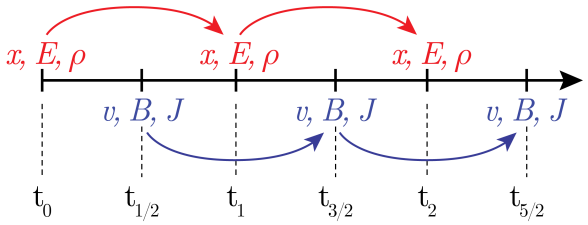


Figure 2.3: Visual representation of the *Leapfrog algorithm*. The position, electric field and charge density are calculated at times n and velocity, magnetic field and current density at times $n + 1/2$

The particle advance can be calculated with the following equations;

$$x_i^{n+1} = x_i^n + \Delta t \mathbf{v}_i^{n+\frac{1}{2}} \tag{2.26}$$

$$\mathbf{v}_i^{n+\frac{3}{2}} = \mathbf{v}_i^{n+\frac{1}{2}} + \Delta t \frac{q_s}{m_s} \mathbf{E^{n+1}}_i(\mathbf{x}^{n+1}) + \Delta t \left(\frac{\mathbf{v}_i^{n+\frac{3}{2}} + \mathbf{v}_i^{n+\frac{1}{2}}}{2} \right) \times \mathbf{B}^{n+1}(\mathbf{x}^{n+1}) \tag{2.27}$$

$$\mathbf{v}_i^{\frac{1}{2}} = \mathbf{v}_i^0 + \Delta t \frac{q_s}{2m_s} \mathbf{E}_i(\mathbf{x}^0) + \Delta t \frac{q_s}{2m_s} \left(\frac{\mathbf{v}_i^{\frac{1}{2}} + \mathbf{v}_i^0}{2} \right) \times \mathbf{B}^0(\mathbf{x}^0) \tag{2.28}$$

This completes the description of the PIC method. The described calculations are separated into steps and solved sequentially in each cycle. The first step is the initialization of the simulation, where the simulation parameters are configured (physical dimensions, grid, timestep, boundary conditions, etc.), and the variables are initiated (particles, initial distribution, external fields, etc.). After the first step, the PIC cycle begins. First, the particles are used to interpolate the values of charge density and current on the grid. Those values are then used to solve the Maxwell equations on the grid and obtain the value of the new electric and magnetic fields. After solving the values of the new fields, they are interpolated from the grid to the particles positions to calculate the force that will affect them. Then, the force is calculated and the particles move according to Newton's equations. Finally, if the Monte-Carlo collisions method is added, after moving the particles the possible collisions are checked and resolved. After that, the time is advanced Δt and the PIC cycle restarts, going

back through the steps already mentioned. Figure 3.3 shows a visual representation of the explicit PIC simulation cycle.

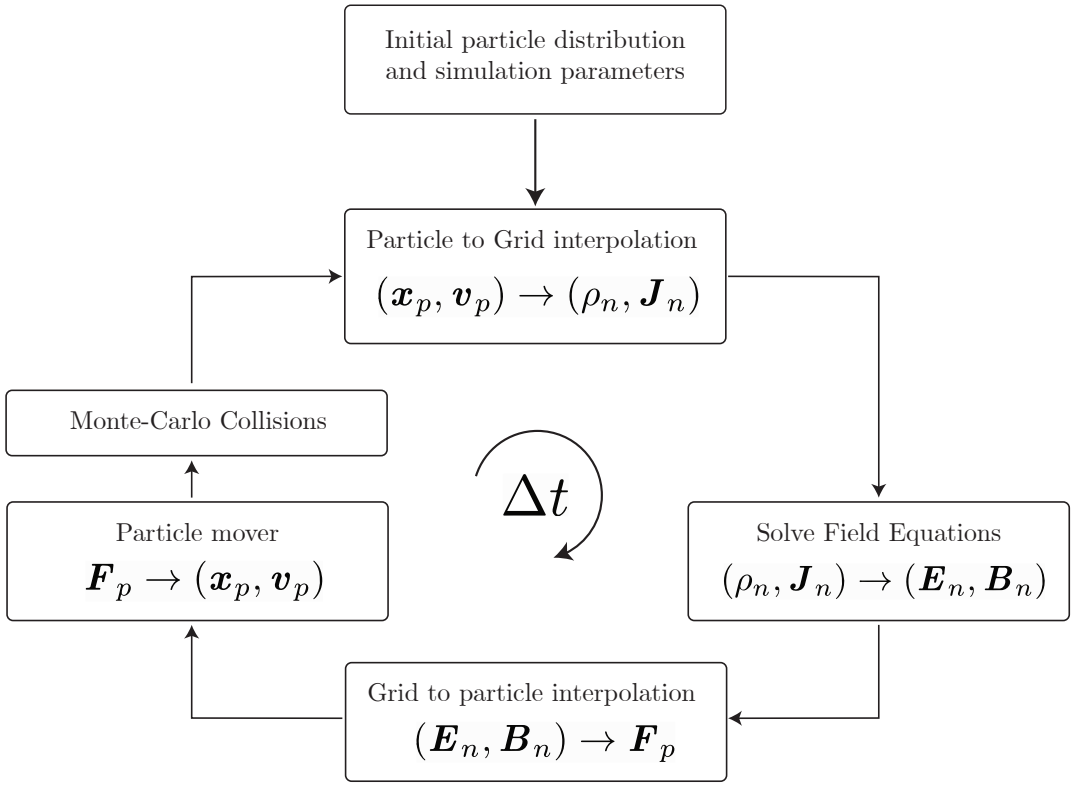


Figure 2.4: PIC simulation cycle including Monte-Carlo Collisions. After the inicial step, the following steps are repeated over the simulation advancing Δt at each cycle.

2.2.2 Stability constraints

The PIC methods introduces several constraints in the simulation. First, as the particle mover is discretized in time, we need to resolve the fastest electron time [23]

$$\omega_{pe}\Delta t < 2 \tag{2.29}$$

where ω_{pe} is the electron plasma frequency, but usually a smaller constant is used to avoid numerical heating [29], typically $\omega_{pe}\Delta t \leq 0.1$[23]. Additionally, as we are using a discretized grid, the differentiation of Maxwell equations requires that the following condition must be satisfied

$$c\Delta t < \Delta x \tag{2.30}$$

Eq. 2.30 is known as the Courant–Friedrich–Levy (CFL) condition. Basically, it states that the timestep should not exceed the time taken by a signal to travel one cell.

Finally, the interpolation between the grid and particles introduces the known *grid instability*

$$\Delta x < \zeta \lambda_D \tag{2.31}$$

where λ_D is the Debye length and ζ is a proportionality constant which is of order one[29]. This condition require us to use a grid spacing smaller o similar to λ_D to maintain the stability.

2.3 Implicit Moment Method

To overcome the restrictions of the explicit PIC method, several alternatives have been considered. Some of them consist of reducing or eliminating a certain part of the simulated physics. Others try to solve the Vlasov equation directly using numerical methods. The latter is unfortunately not feasible for problems in 3 dimensions, since the Vlasov equation becomes a 6-dimensional problem (3 of position and 3 of speed). One of the alternatives are the so-called semi-implicit methods, where an approximation is used to alleviate the computational load and separate the coupling between the Newton and Maxwell equations.

The implicit moment method (IMM) is a type of semi-implicit PIC method developed in Los Álamos, US [30, 31], and allows the use of larger timesteps, eliminating some constrains of the explicit PIC method. It is based on the extrapolation of charge density and current density quantities, by performing a Taylor expansion.

As described by Markidis [22], we start with Maxwell's equations:

$$\nabla \cdot \mathbf{E} = 4\pi\rho \tag{2.32}$$

$$\nabla \times \mathbf{E} = -\frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} \tag{2.33}$$

$$\nabla \times \mathbf{B} = \frac{1}{c}\left(4\pi\mathbf{J} + \frac{\partial \mathbf{E}}{\partial t}\right) \tag{2.34}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.35}$$

Taking the curl in the Maxwell-Faraday equation, and then using the Ampere-Maxwell equation, we get:

$$\nabla^2 \mathbf{E} = \frac{1}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} + \frac{4\pi}{c^2}\frac{\partial \mathbf{J}}{\partial t} + 4\pi\nabla\rho \tag{2.36}$$

Eq. 2.36 can be differentiated in time (n to $n+1$) as [30, 31]:

$$\mathbf{E}^{n+1} - \nabla^2 \mathbf{E}^{n+1} = \mathbf{E}^n + c\Delta t\left(\nabla \times \mathbf{B}^n - \frac{4\pi}{c}\mathbf{J}^{n+1/2}\right) - (c\Delta t)^2 4\pi\nabla\rho^{n+1} \tag{2.37}$$

As already mentioned, this method is based on the extrapolation of the future values of the charge density ρ and current density $\mathbf{J}$. In the IMM PIC, the interpolation function $W(\mathbf{x} - \mathbf{x}_p^{n+1})$ is expanded by Taylor.[2]

$$W(\mathbf{x} - \mathbf{x}_p^{n+1}) \approx W(\mathbf{x} - \mathbf{x}_p^n) + (\mathbf{x} - \mathbf{x}_p^{n+1})\nabla W(\mathbf{x} - \mathbf{x}_p^n) + ... \tag{2.38}$$

Inserting the approximated quantities ρ^{n+1} and $\mathbf{J}^{n+1/2}$ into Eq. 2.37 and after a series of manipulation and keeping second order terms in Δt, an equation for $\mathbf{E}^{n+1}$ is obtained:

$$\begin{aligned}(\mathbf{I} + \chi^n)\cdot\ \mathbf{E}^{n+1} - (c\Delta t)^2\left(\nabla^2\mathbf{E}^{n+1} + \nabla\nabla\cdot\left(\chi^n + \mathbf{E}^{n+1}\right)\right)\\ = \mathbf{E}^n + c\Delta t\left(\nabla\times\mathbf{B}^n - \frac{4\pi}{c}\hat{\mathbf{J}}^n\right) - (c\Delta t)^2\,\Delta 4\pi\hat{\rho}^n\end{aligned} \tag{2.39}$$

Where $\hat{\rho}^n$ and $\hat{J}^n$ where introduced.

$$\hat{\rho}^n = \rho^n - \Delta t\nabla\cdot\hat{J}^n \tag{2.40}$$

$$\hat{J}^n = \sum_s^{n_s} R\cdot\left(\mathbf{J}_s^n - \frac{\Delta t}{2}\nabla\Pi_s^n\right) \tag{2.41}$$

The *pressure tensor* Π_s^n is defined as:

$$\Pi_s^n = \frac{1}{\Delta\mathbf{x}}\sum_p^{N_s} q_s\mathbf{v}_p^2 W(\mathbf{x} - \mathbf{x}_p^n) \tag{2.42}$$

The *implicit susceptibility* χ is introduced, defined as:

$$\chi \equiv \sum_{n_s}\frac{1}{2}(\omega_{ps}\Delta t)R \tag{2.43}$$

Where $\omega_{ps} = \sqrt{4\pi\rho_s q_s/m_s}$

The rotation transform R is defined as:

$$R = \begin{bmatrix} 1+\Omega_{s_x}^2 & \Omega_{s_z}+\Omega_{s_x}\Omega_{s_y} & -\Omega_{s_y}+\Omega_{s_x}\Omega_{s_z} \\ -\Omega_{s_z}+\Omega_{s_x}\Omega_{s_y} & 1+\Omega_{s_y}^2 & \Omega_{s_x}+\Omega_{s_y}\Omega_{s_z} \\ \Omega_{s_y}+\Omega_{s_x}\Omega_{s_z} & -\Omega_{s_x}+\Omega_{s_y}\Omega_{s_z} & 1+\Omega_{s_z}^2 \end{bmatrix} \tag{2.44}$$

where $\mathbf{\Omega}_s = \dfrac{q_s \mathbf{B}^n \Delta t}{2 m_s c}$. The subscript in Ω_s refers to the vector component in $\mathbf{B}^n$.

Eq. 2.39 can be solve with an iterative method, like the generalized minimal residual method (GMRES). After obtaining the value of $\mathbf{E}^{n+1}$, we can calculate the magnetic field using the differentiated version of the Faraday's Law:

$$\mathbf{B}^{n+1} = \mathbf{B} - c\Delta t \nabla \times \mathbf{E}^{n+1} \tag{2.45}$$

2.3.1 Particle mover

The particles can be advanced in time by:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \mathbf{v}_p^{n+1/2} \Delta t \tag{2.46}$$

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \frac{q_s}{m_s} \left(\mathbf{E}_{mid}^{n+1} + \frac{\mathbf{v}_p^{n+1/2} \times \mathbf{B}_{mid}^{n+1}}{c} \right) \Delta t \tag{2.47}$$

where $\mathbf{E}_{mid}$ and $\mathbf{B}_{mid}$ are the electric and magnetic field respectively calculated in the middle of the current and future positions $\mathbf{x}^{n+1/2}$. The midpoint velocity $\mathbf{v}_p^{n+1/2}$ can be calculated with the following equation:

$$\mathbf{v}_p^{n+1/2} = \frac{1}{1 + \left(\alpha \mathbf{B}_{mid}^{n+1}\right)^2} \left(\tilde{\mathbf{v}}_p + \alpha \tilde{\mathbf{v}}_p \times \mathbf{B}_{mid}^{n+1} + \alpha^2 (\tilde{\mathbf{v}}_p \cdot \mathbf{B}_{mid}^{n+1}) \mathbf{B}_{mid}^{n+1} \right) \tag{2.48}$$

where $\alpha = (q_s \Delta t / (2 m_s c))$. The velocity $\tilde{\mathbf{v}}_p$ was introduced for convenience, defined as:

$$\tilde{\mathbf{v}}_p = \mathbf{v}_p + (q_s \Delta t / m_s)\, \mathbf{E}_{mid}^n \tag{2.49}$$

With this equation, we can finally calculate $\mathbf{v}_p^{n+1}$:

$$\mathbf{v}_p^{n+1} = 2\mathbf{v}_p^{n+1/2} - \mathbf{v}_p^n \tag{2.50}$$

2.3.2 Energy integral

The kinectic energy of particles can be easily calculated as the sum of the kinetic energy of individual particles:

$$\text{Total Energy}_{\text{K}} = \sum_{n=1}^{N} \frac{1}{2} m \mathbf{v}_n^2 \tag{2.51}$$

where N is the total number of particles

Energy stored in electromagnetic fields in Gaussian units is given by:

$$\text{Energy}_{\text{E field}} = \frac{1}{2}\frac{1}{4\pi}\mathbf{E}\cdot\mathbf{E} \tag{2.52}$$

$$\text{Energy}_{\text{B field}} = \frac{1}{2}\frac{1}{4\pi}\mathbf{B}\cdot\mathbf{B} \tag{2.53}$$

2.3.3 Boundary conditions

The proper boundary conditions (BC) should be added for each simulation.

The magnetic mirror boundary condition is given by equations $\mathbf{n}\cdot\mathbf{E}^{n+\theta} = 0$ and $\mathbf{n}\times(\mathbf{n}\times\mathbf{E}^{n+\theta})$. Assuming a boundary of this kind in $x = 0$, the magnetic mirror BC is given by:

$$\mathbf{E}_{x_{0,j,z}} = 0 \tag{2.54}$$
$$\mathbf{E}_{y_{0,j,z}} = \mathbf{E}_{y_{1,j,z}} \tag{2.55}$$
$$\mathbf{E}_{z_{0,j,z}} = \mathbf{E}_{z_{1,j,z}} \tag{2.56}$$

Perfect conductor boundary condition is given by $\mathbf{n}\times\mathbf{E} = 0$, where $\mathbf{n}$ is the normal vector. Considering the boundary condition in $x = 0$ along the x axis, the perfect conductor is equivalent to.

$$\mathbf{E}_{x_{0,j,z}} = \mathbf{E}_{x_{1,j,z}} \tag{2.57}$$
$$\mathbf{E}_{y_{0,j,z}} = 0 \tag{2.58}$$
$$\mathbf{E}_{z_{0,j,z}} = 0 \tag{2.59}$$

Periodic boundary condition is straightforward. Considering periodic conditions in the x axis and a L_x dimension in x axis, the periodic boundary condition is defined as:

$$\mathbf{E}_{x_{-1,j,z}} = \mathbf{E}_{x_{L_x-1,j,z}} \tag{2.60}$$

2.3.4 Stability conditions

The PIC-IMM is unconditionally stable as stated by Brackbill [31], and the explicit PIC conditions do not apply. In practice, there has been shown that the following condition should be fullfilled to obtain an accurate description of the evolution in the simulation and keep the energy error bounded:

$$\frac{\Delta t \lambda_D \omega_{pe}}{\Delta x} < 1 \tag{2.61}$$

This makes the PIC-IMM method way more robust against the explicit PIC instabilities, and allows larger time-steps and grid sizes. In the case of plasma fusion devices, those parameters can be $10 - 100$ times larger in comparison with explicit PIC [22].

2.4 Monte-Carlo Collisions

In plasmas, motion of particles are driven primarily by electromagnetic field interaction and inter-particle collisions. When collisions play an important role in plasma dynamics (especially in dense plasmas), we cannot ignore the contribution they make and it is necessary to include collisions in the simulation.

One of the most common and simplest methods is the Monte-Carlo collision method. In PIC we have superparticles, so unlike the direct simulation method (DSMC) method, where individual particle collisions are simulated, in the PIC-MCC method they collide against a "cloud", and it is necessary to calculate collision probability.

The collision probability is given by [32]

$$P = 1 - exp(-\nu \Delta t) = 1 - exp(-\sigma_T(\varepsilon_i) n_t(\mathbf{x_i}) v_i \Delta t) \tag{2.62}$$

where ν is the collision frequency, σ_T is the collision cross-section, n_t is the density of the target species at the incoming particle position $\mathbf{x}_i$, v_i is the velocity of the incoming particle and Δt is the timestep in the simulation.

We could calculate the probability on each particle, but that would be computationally expensive. A more efficient alternative is to use the null-method described by Vahedi[33], in which we first calculate a maximum collision frequency given by

$$\nu_{max} = \max_{\forall \mathbf{x}}(n_t) \max_{\forall \varepsilon}(\sigma_T v) \tag{2.63}$$

Where $\mathbf{x}$ is the position vector and ε is the energy. This gives us a maximum collision probability given by

$$P_{null} = 1 - \exp \nu_{max} \Delta t \tag{2.64}$$

In this way, when we calculate the collisions in each cycle, we take $P_{null} * N$ particles, where N is the total number of particles. This subset of particles will be the ones to check for the collision, and the probability for each type will be calculated and compared to a random number. If the probability in the particular case does not fall into any type of collision, it is considered a null collision (hence the name of the method). This reduces the computational burden considerably, since the number of comparisons is made on a subset of particles instead of the total.

Since this methods allows just one collision per cycle, Δt should be carefully chosen depending on the collision probability. We can notice that the number of missing collision is:

$$R_{missed} = \sum_{n=2}^{\infty} P_i^n = \frac{P_i^2}{1 - P_i} \tag{2.65}$$

If we assume $P_i \ll 1$ (which is the case most of the time), then $R_{missed} \simeq P_i^2$, so, the error is negligible.

Since we already know how to calculate probability, what remains now is to know how to resolve the collision once it occurs. This will depend on each element, and has to be analyzed case by case. In this work, the case of Argon was implemented for 5 types of collisions and Xenon for the charge-exchange collision.

2.4.1 Argon

Argon is a gas commonly used as a background gas in many experiments [34] and PIC-MCC simulations [35, 36]. We will consider a simple model, where the collisions described in Table 2.1 will be included.

Collision type	Reaction
Electron-neutral elastic scattering[37, 38]	$e + Ar \longrightarrow e + Ar$
Electron-neutral excitation[39]	$e + Ar \longrightarrow e + Ar*$
Electron-neutral ionization[37]	$e + Ar \longrightarrow e + Ar + +e$
Ion-neutral elastic scattering[40]	$Ar^+ + Ar \longrightarrow Ar^+ + Ar$
Ion-neutral charge exchange[40]	$Ar^+ + Ar \longrightarrow Ar + Ar^+$

Table 2.1: Collision types in Argon

For each type of collision, we need to know its cross-section, solve the scattering angles and

calculate the energy gain/loss.

The calculation of these parameters will be separated into two subsections, the first considering electron-neutral collisions and the second ion-neutral.

Electron-neutral collisions

The effective section is a parameter that varies according to energy and is different for each species. This data comes from already tabulated experimental results available in databases. In our case, the Phelps database[41, 42, 43] (obtained through the website www.lxcat.net) was used. Figure 2.5 shows a plot of the data retrieved.

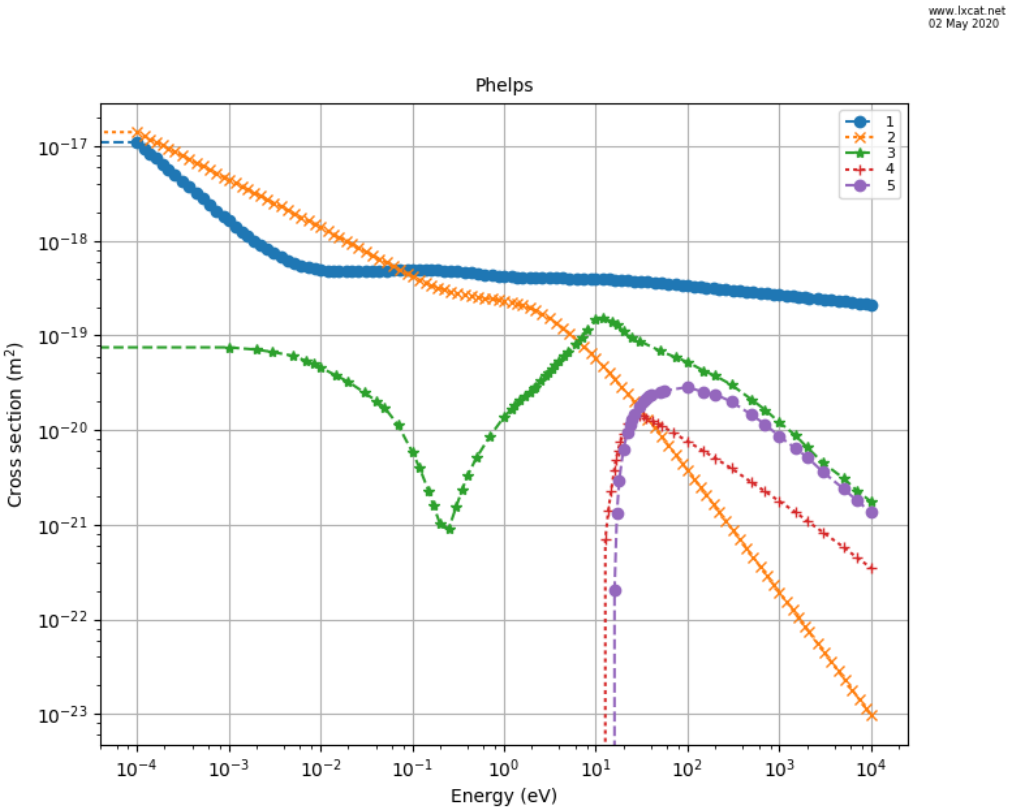


Figure 2.5: Cross sections Ar Phelps database. The curves correspond to the collisions of type (1) ion-neutral charge-exchange, (2) ion-neutral elastic scattering, (3) electron-neutral elastic scattering, (4) electron-neutral excitation and (5) electron-neutral ionization. Image obtained from www.lxcat.net, retrieved on May 02, 2020.

The scattering angle for argon can be approximated with the following equation:

$$\chi = arcos\left(\frac{2+\varepsilon-2(1+\varepsilon)^R}{\varepsilon}\right) \tag{2.66}$$

where $R \in [0,1)$ is a random number and ε is the energy of the incoming particle.

The azimuthal scattering angle can be determined by

$$\phi = 2\pi R \tag{2.67}$$

where $R \in [0, 1)$ is a random number.

The energy loss for the electron-neutral elastic collision is given by

$$\Delta\varepsilon = \frac{2m}{M}(1 - cos\chi) \tag{2.68}$$

For the electron-neutral excitation collision, we just consider the energy loss as

$$\Delta\varepsilon = 11.5eV \tag{2.69}$$

And for ionization, we can consider:

$$\varepsilon_{ejected(new)} \cong R\left(\frac{\varepsilon_{in} - \varepsilon_{threshold}}{2}\right) \tag{2.70}$$

$$\varepsilon_{scattered} = \varepsilon_{in} - \varepsilon_{threshold} - \varepsilon_{ejected} \tag{2.71}$$

$$\varepsilon_{ion} = \varepsilon_{neutral} \tag{2.72}$$

where $\varepsilon_{threshold} \approx 15.759eV$ [44] (considering ionization from ground state). Here we assumed momentum of the incident electron is way less than the ion momentum, hence, the neutral (target particle) does not change its momentum.

The ejected electron velocity is calculated the same way as the scattered electron (using different random number).

Ion-neutral collisions

The cross section is obtained from the database already indicated above.

For the charge-exchange collision, we assume the neutral becomes an ion and the ion becomes a neutral, and their initial velocities are exchanged.

For the elastic ion-neutral collision, the scattering angle is given by

$$\chi = arcos(\sqrt{1 - R}) \tag{2.73}$$

where $R \in [0, 1)$ is a random number.

The energy loss is given by [45]

$$\Delta\varepsilon = \frac{2m_1m_2}{(m_1+m_2)^2}(1-cos\Theta) \tag{2.74}$$

where Θ is the scattering angle in the center of mass frame. For the case $m_1 = m_2$, $\Theta = 2\chi$

The azimuthal scattering angle can be determined by

$$\phi = 2\pi R \tag{2.75}$$

2.4.2 Xenon

At xenon we consider the case of the charge-exchange collision as a test case to verify the correct operation of the code. For this case, we can use the analytic formulas given by Roy[46].

$$n_n(R,\theta) = a\frac{n_0}{2}\left(1-\left(1+\left[\frac{r_T}{R}\right]^2\right)^{-1/2}\right)\cos\theta \tag{2.76}$$

where $a = 1/(1-1/\sqrt{2})$, $R(r,z) = \sqrt{r^2+(z+r_T)^2}$ and $\theta = artan(r/(z+r_T))$.

The charge-exchange cross section is given by [47]

$$\sigma = (k_1 ln(v_i) + k_2)^2 \tag{2.77}$$

where $k_1 = -0.8821$ and $k_2 = 15.1262$.

As in the case of argon, in the charge-exchange collision of xenon the velocities of the ion and neutral are interchanged.

Chapter 3

Results

3.1 Verification tests

All simulation codes must be checked and compared with existing theory to verify that the simulated physics and its results are consistent with the theoretical model. There are two types of tests to which the code is submitted: verification and validation tests. Verification tests consist of checking that the code is capable of reproducing the results of the analytic models. This verification process is essential, as it is usually compared to models of the theory that include basic physics, and a good result in these gives confidence in the code. On the other hand, in validation tests the code is compared in more complex scenarios, and they are compared with other simulations or experiments already carried out.

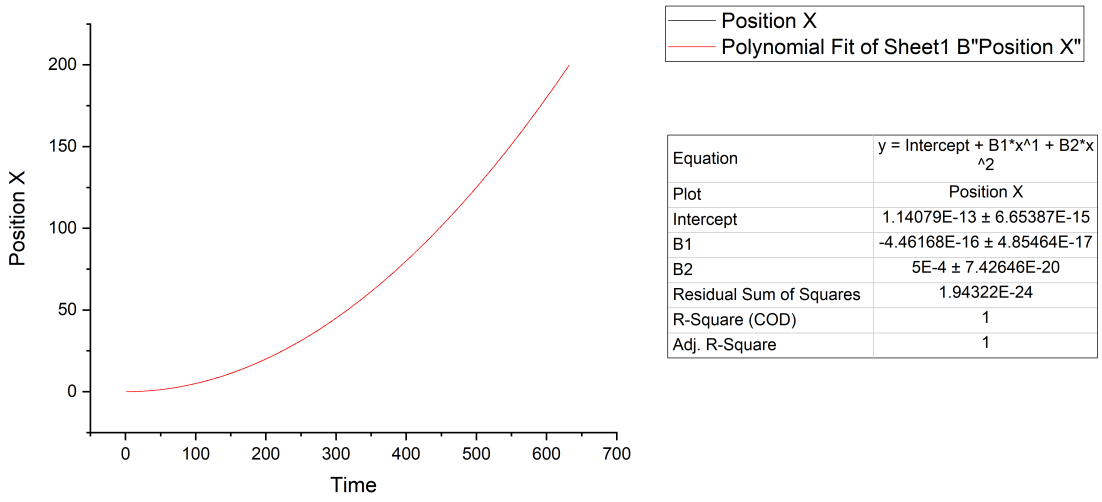
In this section, the results of some of the verification tests that were performed will be detailed.

3.1.1 Acceleration in a constant electric field

To verify that the movement of the particles is well calculated, tests are carried out in simple scenarios where we already know the result. In this case, the path of a simulated particle was followed under a constant uniform electric field E_x along the x axis.

A single charged (positive) particle was placed in the simulation with initial velocity $\mathbf{v} = 0$, and the system was allowed to freely evolve with a timestep $\Delta t = 0.1$. Since the only force acting on the particle is an electrostatic force, the position curve is expected to be equal to the uniformly accelerated motion.

Figure 3.1 shows the evolution of the particle's position over time, and a second order curve fitting was plotted. As we can see, the error between the curve and the data is of the order of the computational precision. This result is expected, since the particle mover algorithm used is stable to the second order.

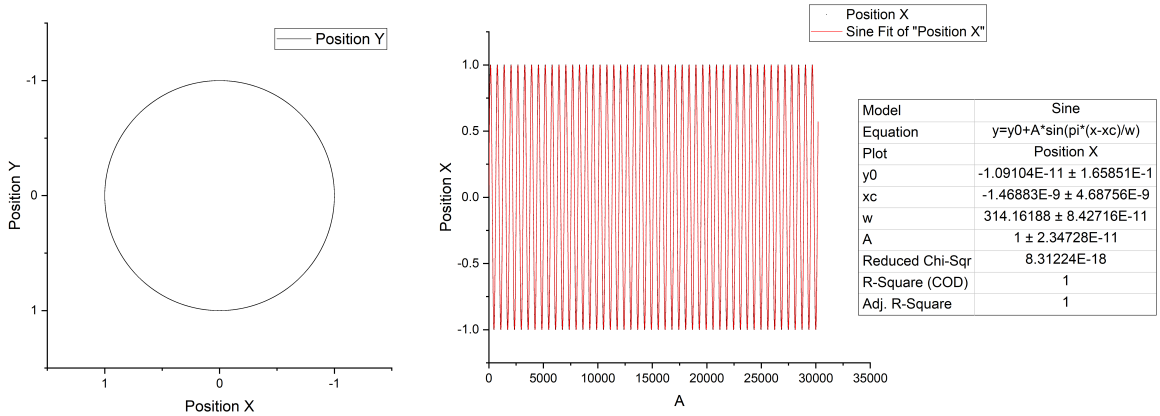


Equation	y = Intercept + B1*x^1 + B2*x^2
Plot	Position X
Intercept	1.14079E-13 ± 6.65387E-15
B1	-4.46168E-16 ± 4.85464E-17
B2	5E-4 ± 7.42646E-20
Residual Sum of Squares	1.94322E-24
R-Square (COD)	1
Adj. R-Square	1

Figure 3.1: Trajectory of a single charged (positive) particle with initial velocity $\mathbf{v} = 0$ under a constant uniform electric field. A second-order curve was fitted and it is shown in red. The fitting parameters are presented in the table.

3.1.2 Movement in a constant magnetic field

In addition to motion caused by the electric field, another essential motion that drives particles is that caused by the magnetic field. To verify this movement, the case of a uniform magnetic field was considered, since it is easier to compare it with the analytic result.



Model	Sine
Equation	y=y0+A*sin(pi*(x-xc)/w)
Plot	Position X
y0	-1.09104E-11 ± 1.65851E-1
xc	-1.46883E-9 ± 4.68756E-9
w	314.16188 ± 8.42716E-11
A	1 ± 2.34728E-11
Reduced Chi-Sqr	8.31224E-18
R-Square (COD)	1
Adj. R-Square	1

Figure 3.2: Trajectory of a single charged (positive) particle with initial velocity $\mathbf{v} = 0.1$ under a constant uniform magnetic field B_Z along the z axis.

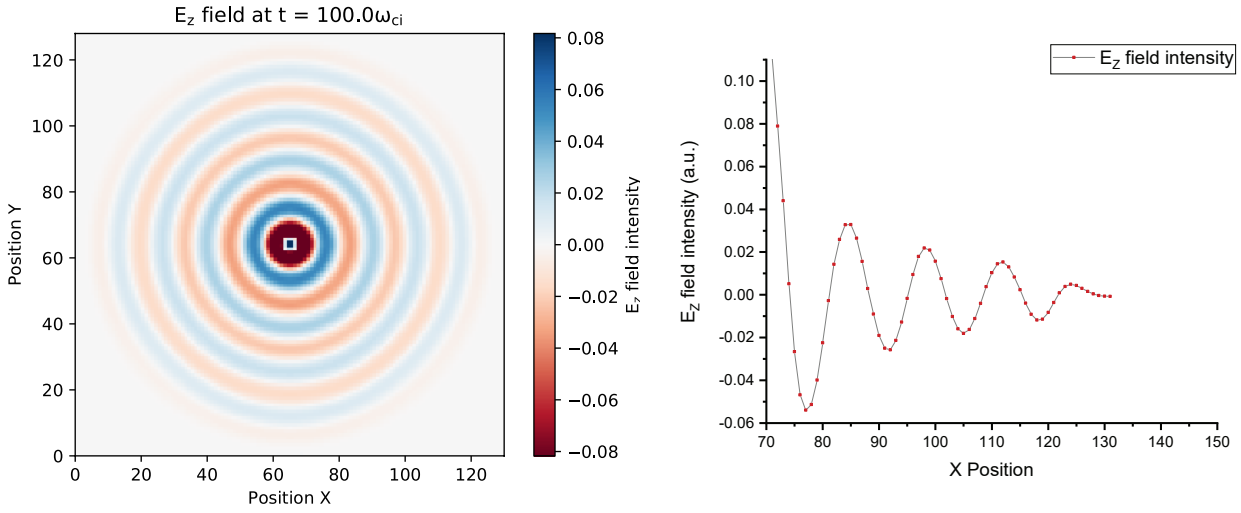
A single charged particle was positioned in the simulation with an initial velocity $v = 0.1\hat{\mathbf{x}}$ and timestep $\Delta t = 0.1$, and it is allowed to evolve freely under the effect of a constant magnetic field along the z axis. Under these simulation parameters, a circular motion is

expected.

Figure 3.2 shows a plot XY of the particle's trajectory. As we can see, the trajectory is stable, follows a circular path, without getting separated from it. This is expected since the particle mover algorithm is second-order stable.

3.1.3 Electric oscillations

To verify the dispersion of the electric field, the generation of an electric field induced by an oscillatory current J_z was tested.



(a) (b)

Figure 3.3: Electric field E_z produced by an oscillating current J_z positioned at the center of the simulation box at $t = 100\omega_{ci}$. (a) shows a 2D plot of the E_z field. (b) shows a profile of the E_z field at $y = 65$.

Fig. 3.3 shows the electric field induced by a A sinusoidal oscillating current J_z at the center of the simulation box after a time $t = 100\omega_{ci}$. As we can see, the electric field is smooth, the electric waves are circular, propagates outwards, the signs are correct, and shows all the characteristic expected from an oscillating electric field.

3.2 Two-stream instability

The two-streams instability is a well known instability caused by two counter-streaming electron beams. A background of ions is added to assure global neutrality of the plasma.

In the simulation, electrons are initialized with a velocity of $\mathbf{v} = \pm 0.2c\hat{\mathbf{x}}$ and distributed uniformly along the $\hat{\mathbf{x}}$ axis. The simulation box dimensions are $3.7l_s \times 1l_s \times 1l_s$ where $l_s = c/\omega_{pi}$ is the skin-depth, and we used a grid of $512 \times 1 \times 1$. The boundary conditions for particles are periodic, and for fields are periodic in the $\hat{\mathbf{x}}$ axis and perfect conductor for the other two axes. For this simulation, the magnetic field has been disabled, therefore this is a pure electrostatic problem.

Using Eq. 2.61, for this simulation $\mathbf{v} = 0.2c$ and $\Delta x = (3.7/512)l_s \approx 0.007l_s$, the maximum timestep we can use is $\Delta t \approx 0.035/\omega_{ci}$. In this simulation we used $\Delta t = 0.002/\omega_{ci}$, because since we are dealing with a low number of particles, the simulation time remains fast.

The simulation results shows a correct behavior of the implemented algorithm. In figure 3.4 we can see a phase-space plot ($\mathbf{x}_x$, $\mathbf{v}_x$) of the simulation at times X1, X2, X3, and X4. The evolution shows a correct formation of vortices, and an stable dynamics of the electrons. We can notice that vortices does not fade in time, due to stability of the implicit moment method.

As we can see in figure 3.6, the kinetic energy shows some dips caused by the energy exchange between particles and the electric field, which stabilize in time due to the thermalization of the particles.

In figure 3.7 we can see a plot of the electric field energy over time. Here we can show a peak which coincide with the loss of kinetic energy of particles, due to the energy exchange from kinetic energy to electric field energy. Like the kinetic energy, the energy of the electric field also stabilizes in time due to the thermalization of the system.

The total energy was plotted in Figure 3.8 (a) . As we can notice, the energy is increasing over time, denoting a numerical heating problem, even when the criteria stated by Brackbill (eq. 2.61) is fulfilled. A second simulation was run using a timestep 4 times lower ($\Delta t = 0.0005/\omega_{ci}$), which result is plotted in Figure 3.8 (b). In this new simulation, the numerical heating was reduced considerably at the cost of a higher simulation time. The noise is due the discretization and the interpolation grid-particle, and is unaffected when lowering the timestep.

The simulation results are satisfactory and coincide with that found in the literature. We can clearly observe the vortices, the energy exchange between particles and the electric field, as well as the thermalization, as observed in [2] (Figure 3.5).

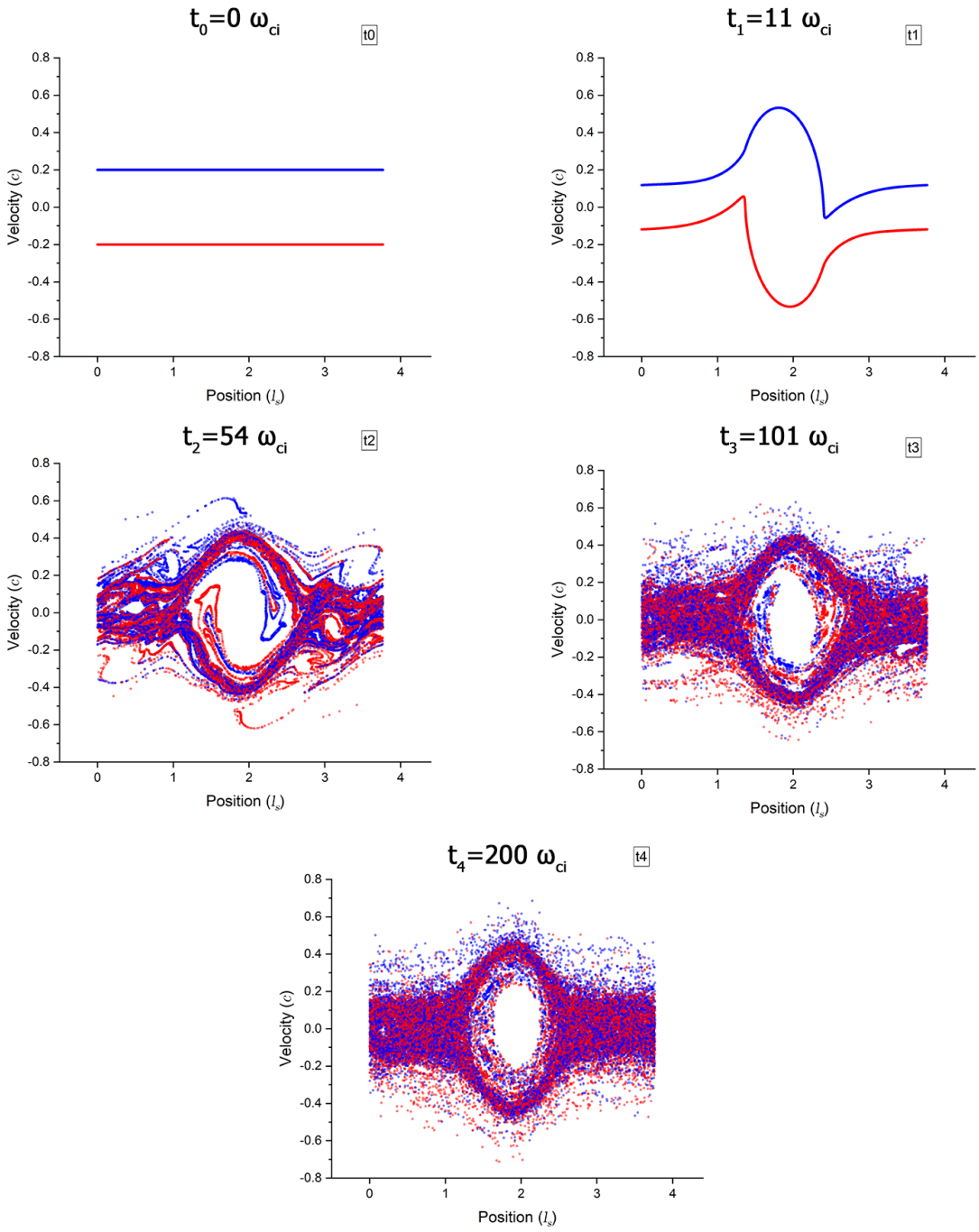


Figure 3.4: Phase space plot of the two stream instability at different times. Red and blue dots represents particles with initial positive and negative velocities respectively.

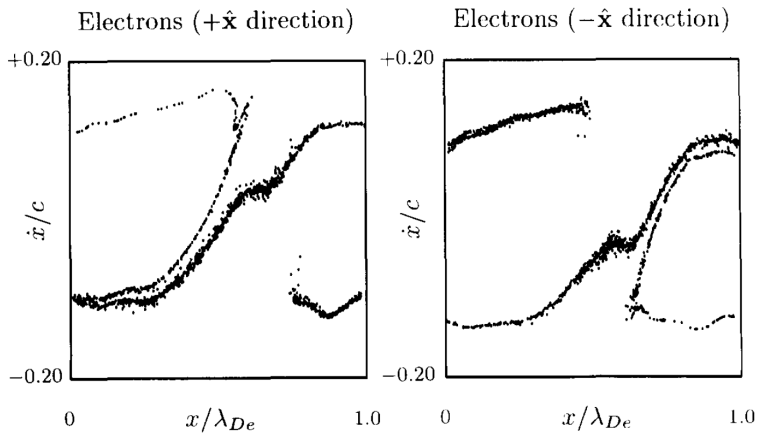


Figure 3.5: Electron distribution in phase space at time $t = 20\omega_{pe}$ obtained by Vu. Figure from his paper[2]

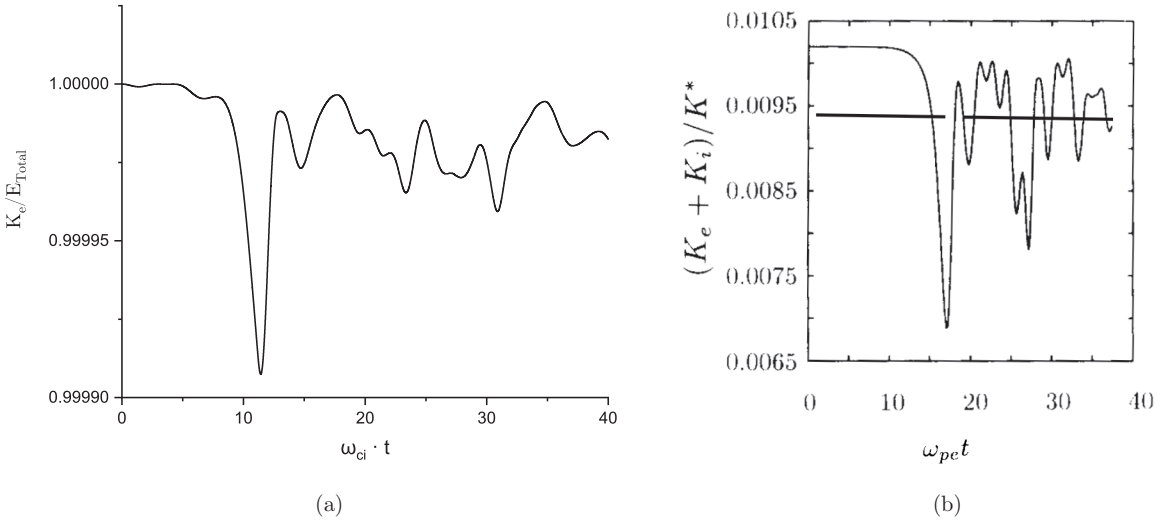
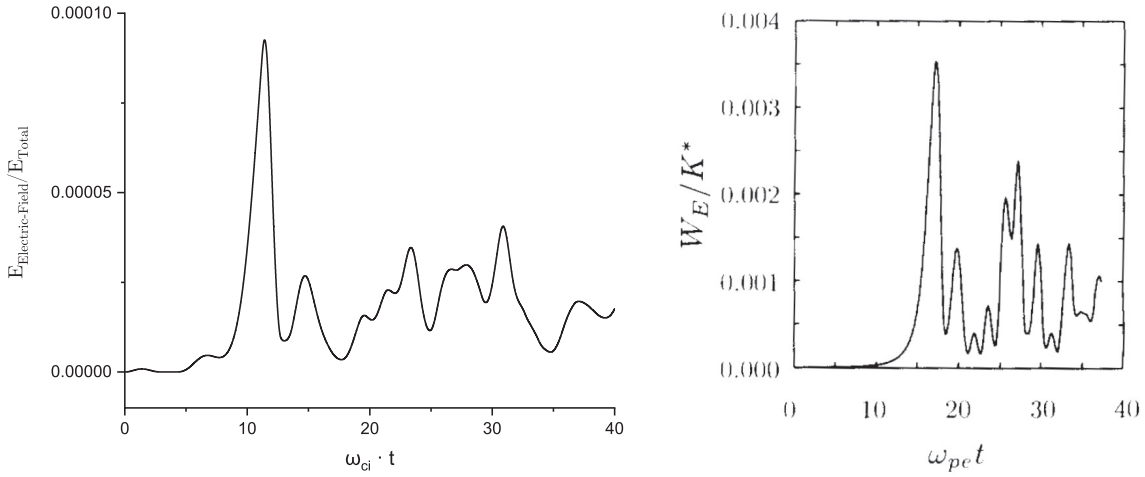


Figure 3.6: (a) Kinetic energy evolution in our simulation, (b) kinetic energy obtained by Vu. Figure from his paper[2]



(a) (b)

Figure 3.7: (a) Electric field energy evolution in our simulation, (b) Electric field energy obtained by Vu. Figure from his paper[2]

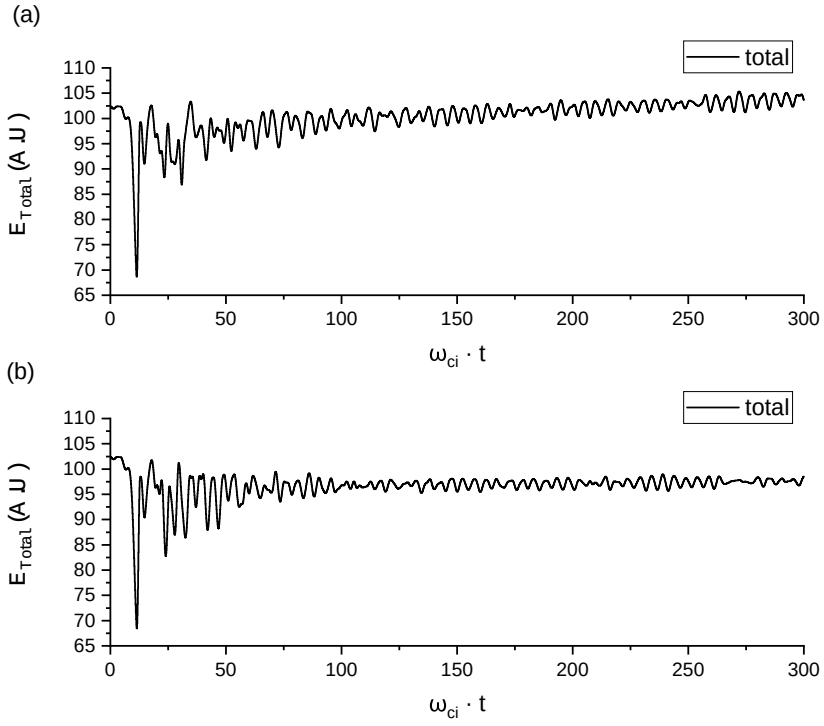


Figure 3.8: Total energy evolution over time using a timestep of $\Delta t = 0.002/\omega_{ci}$ (a) and $\Delta t = 0.0005/\omega_{ci}$ (b) respectively.

3.3 Magnetic reconnection

The magnetic reconnection is a process that occurs in plasmas, where parallel magnetic field lines in opposite direction are reconnected. During this process, magnetic energy is converted into kinetic energy.

This phenomena is present in solar flares, solar corona and Earth's magnetotail [1]. This process is also present in laboratory plasmas [48, 49].

Magnetic reconnection is a fast process once it started, and it has been shown that the dynamics of the electrons play a fundamental role in the process. [50]

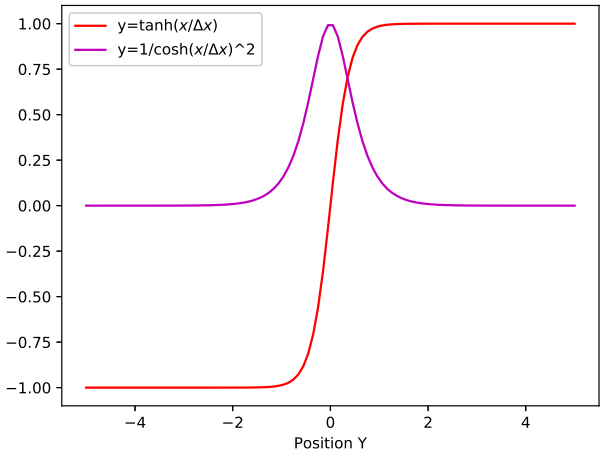


Figure 3.9: Plot of initial particle density (purple) and magnetic field intensity B_x (red).

In the simulation, two regions of anti-parallel magnetic lines are arranged in a 2.5D space, with a null magnetic field in the middle. This configuration is a well-studied case and is called *Harris sheet*, which is a stationary solution to the Vlasov-Maxwell equation. The magnetic field profile is given by:

$$\mathbf{B} = B_0 \tan\left(\frac{2(x - L_y/2)}{L_y}\right)\mathbf{x} \tag{3.1}$$

Particles where initialized with a thermal velocity of $0.05c$ with a Maxwellian distribution. The simulation was performed with 4.9 million particles, with a simulation box of $20l_s \times 10l_s \times 10l_s$ where $l_s = c/\omega_{pi}$ is the skin-depth, and we are using a grid of $120 \times 60 \times 1$ cells. Additionally, the electron density is given by the Eq. 3.2, and the mass ratio of ion-electron used in the simulation was 25.

$$n_e = 1/\cosh\left(\frac{2(x - L_y/2)}{L_y}\right)^2 \tag{3.2}$$

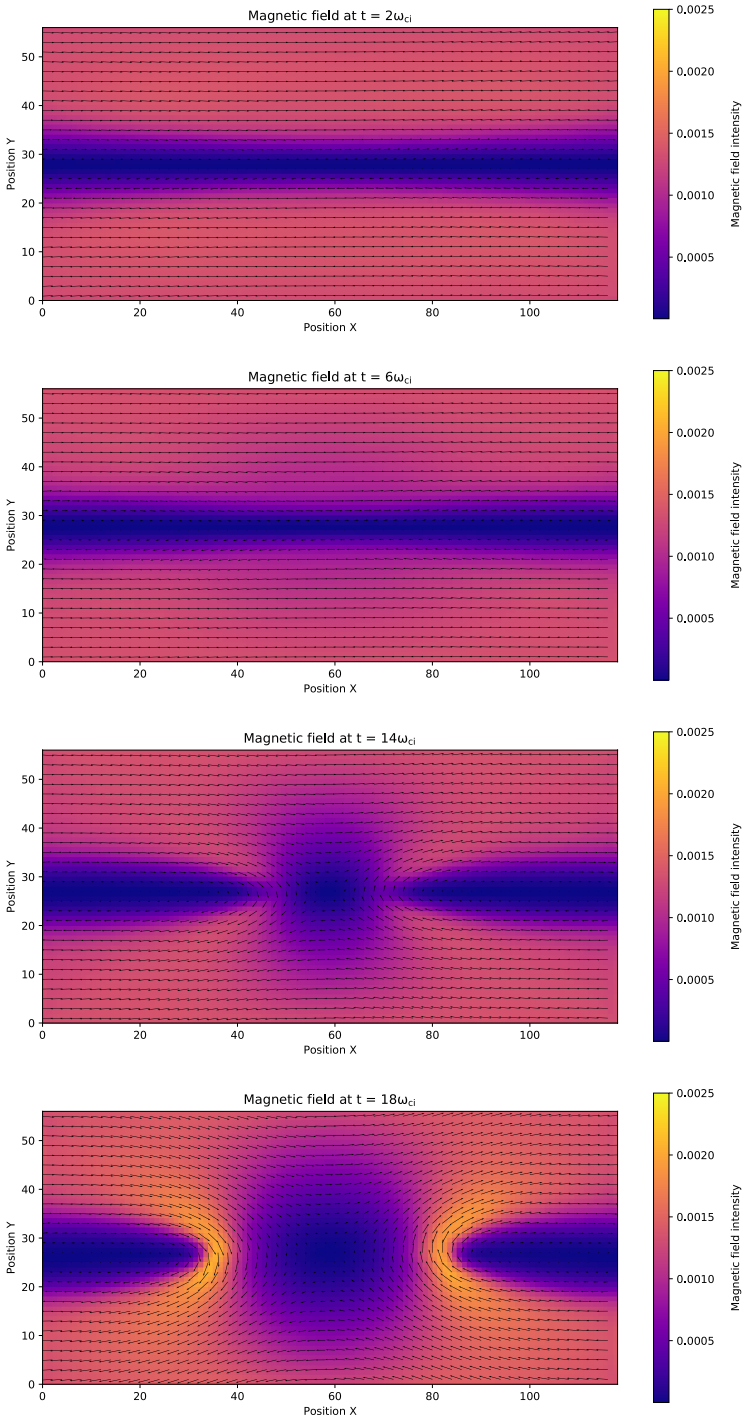


Figure 3.10: Magnetic field evolution at different times. The positions presented are in grid units. The colorscale indicates the magnetic field intensity, and the arrows indicates the magnetic field lines.

We present the results in Fig. 3.10. As we can see, the simulation shows that both regions merge in two points. In those points is where the magnetic reconnection occurs, the magnetic lines connect both regions, changing the initial magnetic topology. We can notice how the magnetic field intensity lowers in the center of the box and intensifies in the reconnection points. As a consequence of this, there is an exchange of energy between the electromagnetic field and the particles, gaining energy the latter. To verify this, the particles and field energy where plotted in figure 3.11.

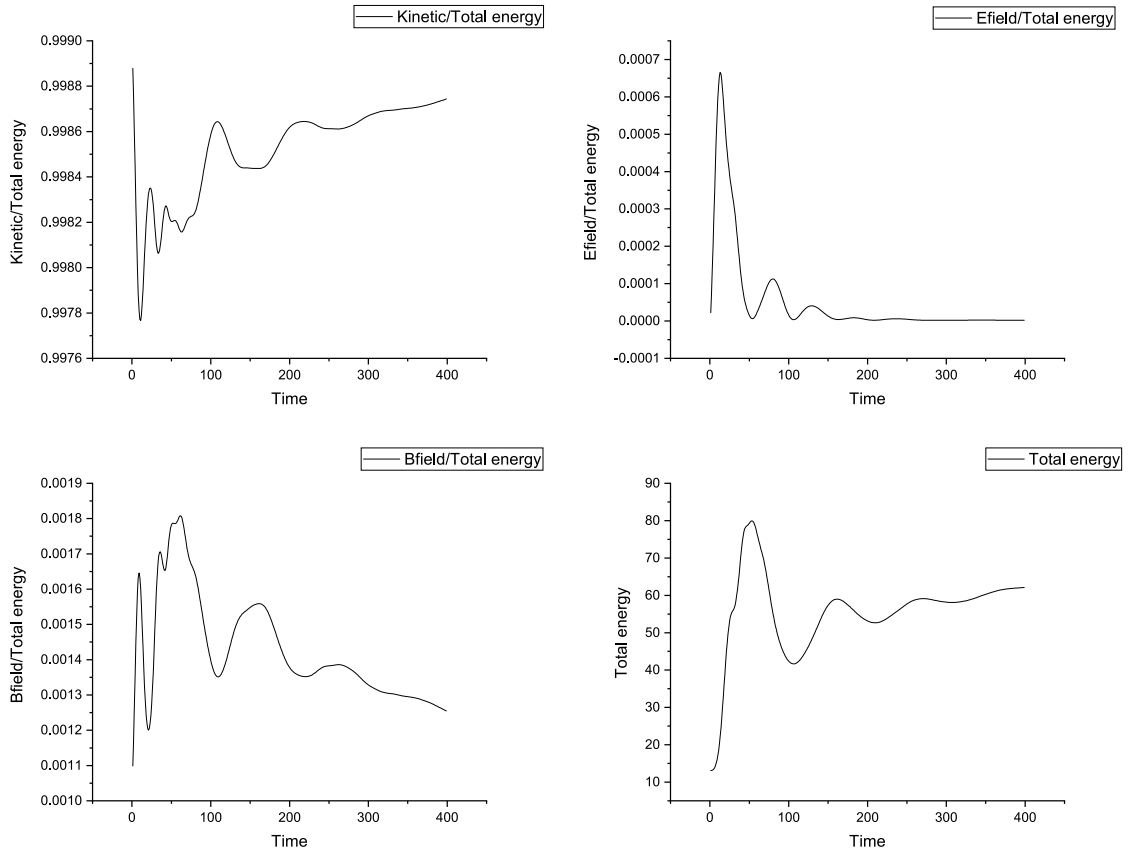


Figure 3.11: Magnetic reconnection energy plots.

In figure 3.11 we can see how the magnetic field energy lowers when the kinetic energy starts to rise. The initial peaks are due to thermalization. This can be explained because in the simulation only the magnetic field is initializated, so, when the simulation starts, there is an initial energy exchange between particle and electric field energy. This can be clearly seen in the second plot of electric field energy over time.

Also, a benchmark was run in this simulation. The time it takes to simulate each part of the cycle is shown in figure 3.12.

The figure 3.12 shows that most of the calculation time is taken by the particle mover and the update-to-grid functions. This is not a surprice, since we are using 5 million particles,

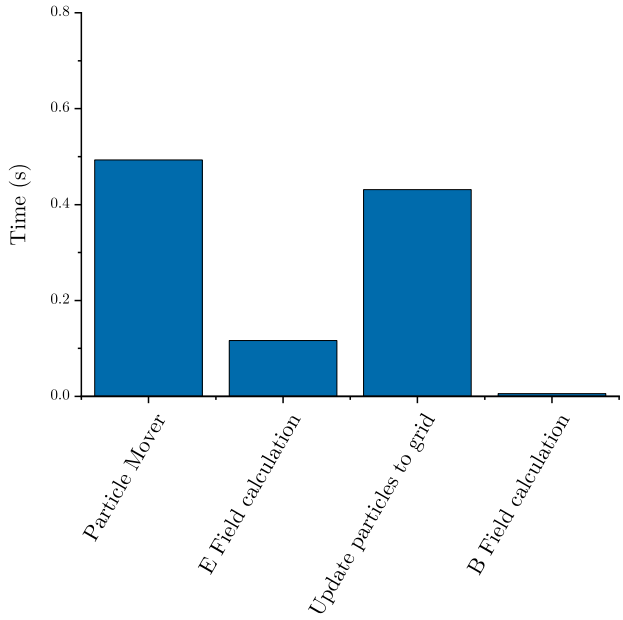


Figure 3.12: Time per cycle step using the magnetic reconnection setup: 4.9 million particles, grid $120 \times 60 \times 1$.

and the number of cells used is not high (7200 cells). In problems with a larger grid, we expect a higher calculation time for the electric field calculation.

On the other hand, the B field calculation is negligible, because it is calculated using the Eq. 2.45, and the algorithm scales as $O(n)$ where n is the number of cells.

3.4 Charge-exchange process

To test Monte-Carlo collisions, we first tested the charge-exchange collisions in ion thrusters plumes. This kind of collision has been studied since it is the cause of contamination due a backflow in spacecraft using electric propulsion thrusters. This case is ideal for verifying the code, since there is an analytical model and simulations made by Roy[46] to be able to contrast the results. Also, we can use an analytic formula to compute the electric field and thus we can focus only on the calculation and analysis of collisions.

This case considers only charge-exchange collisions. Ionization is a also a possible collision type, but, considering 1keV ions, the charge-exchange cross section is $2.5 \cdot 10^{-19} m^2$, while the ionization is an order of magnitude less [47, 51] and it is not included in the simulation.

Using the equations for xenon already mentioned in the previous chapter, a simulation of a thruster plume plasma was performed. The following parameters where considered: ion temperature $T_i = 1000K$, a ion beam velocity $v_{bi} = 30000m/s$ and a background density of $n_0 = 9 \cdot 10^{16} m^{-3}$. The simulation starts empty, without particles or an electromagnetic field. As the simulation progresses, ion particles are constantly injected simulating the thruster exit, and the particles are allowed to evolve freely.

First, it was checked that the probability calculated in the simulation corresponded to the probability given by the analytical formula. For this, the simulation was forced to generate particles with r = 0.1, and the percentage of collided particles in the simulation was compared with the probability curve using r = 0.1 for different values of z. It was tested with different amounts of particles per cell. Figure 3.13 shows a plot of the results.

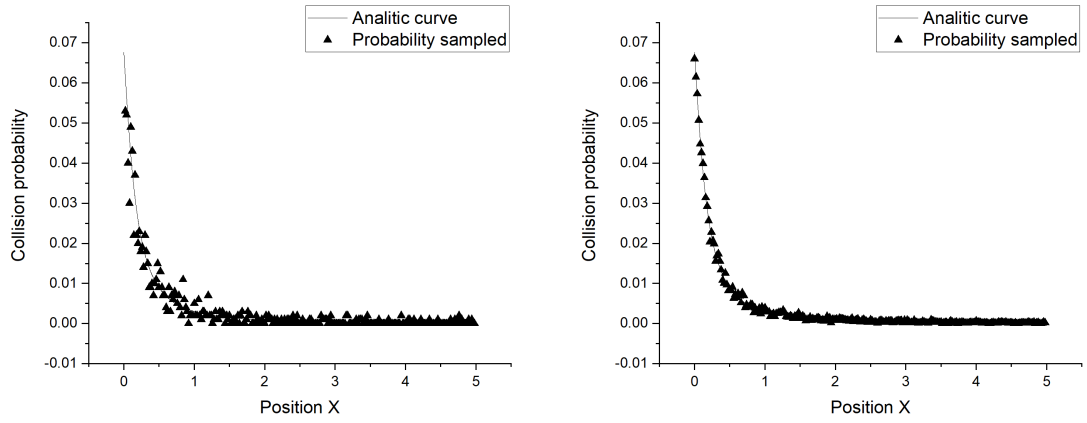


Figure 3.13: Probability sampled with 20 and 200 particles per cell (ppc) respectively.

As we can see from the curve, the number of collisions resemble the probability curve quite well, depending on the number of particles per cell (ppc). The first case is closer to a real simulation setting, since 20 particles is manageable for cases with large grids. The second case, which is more similar to the curve, but is simulated using 200 particles per cell, is a

large number of particles and is restrictive for simulations that require a grid with a larger number of cells.

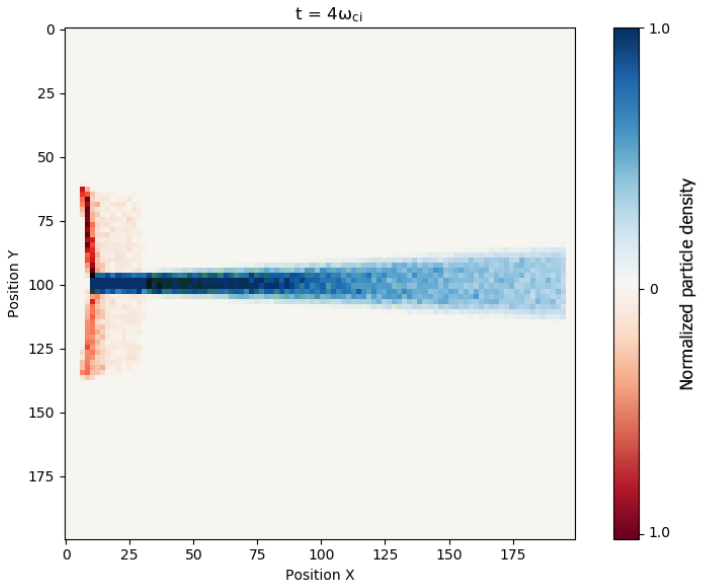


Figure 3.14: 2D plot of the normalized particle density of collided ions (red) and neutral (blue).

A 2D graph of the particle density was made, which we can see in the figure 3.14. The result shows the ion thruster plume backflows with the expected characteristics of the simulation. When a charge-exchange collision occurs, the new ion velocity is slower (compared to the beam velocity) and therefore its movement is greatly affected by the radially-directed electric field. This results are in agreement with the corresponding literature[46].

3.5 Argon collisions in a radiofrequency plasma jet

To validate the argon collision model, an existing result in the literature was tried to replicate. The work done by Charles[3] was chosen, because they use a simple 1D model of argon collisions, which reduces simulation time.

The experiment consists of a vacuum chamber where a flow of argon gas is introduced through an insulating tube (alumina) 18mm long, which is surrounded by a 5mm wide rf copper electrode.

A simulation box of $54 \times 5 \times 5 l_s$ was considered with a grid of $100 \times 1 \times 1$. The simulation time was $10^4 w_{pi}^{-1}$ using a timestep $\Delta t = 0.5 w_{pi}^{-1}$, which is approximately $0.06 w_{rf}$, where $w_{rf} = 13.65 MHz$ is the RF frequency. Neutrals were injected until it reached an equilibrium of approx. $1.5 \cdot 10^5$ macro-particles. The simulation parameters coincide in the order of magnitude with the paper. The approximate ion density of the simulation is $10^9 cm^{-3}$.

In the simulation, an important difference was noted: in the paper, the neutrals are not simulated as particles, and instead act as a fixed energy reservoir. The collision module of our program works with particles exclusively, so we chose to simulate neutrals as particles without a "reservoir", since such a change in our program would mean a greater change in the software structure.

Furthermore, our module includes electron-neutral or ion-neutral collisions, and for these collisions to exist there must be an initial population of electrons or neutrals. For this reason, along with the injection of neutrals, an electron-ion pair is also injected in a proportion of 1.5%. This injection lasts only the first 5000 cycles (or $t > 2.5 \cdot 10^3 w_{pi}^{-1}$), after that the simulation has enough particles to sustain itself.

As a note, in our simulation the electromagnetic grid is 5 times smaller than the grid in the paper, mainly because in our simulation, as we include neutral particles, the ion-electron particles are just a fraction of the total simulated particles, and hence, to have better resolution we could either use more particles ($> 10^6$) or increase the grid resolution. For time-reasons the later option was chosen.

The ion density resulting from the simulation is shown in Figure 3.15. As we can see, the result is quite similar to the result of the paper, but it is slightly moved. This can be due to the particle injection we are using in our simulation.

Figure 3.16 shows the Ion Velocity Distribution Function (IVDF). Despite the simulation differences, the result is mainly in agreement with the results from the paper. The discrepancies are mostly due to the particle injection, and the lower quantity of particles.

A further development and adding particles as a reservoir in the software is proposed to improve the resolution and computation time.

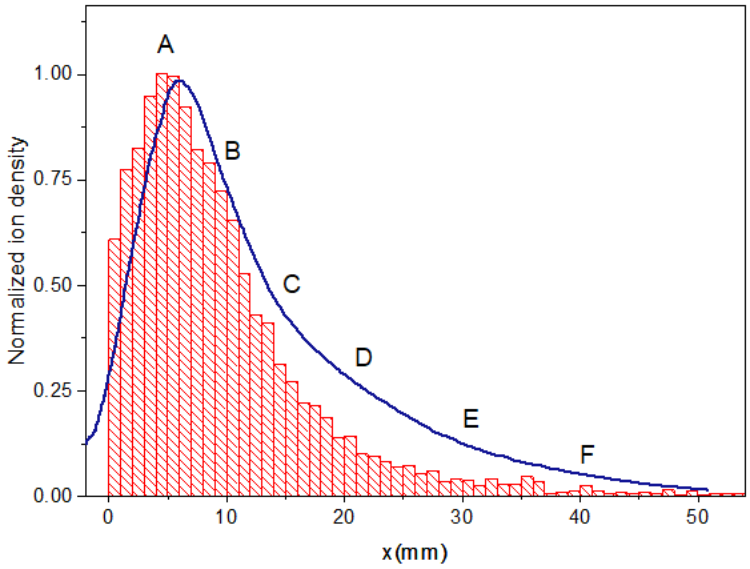


Figure 3.15: Ion density over **x** position (red) and the results obtained by Charles[3] (blue). The labels (A, B, C, D, E, F) correspond to the different sections in the **x** position described in the paper by Charles[3]

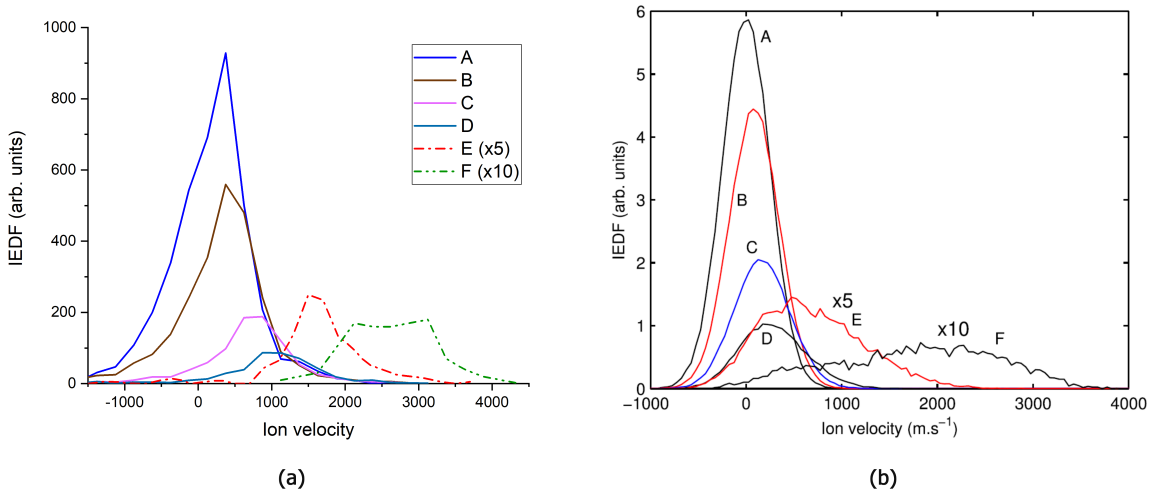


Figure 3.16: (a)Ion velocity distribution for different sections of the experiment, according to the labels described in the paper by Charles[3]. The IVDF in section E and F is multiplied by 5 and 10 respectively. (b) results obtained by Charles[3] (figure from his paper).

Chapter 4

Conclusions and future work

A PIC code using the implicit-moment method with Monte-Carlo collisions has been successfully developed and tested. The code was verified with tests where basic physics was found to be well simulated, with errors of the particles positions of the order of 10^{-18}. The tests included reviewing the energy and trajectory curve of a charged particle under the effect of a constant electric field, also under a constant magnetic field, and the dispersion of the electric field generated by an oscillatory charge density.

Furthermore, the code was tested with the two-streams instability, where one could see the typical vortices of this instability in the space-phase diagram. The simulation also presented particle-field energy transformers as expected, and the total energy of the system was not affected in the long term, demonstrating the stability of the PIC method used.

The code was also tested to analyze magnetic reconnection, an appreciable phenomenon in large amounts of plasma. The simulation showed the reconnection successfully, in addition to the transfer of energy from the magnetic field to the particles, gaining the latter kinetic energy, as observed in previous works [50, 16].

Finally, it was verified that the Monte-Carlo collisions work with probabilities close to the theoretical probability functions found in the literature, despite the discretization of the PIC method.

Despite the development already done, there are still things that could be improved in the code and in the simulated physics. The following improvements are proposed as future work

- Implementation of parallelization using MPI. Parallel computing in clusters is essential if we want to use this code on supercomputers and use its full potential. That is why it is necessary to implement parallelization in architectures with distributed memory, where the industry standard used is MPI.
- A more efficient grid. There are more efficient grid implementations, where the number of cells varies according to the density of the simulated plasma. This can be accomplished using octrees. That is, subdivisions in the cells already created as needed to maintain an adequate amount of particles per cell. Another alternative can be to

use an unstructured mesh[52, 53].

- HDF5 file output: The Hierarchical Data Format version 5 (HDF5) is an standard file format for heterogeneous data[54]. This way, output data could be easily imported into most scientific data visualization softwares, like *Paraview*[55].
- QED effects: A case of interest in the use of PIC codes is in laser-driven ion acceleration from ultra-thin foil targets, where it has been proven essential to consider quantum electrodynamics (QED) effects to obtain more realistic results[56].

Additional tests with data from experiments should also be carried out in the future, especially with dense laboratory plasmas where magnetic reconnection plays an important role. In the PUC plasma group, one of the experiments that could be simulated is the capillary Z-pinch, since this code should be able to simulate the spatial and temporal scales of the experiment and obtain a fully kinetic simulation. Another option, of course, is the RF plasma experiment, although it is recommended to add more collision models for different gases.

Appendix A

Implementation details

This appendix describes the features and implementation details of the developed software.

A.1 General overview

The software was programmed in C++ using an object oriented approach, and we can divide the code in two parts: a part that handles the user interface and another part that handles the simulation. The simulation does not depend on external libraries, unlike the other part that depends on graphic libraries. The graphic library used was OpenFrameworks, due to the simplicity of use and because at the same time it can be used at a low level, optimizing resources for real-time display.

The development was made in Windows 10, using Visual Studio 2018 community version as IDE, which is freely available. Since OpenFrameworks is available for both Linux and OSX, there is nothing limiting its use to Windows only, and there is no need to change code to compile the program on other platforms.

Each class has its own file, and the files were distributed into folders according to their category. Figure A.1 shows the structure of the folders.

A.2 Input file

As a starting point for the simulation it is necessary to write an input file. This file has a defined structure, in which all the simulation parameters are defined. The input file is actually a C header file, where a structure is defined that will be the base for the Config class, which is accessible from the rest of the classes, so the variables defined here will also be.

Among the variables to be defined, are the number of particles, dimensions of the simulation box, grid cells, the number of species as well as their parameters (charge, charge-to-mass ratio and type), the timestep, the seconds or cycles to simulate, smoothing of the electric field, tolerance of the solver and the decentering parameter θ.

Figure A.1: Folder structure and files in the project.

After defining the variables (properties of the structure), the functions (members of the structure) are defined. These methods include particle initialization, electromagnetic field initialization, particle injection during simulation, collision settings, and edge condition settings. Although most of these functions could be described by parameters, it was decided to define functions to provide greater flexibility in use and configuration, allowing complex definitions by the user.

The graphical options are defined as preprocessor directives. Although it may seem slightly hardcoded, this solution was chosen because it allows you to easily separate the part of the user interface with the simulation code. This is especially useful when one wants to run the code without the user interface.

Finally, there are a couple of extra options for special simulations that require custom conditions.

```
1 #pragma once
2 #include "utils/Utils.h"
3
4 struct input {
5
```

```
6     int pn = 1000;  // Number of macro particles
7
8     // == dimensions ==
9     double Lx = 40.0;   // physical dimension x in skin depth units
10    double Ly = 20.0;   // physical dimension y in skin depth units
11    double Lz = 20.0;   // physical dimension z in skin depth units
12
13    // == grid ==
14    int gridx = 160;        // grid size X (number of cells)
15    int gridy = 80;     // grid size Y (number of cells)
16    int gridz = 80;     // grid size Z (number of cells)
17
18    // Number of species
19    int sn = 2;
20
21    // species:
22    vector<string> species = { "electron", "ion" };
23    vector<double> q    = { -1.0, 1.0 };        // charge specie (q)
24    vector<double> qmrs = { -25, 1.0 };     // charge to mass ratio (q/m)
25
26    // == simulation time ==
27    double dt   = 0.02;     // time step
28    double s    = 1;        // seconds to simulate
29    int cycle   = 0;        // cycle of simulation
30    int tCycles = 200;      // total cycles of simulation
31
32    // constants
33    double c = 1;
34
35    // == derived quantities ==
36    double dx = Lx / gridx; // cell length X
37    double dy = Ly / gridy; // cell length Y
38    double dz = Lz / gridz; // cell length Z
39    double invdx = 1.0 / dx; // 1 / dx
40    double invdy = 1.0 / dy; // 1 / dy
41    double invdz = 1.0 / dz; // 1 / dz
42    double invc = 1.0 / c; // 1/c
43    double invdt = 1.0 / dt; // 1/dt
44
45    // params
46    double  smooth              = 0.9;  // Smooth param: (smooth) * Field +
          (1.0-smooth) * Field_neighbors
47    int     smoothRank          = 0;    // How many times smooth() should be
          applied. E.g. 2 => smooth(smooth(E)). 0 will deactivate smooth.
48    double  solverTolerance     = 1E-8; // Solver tolerance used in GMRES
49    double  divCleaningTol      = 1E-5; // Solver tolerance used in GMRES
50    double  theta               = 0.5;
51
52
```

```
53      // boundary conditions initialization
54      void set_boundary_conditions() {};
55
56      // particles initialization
57      void set_particles() {};
58
59      // fields initialization
60      void set_fields() {};
61
62      // particles inject
63      void inject_particles() {}
64
65      // collisions
66      void collisions();
67
68      // output 2d in user interface
69  #   define OUTPUT_BOTTOM \
70          drawMagnitude( -40, wY + 100, mEMField->E, 'y', -1, -1, halfGridZ, 1.0E2
                , true );\
71          drawDensity( wX + 325, wY + 150, 'x', 'y', 1.0, 0 );
72
73      // output 3d in user interface
74  #   define OUTPUT_3D \
75      drawField( mEMField->E, 'N', -1, -1, halfGridZ );
76
77  #   define OUTPUT_B             50 // output magnetic field each 50 frames
78  #   define OUTPUT_E             50 // output electric field each 50 frames
79  #   define OUTPUT_PARTICLES     -1 // output particles disabled
80  #   define OUTPUT_ENERGIES      10 // output energies each 10 frames
81
82      // others
83      bool backgroundIons = false;
84      bool fixConductor = false;
85  //# define DISABLE_B true
86
87      AUX_FUNCTIONS;
88  };
```

Code A.1: Input file structure.

In the Code A.1 we can see an example of the structure of the input file.

A.3 Output

The program developed can be configured to write the results of the simulation

A.4 Parallelization

Parallel computing is imperative nowadays if we want to get the maximum efficiency from modern computers, specially if we are dealing with large-scale simulations. There are two common architectures of parallelization: shared memory and distributed memory. The shared memory arquitecture have multiple CPUs sharing a global memory, hence, we have an uniform memory access, and we can use OpenMP to parallelize it, which has been the standard in the industry for years[57]. In the distributed memory architecture each CPU has its own memory, so, we need to pass the information from one node to the other ones, usually using MPI, which is the standard for this kind of arquitecture [58].

Currently the software is parallelized using OpenMP, for multiple CPU cores with a shared memory. The optimal would be to add MPI to obtain a hybrid parallelization, but that was not done during the development of this thesis, although it should not be complicated since the software structure easily supports the aggregation of messages between the simulation. It is left proposed as future work.

A.5 User interface

The program has a user interface that allows the simulation and some of its parameters to be viewed in real time.

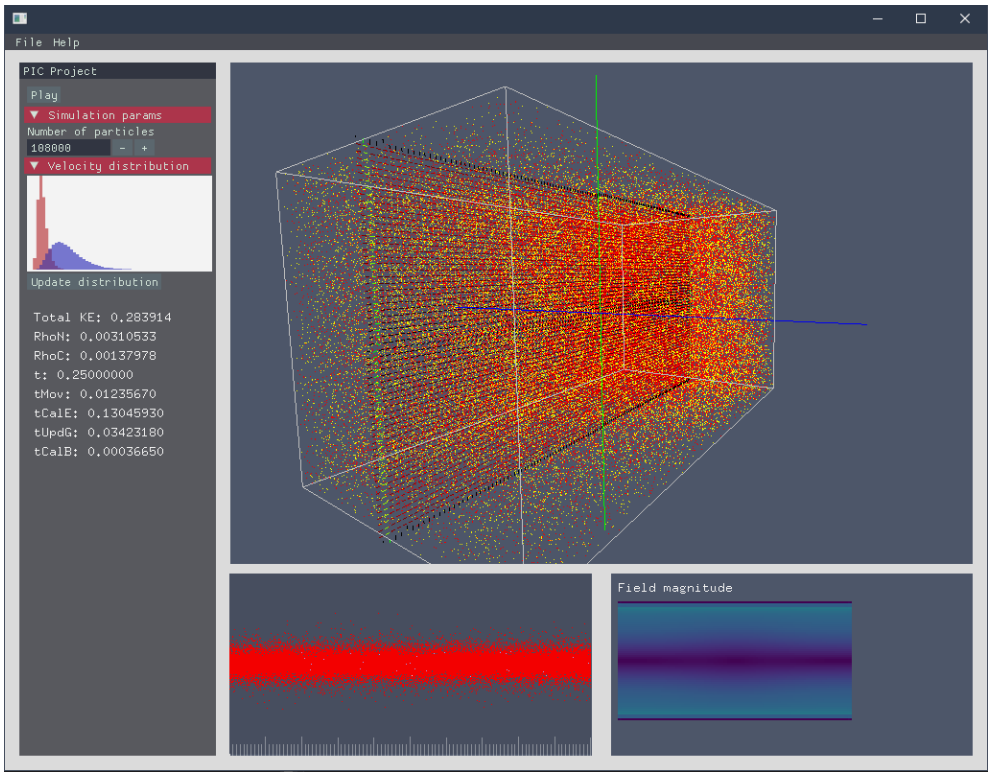


Figure A.2: Graphical user interface

In the figure A.2 we can see an image of the interface. The menu on the left allows you to view some important parameters, such as the number of particles, kinetic energy, current simulation time. A real-time 3D visualization of the particles and the simulation box is displayed on the right. Below is a phase-space diagram on the X axis (configurable).

Initially, it was planned to make a more complete user interface, where the initialization itself could be configured from there, but that limits the possibilities of custom configuration that exist through coding, so it was decided to keep an interface only for data visualization.

A.6 Units

The unit system in the developed code is quite flexible. The units are normalized, where the reference charge is proportional to e (the fundamental charge of an electron), the reference charge-to-mass ratio at q/m_i (with respect to the mass of a proton, but can be also described respect to the mass of an electron) and the reference velocity with respect to c (speed of light).

Units can be described referring to other units. In general, with respect to a frequency of interest in the simulation. In the present work usually ω_{pi} was used. Thus, we can define the quantities

Reference time: $T = 1/\omega_{pi}$

Reference length: $L = c/\omega_{pi}$

A.7 Grid system

As described in Chapter 2, the discretization of electromagnetic fields was performed using a grid. In this software, we chose to replicate the grid configuration used by Markidis [22], where the electric field ($\mathbf{E}$) and charge density (ρ) are calculated in the nodes, while the magnetic field ($\mathbf{B}$) and the current density ($\mathbf{J}$) are calculated in the center of each cell.

A.7.1 Particle-Grid interpolation

The particle-grid interpolation (or grid-particle, since it is the same function to make it stable[59]) is calculated considering weights.

Figure A.3 shows a diagram of weights calculation in 2D.

A.7.2 Ghost cells

As in other PIC codes [60], a ghost cells layers was used, which consists of an extra layer of cells surrounding the grid with the simulation particles. This ghost cells are used for mainly

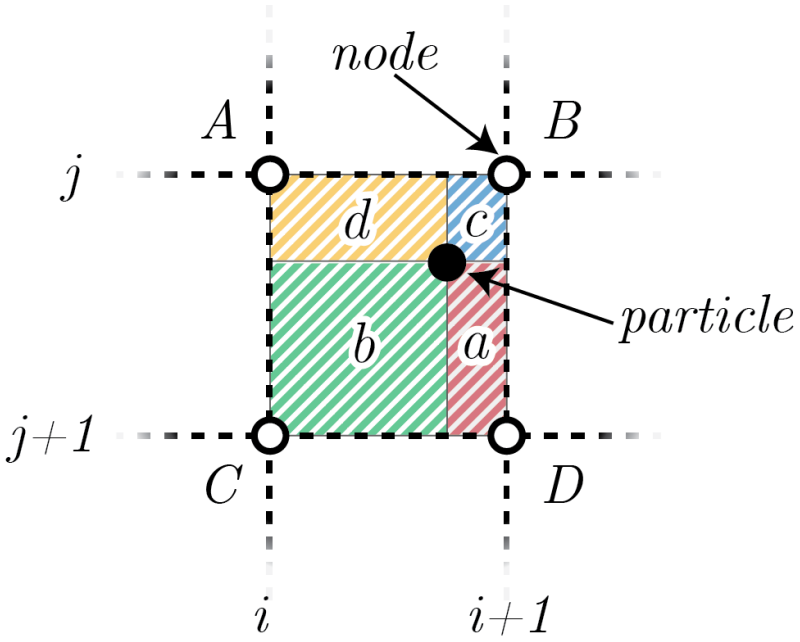


Figure A.3: Grid-particle interpolation. The contribution from each node is weighted with a value equal to the corresponding opposite area (same letter lowercase).

two reasons: as a helper to handle boundary conditions, and as an outer layer for message passing interface(MPI).

References

[1] K. Fujimoto, "Multi-Scale Kinetic Simulation of Magnetic Reconnection With Dynamically Adaptive Meshes," *Frontiers in Physics*, vol. 6, p. 119, oct 2018.

[2] H. X. Vu and J. U. Brackbill, "CELEST1D: an implicit, fully kinetic model for low-frequency, electromagnetic plasma simulation," *Computer Physics Communications*, vol. 69, pp. 253–276, mar 1992.

[3] C. Charles, R. Hawkins, and R. W. Boswell, "Particle in cell simulation of a radiofrequency plasma jet expanding in vacuum," *Applied Physics Letters*, vol. 106, p. 093502, mar 2015.

[4] A. Fridman, *Plasma chemistry*, vol. 9780521847. Cambridge University Press, jan 2008.

[5] T. J. M. Boyd and J. J. Sanderson, *The physics of plasmas*. Cambridge University Press, 2003.

[6] J. U. Brackbill and B. I. B. I. Cohen, *Multiple time scales*.

[7] C. A. Jennings, J. P. Chittenden, A. Ciardi, M. Sherlock, S. V. Lebedev, D. J. Ampleford, S. N. Bland, S. C. Bott, G. Hall, and J. Rapley, "3D resistive, radiative MHD modeling of Z-pinches," in *AIP Conference Proceedings*, vol. 808, pp. 57–60, AIP, jan 2006.

[8] S. H. Langer, I. Karlin, and M. M. Marinak, "Performance characteristics of HYDRA – a multi-physics simulation code from LLNL," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8969, pp. 173–181, 2015.

[9] P. C. Duffell, "DISCO: A 3D MOVING-MESH MAGNETOHYDRODYNAMICS CODE DESIGNED FOR THE STUDY OF ASTROPHYSICAL DISKS," *The Astrophysical Journal Supplement Series*, vol. 226, p. 2, may 2016.

[10] C. Nieter and J. R. Cary, "VORPAL: A versatile plasma simulation code," *Journal of Computational Physics*, vol. 196, pp. 448–473, may 2004.

[11] A. Necas, T. Tajima, S. Nicks, R. Magee, R. Clary, T. Roche, T. A. E. Team, A. Necas, T. Tajima, S. Nicks, R. Magee, R. Clary, T. Roche, and T. A. E. Team, "EPOCH code simulation of a non-thermal distribution driven by neutral beam injection in a high-beta plasma," *APS*, vol. 2016, p. CP10.068, 2016.

[12] K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, and T. J. Kwan, "Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation," *Physics of Plasmas*, vol. 15, p. 055703, may 2008.

[13] H. Burau, R. Widera, W. Hönig, G. Juckeland, A. Debus, T. Kluge, U. Schramm, T. E. Cowan, R. Sauerbrey, and M. Bussmann, "PIConGPU: A fully relativistic particle-in-cell code for a GPU cluster," *IEEE Transactions on Plasma Science*, vol. 38, pp. 2831–2839, oct 2010.

[14] C. Benedetti, A. Sgattoni, G. Turchetti, and P. Londrillo, "ALaDyn: A high-accuracy PIC code for the Maxwell-Vlasov equations," *IEEE Transactions on Plasma Science*, vol. 36, pp. 1790–1798, aug 2008.

[15] M. Grech, J. Derouillat, A. Beck, M. Chiaramello, A. Grassi, F. Niel, F. Perez, T. Vinci, M. Fle, N. Aunai, J. Dargent, I. Plotnikov, G. Bouchard, P. Savoini, C. Riconda, M. Grech, J. Derouillat, A. Beck, M. Chiaramello, A. Grassi, F. Niel, F. Perez, T. Vinci, M. Fle, N. Aunai, J. Dargent, I. Plotnikov, G. Bouchard, P. Savoini, and C. Riconda, "SMILEI: A collaborative, open-source, multi-purpose PIC code for the next generation of super-computers," *APS*, vol. 2016, p. GP10.006, 2016.

[16] A. Schmidt, V. Tang, and D. Welch, "Fully kinetic simulations of dense plasma focus Z-pinch devices," *Physical Review Letters*, vol. 109, p. 205003, nov 2012.

[17] G. Lapenta, J. U. Brackbill, and P. Ricci, "Kinetic approach to microscopic-macroscopic coupling in space and laboratory plasmas," in *Physics of Plasmas*, vol. 13, p. 055904, American Institute of PhysicsAIP, may 2006.

[18] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*. USA: Taylor & Francis, Inc., 1988.

[19] D. S. Harned, "Quasineutral hybrid simulation of macroscopic plasma phenomena," *Journal of Computational Physics*, vol. 47, pp. 452–462, sep 1982.

[20] H. Alfvén, "On the cosmogony of the solar system III," *Stockholms Observatoriums Annaler*, vol. 14, pp. 9.1–9.29, jan 1946.

[21] J. P. Freidberg, *Ideal MHD*. Cambridge: Cambridge University Press, 2014.

[22] S. Markidis, G. Lapenta, and Rizwan-uddin, "Multi-scale simulations of plasma with iPIC3D," *Mathematics and Computers in Simulation*, vol. 80, pp. 1509–1519, mar 2010.

[23] G. Colonna and A. D'Angola, *Plasma Modeling Methods and Applications*. IOP Publishing, 2016.

[24] P. A. Damiano, "Two-dimensional hybrid MHD-kinetic electron simulations of an Alfvén wave pulse," *Journal of Geophysical Research*, vol. 110, p. A01201, jan 2005.

[25] A. A. Vlasov, "On Vibration Properties of Electron Gas," *Journal of Experimental and Theoretical Physics*, vol. 8, no. 3, p. 291, 1938.

[26] P. Londrillo, C. Benedetti, A. Sgattoni, and G. Turchetti, "Charge preserving high order PIC schemes," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 620, pp. 28–35, aug 2010.

[27] G. Lapenta, "The algorithms of the implicit moment method for plasma simulation," feb 2008.

[28] R. Nuter, M. Grech, P. Gonzalez de Alaiza Martinez, G. Bonnaud, and E. D'Humières, "Maxwell solvers for the simulations of the laser-matter interaction," *The European Physical Journal D*, vol. 68, no. 6, p. 177, 2014.

[29] C. K. Birdsall and A. B. Langdon, *Plasma physics via computer simulation*. Taylor & Francis, 2005.

[30] R. J. Mason, "Implicit moment particle simulation of plasmas," *Journal of Computational Physics*, vol. 41, pp. 233–244, jun 1981.

[31] J. U. Brackbill and D. W. Forslund, "An implicit method for electromagnetic plasma simulation in two dimensions," *Journal of Computational Physics*, vol. 46, pp. 271–308, may 1982.

[32] C. K. Birdsall, "Particle-in-Cell Charged-Particle Simulations, Plus Monte Carlo Collisions With Neutral Atoms, PIC-MCC," *IEEE Transactions on Plasma Science*, vol. 19, no. 2, pp. 65–85, 1991.

[33] V. Vahedi and M. Surendra, "A Monte Carlo collision model for the particle-in-cell method: applications to argon and oxygen discharges," *Computer Physics Communications*, vol. 87, pp. 179–198, may 1995.

[34] M. G. Haines, "A review of the denseZ-pinch," *Plasma Physics and Controlled Fusion*, vol. 53, no. 9, p. 93001, 2011.

[35] Y. Takao, K. Matsuoka, K. Eriguchi, and K. Ono, "PIC-MCC simulations of capacitive RF discharges for plasma etching," in *AIP Conference Proceedings*, vol. 1333, pp. 1051–1056, American Institute of PhysicsAIP, may 2011.

[36] M. Frignani, *Simulation of Gas Breakdown and Plasma Dynamics in Plasma Focus Devices Esame finale anno 2007*. PhD thesis, 2007.

[37] L. R. Peterson and J. E. Allen, "Electron impact cross sections for argon," *The Journal of Chemical Physics*, vol. 56, pp. 6068–6076, jun 1972.

[38] J. C. Gibson, R. J. Gulley, J. P. Sullivan, S. J. Buckman, V. Chan, and P. D. Burrow, "Elastic electron scattering from argon at low incident energies," *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 29, no. 14, p. 3177, 1996.

[39] E. Eggarter, "Comprehensive optical and collision data for radiation action. II. Ar," *The Journal of Chemical Physics*, vol. 62, pp. 833–847, feb 1975.

[40] W. H. Cramer, "Elastic and inelastic scattering of low-velocity ions: Ne+ in A, A+ in Ne, and A+ in A," *The Journal of Chemical Physics*, vol. 30, pp. 641–642, mar 1959.

[41] Lawton and Phelps, "PHELPS database," 2013.

[42] A. V. Phelps, "The application of scattering cross sections to ion flux models in discharge sheaths," *Journal of Applied Physics*, vol. 76, pp. 747–753, jul 1994.

[43] C. Yamabe, S. J. Buckman, and A. V. Phelps, "Measurement of free-free emission from low-energy-electron collisions with Ar," *Physical Review A*, vol. 27, pp. 1345–1352, mar 1983.

[44] I. Velchev, W. Hogervorst, and W. Ubachs, "Precision VUV spectroscopy of Ar I at 105 nm," *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 32, no. 17, p. L511, 1999.

[45] E. W. McDaniel, *Atomic collisions : electron and photon projectiles*. Wiley, 1989.

[46] R. I. Samanta Roy, *Numerical simulation of ion thruster plume backflow for spacecraft contamination assessment*. PhD thesis, Massachusetts Institute of Technology, 1995.

[47] D. Rapp and W. E. Francis, "Charge exchange between gaseous ions and atoms," *The Journal of Chemical Physics*, vol. 37, pp. 2631–2645, dec 1962.

[48] W. Fox, A. Bhattacharjee, and K. Germaschewski, "Magnetic reconnection in high-energy-density laser-produced plasmas," *Physics of Plasmas*, vol. 19, p. 056309, may 2012.

[49] L. Yi, B. Shen, A. Pukhov, and T. Fülöp, "Relativistic magnetic reconnection driven by a laser interacting with a micro-scale plasma slab," *Nature Communications*, vol. 9, pp. 1–7, dec 2018.

[50] Y. Kuramitsu, T. Moritaka, Y. Sakawa, T. Morita, T. Sano, M. Koenig, C. D. Gregory, N. Woolsey, K. Tomita, H. Takabe, Y. L. Liu, S. H. Chen, S. Matsukiyo, and M. Hoshino, "Magnetic reconnection driven by electron dynamics," *Nature Communications*, vol. 9, pp. 1–6, dec 2018.

[51] H. S. W. Massey, E. H. S. Burhop, and H. B. Gilbody, *Electronic and Ionic Impact Phenomena: Collision of electrons with atoms, by H. S. W. Massey and E. H. S. Burhop*. Electronic and Ionic Impact Phenomena V.1: Collision of Electrons with Atoms, Clarendon P., 1969.

[52] G. B. Jacobs and J. S. Hesthaven, "High-order nodal discontinuous Galerkin particle-in-cell method on unstructured grids," *Journal of Computational Physics*, vol. 214, pp. 96–121, may 2006.

[53] A. Hilgers, S. Clucas, B. Thiébault, J. F. Roussel, J. C. Matéo-Vélez, J. Forest, and D. Rodgers, "Modeling of plasma probe interactions with a PIC code using an unstructured mesh," *IEEE Transactions on Plasma Science*, vol. 36, no. 5 PART 2, pp. 2319–2323, 2008.

[54] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An overview of the HDF5 technology suite and its applications," in *ACM International Conference Proceeding Series*, (New York, New York, USA), pp. 36–47, ACM Press, 2011.

[55] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application*. Clifton Park, NY, USA: Kitware, Inc., 2015.

[56] T. D. Arber, K. Bennett, C. S. Brady, A. Lawrence-Douglas, M. G. Ramsay, N. J. Sircombe, P. Gillies, R. G. Evans, H. Schmitz, A. R. Bell, and C. P. Ridgers, "Contemporary particle-in-cell approach to laser-plasma modelling," *Plasma Physics and Controlled Fusion*, vol. 57, no. 11, p. 113001, 2015.

[57] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[58] D. Walker and J. Dongarra, "MPI: A Standard Message Passing Interface," *Supercomputer*, vol. 12, pp. 56—-68, dec 1995.

[59] A. Pukhov, "Particle-in-cell codes for plasma-based particle acceleration," in *CAS-CERN Accelerator School: Plasma Wake Acceleration 2014, Proceedings*, vol. 1, pp. 181–206, CERN, feb 2014.

[60] J. Derouillat, A. Beck, F. Pérez, T. Vinci, M. Chiaramello, A. Grassi, M. Flé, G. Bouchard, I. Plotnikov, N. Aunai, J. Dargent, C. Riconda, and M. Grech, "SMILEI: A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation," *Computer Physics Communications*, vol. 222, pp. 351–373, jan 2018.