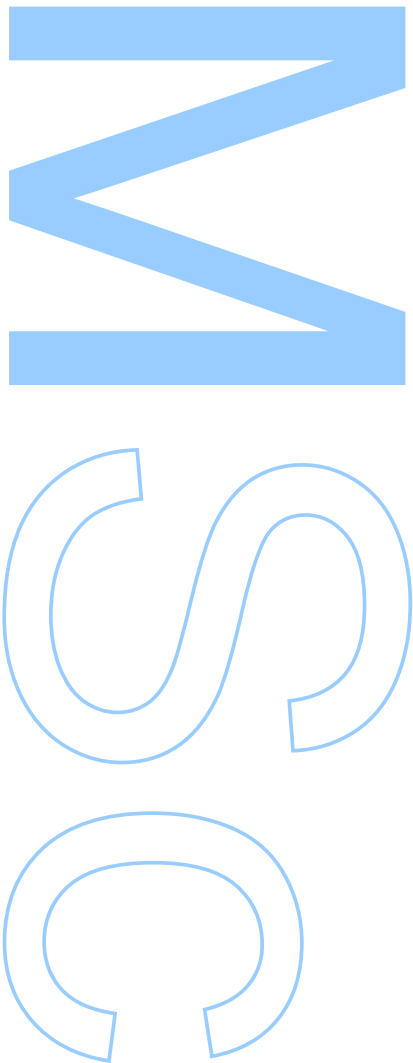
Development of a simulator of light-matter interaction using GPGPU: from plasmas to atomic gases

Miguel Boaventura Teixeira Gomes
Mestrado Integrado em Engenharia Física
Departamento de Física e Astronomia
2017

Orientador
Ariel Guerreiro, Faculdade de Ciências



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

UNIVERSIDADE DO PORTO

MASTERS THESIS

Development of a simulator of light-matter interaction using GPGPU: from plasmas to atomic gases

Author:
Miguel Gomes

Supervisor:
Prof. Dr. Ariel Guerreiro

A thesis submitted in fulfilment of the requirements
for the degree of Integrated Master's in Engineering Physics

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Física e Astronomia

October 2017

Acknowledgements

I would like to thank my supervisor, Prof. Dr. Ariel Guerreiro, for guiding me through the development of this thesis, and always being available to help and motivate me during this endeavour.

I would also like to thank Nuno Azevedo Silva for his support and advice, which were vital to the materialization of this thesis.

Finally, I would like to thank my friends and colleagues João Costa and Rúben Alves, who worked on their thesis at the same time and in the same team, and whose help and friendship was crucial during this thesis.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Física e Astronomia

Integrated Master's in Engineering Physics

Development of a simulator of light-matter interaction using GPGPU: from plasmas to atomic gases

by Miguel Gomes

This thesis describes the development of a solver for plasmas and atomic gases, with the PIC method from Plasma Physics. The software was developed in C++ using GPGPU computing. The code is also constructed with a modular structure that allows the combination with other solvers to describe a wide range of phenomena and physical systems. In this work, we also analyse the analytical and numerical methods used to describe the electromagnetic field, and fluids of charged and neutral particles. We present the performance analysis of our code by comparison with the CPU performance. We also validate our code with a series of tests, including three test cases that include a direct application of a PIC code, a simulation of a gas under gravitational interaction, and a dipolar gas displaying quantum-optomechanical effects. These last two simulations serve to demonstrate the modularity of the code.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto
Departamento de Física e Astronomia

Mestrado Integrado em Engenharia Física

Development of a simulator of light-matter interaction using GPGPU: from plasmas to atomic gases

por Miguel Gomes

Esta tese descreve o desenvolvimento de um *solver* para plasmas e gases atómicos, com o método PIC de Física de Plasmas. O software foi desenvolvido em C++ usando computação em GPGPU. O código também é construído com uma estrutura modular que permite a combinação com outros *solvers* para descrever uma grande variedade de fenómenos e sistemas físicos. Neste trabalho, também analisamos os métodos analíticos e numéricos utilizados para descrever o campo eletromagnético, e fluidos de partículas carregadas e neutras. Apresentamos a análise de desempenho do nosso código em comparação com o desempenho em CPU. Validamos ainda o nosso código com um conjunto de testes, incluindo três casos de teste que incluem uma aplicação direta de um código PIC, uma simulação de um gás sob interação gravitacional, e um gás dipolar exibindo efeitos optomecânicos quânticos. Estas duas últimas simulações servem para demonstrar a modularidade do código.

Contents

List of Figures

List of Tables

Acronyms

ABC Absorbing Boundary Conditions. 22, 24, 35, 36, 47, 71

ALU Arithmetic Logic Unit. 40

API Application Programming Interface. 40, 49, 53

APIC Affine Particle-In-Cell. 26

CBE Collisionless Boltzmann Equation. 10, 28, 29, 75

CFD Computational Fluid Dynamics. 25

CPU Central Processing Unit. 5, 37, 39, 40, 42, 49, 53

DCT Discrete Chebyshev Transform. 16

DFT Discrete Fourier Transform. 16

DNS Direct Numerical Simulation. 25

DSMC Direct Simulation Monte Carlo. 5, 25–27, 72

FDTD Finite-Difference Time-Domain. 15–17, 20, 22, 25–27, 31, 35, 36, 45–47, 50, 53, 71, 73, 74

FEM Finite-Element Method. 16, 17, 25

FFT Fast Fourier Transform. 16, 21, 22, 40, 74

FLIP Fluid Implicit Particle. 26

GPGPU General Purpose Graphics Processing Unit. 5, 40

GPU Graphics Processing Unit. 5, 6, 15, 35, 37, 39, 40, 42, 49, 53, 72

MABC Multiplicative Absorbing Boundary Conditions. 22, 45, 46, 71

MAC Marker-And-Cell. 25, 26

MD Molecular Dynamics. 4, 26

MHD Magnetohydrodynamic. 5, 7, 12, 25, 26

NSE Navier-Stokes Equation. 4

ODE Ordinary Differential Equation. 31

PBC Periodic Boundary Conditions. 22, 45

PIC Particle-In-Cell. 5, 6, 15, 25, 27, 28, 30, 31, 33, 35, 36, 47, 48, 50, 61, 62, 71, 72, 75

PSTD Pseudospectral Time-Domain. 16, 73, 74

RBC Reflecting Boundary Conditions. 22

UPML Uniaxial Perfectly Matched Layer. xi, 22, 23, 37, 45, 46

Dedicated to my family for their constant support during this five years of study.

Chapter 1

Introduction

Many-body systems occur in many branches of physics, from the motion of celestial bodies to particle physics. They establish a connection between the fundamental principles that determine the interaction between two objects, typically at a microscopic scale, and the apparent complexity of macroscopic systems composed of countless and mutually interacting objects.

Strictly speaking, one usually distinguishes between many-body and n-body systems, where the first term refers to systems with a large number of interacting particles described according to a quantum model, whereas the second is usually used to describe classical systems, typically associated with the description of the motion of celestial bodies under gravitational interaction.

However, the nature of these types of physical systems is so diverse that it is possible to find examples that can be modelled by a hybrid description, which combines both quantum and classical features. Examples of these types of systems are plasmas and atomic gases. While the position and velocity of the atoms can be described as if they were classical point particles, the ionization and recombination processes, and the electronic transitions must necessarily be described using a quantum model.

This thesis addresses these hybrid (or semi-classical) systems and focuses on the interaction between the electromagnetic field and a many-body system composed of atoms, ions, and electrons. Of course, this is still an extremely loose definition, and many different systems fit this description. The most obvious is a plasma, which is the main ingredient of the Universe and accounts for 99% of its mass and volume[1]. Naturally, understanding plasmas becomes crucial in Astrophysics to study stars, the solar wind, and the accretion disks around black holes, just to give a few examples[2].

Today, research in plasmas ranges over a wide variety of topics and remarkable phenomena in fundamental Physics, such as *photon bubbles*[3], *photon charge*[4], and generation or amplification of laser beams with high orbital angular momentum[5]. They also play a pivotal role in various technologies that encompass thin film deposition[6], particle acceleration[7], and fusion power research, including both magnetic and laser-based inertial confinement[8–10].

Neutral fluids are equally rich in phenomenology. Even common phenomena such as turbulence are still poorly understood[11]. But of special interest is the interaction of intense coherent electromagnetic field with atomic gases, which can give rise to exotic phenomena like electromagnetically induced transparency[12], slow-light[13] and other nonlinear effects[14,15]. Additionally, it is also possible for the field to produce optomechanical phenomena such as pattern formation in the gas density, and optical trapping.[16,17].

Recently, there has been a growing interest in the study of quantum many-body systems out of equilibrium, including the research into the processes of thermalization, transport phenomena, and quantum phase transitions. Another important field of research includes quantum simulations, which refers to the use of controllable physical systems to mimic the natural interaction of other less tractable systems, thus providing an analogue quantum simulator[18]. Although all these problems are of great importance, they are also very difficult to study using a complete quantum description. Therefore, an approach that is half way between the classical and the quantum approaches may provide a sufficiently simple yet interesting solution.

Finding exact analytical solutions for many-body problems in both equilibrium and out of equilibrium situations is extremely difficult and in the majority of cases even impossible. This has thwarted a full analytical approach and favoured the introduction of simplifications, that are often specific to the context of each problem, or the adoption of numerical methods, given the calculation power of modern computer systems.

An analytical or numerical model can usually be built from first principles, starting with the most fundamental equations or laws and introducing simplifications until the problem is tractable. Conversely, it is possible to start with a very simple and broad description and add features to introduce more detailed effects.

Using a numerical approach allows us to remove simplifications or add more features beyond what is possible analytically. This extra advance comes at a cost. Numerical solutions tend to be case specific and give fewer physical insights into broad problems.

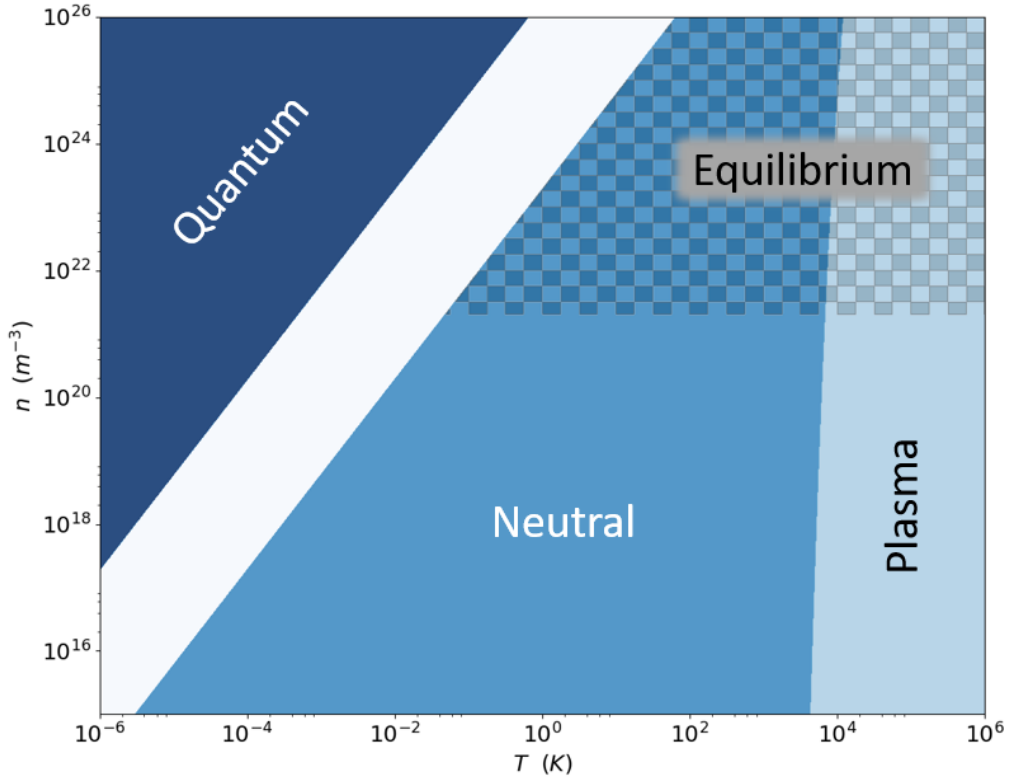


FIGURE 1.1: The various regimes of the many-body problem for atoms, ions, and electrons. The plot is intended to be qualitative but the actual borders were calculated for a hydrogen gas with the thermal de Broglie wavelength, Saha ionization equation, and Knudsen number for a length scale of 1 mm.

However, when introducing simplifications in an analytical model, it may not be clear in which conditions the model remains valid. Verifying the validity of analytical models through simulation is one of the many uses of numerical analysis.

The collective behaviour of many-body systems, such as plasma and gases, can be very diverse and exhibit different regimes, as shown in figure 1.1, typically depending on temperature and particle concentration.

For neutral fluids composed of atoms at normal conditions of temperature and pressure, the electromagnetic force is the main driver behind the dynamics of the atoms. This interaction force between atoms only becomes significant at close distances, when two particles collide. In most cases, these collisions force local groups of atoms to acquit velocities that follow closely the Maxwell-Boltzmann distribution. In these cases, the gas can be modelled by the hydrodynamic equations, corresponding to the intersection between the *Equilibrium* and *Neutral* regimes in figure 1.1. One aspect not included in figure 1.1 is the

timescale. In principle, the hydrodynamic equations could be used to study something as diluted as the interstellar medium, provided we look at a volume sufficiently large over a period long enough such that a large number of binary collisions can occur[2].

So far we have only considered binary collisions, which are characteristic of ideal gases, however, the validity of the hydrodynamic equations extends beyond this type of gases. In fact, since they can be derived exclusively from macroscopic conservation laws, the actual limitation is that the local distributions of atoms remain in thermodynamic equilibrium as described by an equation of state (of which the law of perfect gases is the simplest example).

In situations where the conditions of local equilibrium are no longer maintained, it is necessary to adopt a description based on the Kinetic Theory, and use the Boltzmann equation. This equation considers the local distributions of velocities explicitly. In fact, the hydrodynamic equations can be derived from the Boltzmann equation by taking the first three moments on momentum space and obtaining the equations of conservation of mass, conservation of momentum (also known as Navier-Stokes Equation (NSE)), and conservation of energy, respectively. This process of taking higher order moments could be continued indefinitely without ever closing the problem. Closing the problem at the second order moment requires the *ad-oc* inclusion of a state equation.

Non-equilibrium situations can arise from external forces that act on spatial scales comparable to or smaller than the mean free path of the particles. As shown in figure 1.1, the *Neutral* regime extends beyond the *Equilibrium* to lower temperatures and concentrations, where the mean free path is larger. However, the Boltzmann equation is statistical in nature, and still requires a number of particles large enough for this description to be statistically significant. If we decrease the concentration of particles beyond this limit, describing each particle individually with Molecular Dynamics (MD) may be the only option.

If we increase the temperature, the forces involved in the collisions of the atoms will be large enough for ionization to occur, corresponding to the *plasma* section of figure 1.1. A model based on binary collisions is inadequate to describe this state of matter due to the long-range nature of the interaction between charged particles. As noted by Vlasov, phenomena such as the natural vibrations of electron plasmas, and the anomalous electron scattering in plasmas, can only be well described after removing the binary collision term

from the Boltzmann equation, introducing instead the Lorentz force as the main driving mechanism for charged particles[19].

Unlike neutral fluids, plasmas are harder to describe in local equilibrium due to the long-range interactions. For most plasmas of interest, it would be necessary to look at exceedingly large volumes to find a population corresponding to a thermodynamic equilibrium distribution. Nevertheless, when such conditions are met, the equilibrium description of a plasma is described by the Magnetohydrodynamic (MHD) equations. Even in some situations where these descriptions are not completely valid, the MHD equations still provide good qualitative results and interesting physical insight.

For extremely low temperatures, the thermal de Broglie wavelength of the particles will become on the order of or larger than the average distance between particles and the kinetic description is no longer valid. Instead, the Fermi-Dirac or the Bose-Einstein statistics apply, producing exotic states of matter such as Bose-Einstein condensates and superfluid Helium. This corresponds to the quantum regime in figure 1.1, where all aspects of the gas have a quantum character.

As previously discussed, this thesis is focused on the development of numerical tools that can simulate many-body systems composed of atoms, ions, and electrons interacting with the electromagnetic field, based on a classical model for the motion of these particles but capable of incorporating quantum features. In particular and as a starting point, it is focused on modelling plasmas through the Particle-In-Cell (PIC) method. Although this method was devised for plasmas it has enough similarities with other numerical methods, like Direct Simulation Monte Carlo (DSMC), to be able to simulate more generic systems.

Another important aspect in developing a simulation code is the choice of hardware. Because PIC codes tend to be extremely demanding in terms of computational power, we chose to run the simulations on General Purpose Graphics Processing Units (GPGPUs) (or just Graphics Processing Units (GPUs)), which can provide computing power at a reduced cost when compared to regular CPUs.

Another objective of this thesis is that the simulation tools must have a modular structure, constituted by individuals solvers for different physical phenomena that can operate together. This will allow these modules to be combined in different ways to provide models for a wide diversity of systems, such as plasmas, atomic gases, and others.

In this thesis we describe the development of a C++ library for the simulation in GPU of the many-body problem for atoms, ions, and electrons in the classical limit based on the

PIC method. We develop it in a modular fashion so that it is possible to model other systems, such as atomic gases interacting with resonant electromagnetic radiation or clouds of particles under gravitational interaction. In chapter 2 there is a more detailed description of the analytical models to describe these systems, chapter 3 discusses the various numerical models to simulate them, in chapter 4 the details of the GPU based implementation are described and finally, several test cases of the code and final conclusion are presented in chapters 5 and 6 respectively.

Chapter 2

Physical model

Plasmas and atomic gases are composed of numerous and extremely small mutually interacting particles. The interaction between charged ions and electrons, and neutral atoms is essentially mediated by the electromagnetic field. For both charged and neutral fluids, we can separate the system into two parts: the electromagnetic field and the particles. For both charged and neutral fluids, the microscopic descriptions turn out to be quite similar. The main distinction is that, although the interaction between neutral atoms is mediated by the electromagnetic field, it is usually treated as an instantaneous collision. Plasmas, on the other hand, are composed of charged particles whose long-range interaction cannot be ignored. In some cases, cold gases of neutral atoms also have long-range interactions through higher order electric or magnetic moments.

To begin, we restrict ourselves to a classical description for both the electromagnetic field and the particles. In the next sections, we look at some of the models to describe plasmas and neutral gases, both in and out of equilibrium, and choose a model to best reproduce the microscopic and macroscopic behaviour of these systems. First, we concisely state the Maxwell equations and discuss their validity in section 2.1. For the fluids, we start with the kinetic description, showing the Boltzmann and Vlasov equations in section 2.2. Next, we look at the Hydrodynamic equations as a simplification of the Boltzmann equation, and similarly, at the MHD equations as a simplification of the Vlasov equation in section 2.3.

2.1 Electromagnetic field

The classical description of electrodynamics is provided by the microscopic Maxwell equations

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0} \tag{2.1a}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.1b}$$

$$\dot{\mathbf{B}} = -\nabla \times \mathbf{E} \tag{2.1c}$$

$$\dot{\mathbf{E}} = c^2 \nabla \times \mathbf{B} - \frac{1}{\varepsilon_0}\mathbf{J}_f, \tag{2.1d}$$

where $\mathbf{E}$ is the electric field, $\mathbf{B}$ is the magnetic induction field (commonly called just magnetic field), $\mathbf{J}$ is the current density, ρ is the charge density, ε_0 is the vacuum permittivity, and c is the speed of light. For continuous media it is common to use the macroscopic Maxwell equations instead

$$\nabla \cdot \mathbf{D} = \rho_f \tag{2.2a}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.2b}$$

$$\dot{\mathbf{B}} = -\nabla \times \mathbf{E} \tag{2.2c}$$

$$\dot{\mathbf{D}} = \nabla \times \mathbf{H} - \mathbf{J}_f, \tag{2.2d}$$

together with the constitutive relations

$$\mathbf{E} = \frac{1}{\varepsilon_0}(\mathbf{D} - \mathbf{P}) \tag{2.3}$$

$$\mathbf{H} = \frac{1}{\mu_0}\mathbf{B} - \mathbf{M}, \tag{2.4}$$

where $\mathbf{D}$ is the electric displacement field, $\mathbf{H}$ is the magnetic field, $\mathbf{J}_f$ is the current density of free charges, ρ_f is the density of free charges, $\mathbf{P}$ is the polarization field, $\mathbf{M}$ is the magnetization field, and μ_0 is the vacuum permeability.

The classical formulation of electrodynamics provided by either the microscopic or the macroscopic Maxwell equations fails for fields with very small wavelength λ and for very large electric field strength $|\mathbf{E}|$ [20]. Specifically, these equations are valid for

$$\lambda \gg r_e = \frac{1}{4\pi\varepsilon_0}\frac{e^2}{m_e c^2} = 2.818\,\text{fm},$$
$$|\mathbf{E}| \ll \frac{1}{\sqrt{4\pi\varepsilon_0}}\frac{e}{r_e^2} = 5.734 \times 10^{24}\ \text{Vm}^{-1},$$

where e is the elementary charge, and m_e is the electron mass. Beyond this range of validity, it is necessary to adopt a quantum description of the field, which goes beyond the

scope of this thesis.

2.2 Particle dynamics and kinetic description

Both the microscopic and macroscopic Maxwell equations contain source terms, such as ρ, $\mathbf{J}$, $\mathbf{P}$, and $\mathbf{M}$, which describe the influence of the particles that comprise a gas or plasma on the electromagnetic field.

Similarly to the case of the fields, the most fundamental description of the particles is based on a quantum approach. However, it is possible to adopt a classical description when the particles have an average distance from each other that is much greater than the thermal de Broglie wavelength λ_{th}

$$n^{-\frac{1}{3}} \gg \lambda_{th}, \tag{2.5}$$

where n is the density and the thermal de Broglie wavelength λ_{th} for non-relativistic particles is

$$\lambda_{th} = \frac{h}{2\pi m k_B T},$$

where m is the mass of the particles, T is the temperature, h is the Plank constant, and k_B is the Boltzmann constant. The thermal de Broglie wavelength basically describes the length above which the particles no longer exhibit coherent behaviour. When condition (2.5) is not met, the particles experience coherent interaction and the gas or plasma may display a macroscopic quantum behaviour.

The state of this system of particles can now be represented by a set of positions $\{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_N\}$ and momenta $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\}$, with each pair $\mathbf{r}_i$, $\mathbf{p}_i$ being a point in 6-dimensional phase space. In this context, the evolution of the system follows the dynamical equations

$$\dot{\mathbf{r}}_i = \mathbf{v}_i \tag{2.6a}$$

$$\dot{\mathbf{p}}_i = \mathbf{F}_i \tag{2.6b}$$

where $\mathbf{F}_i$ is the applied force, and $\mathbf{v}_i$ is the velocity of each particle. For a fully relativistic description, the velocity of the particles is given by

$$\mathbf{v}_i = \frac{\mathbf{p}_i}{m\gamma_i}, \tag{2.7}$$

with the Lorentz factor

$$\gamma_i = \sqrt{1 + \left(\frac{|\mathbf{p}_i|}{mc}\right)^2}.$$

With this groundwork, we can now introduce a statistical description of the phase space by defining a probability density function f, that is defined by

$$dN = f\left(t, \mathbf{r}, \mathbf{p}\right) d^3\mathbf{r}\, d^3\mathbf{p},$$

where dN is the number of particles located within the volume $d^3\mathbf{r}$ and with momentum within the volume $d^3\mathbf{p}$. The evolution of the system is given by the Collisionless Boltzmann Equation (CBE)

$$\dot{f} + \mathbf{v} \cdot \nabla_{\mathbf{r}} f + \mathbf{F} \cdot \nabla_{\mathbf{p}} f = 0,$$

where $\nabla_{\mathbf{r}}$ and $\nabla_{\mathbf{p}}$ are the gradients taken only along position and momentum space, respectively. Boltzmann introduced the binary collision term into the right hand side of the equation as the main form of interaction between the particles

$$\left(\frac{\partial f}{\partial t}\right)_{coll} = \int g I\left(g, \Omega\right) \left(f(t, \mathbf{x}, \mathbf{p}_1') f(t, \mathbf{x}, \mathbf{p}_2') - f(t, \mathbf{x}, \mathbf{p}_1) f(t, \mathbf{x}, \mathbf{p}_2)\right) d\Omega d^3\mathbf{p}_1\, d^3\mathbf{p}_2, \quad (2.8)$$

where $g = |\mathbf{p}_2 - \mathbf{p}_1|$, and $I\left(g, \Omega\right)$ is the differential cross section.

For charged particles, as Vlasov found, the collision term can usually be discarded in favour of the long distance interaction through the Lorentz force[19]. Thus, $\mathbf{F}$ is replaced by

$$\mathbf{F} = q\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right), \quad (2.9)$$

where q is the charge of the particles. The Lorentz force finally closes the feedback loop between the particles and the electromagnetic field.

Neglecting collisions and generalising the model to take into account that a plasma is usually composed of several species of particles (ions, electrons, etc.), a full dynamical theory for plasma behaviour is given by the following equations

$$\begin{aligned}
\dot{f}_s &= -\mathbf{v} \cdot \nabla_{\mathbf{r}} f_s - q_s \left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) \cdot \nabla_{\mathbf{p}} f_s \\
\dot{\mathbf{B}} &= -\nabla \times \mathbf{E} \\
\dot{\mathbf{E}} &= c^2 \nabla \times \mathbf{B} - \frac{1}{\varepsilon_0} \sum_s q_s \int_{-\infty}^{\infty} f_s \mathbf{v}\, d^3\mathbf{p}
\end{aligned}$$

where we introduced the expression for the current density

$$\mathbf{J} = \sum_s q_s \int_{-\infty}^{\infty} f_s \mathbf{v}\, d^3\mathbf{p}.$$

The remaining Maxwell equations are unnecessary, as long as they are satisfied at some initial time. Finally, it is important to notice that, with the exception of the collision term in equation (2.8), the models in this section are fully relativistic with the definition in equation (2.7).

2.3 Fluids in equilibrium

In the previous section, we described models for the dynamics of particles in gases and plasmas that are valid in conditions out of equilibrium. However, when the particle distributions are in local thermodynamic equilibrium, it is possible to adopt alternative models which are simpler or provide better physical insight. In equilibrium, we know that the local momentum distributions will be close to the Maxwell-Jüttner distribution

$$f(\mathbf{p}) = \frac{1}{4\pi m^3 c^3 \theta K_2\left(\frac{1}{\theta}\right)} \exp\left(-\frac{\gamma}{\theta}\right),$$

where $\theta = \frac{k_B T}{mc^2}$ and K_2 is the modified Bessel function of the second kind, or in the non-relativistic limit, the Maxwell-Boltzmann distribution

$$f(\mathbf{p}) = \sqrt{\frac{2}{\pi m T^3 k_B^3}}\, \mathbf{p}^2 \exp\left(-\frac{\mathbf{p}^2}{2mTk_B}\right).$$

For neutral fluids, the Knudsen number provides a useful heuristic to know whether an equilibrium description is valid. It is defined as $Kn = \lambda / L$, where λ is the mean free path and L is a representative length scale of the problem. If $Kn \gtrsim 1$, the hydrodynamic model does not provide a good description of the system. On the other hand, for $Kn \ll 1$ the equilibrium assumption is valid and it is possible to use the hydrodynamic equations

$$\dot{\rho} = -\nabla \cdot (\rho \mathbf{v}) \tag{2.11a}$$

$$\dot{\mathbf{v}} = -(\mathbf{v} \cdot \nabla)\mathbf{v} - \frac{1}{\rho}\nabla p + \frac{1}{\rho}\mathbf{F} + \frac{\mu}{\rho}\left(\nabla^2 \mathbf{v} + \frac{1}{3}\nabla(\nabla \cdot \mathbf{v})\right) \tag{2.11b}$$

$$\dot{\epsilon} = -\mathbf{v} \cdot \nabla \epsilon + \frac{1}{\rho}\nabla \cdot (K \nabla T) - \frac{p}{\rho}\nabla \cdot \mathbf{v} \tag{2.11c}$$

where ρ is the mass density, $\mathbf{v}$ is the velocity field, p is the pressure, $\mathbf{F}$ is the external force, μ is the viscosity coefficient, ϵ is the internal energy, K is the thermal conductivity, and T

is the temperature. These equations correspond respectively to the conservation of mass, momentum, and energy.

A common simplification for the fluid equations is the assumption of incompressibility, mathematically stated as

$$\nabla \cdot \mathbf{v} = 0.$$

This assumption is valid when the flow speed $\mathbf{v}$ is much smaller than the speed of sound of the fluid (corresponding to a low Mach number). In this case, equation (2.11b) simplifies to

$$\dot{\mathbf{v}} = -(\mathbf{v} \cdot \nabla)\,\mathbf{v} - \frac{1}{\rho}\nabla p + \frac{1}{\rho}\mathbf{F} + \frac{\mu}{\rho}\nabla^2 \mathbf{v}. \tag{2.12}$$

Further simplification can be achieved for inviscid flows where the viscosity of the fluid is close to zero. The validity of this assumption can be substantiated in terms of the dimensionless Reynolds number, which measures the ratio between the inertial and viscous forces, and is defined as

$$Re = \frac{\rho v L}{\mu},$$

where ρ is the density, v is the flow velocity, L is the characteristic length scale, and μ is the dynamic viscosity.

In particular, for fluids with small viscosity, corresponding to $Re \gg 1$, the fluid can be considered inviscid and equation (2.12) further simplifies to

$$\dot{\mathbf{v}} = -(\mathbf{v} \cdot \nabla)\,\mathbf{v} - \frac{1}{\rho}\nabla p + \frac{1}{\rho}\mathbf{F}. \tag{2.13}$$

In the case of plasmas it is also possible to obtain an equilibrium description, For example, the ideal MHD equations include equation (2.11a) for mass conservation, and equation (2.13) for momentum conservation. However, the force term is given by

$$\mathbf{F} = \mathbf{J} \times \mathbf{B} = \frac{(\mathbf{B} \cdot \nabla)\,\mathbf{B}}{\mu_0} - \nabla\left(\frac{\mathbf{B}^2}{2\mu_0}\right).$$

Under these conditions, equation (2.13) becomes

$$\dot{\mathbf{v}} = -(\mathbf{v} \cdot \nabla)\,\mathbf{v} - \frac{1}{\rho}\nabla p + \frac{(\mathbf{B} \cdot \nabla)\,\mathbf{B}}{\mu_0 \rho} - \frac{1}{\rho}\nabla\left(\frac{\mathbf{B}^2}{2\mu_0}\right).$$

This single-fluid model is only applicable in a narrow range of parameters where the collisions are sufficient to prompt local equilibriums, but do not cause resistivity. Still, these equations give good qualitative results, and remain a good first approach to study plasmas[2]. There are several extensions to these equations, including the addition of

resistive effects and viscosity, as well as two-fluid models, which include the electric field explicitly, and three-fluid models for plasmas that are not fully ionized.

2.4 Conclusions

This chapter provided a brief discussion of the fundamental models that describe the interaction between the electromagnetic field and the charged particles in a plasma, or the neutral atoms in a gas. The phenomenology behind these systems is very rich, allowing for a wide variety of regimes, and naturally, an equally rich diversity of models to describe them. Here, we have focused on the fundamental ones, which will be relevant in the next chapter when we discuss the numerical models used to simulate these systems, justify the specific choice of the numerical model adopted, and analyse the results of the simulations.

Chapter 3

Numerical models

The previous chapter presented several physical models that capture the fundamental dynamics of plasmas and gases. Although all the systems, including the electromagnetic field, are described by partial differential equations, the models for fluids out of equilibrium operate in a 6-dimensional space and are very difficult to solve directly in their normal formulation. Furthermore, as a design objective, the goal is to develop distinct solvers that can interact with each other and exchange information.

In this chapter, several numerical methods are presented to address the physical models of chapter 2. The main focus will be on the PIC method from plasma physics. We will start by looking at the methods available for the the electromagnetic field in section 3.1, focusing on time-domain methods and more specifically the Finite-Difference Time-Domain (FDTD) method. In section 3.3, several methods for describing fluid in and out of equilibrium are discussed and the choice of the PIC method for plasmas is justified. The last section explores this method in detail.

3.1 Numerical methods for electrodynamics

There is a wide diversity of solvers for the Maxwell equations and choosing a method can be a challenge. Since the objective is to couple a solver of the Maxwell equations with solvers of other equations, such as the Vlasov or the Bloch equations, as discussed in chapter 1, a simple and robust method is desirable. Furthermore, the method must be parallelizable on GPU.

Frequency-domain techniques make it very difficult to include material nonlinearities, which conflicts with the goal of combining with the Bloch and Vlasov equations[21].

Moreover, assuming a time-domain method, a reasonably regular mesh is preferable since it will allow for fast interpolation of field quantities in the positions of the particles and conversely, fast deposition of particle quantities to the grid[1]. This excludes Finite-Element Methods (FEMs). We are left with FDTD method as the prime choice. The FDTD method also has other advantages of which the following are some of the most important[21]:

- FDTD does not require linear algebra solvers as it is a direct method, unlike integral-equation method and FEM;
- FDTD is robust as the sources of error for this method are well studied and can be restricted for a wide variety of problems;
- FDTD is systematic in the sense that it is easy to adapt to any problem without much effort. It does not require, for example, the reformulation of the mesh like integral-equation method and FEM;
- FDTD handles nonlinearities and impulse responses naturally.

However, there is one other option similar to FDTD that conserves some of the advantages listed above, while solving the problem of numerical phase velocity anisotropy found in FDTD: the Pseudospectral Time-Domain (PSTD) method. In FDTD, this anisotropy can be mitigated by using the different meshes such as the truncated octahedron lattice or adopting higher order finite-difference derivatives. The PSTD method entirely changes the way the spatial derivatives are evaluated, using Discrete Fourier Transform (DFT) with Fast Fourier Transform (FFT), or Discrete Chebyshev Transform (DCT), to obtain an almost isotropic numerical dispersion relation. However, we chose to foucus on the FDTD method since it has better performance. A brief discussion of the PSTD method is given in appendix A.

3.2 The Finite-Difference Time-Domain method

3.2.1 Yee cell and the leapfrog scheme

The FDTD method samples the simulation domain in a regular mesh composed of many cells, each of which contains values for the components of the electric and magnetic

[1] A formal definition of *depositing* particle quantities will be given ahead, but essentially it is the process of transforming a particle related quantity, such as charge and velocity, and transforming it into a field on the mesh, like charge density or current density.

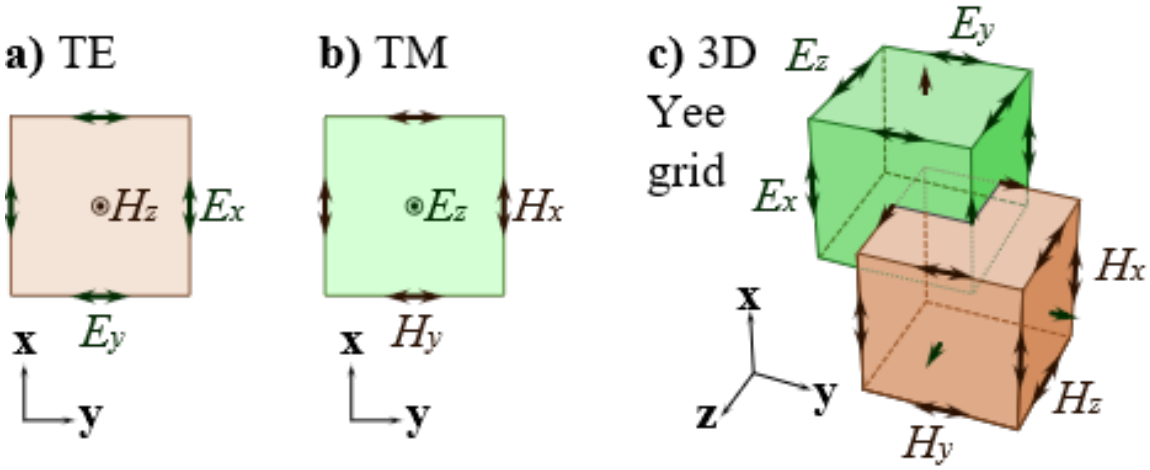


FIGURE 3.1: The Yee cell. The electric field components are stored in the centre of the minimal faces of the cell and the magnetic field components are stored in the centre of the minimal edges.

vector fields, as well as, scalar fields such as the charge density, and the dielectric constant. Each of these values can be sampled at a different location within the cell. In most cells, the components of the electric and magnetic fields will be positioned so as to ease the calculation of the curl of one of the fields and placing the result on the other field. Also, the charge density sampling will usually be aligned with the components of the electric field to facilitate the application of the Gauss law.

Given that a complex cell shape falls under the same problems of FEMs described above, the original Yee cell[22] is the best option and can be seen in figure 3.1. Compared with collocating all components at the centre of the cell, this method reduces the spacing in the finite-difference by half while not incurring in any additional calculations, thus maintaining the truncation error.

Equipped with the Yee cell, we can now outline the FDTD method. We consider the minimal edges and faces, and their respective vector field components, to belong to the cell indexed with i, j, and k. We also consider the fields $\left(E_x|_{i,j,k}, E_y|_{i,j,k}, E_z|_{i,j,k}\right)$ and $\left(D_x|_{i,j,k}, D_y|_{i,j,k}, D_z|_{i,j,k}\right)$ to be sampled at the faces, the $\left(B_x|_{i,j,k}, B_y|_{i,j,k}, B_z|_{i,j,k}\right)$ and $\left(H_x|_{i,j,k}, H_y|_{i,j,k}, H_z|_{i,j,k}\right)$ to be sampled at the edges, and scalar fields to be stored at the centre of the cell. With this scheme we can formulate the temporal evolution of the electromagnetic field based on equations (2.2c) and (2.2d) as a set of finite-difference equations

$$D_x|_{i,j,k}^{t+\Delta_t} = D_x|_{i,j,k}^{t} + \Delta_t \left(\frac{H_z|_{i,j+1,k}^{t+\frac{\Delta_t}{2}} - H_z|_{i,j,k}^{t+\frac{\Delta_t}{2}}}{\Delta_y} + \frac{H_y|_{i,j,k}^{t+\frac{\Delta_t}{2}} - H_y|_{i,j,k+1}^{t+\frac{\Delta_t}{2}}}{\Delta_z} - J_x|_{i,j,k}^{t+\frac{\Delta_t}{2}} \right) \tag{3.1a}$$

$$D_y|_{i,j,k}^{t+\Delta_t} = D_y|_{i,j,k}^{t} + \Delta_t \left(\frac{H_x|_{i,j,k+1}^{t+\frac{\Delta_t}{2}} - H_x|_{i,j,k}^{t+\frac{\Delta_t}{2}}}{\Delta_z} + \frac{H_z|_{i,j,k}^{t+\frac{\Delta_t}{2}} - H_z|_{i+1,j,k}^{t+\frac{\Delta_t}{2}}}{\Delta_x} - J_y|_{i,j,k}^{t+\frac{\Delta_t}{2}} \right) \tag{3.1b}$$

$$D_z|_{i,j,k}^{t+\Delta_t} = D_z|_{i,j,k}^{t} + \Delta_t \left(\frac{H_y|_{i+1,j,k}^{t+\frac{\Delta_t}{2}} - H_y|_{i,j,k}^{t+\frac{\Delta_t}{2}}}{\Delta_x} + \frac{H_x|_{i,j,k}^{t+\frac{\Delta_t}{2}} - H_x|_{i,j+1,k}^{t+\frac{\Delta_t}{2}}}{\Delta_y} - J_z|_{i,j,k}^{t+\frac{\Delta_t}{2}} \right) \tag{3.1c}$$

$$B_x|_{i,j,k}^{t+\frac{3\Delta_t}{2}} = B_x|_{i,j,k}^{t+\frac{\Delta_t}{2}} + \Delta_t \left(\frac{E_z|_{i,j-1,k}^{t+\Delta_t} - E_z|_{i,j,k}^{t+\Delta_t}}{\Delta_y} + \frac{E_y|_{i,j,k}^{t+\Delta_t} - E_y|_{i,j,k-1}^{t+\Delta_t}}{\Delta_z} \right) \tag{3.2a}$$

$$B_y|_{i,j,k}^{t+\frac{3\Delta_t}{2}} = B_y|_{i,j,k}^{t+\frac{\Delta_t}{2}} + \Delta_t \left(\frac{E_x|_{i,j,k-1}^{t+\Delta_t} - E_x|_{i,j,k}^{t+\Delta_t}}{\Delta_z} + \frac{E_z|_{i,j,k}^{t+\Delta_t} - E_z|_{i-1,j,k}^{t+\Delta_t}}{\Delta_x} \right) \tag{3.2b}$$

$$B_z|_{i,j,k}^{t+\frac{3\Delta_t}{2}} = B_z|_{i,j,k}^{t+\frac{\Delta_t}{2}} + \Delta_t \left(\frac{E_y|_{i-1,j,k}^{t+\Delta_t} - E_y|_{i,j,k}^{t+\Delta_t}}{\Delta_x} + \frac{E_x|_{i,j,k}^{t+\Delta_t} - E_x|_{i,j-1,k}^{t+\Delta_t}}{\Delta_y} \right), \tag{3.2c}$$

where Δ_t is the temporal discretization, and Δ_x, Δ_y, and Δ_z are the spatial discretizations and side lengths of the Yee cell. Using these equations to evolve the field satisfies automatically equations (2.2a) and (2.2b), as long as they are obeyed by the initial conditions. The superscripts $t - \frac{1}{2}$, t, $t - \frac{\Delta_t}{2}$ and $t + \Delta_t$ represent the time at which the fields are evaluated, and so, the electric and magnetic fields are interleaved both in space and time. As a result, the time derivatives are centred, thus improving the error from $\mathcal{O}(\Delta_t)$ to $\mathcal{O}(\Delta_t^2)$. This technique is called *leapfrog*.

A full step also includes equations (2.3) and (2.4) whose numerical application is direct. The cycle starts with $\mathbf{D}^t$, $\mathbf{E}^t$, $\mathbf{B}^{t+\frac{\Delta_t}{2}}$, $\mathbf{H}^{t+\frac{\Delta_t}{2}}$ and has the following steps

1. apply equations (3.1a) to (3.1c): $\mathbf{D}^t \to \mathbf{D}^{t+\Delta_t}$;
2. apply equation (2.3): $\mathbf{E}^t \to \mathbf{E}^{t+\Delta_t}$;
3. apply equations (3.2a) to (3.2c): $\mathbf{B}^{t+\frac{\Delta_t}{2}} \to \mathbf{B}^{t+\frac{3\Delta_t}{2}}$;
4. apply equation (2.4): $\mathbf{H}^{t+\frac{\Delta_t}{2}} \to \mathbf{H}^{t+\frac{3\Delta_t}{2}}$.

Although this discretization primes for its simplicity, electromagnetic waves evolved using this algorithm no longer satisfy the normal dispersion relation of electromagnetic

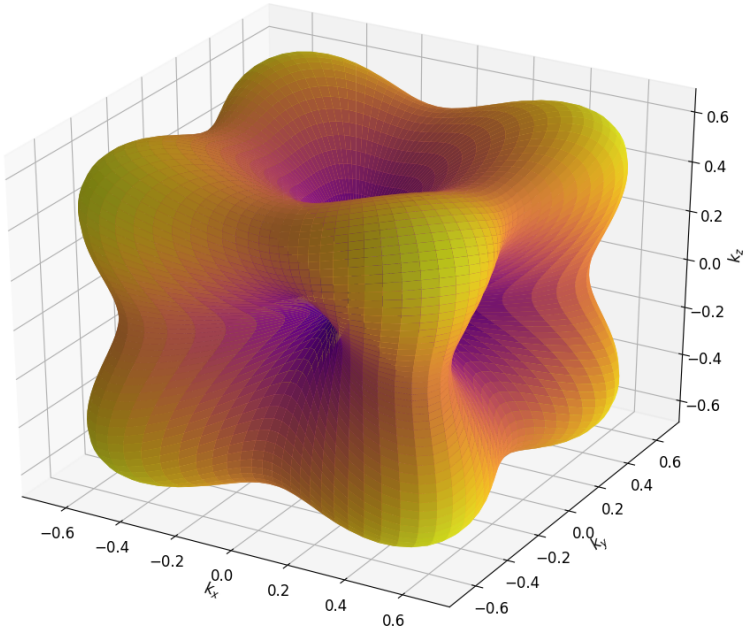


FIGURE 3.2: Spherical plot of the velocity anisotropy of the Yee cell calculated for $\lambda = 500\,\text{nm}$, $\Delta_x = \Delta_y = \Delta_z = \lambda/50$, and $\Delta_t = 0.1\Delta_x/c$. The distance from the origin to the surface represents the velocity of the wave for that direction in space. The values are renormalized and the distance to the centre is equal to $\frac{v-\min\{v\}}{\max\{v\}-\min\{v\}}$ for velocity v. As shown, the directions of propagation aligned with the axes of the grid have the lowest velocity. The velocity increases to its maximum along the the directions $(\pm 1, \pm 1, \pm 1)$.

radiation

$$\left(\frac{\omega}{c}\right)^2 = k_x^2 + k_y^2 + k_z^2,$$

following instead a distorted version given by

$$\left[\frac{1}{c\Delta_t}\sin\left(\frac{\omega\Delta_t}{2}\right)\right]^2 = \left[\frac{1}{\Delta_x}\sin\left(\frac{k_x\Delta_x}{2}\right)\right]^2 + \left[\frac{1}{\Delta_y}\sin\left(\frac{k_y\Delta_y}{2}\right)\right]^2 + \left[\frac{1}{\Delta_z}\sin\left(\frac{k_z\Delta_z}{2}\right)\right]^2.$$

This dispersion relation causes a spatial anisotropy of the velocity of propagation of waves along different directions, as shown in figure 3.2. Figure 3.3, on the other hand, shows the dependence of the anisotropy on Δ_x.

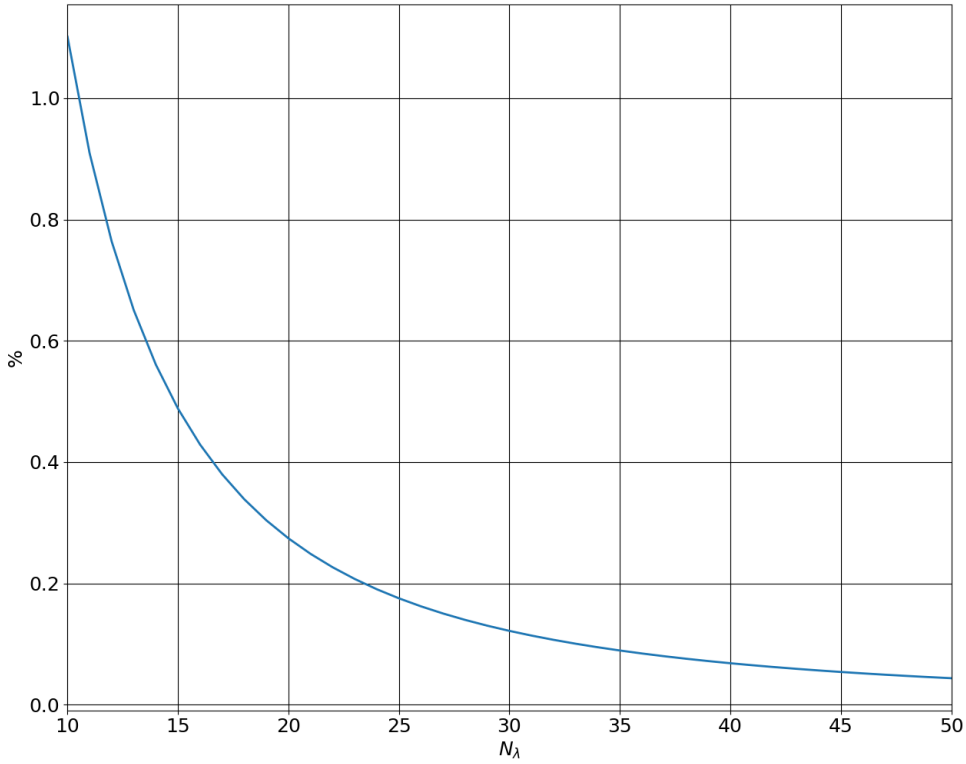


FIGURE 3.3: Percentage difference between the maximum and minimum velocities $\frac{v_{max}-v_{min}}{2(v_{max}+v_{min})}$ as a function of $N_\lambda = \lambda/\Delta_x$. For small N_λ the velocity anisotropy is very large, meaning that waves with the same wavelength will propagate with different velocities in different directions. This effect can introduce numerical distortions in complex wavefronts.

The FDTD method, as outlined above, is only stable if the following Courant relation is obeyed

$$\Delta_t \leq \frac{1}{c\sqrt{\frac{1}{\Delta_x^2}+\frac{1}{\Delta_y^2}+\frac{1}{\Delta_z^2}}}.$$

The relation is better understood in this simpler form for a cubic cell ($\Delta_x = \Delta_y = \Delta_z = \Delta$)

$$\Delta_t \leq \frac{\Delta}{c\sqrt{3}}.$$

Furthermore, the minimum number of grid points per wavelength N_λ for the smallest wavelength is considered to be around 10[23]. There are methods to overcome the Courant limit but they only apply in situations where the maximum Δ is limited by the geometrical features of the simulation[21].

3.2.2 Initial conditions

As mentioned before, evolving the electromagnetic field through the Faraday and Ampere laws (equations (2.2c) and (2.2d)) ensures the validity of the Gauss laws for electric and magnetic fields (equations (2.2a) and (2.2b)), as long as they are met by the initial conditions. The initial fields **D**, **E**, **B**, and **H** can not be uniquely determined for a general initial situation with charge density ρ, current density **J**, polarization **P**, and magnetization **M**. This can be understood by considering the Jefimenko equations, which contain the retarded time term[24]. Completely determining the fields at the initial time $t = 0$ would require knowing ρ and **J** at times $t < 0$. Since that is impossible, the best option is to use electrostatic and magnetostatic approximations where the fields can be obtained by solving four Poisson equations, specifically for a scalar potential ϕ, and the three components of the vector potential **A**

$$\nabla^2\phi = -\rho_f$$

$$\nabla^2\mathbf{A} = \mu_0\mathbf{J}_f,$$

and then applying the gradient and curl according to

$$\mathbf{D} = -\nabla\phi \tag{3.3}$$

$$\mathbf{B} = \nabla \times \mathbf{A}. \tag{3.4}$$

Numerically, the calculation of equations (3.3) and (3.4) can be done using FFTs as

$$\mathbf{D} = \nabla\left(\mathcal{F}^{-1}\left\{-\frac{1}{\mathbf{k}^2}\mathcal{F}\left\{\rho_f\right\}\right\}\right) \tag{3.5}$$

$$\mathbf{B} = \nabla \times \left(\mathcal{F}^{-1}\left\{-\frac{1}{\mathbf{k}^2}\mathcal{F}\left\{\mathbf{J}_f\right\}\right\}\right), \tag{3.6}$$

where **k** is the spatial frequency in reciprocal space. This solution will be valid if

$$\frac{\left|\rho_f(t_r, \mathbf{r}') - \rho_f(0, \mathbf{r})\right|}{\left|\rho_f(0, \mathbf{r})\right|} \ll 1$$
$$\frac{\left|\mathbf{J}_f(t_r, \mathbf{r}') - \mathbf{J}_f(0, \mathbf{r})\right|}{\left|\mathbf{J}_f(0, \mathbf{r})\right|} \ll 1,$$

with retarded time

$$t_r = \frac{|\mathbf{r} - \mathbf{r}'|}{c},$$

for any positions **r** and **r**′ in the simulation domain.

Another option is to create simulation scenarios where the initial conditions correspond to cases with simple solutions. For example, starting with a globally neutral distribution of particles, where in each point of space there is as much positive as negative charge by matching charges with opposite signs. Also, when simulating intense electromagnetic pulses that will rapidly increase the kinetic energy of the particles, it is possible to assume that the thermal velocities of the particles are insignificant, and all the charges are initially at rest. This type of initial condition is known as *quiet start*.

3.2.3 Boundary conditions

Typically, there are 3 types of boundary conditions that are implemented with FDTD:

- Periodic Boundary Conditions (PBC);
- Absorbing Boundary Conditions (ABC);
- Reflecting Boundary Conditions (RBC).

With PBC, a wave that exits the simulation domain at some boundary immediately reenters the domain in the opposite boundary. This is the easiest boundary condition to implement and is formally equivalent to replacing the indices in equations (3.1a) to (3.1c) and (3.2a) to (3.2c) according to

$$i \to i \bmod N_x, \tag{3.7}$$

$$j \to j \bmod N_y, \tag{3.8}$$

$$k \to k \bmod N_z. \tag{3.9}$$

where N_x, N_y, and N_z are the number of Yee cells along each dimension. Furthermore, for the initial conditions, the method outlined using FFTs naturally follows the periodic boundary conditions.

With the ABC, a wave that exits the simulation domain is absorbed at the boundary and never reenters the domain again. This type of boundary condition is used many times to simulate the effects of a physically boundless domain. Here we will look into two different types of ABC: the Uniaxial Perfectly Matched Layer (UPML), a standard with the FDTD method but difficult to implement and computationally costly, and a method we called Multiplicative Absorbing Boundary Conditions (MABC) which was created as a simple and fast algorithm to obtain some crude results fast. However, we found that

the method performed surprisingly well (see section 4.2.1). Finally, the RBC are used to simulate, for example, waveguides, plasmas contained by a confinement field, or other material boundaries. Although they are not especially difficult to implement, they are seldom used to simulate models of propagating electromagnetic fields and will not be considered here.

3.2.3.1 Uniaxial Perfectly Matched Layer (UPML)

Starting with the UPML, the Maxwell equations have to be modified to accommodate the boundary conditions. Equation (3.1a) becomes

$$D_x|_{i,j,k}^{t+\Delta_t} = \left(\frac{2\varepsilon_0\kappa_y - \sigma_y\Delta_t}{2\varepsilon_0\kappa_y + \sigma_y\Delta_t}\right) D_x|_{i,j,k}^{t} + \left(\frac{2\varepsilon_0\Delta_t}{2\varepsilon_0\kappa_y + \sigma_y\Delta_t}\right) \times \left(\frac{H_z|_{i,j+1,k}^{t+\frac{\Delta_t}{2}} - H_z|_{i,j,k}^{t+\frac{\Delta_t}{2}}}{\Delta_y} + \frac{H_y|_{i,j,k}^{t+\frac{\Delta_t}{2}} - H_y|_{i,j,k+1}^{t+\frac{\Delta_t}{2}}}{\Delta_z} - J_x|_{i,j,k}^{t+\frac{\Delta_t}{2}}\right). \tag{3.10}$$

The constitutive relations also have to be changed

$$E_x|_{i,j,k}^{t+\Delta_t} = \left(\frac{2\varepsilon_0\kappa_z - \sigma_z\Delta_t}{2\varepsilon_0\kappa_z + \sigma_z\Delta_t}\right) E_x|_{i,j,k}^{t} + \left(\frac{1}{(2\varepsilon_0\kappa_z + \sigma_z\Delta_t)\,\varepsilon_0}\right) \times \left[(2\varepsilon_0\kappa_x + \sigma_x\Delta_t)\, D_x|_{i,j,k}^{t+\Delta_t} - (2\varepsilon_0\kappa_x - \sigma_x\Delta_t)\, D_x|_{i,j,k}^{t} - P_x|_{i,j,k}^{t+\Delta_t}\right]. \tag{3.11}$$

Equation (3.2a) becomes

$$B_x|_{i,j,k}^{t+\frac{3\Delta_t}{2}} = \left(\frac{2\varepsilon_0\kappa_y - \sigma_y\Delta_t}{2\varepsilon_0\kappa_y + \sigma_y\Delta_t}\right) B_x|_{i,j,k}^{t+\frac{\Delta_t}{2}} + \left(\frac{1}{(2\varepsilon_0\kappa_z + \sigma_z\Delta_t)\,\varepsilon_0}\right) \times \left(\frac{E_z|_{i,j-1,k}^{t+\Delta_t} - E_z|_{i,j,k}^{t+\Delta_t}}{\Delta_y} + \frac{E_y|_{i,j,k}^{t+\Delta_t} - E_y|_{i,j,k-1}^{t+\Delta_t}}{\Delta_z}\right), \tag{3.12}$$

and the final constitutive relation becomes

$$H_x|_{i,j,k}^{t+\Delta_t} = \left(\frac{2\varepsilon_0\kappa_z - \sigma_z\Delta_t}{2\varepsilon_0\kappa_z + \sigma_z\Delta_t}\right) H_x|_{i,j,k}^{t} + \left(\frac{1}{(2\varepsilon_0\kappa_z + \sigma_z\Delta_t)\,\mu_0}\right) \times \left[(2\varepsilon_0\kappa_x + \sigma_x\Delta_t)\, B_x|_{i,j,k}^{t+\Delta_t} - (2\varepsilon_0\kappa_x - \sigma_x\Delta_t)\, B_x|_{i,j,k}^{t}\right] - M_x|_{i,j,k}^{t+\Delta_t}. \tag{3.13}$$

The equations for the missing components can be obtained by permuting the (i, j, k) and (x, y, z) indices[21].

By inspecting the equations above, it is clear that, outside the borders $\kappa_x, \kappa_y, \kappa_z = 1$ and $\sigma_x, \sigma_y, \sigma_z = 0$, so that the equations simplify to the normal equations from section 3.2. At the borders, the numerical parameters κ_x and σ_x must vary smoothly to avoid numerical

reflections

$$k_x(x) = \begin{cases} 1 + (\kappa^-_{x,max} - 1) \cdot \left(\frac{d^-_x - x}{d^-_x}\right)^m, & \text{for } 0 \leq x < d^-_x \\ 1, & \text{for } d^-_x \leq x \leq L_x - d^+_x \\ 1 + (\kappa^+_{x,max} - 1) \cdot \left(\frac{x + L_x - d^+_x}{d^+_x}\right)^m, & \text{for } L_x - d^+_x < x \leq L_x \end{cases} \tag{3.14}$$

$$\sigma_x(x) = \begin{cases} \left(\frac{d^-_x - x}{d^-_x}\right)^m \sigma^-_{x,max}, & \text{for } 0 \leq x < d^-_x \\ 0, & \text{for } d^-_x \leq x \leq L_x - d^+_x \\ \left(\frac{x + L_x - d^+_x}{d^+_x}\right)^m \sigma^+_{x,max}, & \text{for } L_x - d^+_x < x \leq L_x \end{cases}, \tag{3.15}$$

where $L_x = N_x \Delta_x$, d^- and d^+ are the thicknesses of the boundaries at the start and end of the simulation domain, and m is an integer power typically chosen to be 3 or 4. Again, the remaining equations are obtained by cycling through the indices (x, y, z).

With a chosen d, m, and reflection coefficient R, for a wave placed perpendicular to the boundary, the optimal $\sigma_{x,max}$ is

$$\sigma_{x,max} = -\frac{(m+1)\log(R)}{2\mu_0 c d}.$$

3.2.3.2 Multiplicative Absorbing Boundary Conditions (MABC)

As briefly discussed, this method was created in the course of this thesis as a simple and fast method to obtain rudimentary ABC, but produced unexpectedly good results, as will be discussed in section 4.2.1. The method consists of simply multiplying the fields **D**, **E**, **B**, and **H** by a spatially varying coefficient

$$\alpha = \alpha_x(x)\alpha_y(y)\alpha_z(z),$$

where

$$\alpha_x(x) = \begin{cases} \left(\frac{d^-_x - x}{d^-_x}\right)^m \alpha^-_{x,max}, & \text{for } 0 \leq x < d^-_x \\ 0, & \text{for } d^-_x \leq x \leq L_x - d^+_x \\ \left(\frac{x + L_x - d^+_x}{d^+_x}\right)^m \alpha^+_{x,max}, & \text{for } L_x - d^+_x < x \leq L_x \end{cases}, \tag{3.16}$$

with $\alpha^-_{x,max}, \alpha^+_{x,max} \in]0, 1]$. The remaining equations are, again, obtained by permuting the lower indices.

3.3 Numerical methods for fluids

To model the transport dynamics of the fluids, it is necessaary to take into account, not only the nature of the fluid and whether it is neutral or charged, but also whether the

conditions of the fluid are close to local equilibriums or not. In the case of neutral fluids in equilibrium, the field of study of the hydrodynamic equations and their numerical solutions is usually known as Computational Fluid Dynamics (CFD). When combined with methods of computational electrodynamics, many of the numerical methods used in CFD can be adapted to model charged fluids in equilibrium and obtain solutions of the MHD equations.

For situations that clearly are out of equilibrium, such as the effects produced by ultrashort laser pulses propagating through a fluid, there are specific approaches. For example, neutral fluids out of equilibrium are described by the Boltzmann equation, which considers the evolution of the probability function of finding a particle in a 6-dimensional phase space. Although direct simulations of the 6-dimensional phase-space are sometimes used, the main method is DSMC. Finally, charged fluids out of equilibrium are described by the Vlasov equation, for which the main numerical method is the PIC. Again, direct simulations of the entire phase-space are rarely used.

It is important to note that the intention of this work is to combine a numerical model describing the fluid with a model for the electromagnetic field, in particular using the FDTD method. This will automatically exclude the usage of FEMs and other complex methods and favour finite-difference methods. For the hydrodynamic equations two of the classical examples of methods are the Marker-And-Cell (MAC)[25] and PIC[26] for incompressible and compressible fluids, respectively. The main distinction between these algorithms and directly integrating the hydrodynamic equations with finite-difference (Direct Numerical Simulation (DNS)), is the inclusion of free surfaces in the fluid[2]. Essentially, both methods employ the MAC cell illustrated in figure 3.4, but the particles play different roles. In the MAC method the particles are moved according to the velocity field (interpolated to their position) and any cell that contains particles is considered to contain fluid. This solves the problem of division by zero when dividing by the density ρ in equation (2.12). In the original PIC method, the particles take a more prominent role as they actually automatically ensure the conservation of mass (corresponding to equation (2.11a)) and account for the advection terms, $\mathbf{v} \cdot \nabla \mathbf{v}$ and $\mathbf{v} \cdot \nabla \epsilon$, in equations (2.11b) and (2.11c) by transferring properties from the previous cell to the next as particles move between them. Although the MAC has become obsolete, the PIC method is still widely

[2] A free surface in the fluid is a boundary between that fluid and another fluid or vacuum.

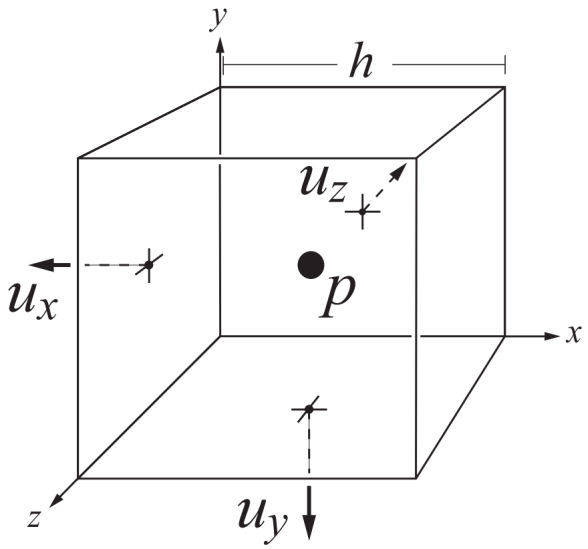


FIGURE 3.4: The MAC grid cell. Velocity components are stored in the minimal faces of the cell and the pressure p is represented at the center.

used and has several variations, such as Fluid Implicit Particle (FLIP) and Affine Particle-In-Cell (APIC)[27].

Some numerical methods for MHD adopt methods mentioned above and couple them with FDTD. Examples of this are the FLIP-MHD method[28] and the MAC-Yee scheme[29], where the Yee and MAC cells are combined.

Neutral fluids out of equilibrium are usually modelled by the Boltzmann equation. A direct approach would be to discretize the 6-dimensional phase-space, for example with finite-differences, and integrate the Boltzmann equation. However, for most situations, this requires huge amounts of computer memory and consequently very time-consuming computations. A normally favoured approach is DSMC, where the motion of several particles is tracked. Up to now, the description of DSMC is indistinct from the methods of MD. The difference is that in DSMC the number of numerical particles is much smaller than the number of particles in the fluid, with each simulated particle representing many real particles. This requires the collision cross section of each numerical particle to be the sum of the cross sections of the particles it represents, to maintain the overall number of collisions. The generic DSMC method starts by moving the particles without collision (in most cases this is a simple rectilinear uniform motion). Afterwards, the simulation domain is divided into cells with each particle pair in the same cell having a probability of collision proportional to the cross sections, velocity difference, and time step. The new velocities after the collision are updated according to a model, for example, the Hard

Sphere model. The smaller the cells the closer to $\mathcal{O}(n)$ the time complexity of algorithm will be, with n particles. This method is used, for example, in atmospheric vehicle re-entry, and upper atmosphere dynamics simulations.

Finally, plasmas out of equilibrium can be described by the Vlasov equation, which also admits a direct approach like the Boltzmann equation. Again, a more efficient method is based on simulating particles. The PIC method for plasmas, despite sharing its name with the method for compressible fluids, is very different in its usage of particles. While in the hydrodynamic PIC the information of the state of the fluid is split between fields and particles, the plasma PIC uses them to represents all the characteristics of the fluid. Furthermore, the particles have a finite size, which acts as a low-pass filter on the probability density function f. Despite not being point-like, they can pass through each other, and as they intersect their mutual interaction decreases, following the collisionless assumption made by Vlasov. The most common version of this method calculates the electromagnetic field through a FDTD algorithm. Then, the remaining components are a particle pusher, algorithms for interpolating the field quantities onto the particles, and conversely, algorithms for *depositing* the particle properties onto the mesh, for example, charge density and current density (the latter is the subject of various specialized methods).

In this thesis we chose to focus on the PIC method as it is one of the most general methods for describing fluids. The usage of particles is in line with the DSMC and adding collisions is possible with minor adjustments. Furthermore, the PIC method can easily be adapted to include other forces and potentials between the particles. This scheme will not be ideal in situations where the particles are in equilibrium but, its generality compensates for this drawback. The method will be examined in detail in the next section.

3.4 The Particle-In-Cell method

Simulating plasmas by following the paths of several particles began in the late 1950's and early 1960's, partly as an extension of electron beam simulations done during the 1950's. In PIC codes, each simulated particle actually represents several real particles in the plasma. This is a necessary reduction of the degrees of freedom since most plasmas of interest are several Debye lengths $\lambda_D \propto \sqrt{T/n}$ in size and the number of particles in a single Debye cube $N_D = n_e \lambda_D^3$ tends to be very large as seen in table 3.1 [30].
Furthermore, the simulated particles are usually not point-like but are instead a distribution of charge or *cloud*, normally called *super-particle*. This characteristic arises naturally

TABLE 3.1: The electron concentration n_e, electron temperature T_e, Debye length λ_D, and number of particles per Debye cube N_D listed for several mediums.

Plasma	$n_e\ (\mathrm{m}^{-3})$	$T_e\ (\mathrm{K})$	$\lambda_D\ (\mathrm{m})$	N_D
Solar core	10^{32}	10^{7}	10^{-11}	10^{-1}
Tokamak	10^{20}	10^{8}	10^{-4}	10^{8}
Gas discharge	10^{16}	10^{4}	10^{-4}	10^{4}
Ionosphere	10^{12}	10^{3}	10^{-3}	10^{3}
Magnetosphere	10^{7}	10^{7}	10^{2}	10^{13}
Solar wind	10^{6}	10^{5}	10	10^{9}
Interstellar medium	10^{5}	10^{4}	10	10^{8}
Intergalactic medium	1	10^{6}	10^{5}	10^{15}

from the usage of a grid for calculating the charge and current densities from the super-particles, but is also a useful trait that attenuates inter-particle forces has they overlap, thus obeying the collisionless assumption made by Vlasov. Of course, depending on the size of the super-particles, the fine details of the plasma behaviour will be obfuscated, leaving only the overall features[31,32].

3.4.1 Discretization of phase-space

The PIC method is a form of integrating the Vlasov equations as shown in appendix B. We start by sampling the 6-dimensional phase space of the CBE with *super-particles* p each representing w_p real particles

$$f(\mathbf{r},\mathbf{p},t) = \sum_p f_p(\mathbf{r},\mathbf{p},t) = \sum_p S\left(\mathbf{r}-\mathbf{r}_p(t)\right)\delta^3\left(\mathbf{p}-\mathbf{p}_p(t)\right), \tag{3.17}$$

where S is the *shape factor* and δ is the Dirac delta function. We will further assume that

1. $\int_{-\infty}^{+\infty} S_i(x)\,dx = 1$ for $i \in \{x, y, z\}$;
2. $S(\mathbf{r}) = S_x(x)\,S_y(y)\,S_z(z)$;
3. S_x, S_y and S_z are even;
4. $\lim_{i\to\pm\infty} S_i(i) = 0$ for $i \in \{x, y, z\}$.

Assumptions 2 and 3 simplify the derivations while not affecting the generality of the results.

The objective now is to deduce the equations governing $\mathbf{r}_p(t)$ and $\mathbf{p}_p(t)$ from the Vlasov equation. We start with the CBE expressed here in 3D Cartesian coordinates

$$\partial_t f + \sum_i v_i \partial_i f + \sum_i F_i \partial_{p_i} f = 0, \tag{3.18}$$

with $i = x, y, z$, and restate equation (3.17) in the same form, dropping the time dependence from notation

$$f_p = w_p \prod_i S_i \left(i - i_p\right) \delta \left(p_i - p_{i,p}\right).$$

The first argument comes from the fact that the CBE is linear (Vlasov's equation's nonlinearity is due to the force term depending on f). With this argument, we can treat each f_p independently.

Taking the first moment of the CBE along the position by multiplying both sides of equation (3.18) by position coordinate i and integrating over the entire phase space we get

$$w_p \partial_t i_p - w_p v_{i,p} + 0 = 0,$$

and joining for all i

$$\dot{\mathbf{r}}_p = \mathbf{v}_p.$$

The next equation comes from taking the first moment along the momentum p_i. Obtaining

$$w_p \partial_t p_{i,p} + 0 - w_i F_i = 0$$

or in vector form

$$\dot{\mathbf{p}}_p = \mathbf{F}.$$

The full derivation is shown in appendix B.

3.4.1.1 Equations of motion for the super-particles

Joining the results of the previous section we get

$$\dot{\mathbf{r}}_p = \mathbf{v}_p$$
$$\dot{\mathbf{p}}_p = \mathbf{F},$$

which is exactly the same as equations (2.6a) and (2.6b). Thus, the super-particles move as if they were point particles.

The manipulations of the third term were made assuming that **F** is constant. Relaxing this condition to include a dependence of the force on the position, and following through the same derivation, will yield the system of equations

$$\dot{\mathbf{r}}_p = \mathbf{v}_p \tag{3.20a}$$

$$\dot{\mathbf{p}}_p = \int_{-\infty}^{+\infty} \mathbf{F}(\mathbf{r}) S(\mathbf{r} - \mathbf{r}_p) d^3\mathbf{r} \tag{3.20b}$$

3.4.1.2 The *shape factors*

An appropriate choice for shape factor must meet the conditions already established earlier, both analytically and under the discretization of the Yee cell. The common choice is the family of b-spline functions

$$b_0(x) = \begin{cases} 1, & \text{if } -\frac{1}{2} < x \leqslant \frac{1}{2} \\ 0, & \text{else} \end{cases}$$

$$b_n(x) = b_0(x) * b_{n-1}(x),$$

where $*$ denotes the convolution. Then, the shape factor is

$$S_n(\mathbf{r}) = b_n\left(\frac{x}{\Delta_x}\right) b_n\left(\frac{y}{\Delta_y}\right) b_n\left(\frac{z}{\Delta_z}\right).$$

The b-splines ensure that the conversion from continuous space to the Yee mesh does not change the property

$$\int_{-\infty}^{+\infty} S_n(\mathbf{r} - \mathbf{r}_p) d^3\mathbf{r} = 1 \;\rightarrow\; \sum_{i,j,k} S_n(\mathbf{r}_{ijk} - \mathbf{r}_p) = 1$$

for any particle position $\mathbf{r}_p$, with $\mathbf{r}_{ijk}$ denoting the positions of the grid points. The most commonly used in PIC codes are the S_0 and S_1 as they match *nearest point interpolation* and *trilinear interpolation* respectively, in equation

$$\varphi_p = \sum_{i,j,k} \varphi_{ijk} S_n\left(x_{ijk} - x_p\right) S_n\left(y_{ijk} - y_p\right) S_n\left(z_{ijk} - z_p\right), \tag{3.21}$$

where φ is a generic scalar field, φ_p is the value of the field at the position of the particle, φ_{ijk} is the sampled value in the Yee cell, $(x_{ijk}, y_{ijk}, z_{ijk})$ is the position of the sampling points, and (x_p, y_p, z_p) is the position of the particle. For vector fields, this process is repeated for each component with the correct grid point positions. Conversely, to deposit

particle quantities onto the mesh the standard method is

$$\varphi_{ijk} = \frac{1}{\Delta_x \, \Delta_y \, \Delta_z} \sum_p \varphi_p S_n \left(x_{ijk} - x_p \right) S_n \left(y_{ijk} - y_p \right) S_n \left(z_{ijk} - z_p \right), \tag{3.22}$$

where φ_p is the quantity of interest of particle p. Again, for vector quantities, the method is repeated for each component.

3.4.2 Particle pusher

One of the key components of any PIC code is the particle pusher, used to integrate the Vlasov equation. Equations (3.20a) and (3.20b) are Ordinary Differential Equations (ODEs) and many general methods could be used, such as Adams-Bashforth or Runge-Kutta methods. However, the statistical nature of the phase space sampling does not justify using high order instances of these methods. Furthermore, the coupling with the FDTD method calls for a similar method for the particles. The standard discretization, as established by Boris[33], is

$$\frac{\mathbf{r}_{t+\Delta_t} - \mathbf{r}_t}{\Delta_t} = \mathbf{v}_{t+\frac{\Delta_t}{2}} \tag{3.23a}$$

$$\frac{\gamma_{t+\frac{\Delta_t}{2}} \mathbf{v}_{t+\frac{\Delta_t}{2}} - \gamma_{t-\frac{\Delta_t}{2}} \mathbf{v}_{t-\frac{\Delta_t}{2}}}{\Delta_t} = \frac{q}{m} \left(\mathbf{E}_t + \bar{\mathbf{v}}_t \times \mathbf{B}_t \right), \tag{3.23b}$$

where $\bar{\mathbf{v}}_t$ is the the velocity at time t and is obtained as a function of $\mathbf{v}_{t-\frac{\Delta_t}{2}}$ and $\mathbf{v}_{t+\frac{\Delta_t}{2}}$. Different pushers will have different definitions for $\bar{\mathbf{v}}_t$.

3.4.2.1 Boris pusher

The particle pusher developed by Boris is considered the *de facto* standard as it conserves energy while requiring no linear algebra solvers since it is a direct method[33]. The closing relation for $\bar{\mathbf{v}}_t$ proposed by Boris is

$$\bar{\mathbf{v}}_t = \frac{\gamma_{t+\frac{\Delta_t}{2}} \mathbf{v}_{t+\frac{\Delta_t}{2}} + \gamma_{t-\frac{\Delta_t}{2}} \mathbf{v}_{t-\frac{\Delta_t}{2}}}{2\sqrt{1 + \left(\gamma_{t-\frac{\Delta_t}{2}} \mathbf{v}_{t-\frac{\Delta_t}{2}} + \frac{q \, \Delta_t}{2m} \mathbf{E}_t \right)}},$$

This pusher conserves energy for $\mathbf{E} = \mathbf{0}$ and conserves phase-space volume according to Liouville's theorem[34].

Although this pusher is very robust, and is the best choice for non-relativistic PIC codes, it does present a flaw. When the electromagnetic field and the velocity of the particle are such that $\mathbf{E} + \mathbf{v} \times \mathbf{B} = \mathbf{0}$, with $\mathbf{E}, \, \mathbf{B} \neq \mathbf{0}$, it results in a spurious force[35].

3.4.2.2 Vay pusher

The Vay pusher eliminates the spurious force of the Boris pusher by introducing the alternative closing relation[35]

$$\bar{\mathbf{v}}_t = \frac{\mathbf{v}_{t-\frac{\Delta_t}{2}} + \mathbf{v}_{t+\frac{\Delta_t}{2}}}{2}.$$

Substituting in equation (3.23b) yields the following leapfrog pusher

$$\begin{aligned}
\mathbf{u}_t &= \mathbf{u}_{t-\frac{\Delta_t}{2}} + \frac{q\,\Delta_t}{2m}\left(\mathbf{E}_t + \mathbf{v}_{t-\frac{\Delta_t}{2}} \times \mathbf{u}_t\right) \\
\mathbf{u}_{t+\frac{\Delta_t}{2}} &= s\left(\mathbf{u}' + \left(\mathbf{u}' \cdot \mathbf{p}\right)\mathbf{p} + \mathbf{u}' \times \mathbf{p}\right),
\end{aligned}$$

with the following relations

$$\begin{aligned}
\mathbf{u} &= \gamma \mathbf{v} \\
\mathbf{u}' &= \mathbf{u}_t + \frac{q\,\Delta_t}{2m}\mathbf{E}_t \\
\mathbf{t} &= \frac{\boldsymbol{\tau}}{\gamma_{t+\frac{\Delta_t}{2}}} \\
s &= \frac{1}{1+\mathbf{t}^2} \\
\boldsymbol{\tau} &= \frac{q\,\Delta_t}{2m}\mathbf{B}_t = \frac{q\,\Delta_t}{4m}\left(\mathbf{B}_{t-\frac{\Delta_t}{2}} + \mathbf{B}_{t+\frac{\Delta_t}{2}}\right) \\
\gamma_{t+\frac{\Delta_t}{2}} &= \sqrt{\frac{\sigma + \sqrt{\sigma^2 + 4\left(\boldsymbol{\tau}^2 + u^{*2}\right)}}{2}} \\
\sigma &= 1 + \frac{u'^2}{c^2} - \boldsymbol{\tau}^2 \\
u^* &= \frac{\mathbf{u}' \cdot \boldsymbol{\tau}}{c}.
\end{aligned}$$

This particle pusher can maintain good agreement with analytical results up to $\gamma = 10^4$, while the Boris pusher experiences a significant discrepancy from the analytical solution at $\gamma = 3$. There is, however, a larger amount of calculations to be made, which impacts the run time of this algorithm. The Vay pusher is also reported as not being volume preserving and as having larger ($\approx 10^2$) energy fluctuations[34].

3.4.3 Current deposition

Using the standard method for depositing particle quantities onto the grid outlined in equation (3.22), and specifying it for current density **J** at the faces of the Yee cell (since it is added to **E** in equations (3.1a) to (3.1c)), yields

$$
\begin{aligned}
J^x_{ijk} &= \frac{1}{\Delta_x\,\Delta_y\,\Delta_z}\sum_p w_p q_p v^x_p S_n\left(x^x_{ijk}-x_p\right) S_n\left(y^x_{ijk}-y_p\right) S_n\left(z^x_{ijk}-z_p\right)\\
J^y_{ijk} &= \frac{1}{\Delta_x\,\Delta_y\,\Delta_z}\sum_p w_p q_p v^y_p S_n\left(x^y_{ijk}-x_p\right) S_n\left(y^y_{ijk}-y_p\right) S_n\left(z^y_{ijk}-z_p\right)\\
J^z_{ijk} &= \frac{1}{\Delta_x\,\Delta_y\,\Delta_z}\sum_p w_p q_p v^z_p S_n\left(x^z_{ijk}-x_p\right) S_n\left(y^z_{ijk}-y_p\right) S_n\left(z^z_{ijk}-z_p\right),
\end{aligned}
$$

where the grid points are located at

$$
\begin{aligned}
\left(x^x_{ijk},\, y^x_{ijk},\, z^x_{ijk}\right) &= \left(i\Delta_x,\, j\Delta_y+\frac{1}{2},\, k\Delta_z+\frac{1}{2}\right)\\
\left(x^y_{ijk},\, y^y_{ijk},\, z^y_{ijk}\right) &= \left(i\Delta_x+\frac{1}{2},\, j\Delta_y,\, k\Delta_z+\frac{1}{2}\right)\\
\left(x^z_{ijk},\, y^z_{ijk},\, z^z_{ijk}\right) &= \left(i\Delta_x+\frac{1}{2},\, j\Delta_y+\frac{1}{2},\, k\Delta_z\right).
\end{aligned}
$$

It is well known that this "naive" method leads to the violation of the equation of charge conservation[36–39]

$$
\dot{\rho}+\nabla\cdot\mathbf{J}=0, \tag{3.25}
$$

despite the charge itself being exactly conserved as long as no super-particle disappears. The error in the numerical estimation of the current density **J** will lead to the violation of the Gauss equation for the electric field. The first methods developed to tackle this problem were based on solving the Poisson equation at each step[31] or adding a *pseudo-current*[40]. The first method that focused on calculating a *charge conserving* current density was devised by Villasenor and Buneman[37], and it was conceived for super-particles of shape factor $n = 0$. Later, Esirkepov developed a general deposition method for any shape factor[38] that has become the standard in many PIC codes[41–43]. There are, however, other *charge conservation methods* such as the *zig-zag* method of Umeda developed for $n = 1,\, 2$[39,44] and later generalized for any order[45]. In this thesis we chose to work with the Esirkepov method as it is more general and used by many PIC codes.

With the Esirkepov method, the current density **J** is evaluated with the following equations

$$J^x_{i+1,j,k} - J^x_{ijk} = -\frac{\Delta_x}{\Delta_t} W^x_{ijk} \tag{3.26a}$$

$$J^y_{i,j+1,k} - J^y_{ijk} = -\frac{\Delta_y}{\Delta_t} W^y_{ijk} \tag{3.26b}$$

$$J^z_{i,j,k+1} - J^z_{ijk} = -\frac{\Delta_z}{\Delta_t} W^z_{ijk}, \tag{3.26c}$$

where

$$\begin{aligned} W^x_{ijk} = \frac{w_p q_p}{\Delta_x \Delta_y \Delta_z} \sum_p \Bigg[& \frac{1}{3}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t+\Delta_t}, z_p^{t+\Delta_t}\right) - \frac{1}{3}\mathcal{S}\left(x_p^{t}, y_p^{t+\Delta_t}, z_p^{t+\Delta_t}\right) + \\ & \frac{1}{6}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t}, z_p^{t+\Delta_t}\right) - \frac{1}{6}\mathcal{S}\left(x_p^{t}, y_p^{t}, z_p^{t+\Delta_t}\right) + \\ & \frac{1}{6}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t+\Delta_t}, z_p^{t}\right) - \frac{1}{6}\mathcal{S}\left(x_p^{t}, y_p^{t+\Delta_t}, z_p^{t}\right) + \\ & \frac{1}{3}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t}, z_p^{t}\right) - \frac{1}{3}\mathcal{S}\left(x_p^{t}, y_p^{t}, z_p^{t}\right) \Bigg] \end{aligned} \tag{3.27a}$$

$$\begin{aligned} W^y_{ijk} = \frac{w_p q_p}{\Delta_x \Delta_y \Delta_z} \sum_p \Bigg[& \frac{1}{3}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t+\Delta_t}, z_p^{t+\Delta_t}\right) - \frac{1}{3}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t}, z_p^{t+\Delta_t}\right) + \\ & \frac{1}{6}\mathcal{S}\left(x_p^{t}, y_p^{t+\Delta_t}, z_p^{t+\Delta_t}\right) - \frac{1}{6}\mathcal{S}\left(x_p^{t}, y_p^{t}, z_p^{t+\Delta_t}\right) + \\ & \frac{1}{6}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t+\Delta_t}, z_p^{t}\right) - \frac{1}{6}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t}, z_p^{t}\right) + \\ & \frac{1}{3}\mathcal{S}\left(x_p^{t}, y_p^{t+\Delta_t}, z_p^{t}\right) - \frac{1}{3}\mathcal{S}\left(x_p^{t}, y_p^{t}, z_p^{t}\right) \Bigg] \end{aligned} \tag{3.27b}$$

$$\begin{aligned} W^y_{ijk} = \frac{w_p q_p}{\Delta_x \Delta_y \Delta_z} \sum_p \Bigg[& \frac{1}{3}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t+\Delta_t}, z_p^{t+\Delta_t}\right) - \frac{1}{3}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t+\Delta_t}, z_p^{t}\right) + \\ & \frac{1}{6}\mathcal{S}\left(x_p^{t}, y_p^{t+\Delta_t}, z_p^{t+\Delta_t}\right) - \frac{1}{6}\mathcal{S}\left(x_p^{t}, y_p^{t+\Delta_t}, z_p^{t}\right) + \\ & \frac{1}{6}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t}, z_p^{t+\Delta_t}\right) - \frac{1}{6}\mathcal{S}\left(x_p^{t+\Delta_t}, y_p^{t}, z_p^{t}\right) + \\ & \frac{1}{3}\mathcal{S}\left(x_p^{t}, y_p^{t}, z_p^{t+\Delta_t}\right) - \frac{1}{3}\mathcal{S}\left(x_p^{t}, y_p^{t}, z_p^{t}\right) \Bigg], \end{aligned} \tag{3.27c}$$

with

$$\mathcal{S}(x, y, z) = S_n\left(\Delta_x\left(i+\frac{1}{2}\right) - x\right) S_n\left(\Delta_y\left(j+\frac{1}{2}\right) - y\right) S_n\left(\Delta_z\left(k+\frac{1}{2}\right) - z\right).$$

Here, the method is outlined for multiple particles since this form will be useful latter for the GPU implementation. It is important to note that the method assumes that

$$\left|x_p^{t+\Delta_t} - x_p^{t+\Delta_t}\right| < \Delta_x$$
$$\left|y_p^{t+\Delta_t} - y_p^{t+\Delta_t}\right| < \Delta_y$$
$$\left|z_p^{t+\Delta_t} - z_p^{t+\Delta_t}\right| < \Delta_z.$$

This is not very limiting considering it leads to a constraint similar to the Courant condition

$$\Delta_t < \frac{1}{c}\min\left\{\Delta_x, \Delta_y, \Delta_z\right\}, \tag{3.28}$$

and is in fact less limiting for a cubic cell, where it simplifies to

$$\Delta_t < \frac{\Delta}{c}.$$

3.4.4 Boundary conditions

For particles, the boundary conditions are simpler to implement than for the electromagnetic field. Again, the periodic boundary conditions are the default mechanism and the other useful boundary type is the ABC.

For the periodic boundaries, first we have to take the new positions $\left(x_p^{t+\Delta_t}, y_p^{t+\Delta_t}, z_p^{t+\Delta_t}\right)$ and replace them, at the end of the PIC loop, by

$$\left(x_p^{t+\Delta_t}, y_p^{t+\Delta_t}, z_p^{t+\Delta_t}\right) \rightarrow \left(x_p^{t+\Delta_t} \bmod L_x, y_p^{t+\Delta_t} \bmod L_y, z_p^{t+\Delta_t} \bmod L_z\right).$$

We must be careful, however, not to do this before calculating the current density since traversing the entire length of the simulation domain, due to the periodicity, will surely not meet condition (3.28). Next, to obtain the charge density necessary for the initial conditions of the FDTD method, in periodic boundary conditions, we use equation (3.22) but make the same index replacement of equations (3.7) to (3.9). Finally, for the current density, equations (3.26a) to (3.26c) are finite-difference equations that require the boundary condition $W(x_{min}) = W(x_{max}) = 0$. Thus, equations (3.27a) to (3.27c) must have a larger domain than the simulation by $n + 1$ at both ends along the indexed dimension, where n

is the shape factor order

$$\begin{aligned} W^x_{ijk}: &\quad -n-1 \leq i \leq N_x+n+1, \quad x_p \to x_p + \Delta_x(n+1) \\ W^y_{ijk}: &\quad -n-1 \leq j \leq N_y+n+1, \quad y_p \to y_p + \Delta_y(n+1) \\ W^z_{ijk}: &\quad -n-1 \leq k \leq N_z+n+1, \quad z_p \to z_p + \Delta_z(n+1). \end{aligned}$$

Only after calculating **J** can we apply the boundary conditions, essentially wrapping the excess cells onto the other side

$$J^{x,periodic}_{i+n+1,j,k} = J^x_{i+n+1,j,k} + J^x_{i+N_x+n+1,j,k}\,, \quad 0 \leq i \leq n+1,$$

with similar equations for the remaining dimensions.

For the ABC, the case is simpler and it consists of simply eliminating particles with positions exceeding the boundaries. The non-physical electromagnetic field resultant from eliminating the particles should be absorbed by the ABC of the field.

3.4.5 The PIC loop

The equations presented in sections 3.2 and 3.4 to push both the electromagnetic field and the particles by an interval Δ_t are summarized in figure 3.5. This is the best organization of the steps considering we are using the macroscopic Maxwell equations. If the microscopic formulation was used, steps 2 and 4 would not be necessary and the equation of steps 1 and 3 would the microscopic counterparts. Moreover, if the code was not relativistic, the previous positions of the particles would not be needed since they could be calculated effortlessly from the velocity. For the relativistic case, since the velocity that is saved is $\mathbf{u} = \gamma\mathbf{v}$[3], it is simpler to save the previous positions and reduce the computation time.

3.5 Conclusions

This chapter revised the main numerical methods used to simulate a plasma or gas, as a many-body problem involving atoms, ions, and electrons interacting with the electromagnetic field. As the result of a comparative study between the different methods, we have adopted the FDTD and PIC methods for the implementation of this code. We also

[3]The normalized velocity is employed to use the full range of the floating point representation. Using the velocity **v** would squeeze the values of ultra- relativistic particles close to c.

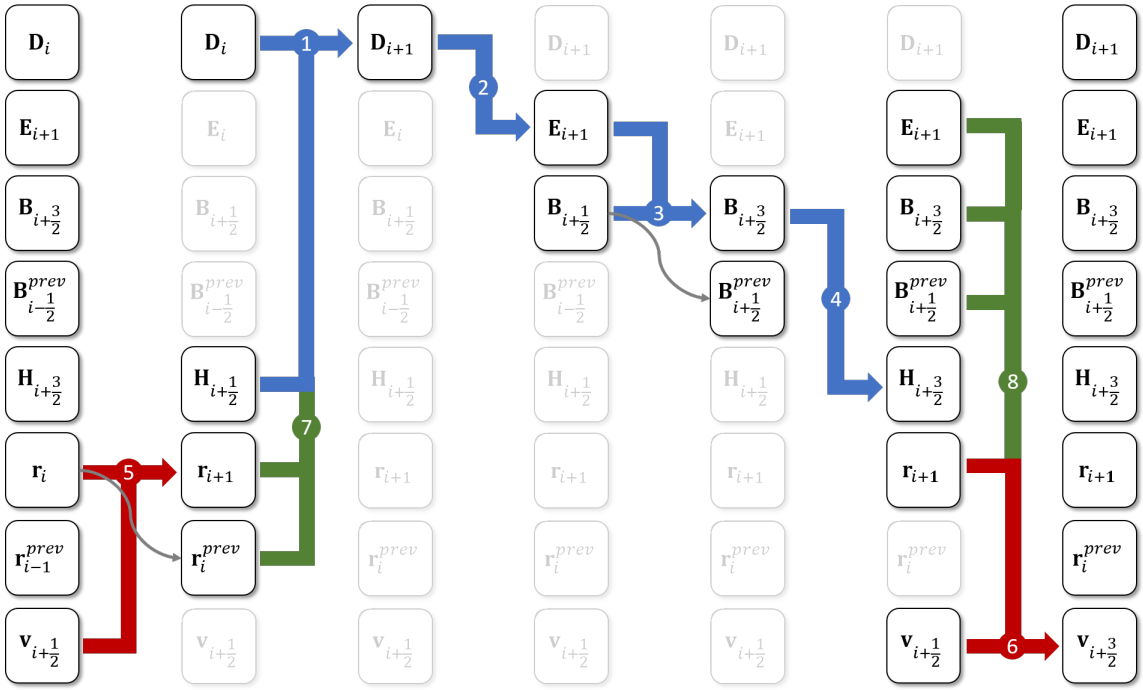


FIGURE 3.5: Diagram of the steps in the PIC loop. Each column contains all the necessary fields and particle quantities at different times, the blue arrows represent time evolutions related to the electromagnetic field, and the red arrows represent time evolutions related to the particles. The green arrows are the exchange of information between the field and the particles. Finally, the grey arrows represent the update of the previous quantities necessary for temporal interpolation and do not encompass any actual calculations. For the blue arrows, number 1 in the figure corresponds to equations (3.1a) to (3.1c) or equation (3.10) for the UPML, number 2 corresponds to equation (2.3) or equation (3.11), number 3 corresponds to equations (3.2a) to (3.2c) or equation (3.12), and number 4 corresponds to equation (2.4) or equation (3.13). Numbers 5 and 6, in the blue arrows, match equations equations (3.23a) and (3.23b), respectively. The remaining numbers, 7 and 8, in the green arrows correspond to the Esirkepov current deposition and the interpolation of the electromagnetic field.

analysed the limitations and range of validity of this methods, which are consistent with the ranges of validity discussed in the previous chapter.

In the following chapter we will discuss how to adopt this numerical model implemented as a parallel algorithm capable of running in either GPUs and CPUs.

Chapter 4

Implementation

The two previous chapters discussed the physical and numerical models used to describe the dynamical processes in plasmas and neutral atomic gases.

The current chapter summarizes the most important aspects associated with the development of the simulation code, including the most relevant technical features. The outcome of this chapter can be viewed as the result of a software engineering project that balances constraints, objectives, and technological solutions.

Section 4.1 presents a brief review of the current GPU computing technologies and software resources used in the development of this code. Section 4.2 addresses the implementation of the code, by providing details of the different classes and libraries behind its modular structure, as well as, how they are integrated to function together. Finally, section 4.3 discusses the tests used to validate the solver and to determine the performance increase of GPUs relative to CPUs.

4.1 GPGPU computing

The use of GPUs to perform numerical calculations represented a revolution in scientific computing. On one hand, it allowed to run on a single piece of hardware simulations that previously required computer clusters or supercomputers, while reducing the communication latency in the calculations. On the other hand, it reduced the cost of computing power, thus allowing anyone with a few thousand euros to have a supercomputing system on a desktop. This revolution, however, also implied a change in the way we program simulation codes.

For many years the field of computational physics was mostly dominated by distributed memory systems, involving several computers or nodes linked in a network. With the appearance of multi-core processors, numerical simulations evolved into hybrid parallel computing schemes in order to take advantage of the new technology. The landscape of scientific computing changed again with the introduction of GPGPUs[46]. This type of processor is based on the Single Instruction Multiple Data (SIMD) architecture, as identified by Flynn's taxonomy[47], and has arrays of processors that can execute an instruction on several data entries at the same time. Compared to other parallel computing technologies, such as multi-core CPUs and clusters of single or multi-core CPUs, the GPU not only packs more computing power onto a single chip, but it also allows easier integration of several such devices into a single machine, due to the master-slave relation with the CPU.

When compared to CPUs, which use the Multiple Instructions Multiple Data (MIMD) paradigm, GPUs do not have as many control units, which allows the inclusion of more Arithmetic Logic Units (ALUs) in their place (see figures 4.1 to 4.3). Put simply, the GPU dedicates more transistors to do calculations than the CPU (see figure 4.3). This type of processor usually has specialized memories, such as constant memory, which performs a single access when several processors request the same memory location, and *texture* memory which accelerates accesses based on the proximity of the processors and of the data[48–50]. The devices inside a single machine already work in a distributed memory scheme but intercommunication can be faster and easier to program, depending on the particular device and API being used. A second level shared memory parallelism can still be added using multiple machines.

The two main APIs available are NVidia's CUDA, and OpenCL. The first is proprietary and works only on NVidia GPUs, while the other is open source and works on most GPUs and CPUs. CUDA was launched in 2007 and is often the choice for scientific computing due to its better support and efficient libraries of numerical methods such as linear algebra, FFTs, machine learning, and others. OpenCL primes for its portability but is far more prone to errors.

Programming on either of these APIs involves writing kernel[1] code in C/C++ and

[1]The runtime used by CUDA and other APIs for GPU computing usually rely on master-slave structure where host (the CPU) runs a "master" program that controls the execution of "slave" programs on the different devices which perform the majority of the calculations. The "slave" programs that run on the the device are commonly called *kernels*.

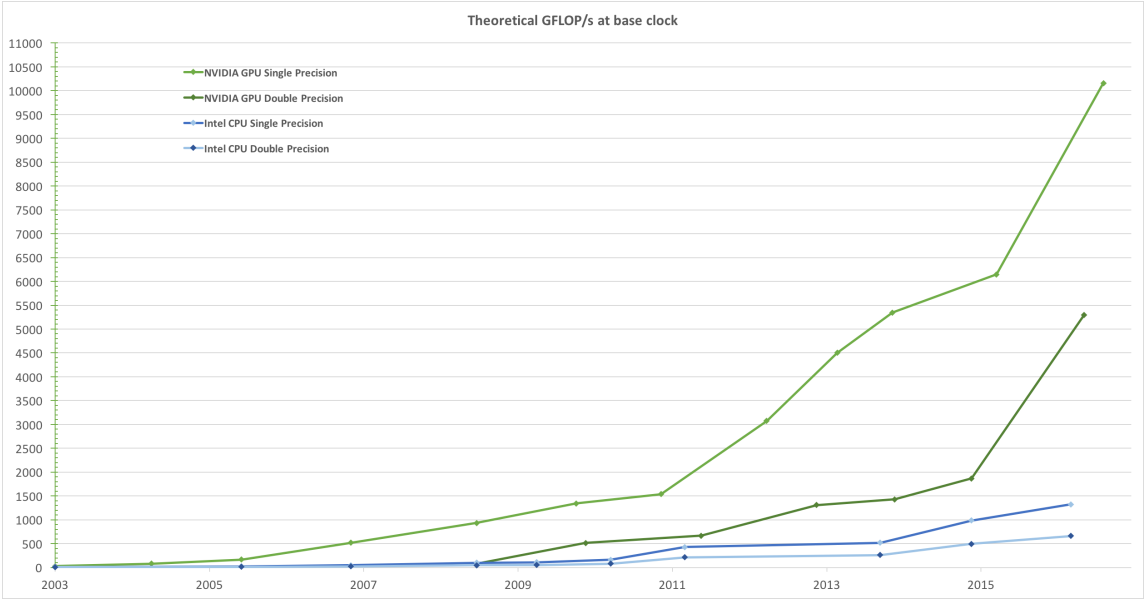


FIGURE 4.1: Comparison between the theoretical performances of CPUs and GPUs in FLOPS. We can see that the GPUs outperform CPUs, especially for single precision floating point numbers. *Adapted from CUDA Toolkit Documentation v8.0.*

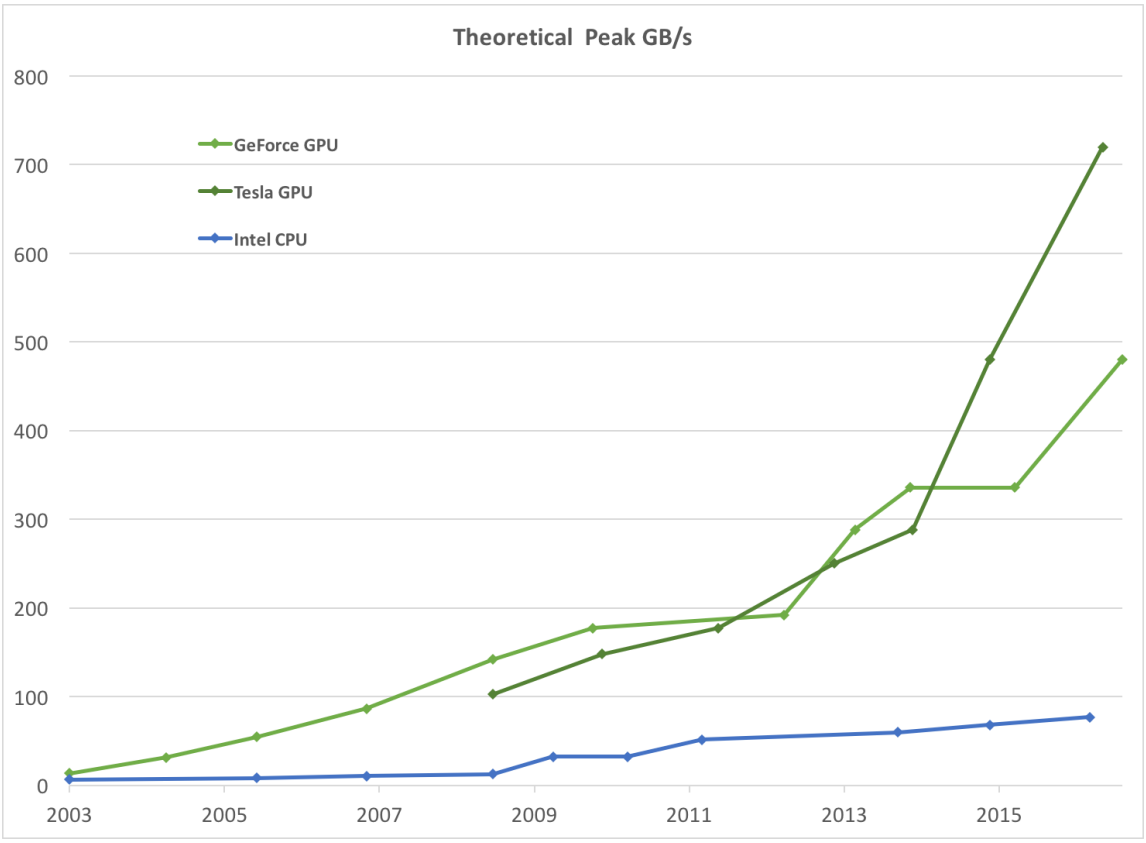


FIGURE 4.2: Comparison of the memory bandwidths of CPUs and GPUs. The GPUs have an higher memory bandwidth, which is of great importance in scientific computing. *Adapted from CUDA Toolkit Documentation v8.0.*

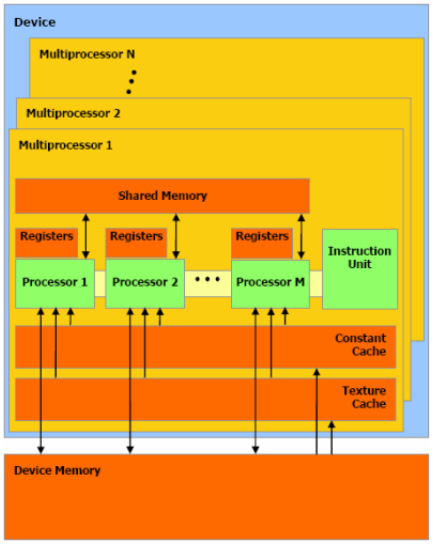


FIGURE 4.3: The CUDA architecture. We can see that a single instruction unit controls processors. The reduction of the amount of control units permits a larger percentage of transistors dedicated to data processing. *Adapted from CUDA Toolkit Documentation v8.0.*

handling raw memory arrays. To expedite the development of our solver while maintaining performance, we used the *ArrayFire* library [51]. The abstraction provided by this framework allows writing code for a single GPU that is modular and whose components can be easily integrated with each other to make many distinct simulations. Also, it can run on the CPU, allowing for easy comparison of performance between the two platforms.

4.2 Implementation model

Although the numerical model and algorithms presented in chapter 3 are parallelizable, they cannot be implemented directly in GPU without adaptations to the specific characteristics and runtime model of these devices. These adaptations must balance performance, memory requirements, and other tradeoffs which will depend on simulation parameters, such as the number of particles and Yee cells. Furthermore, the code must have a modular structure and be able to interact with other simulation models associated with other physical processes, which lead to the adoption of the library *ArrayFire*. The combination of these constraints implies that the numerical model and the implementation scheme have important differences in structure and organization.

In particular, the SIMD architecture of the GPU forces us to think of programs differently. For example, the code must be written to take into account the synchronization of

the calculations at each time step and how these calculations are distributed and organized in the hardware.

To address these issues, the code is built under the imperative and object oriented programming paradigms. The C++ classes were mostly used to provide abstraction and automatic error checking to an otherwise imperative code. The classes and their interdependencies are presented in figure 4.4. The classes in grey are auxiliary and shown only for completeness. The arrows show the interdependence of the classes, each requiring an instance of the classes that point to it. The main classes in the figure are:

- `Units` - this class defines the units system in which the simulation is working, The system of units can be defined by giving a variable representing one of the available natural units systems or by giving the value of the units of length, mass, time, temperature, and charge in SI. This class also contains the values of the fundamental physical constants in the chosen units and in SI, the latter being accessible from the class without an instance, since they are static member functions[2].
- `YeeMesh` - this class defines an Yee mesh from values N_x, N_y, N_z, Δ_x, Δ_y, Δ_z, Δ_t, and an instance of `Units`. Additionally, it can receive an `af::dtype` value to choose floating point precision. This class provides the methods for calculating the sampling locations for fields on the faces, edges and centre of the mesh.
- `Field` - this class is the basic unit of most calculations in the code. It can either represent a scalar field sampled at the centre, or a vector field sampled at the centre, faces, or edges of the Yee mesh. This class includes overloads of most operations for abstraction, as well as, runtime error catching for unavailable operations such as adding a centre field with an edge field. Taking this example, there is also the `Field.as` method allowing the interpolation between the different types of vector fields.
- `EMF` - this class takes an `YeeMesh` instance and generates the necessary fields to simulate the electromagnetic field. Boundary conditions can be added and the evolution of the electromagnetic field is achieved by calling the method `YeeMesh.push` that can take source terms of current density, polarization, and magnetization. This class also checks the Courant condition from the characteristics of the mesh.

[2]This means that the functions or members within the class can be called outside the class without first creating an instance of it.

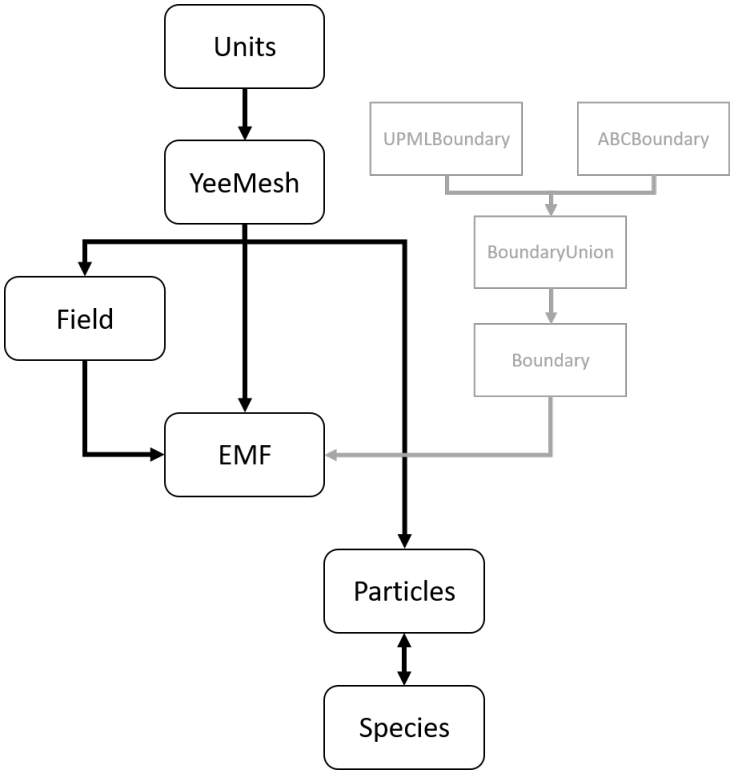


FIGURE 4.4: Class structure of our code. The arrows represent the interdependencies of the classes, with each class requiring an instance of the classes that point to it. The structure in grey is included for completeness but contains only some auxiliary classes for the boundary conditions of the FDTD method.

- `Particles` - this class acts mostly as container for several species of particles, it requires an `EMF` instance and a list of `Species` instances. It also acts as a shortcut for advancing all the species in time from a single command.
- `Species` - this class contains the mass, charge, and *weight* w_p of a species of super-particles, as well as, arrays containing the positions, previous positions and velocities of the super-particles. Each `Species` requires a reference to a `Particles` instance, ultimately having access to the `Units` class. This class also contains member functions for the position and momentum pushes (Vay pusher).

The `Field` and `Species` classes are the only two to handle numeric arrays directly. Together with several auxiliary methods such as

- `div` - for calculating the divergent of `Fields`,
- `curl` - for calculating the curl of `Fields`,
- `grad` - for calculating the gradient of `Fields`,
- `cross` - for calculating the cross product between two vector `Fields`,

they constitute the core of the numerical calculations and would have to be adapted to extend the code to different hardware configurations. The abstraction provided by the object oriented programming would allow the remainder of the code to be unchanged.

Sections 4.2.1 to 4.2.3 detail the various adaptations from the numerical methods to the program, made respectively for the FDTD method, the Vay pusher, and the interpolation and deposition of particle and field quantities.

4.2.1 FDTD method

Within the class `EMF`, the simple electromagnetic push is achieved by applying the auxiliary function `curl` which receives a source `Field` and a target `FieldType` and calculates the curl with first order finite-differences in PBC. Since the `EMF` automatically initializes **D** and **E** as face fields, and **B** and **H** as edge fields, the code

```
Dfield += dt * curl(Hfield, FieldType::FACE) - Jfield;
Bfield -= dt * curl(Efield, FieldType::EDGE);
```

summarizes equations (3.1a) to (3.1c) and (3.2a) to (3.2c).

Initializing the simulation of the electromagnetic field is achieved by calling the method `initialize` from `EMF`, and providing the charge density, current density, polarization and magnetization fields at time $t = 0$. This will generate the electric and magnetic fields according to equations (3.5) and (3.6) and advance the magnetic fields to time $t = \frac{\Delta_t}{2}$.

For the boundary conditions, the method `addBoundaryCondition` from `EMF` must be called prior to initialization. In the case of UPMLs, equations (3.10) to (3.13) are used instead of the regular push, but equations (3.14) and (3.15) only apply to the faces of the simulation domain chosen by the user. Furthermore, since the UPMLs require 6 scalar fields to be known at 6 different positions in the Yee cell (equating to 36 scalar fields), we compute the values of κ and σ at each step instead of saving them in memory. The MABC, on the other hand, are applied by simply multiplying the fields after the push. These Boundary conditions only require a single scalar field to be known at 6 locations in the cell and so we chose to calculate them once and save it in memory. It is important to note that the `EMF` class automatically guarantees that two types of boundary conditions are not placed on the same face.

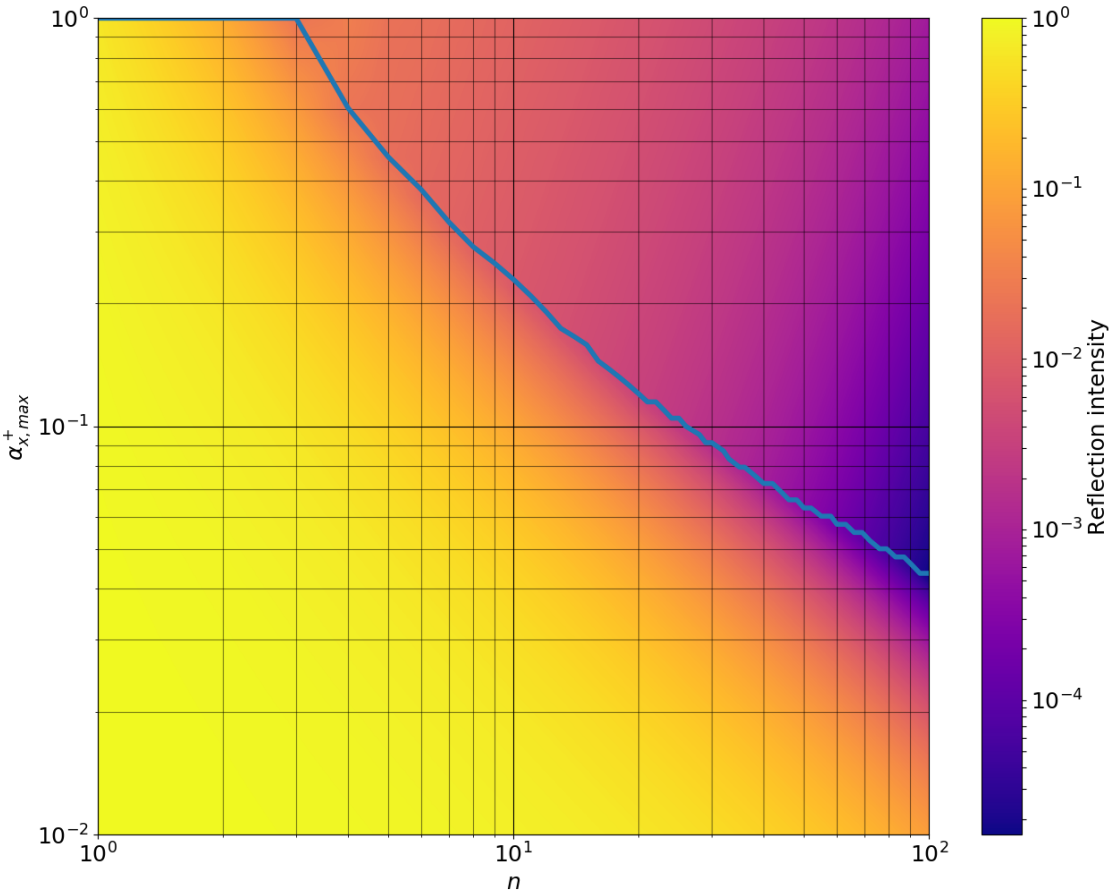
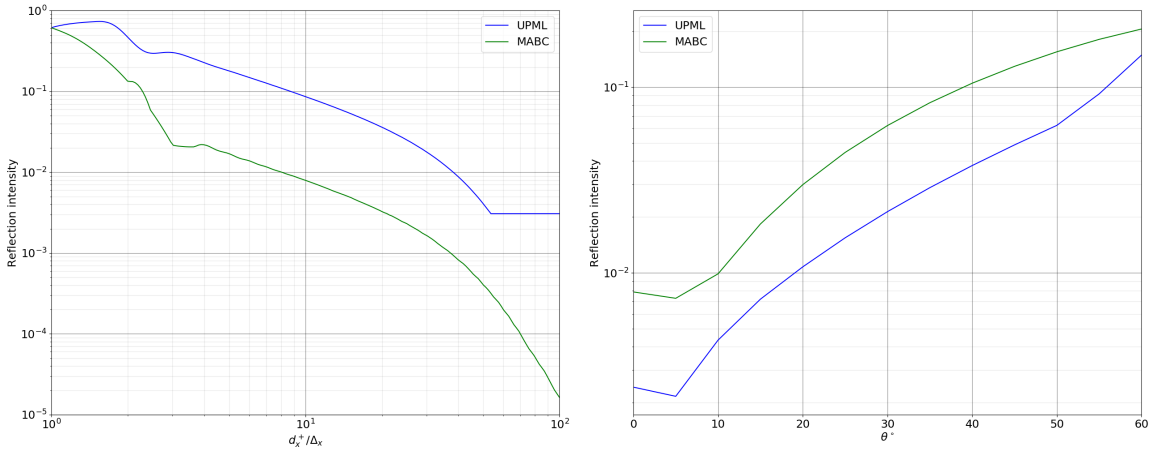


FIGURE 4.5: Characterization of the attenuation of the MABC as a function of d_x^+ and $\alpha_{x,max}^+$, the latter expressed more conveniently in number of cells $n = \frac{d_x^+}{\Delta_x}$. The blue line matches the value of $\alpha_{x,max}^+$ that minimizes the reflection for each n

Figure 4.5 shows the characterization of the MABC. A plane wave was directed perpendicularly at the boundary for different values of d_x^+ and $\alpha_{x,max}^+$ in equation (3.16). Depending on how much attenuation we want for an incident wave, this plot allows us to chose the appropriate values. As said before, these boundary conditions performed surprisingly well and figure 4.6 shows the comparison between them and the UPMLs for different boundary thickness and angles of incidence of the wave. We observed that the MABC actually outperformed the UPMLs for a boundary of the same thickness with optimal parameters for each. Although this result is surprising and the UPMLs did not perform as reported in the literature[21] we did not find any inconsistencies in our implementation.

An important tool in FDTD left unmentioned in chapter 3 is the sources of electromagnetic field. The `EMF` class includes the method `addExternalCurrent` which allows adding external sources of current as defined by three functions (one for each component) that



(A) (B)

FIGURE 4.6: Comparison between the attenuation of the UPML boundary conditions and MABC as a function of (A) the thickness of the boundary, and (B) the angle of incidence of the wave.

depend on space and time. This is also the first example of the small scale usage of functional programming in our code.

4.2.2 Particle push

The particles are organized into different species each with a specific charge and mass. This reduces the memory consumption of saving these constants for each particle and matches the usual situations in a PIC code where many particles of the same type are used. The other characteristic of the super-particles is the *weight*. For simplicity, we also chose to make the weight a property of the species, although some PIC codes make this a property of each super-particle within the species[41]. This is beneficial in highly heterogeneous initial particle distribution.

With these definitions, the Vay pusher is applied directly, according to the equations in section 3.4.2.2, since all particles are independent of each other. The `Particles` class has two methods to apply the Vay pusher: `pushPositions` and `pushVelocities`. This is necessary since these two steps are interceded by the FDTD push, as seen in figure 3.5.

The boundary conditions for the particles are applied automatically after the velocities are pushed forward in time. For the periodic boundary conditions the application is direct. For the ABC, the *ArrayFire* function `where` is used to locate the indices of the particles that are inside the simulation domain, and the array indexing is used to create the new velocity array.

4.2.3 Particle-grid interaction

There are two situations where the fields and the particles must interact to achieve a PIC code: interpolations of the fields to the positions of the particles and deposition of particle quantities onto the grid as fields. For the interpolation we use equation (3.21) and replace the summation over all cells with a summation only over the cells where the shape factor of order n is nonzero

$$\sum_{i,j,k} \longrightarrow \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} \sum_{k=k_1}^{k_2},$$

where

$$\begin{aligned} i_1 &= X_p - n', \quad i_2 = X_p + n' + 1 \\ j_1 &= Y_p - n', \quad j_2 = Y_p + n' + 1 \\ k_1 &= Z_p - n', \quad k_2 = Z_p + n' + 1, \end{aligned}$$

and

$$X_p = \left\lfloor \frac{x_p}{\Delta_x} \right\rfloor, \quad Y_p = \left\lfloor \frac{y_p}{\Delta_y} \right\rfloor, \quad Z_p = \left\lfloor \frac{z_p}{\Delta_z} \right\rfloor, \quad n' = \left\lfloor \frac{n}{2} \right\rfloor.$$

These equations are applied to every particle p at the same time through the *ArrayFire* arrays.

The other interaction, the deposition, is achieved by applying equation (3.22) or equations (3.27a) to (3.27c). Because several particles may occupy the same space, the equations can not be applied to every particle at the same time due to data writing collision. On the other hand, following those equations directly would involve an iteration through every particle for every Yee cell, when only a small fraction of the particles contributes to each cell.

When there are no collisions, the deposition would be achieved in *ArrayFire* by indexing an array with another array

```
phi[indices] += values;
```

If `values` had only a single value this could be achieved with the `histogram` function. The more general version that we need, essentially a histogram with weighted entries, is not available. In our solver, we used two different solutions. The first uses CUDA

kernels directly and atomic operations to solve the data collision. The atomic operations provided by CUDA ensure that the writing collisions in the data are resolved, but we found several inconveniences. First, atomic operations for double precision floating point numbers are only available for GPUs of computing capability 6.0[3] or higher. For lower values (as is the case for the GPUs we have available) the solution from NVidia, based on `atomicCAS` (atomic Compare And Swap), lowers performance considerably. The second problem appears when using a GPU that is also connected to a display device since for large simulations the video card driver prompts a timeout due to the kernel taking too long to run. Solving this problem is possible by dividing the data into several blocks and running the kernel sequentially for each block, incurring in more loss of performance. Finally, despite the CPU version being far simpler to implement, we were unable to avoid memory leaks when transferring the data from *ArrayFire* to common C++ pointers in the CPU backend, despite following the documentation available. This would keep us from running the program on the CPU during development and for comparing performances between CPUs and GPUs

The second solution is the default mechanism in the code and uses several *ArrayFire* functions:

- `sort(out_keys, out_values, keys, values)` - places the equally rearranged `keys` and `values` in `out_keys` and `out_values` for sorted `keys`;
- `scanByKey(keys, values)` - similar to `scan` where each value of the returned array is the sum of all values up to that index of the input array, this function returns the scan of `values` while the integers in `keys` remain the same. When they change the summation resets;
- `where(input)` - returns an array with the indices of the entries of `input` with boolean value `true`;
- `shift(input, n)` - returns an array with all entries shifted cyclically by `n`.

With these functions, the algorithm is as follows

```
sort(keys\_s, vals\_s, indices, values);
sbk = scanByKey(keys\_s, vals\_s);
idx = where((shift(keys\_s, -1) - keys\_s) != 0);
```

[3]The computing capability is the versioning of the NVidia cards. Even if the API is updated the card will not be able to perform certain operation if they are beyond its computing capability.

```
aux = keys\_s[idx];
out[aux] += sbk[idx];
```

This algorithm is expressed only for 1-dimensional arrays but this is enough for any array by calculating the 1-dimensional indices from the 3-dimensional ones with Column-Major order.

To complete the Esirkepov current deposition method, after completing the calculations in equations (3.27a) to (3.27c), we still need to solve the finite-difference equations (3.26a) to (3.26c). This is done using the `scan` function (in particular, we used the exclusive scan)

$$\texttt{scan} : \{a_0, a_1, a_2, ..., a_i, ..., a_{n-1}\} \rightarrow \left\{0, a_0, a_0 + a_1, ..., \sum_{j=0}^{i-1} a_j, ..., \sum_{j=0}^{n-2} a_j\right\},$$

along the relevant dimension of the array. This means that even if we only have a single particle this calculation is still done over the entire simulation domain. Thus, our implementation is unsuited for small number of particles when compared to the number of Yee cells. However, this method is more efficient if there are several particles per Yee cell on average.

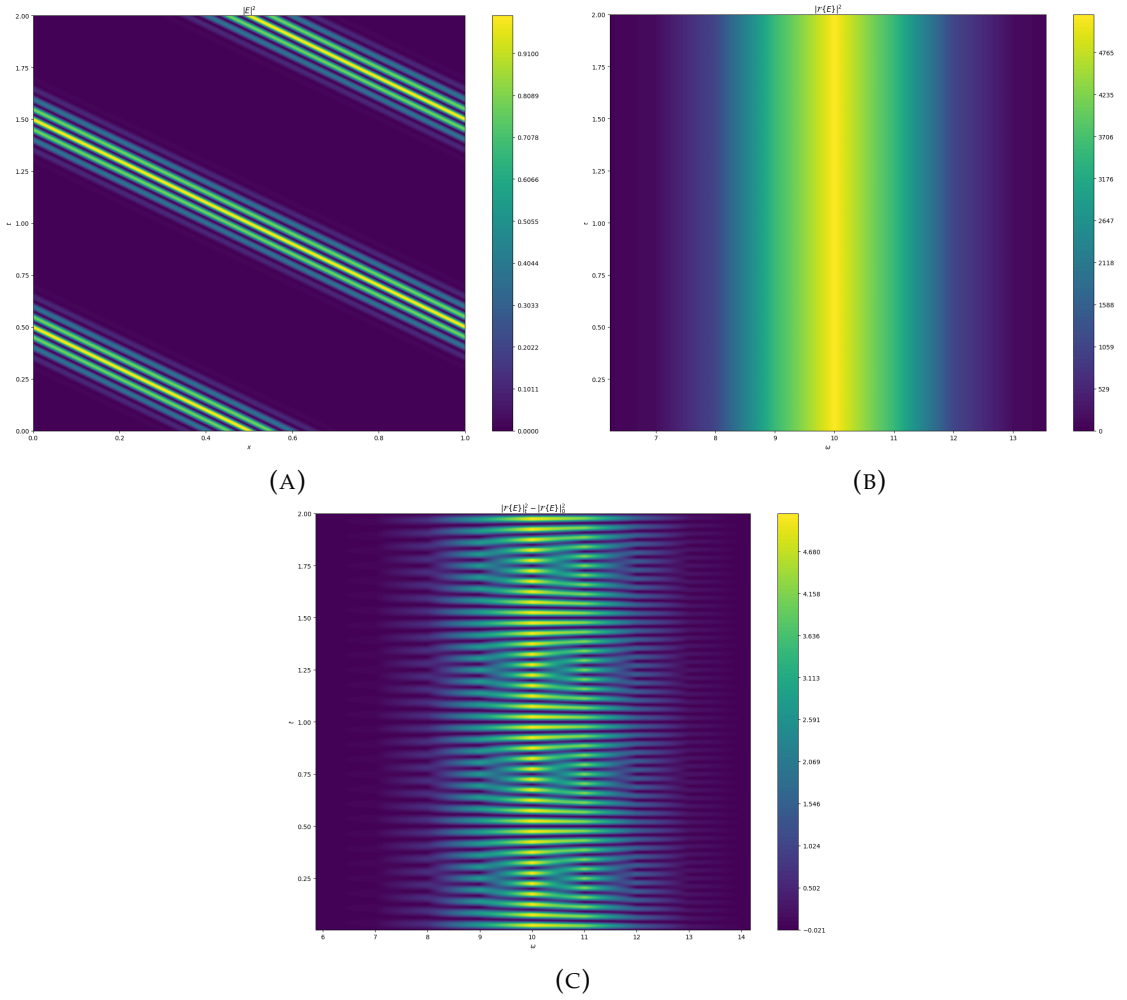
This solution allows us to keep all calculations within *ArrayFire* and outperformed the previous solution in all the devices we could test.

4.3 Testing and performance analysis

4.3.1 Fundamental tests

To test the validity of this software, several tests were performed. The first test concerns the FDTD methods implemented in the `EMF` class. Figure 4.7 shows a 1-dimensional Gaussian pulse propagating and having small oscillations of the spectral content of roughly 10^{-3}, relative to the maximum power of the spectrum. The actual test was done for several thousand passages through the simulation domain but, for clarity, only two are shown. However, the oscillations of the power spectrum remain the same even after long simulation times, proving the stability of the method.

The second test concerns the entire PIC method and checks the validity of the Gauss and charge conservation laws of equations (2.1a) and (3.25), respectively. Moreover, the



(A) (B)

(C)

FIGURE 4.7: Validation test of the FDTD method implemented in the `EMF` class. A Gaussian pulse was propagated in a 1-dimensional simulation domain with periodic boundary conditions. The vertical axis represents time in all plots but the horizontal axis represents the coordinate x in plot (A) and the frequency ω for plots (B) and (C). Figure (A) shows the intensity of the electric field over two passages in the domain, figure (B) shows the evolution of the power spectrum of the field ($|\mathcal{F}\{\mathbf{E}\}|_t^2$) and figure (C) is the variation of the spectrum over time ($|\mathcal{F}\{\mathbf{E}\}|_t^2 - |\mathcal{F}\{\mathbf{E}\}|_0^2$). The variations of the spectrum represent an relative error of roughly 10^{-3}.

test also evidences the improvement made by using the Esirkepov deposition method discussed in section 3.4.3 over the "naive" method of equation (3.22). Figure 4.8 shows the validity of the two laws over the simulation time using the two current deposition methods. We can see that the Esirkepov method (or other charge conserving methods) is fundamental to get physically valid results in the interaction between the field and the particles. The initially larger error is due to the normalization used on the calculation of the errors. Since a *quiet start* was used, the initial $\max\{|\rho|\}$ and $\max\{|\dot{\rho}|\}$ are very small and the error measure becomes irrealistically high.

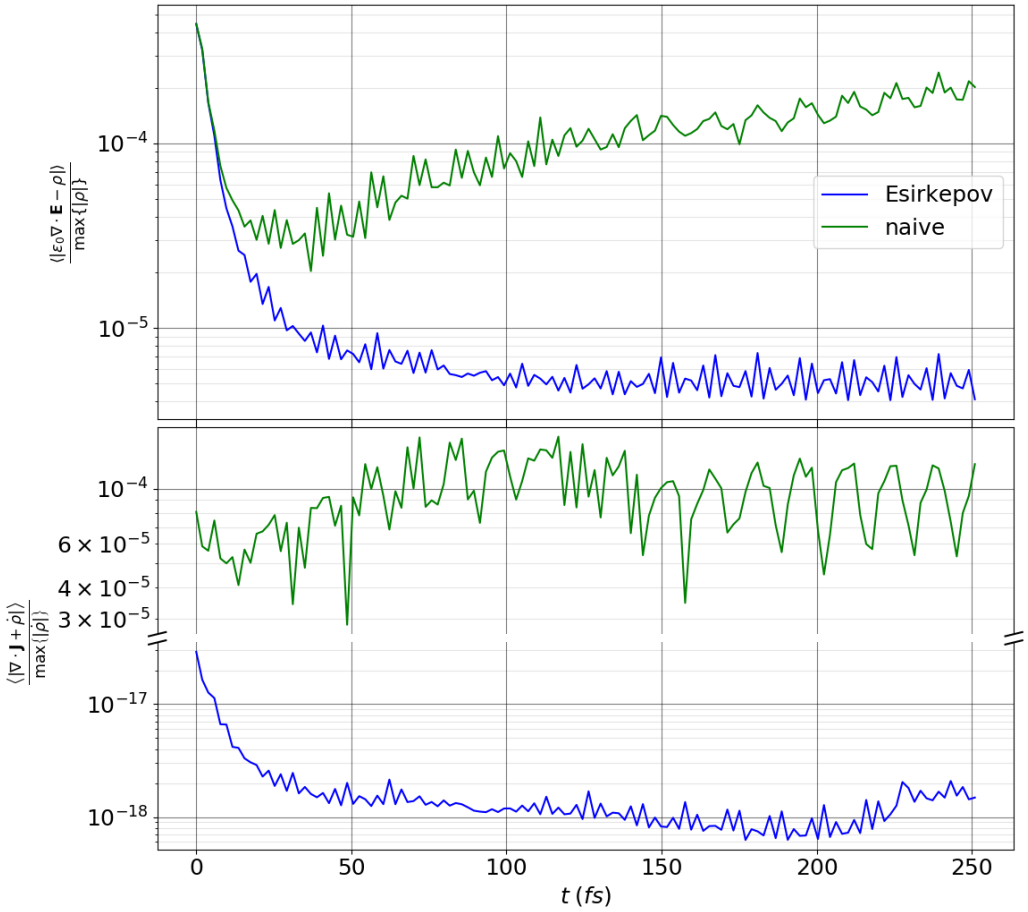


FIGURE 4.8: Validity of the Gauss and charge conservation laws of equations (2.1a) and (3.25) over the simulation time. The top and bottom graphs corresponds to the Gauss and charge conservation laws respectively. The errors displayed were calculated by $\frac{\langle|\varepsilon_0\nabla\cdot\mathbf{E}-\rho|\rangle}{\max\{|\rho|\}}$ and $\frac{\langle|\nabla\cdot\mathbf{J}+\dot{\rho}|\rangle}{\max\{|\dot{\rho}|\}}$, respectively. From these plots we conclude that the Esirkepov method is far better at maintaining good agreement with the Gauss and charge conservation laws.

4.3.2 Computing environment

To test our code and run the simulations we used the following three system setups:

- A) Intel core i7-4930K, NVidia GeForce GTX Titan, 64 GB DDR3;
- B) Intel core i7-4790K, NVidia GeForce GTX 970, 16 GB DDR3;
- C) Intel core i7-6500U, NVidia GeForce 920M, 8 GB DDR3.

They correspond to an high-end desktop, a desktop, and a laptop, respectively. With these three system we were able to compare the impact on performance of GPU computing for very different systems.

4.3.3 Performance analysis

Measuring the performance of our code is fundamental to understand the distribution of computing power among the components of the algorithm, for measuring the change in performance of different computing devices, and for making predictions on how much time a particular simulation will take.

In this section, we separate the code into its main components: the FDTD method, the Vay pusher, the interpolation method, and the Esirkepov current deposition method; and analyse their performance individually. The results are displayed in figure 4.9. In general, we used a fixed Yee mesh of $100 \times 100 \times 100$ cells when we wanted to study the effect of varying the number of particles.

The particles are obviously the most demanding component of the algorithm, with the Esirkepov method being the most time consuming, occupying over 95% of the overall time for a standard simulation of $N = p = 10^6$ on the GTX Titan. This method also presents a high base time for small p due to the integration of equations (3.26a) to (3.26c) as already discussed in section 4.2.3. For the standard simulation, figure 4.10 shows the speedup metric comparing GPUs and GPUs. The results show that even for the low-end hardware the speedup is over 15, reaching a maximum of 108 for the high-end desktop. Of course, this comparison is not completely fair since the CPU versions were single-threaded. However, even if we divide the speedup obtained, by the number of cores of the CPUs (6, 4 and 2 for i7-4930K, i7-4790K and i7-6500K) we still have an impressive amount of performance gain. However, the actual performance of a multi-threaded version would not be that high, mainly due to the lower memory bandwidth of the CPUs when compared with the GPUs (see figure 4.2).

4.4 Conclusions

This chapter presented a brief analysis of GPU computing, the different APIs available, and the main concepts that differ from common processors. The details of the implementation of our solver were presented and justified within the context of our objective of modularity, and the GPU technologies available.

Furthermore, the code was validated through a series of physical test such as verifying the Gauss law and the law of conservation of charge. Finally, the performance of the various components of the code was presented for the devices available.

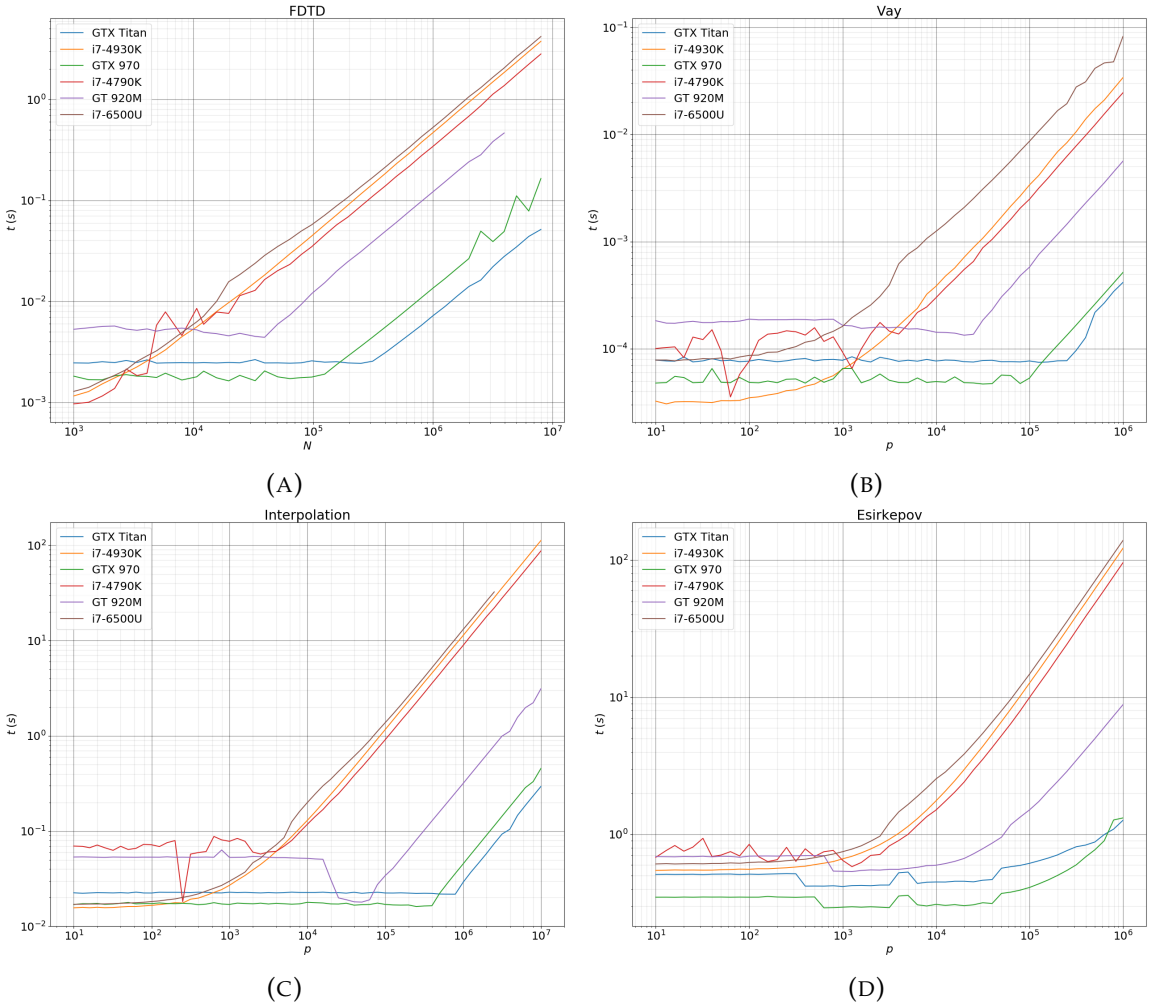


FIGURE 4.9: Performance analysis of the four components of the PIC algorithm for the CPUs and GPUs from system setups from section 4.3.2. Figure (A) features the time per step of the FDTD method as a function of the number of cells in a cubic grid, averaged over 100 repetitions with PBC. Figure (B) shows the time per step of the Vay pusher as a function of the number of particles p averaged over 10000 steps. Figure (C) displays the time taken to interpolate a face vector field of dimensions $100 \times 100 \times 100$ onto the positions of p particles for shape factor order $n = 1$. The results were averaged over 100 repetitions. Finally, figure (D) shows the time taken to deposit the current of p particles onto a $100 \times 100 \times 100$ face vector field with the Esirkepov current deposition method, again with shape factor order $n = 1$.

The next chapter uses the implementation described in this chapter to run three distinct physical test cases that evidence the modularity and plasticity of our code.

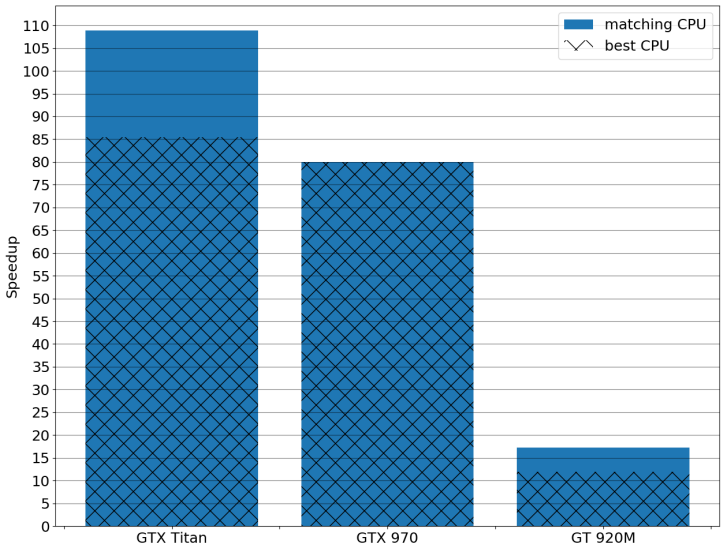


FIGURE 4.10: Speedups for the different setups. The blue bars represent the speedup $S = t_{\text{CPU}}/t_{\text{GPU}}$ compared with the CPU in the same machine, and the cross pattern is the comparison with the best CPU (i7-4790K).

Chapter 5

Physical test cases

This chapter presents three examples that correspond to adaptations of the original simulation software developed during this thesis to model distinct physical problems and systems, which demonstrate the versatility of the code itself and of its modular structure. Indeed, while the first application considers the interaction between an ultra-short laser pulse with a plasma cloud, which corresponds to a situation for which the code was originally designed, the second and third examples required small adaptations for it to simulate the gravitational interaction in a gas cloud, and the interaction between light and a quantum dipolar gas.

In particular, the first application investigates direct electron acceleration produced by an high-intensity ultra-short laser pulse sweeping through a cluster of ionized atoms and electrons, and reproduces the results previously identified in the literature[52,53]. In the second example, the solver of the electromagnetic field was adapted to calculate the solution of the Einstein field equations of General Relativity in the weak field assumption, and simulate the process of mass accretion due to gravitational attraction in a cloud of hydrogen atoms. In the third and final example we coupled our code with a solver of the optical Bloch equations to investigate quantum optomechanical processes, and transport phenomena in dipolar gases.

The examples chosen also allow us to explore the range from a fully classical model, as in the cases of the gravitoelectromagnetism and the plasma, to a hybrid quantum-classical many-body problem, as in the case of the quantum dipolar gas.

Finally, we emphasize that the objective of this chapter is not to explore all the physical implications and results provided by these examples, but instead to demonstrate the versatility, features, and functionalities of our code.

5.1 Laser-plasma interaction

The first example of an application of the simulation software considers the interaction of an ultra-short laser pulse with a cluster of atoms, and the plasma formed after the leading part of the pulse hits the cluster. This system has been already studied in the literature, both experimentally and with simulations, particularly for the production of accelerated particles via laser-based plasma acceleration[7] as an alternative to cyclotrons in the excitation of radioactive isotopes for medical applications. Therefore, the phenomenology of this system is well characterized and constitutes a good test to validate the simulation software.

For simplicity, the simulation scenario considers a 2-dimensional Gaussian distribution of charged particles where the positive (ions) and negative (electrons) charges are perfectly matched at each point of the simulation domain, constituting a so called *quiet start*, as explained in section 3.2.2. An electromagnetic Gaussian pulse is initialized at $t = 0$ on one side of the simulation domain and propagates towards the distribution of charges. The initial pulse is linearly polarized along the plane of the simulation and is defined by

$$E_x\left(x,\, t=0\right) = E_0 \sin\left(\frac{2\pi x}{\lambda}\right) \exp\left(-\frac{4\left(x-x_0\right)^2 \log 2}{2\sigma_{FWHM}^2}\right) \hat{\mathbf{y}}$$

$$B_x\left(x,\, t=0\right) = \frac{1}{c} E_x\left(x,\, t=0\right) \hat{\mathbf{z}},$$

where x_0 is the position of the centre of the pulse, and σ_{FWHM} is the full width at half maximum of the pulse.

For low intensities the main force acting on the charges is the electric field of the pulse. Then, electrons and ions are pulled in opposite directions resulting in some charge separation, but few particles are accelerated forward with the pulse. Instead, they oscillate with the electric field of the pulse as it passes through it (see figure 5.1), and afterwards, due to the electrostatic attraction between electrons and ions (see figures 5.2 and 5.3). In practice, electrons and ions behave as two droplets of a charged fluid oscillating around each other. For this reason, this case is known as the hydrodynamic regime.

Increasing the intensity of the pulse 10 times, a new behaviour emerges that is dominated by radiation pressure. In this case the slope of the intensity profile of the pulse in the lead is too steep for the particles to begin to oscillate with the field. Instead, the electrons (which are lighter) are ripped from the cluster by the radiation pressure and move

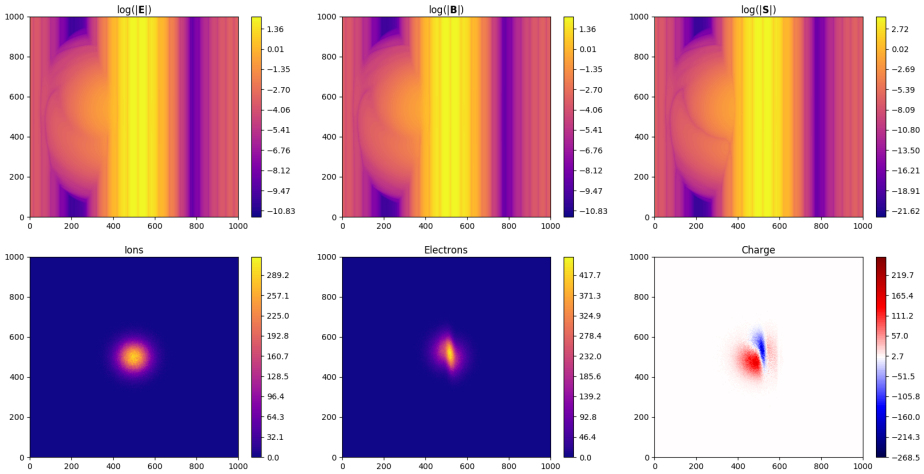


FIGURE 5.1: State of the simulation with the pulse in its first pass through the plasma, in the hydrodynamic regime (u. arb.).

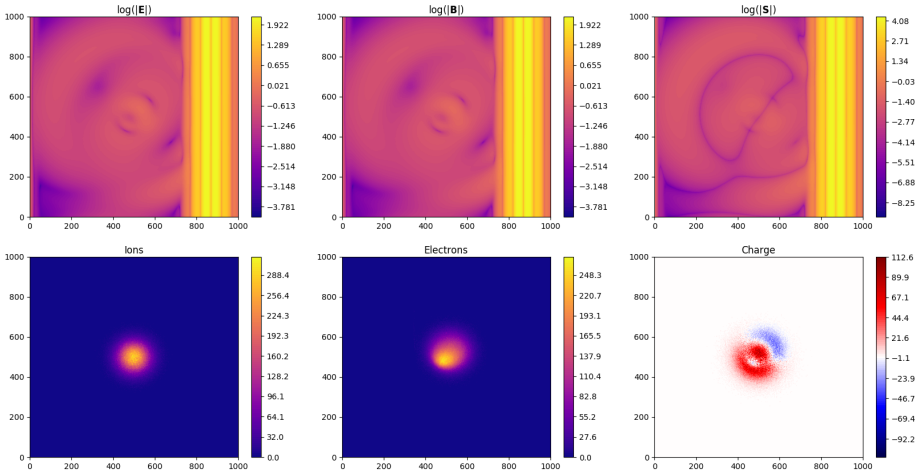


FIGURE 5.2: State of the simulation after the passage of the pulse, in the hydrodynamic regime (u. arb.).

in front of the pulse (see figures 5.4 and 5.5), gaining velocities very close to the speed of light. This is known as the snowplough effect, for obvious reasons.

As the electrons move together with the pulse, they experience a quasi-static electric field in their own reference frame. This causes them to start moving upwards against this field.

In the simulations, this motion is described in the rest frame and the motion of the

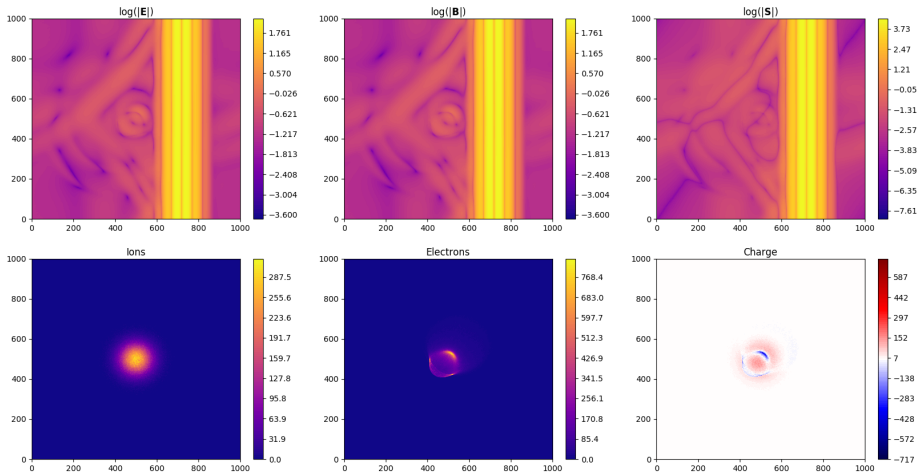


FIGURE 5.3: State of the simulation after the second passage of the pulse, in the hydrodynamic regime (u. arb.).

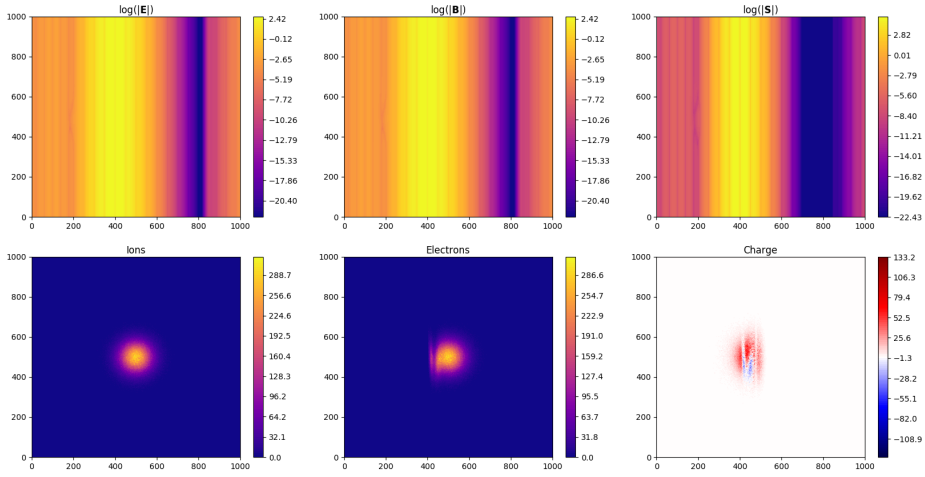


FIGURE 5.4: State of the simulation at the middle of the first passage of the pulse in the snowplough regime (u. arb.).

electrons incorporates both the motion in the direction of the pulse and, to a smaller degree, along the direction of the field, resulting in the small ejection angle observed in figure 5.5. With more intense pulses the radiation pressure should completely overcome the direct effect of the electric field. In figure 5.6, the electrons have fallen out of the faster moving pulse and are attracted back to the cluster of ions, which in turn have not moved significantly due to their higher mass.

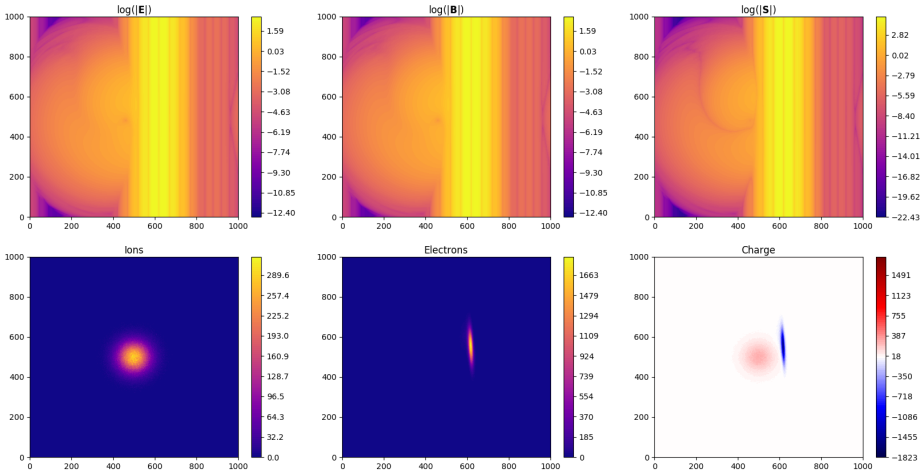


FIGURE 5.5: State of the simulation just after the first passage of the pulse in the snowplough regime (u. arb.).

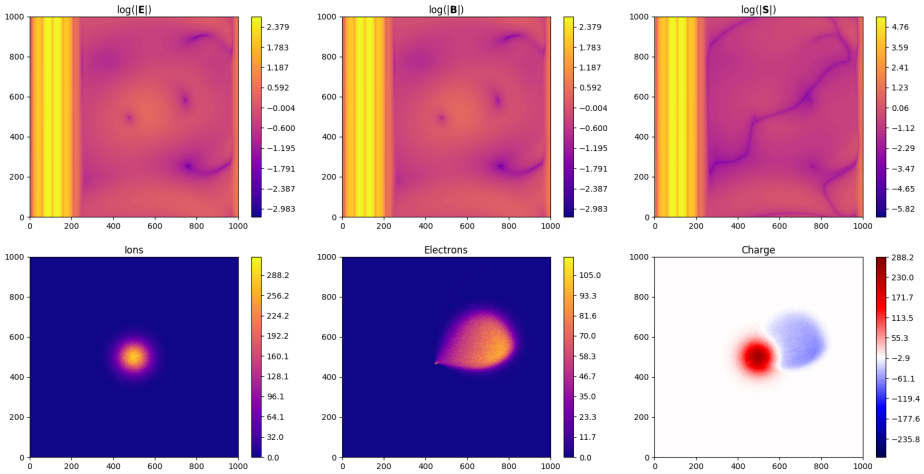


FIGURE 5.6: State of the simulation shortly after the pulse traverses the simulation domain in the snowplough regime (u. arb.).

These qualitative results match the results obtained by other PIC codes[52] and by experimental confirmations[53].

5.2 Gravitoelectromagnetism: gas accretion

For weak fields and slow moving objects, Einstein's field equations of General Relativity admit an approximation that looks similar to the Maxwell equations. Gravitoelectromagnetism, as it is called, started as an extension of Newtons gravitation and was

later deduced from General Relativity in the weak field assumption. Today, Gravitoelectromagnetism admits different formulations, each corresponding to a different type of approximation and having distinct ranges of validity[54–56]. One of such formulations allows to recast the Einstein field equations as

$$\nabla \cdot \mathbf{E}_g = -4\pi G \rho_g \tag{5.2a}$$

$$\nabla \cdot \mathbf{B}_g = 0 \tag{5.2b}$$

$$\dot{\mathbf{B}}_g = -\nabla \times \mathbf{E}_g \tag{5.2c}$$

$$\dot{\mathbf{E}}_g = c^2 \nabla \times \mathbf{B}_g + 4\pi G \mathbf{J}_g \tag{5.2d}$$

where $\mathbf{E}_g$ is the gravitoelectric field, $\mathbf{B}_g$ is the gravitomagnetic field, ρ_g is the mass density, and $\mathbf{J}_g$ is the mass current density.

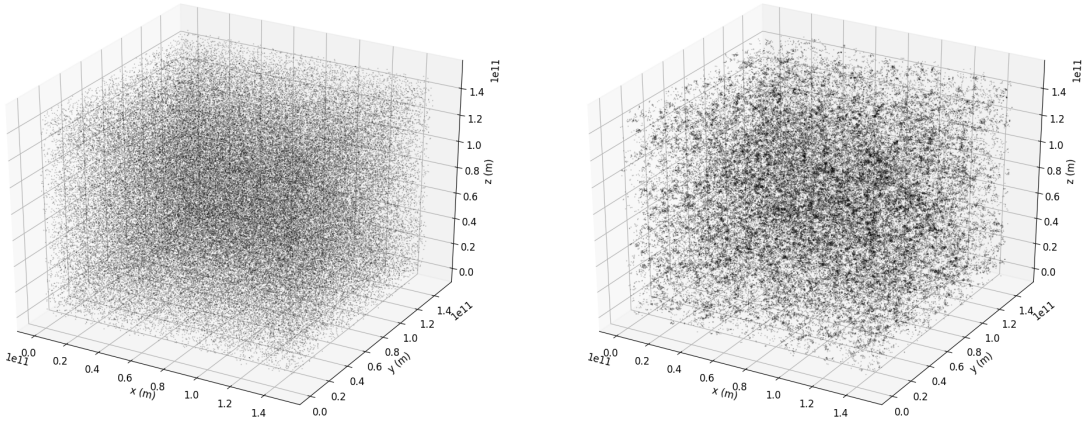
Like in the Einstein field equations, the source of the gravitoelectric and gravitomagnetic fields are the mass density and current density. These fields, in turn, produce forces that act on mass particles, moving them around, and are described by

$$\mathbf{F}_g = m\left(\mathbf{E}_g + 4\mathbf{v} \times \mathbf{B}_g\right), \tag{5.3}$$

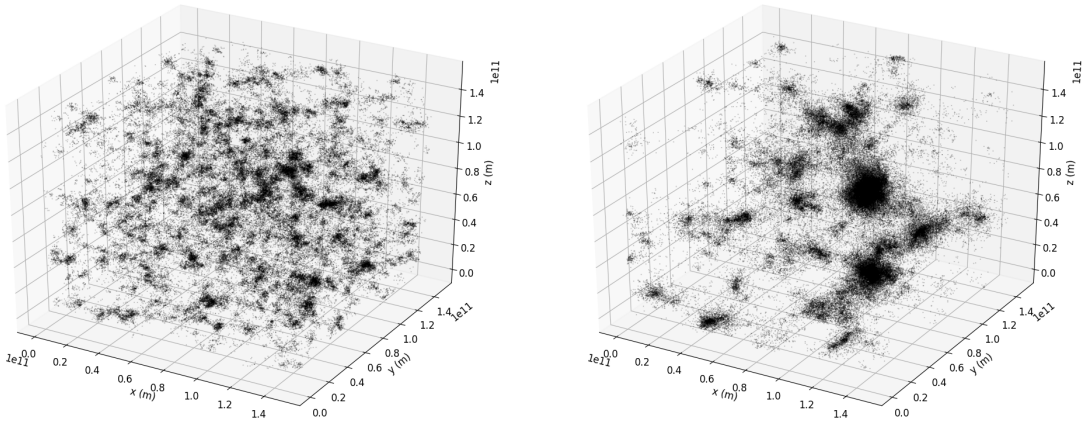
where m is the mass of the particle.

Clearly, equations (5.2a) to (5.2d) and (5.3) present a striking resemblance with equations (2.1a) to (2.1d) and (2.9) used in chapter 2 to model a plasma. As a result, it takes little adaptation to use the PIC code to model a system of particles under gravitational interaction. Figure 5.7 shows several states of a simulation, of a gas of hydrogen atoms in a cubic simulation domain with dimensions 1 ua $\times$ 1 ua $\times$ 1 ua and uniform initial density $n = 10^{31}$, an obvious exaggeration aimed at reducing the computation time. The results of the simulation show the formation of clusters of gas in the early stages. Later, most of the particles join a single large cloud. For more realistic densities this simulation would take much more time due to the time step being constrained by the Courant limit. However, here, we mainly want to show that the code can be easily adapted to model different physical systems.

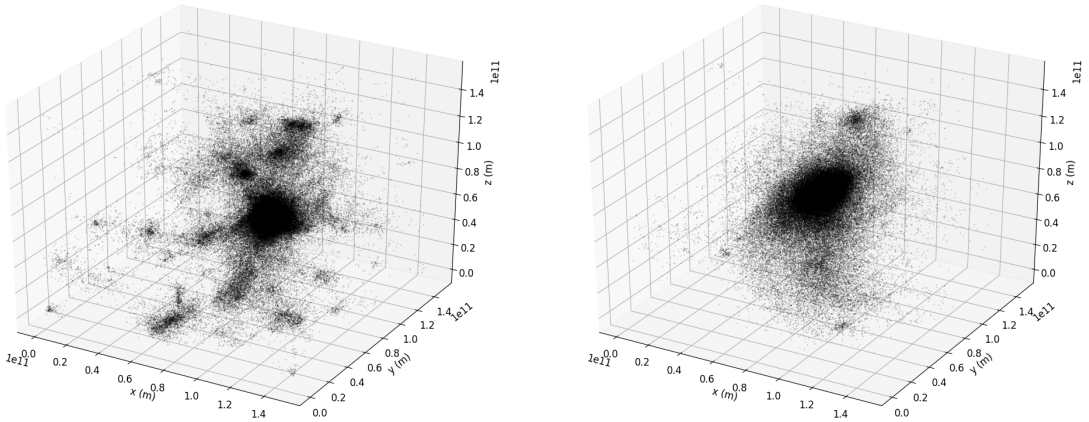
Additionally, since the graphical representation of the 3-dimensional gravitoelectric and gravitomagnetic fields is difficult to interpret, figures 5.8 to 5.11 show the state for a similar simulation at different times, now in 2-dimensional space. The same clustering is observed but now we can also see the appearance of intense gravitoelectric fields around



(A) (B)



(C) (D)



(E) (F)

FIGURE 5.7: Gravitoelectromagnetic simulation of uniform hydrogen gas at times $t = 0\,\mathrm{s}$, $t = 500.35\,\mathrm{s}$, $t = 1000.69\,\mathrm{s}$, $t = 1526.06\,\mathrm{s}$, $t = 1826.26\,\mathrm{s}$, $t = 2151.49\,\mathrm{s}$ for images A, B, C, D, E, and F, respectively. The particles are initialized with zero velocity and uniformly distributed throughout the simulation domain.

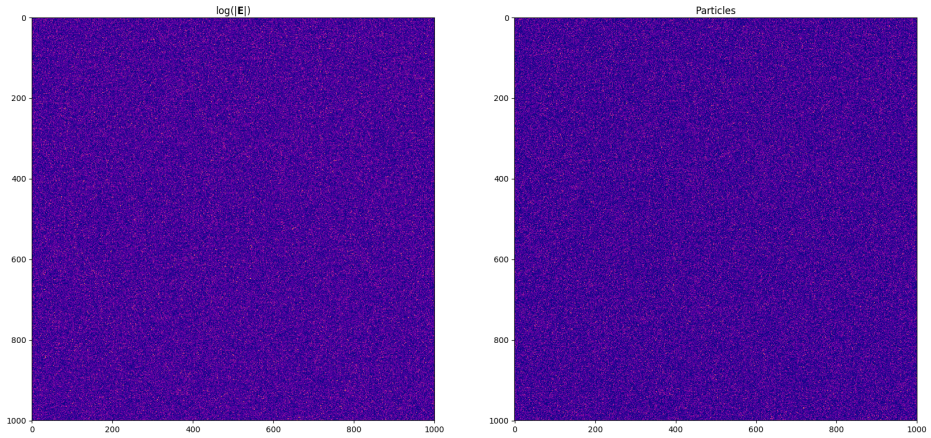


FIGURE 5.8: Initial state of a 2-dimensional Gravitoelectromagnetic simulation. The left tile of each figure shows the magnitude of the electric field in logarithmic scale, while the right tile shows the particle density. The particles are at rest and uniformly distributed throughout the simulation domain (u. arb.).

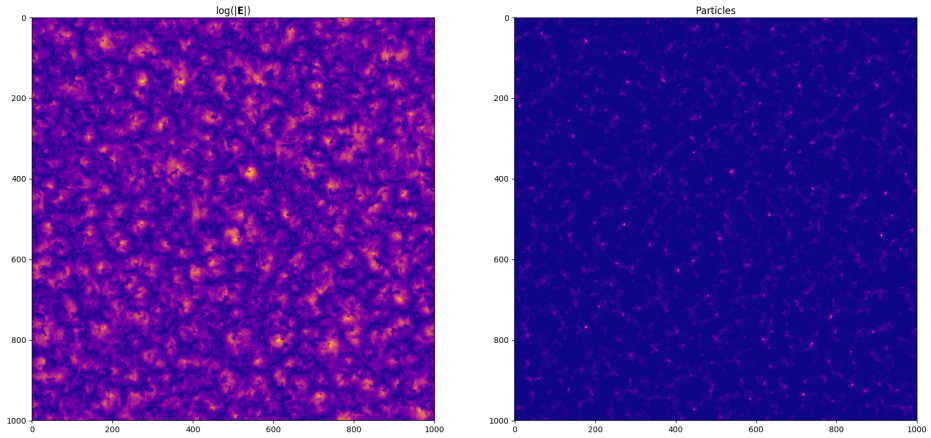


FIGURE 5.9: The simulation figure 5.8 after the formation of many small clusters (u. arb.).

the clusters. In particular, in figure 5.11 we can even see the zero field in the centre of mass of the right upper cluster.

5.3 Transport phenomena in quantum dipolar gases

The next test case deals with neutral gases and includes the internal electronic states of the atoms, as discussed in chapter 1. To describe the internal degrees of freedom and

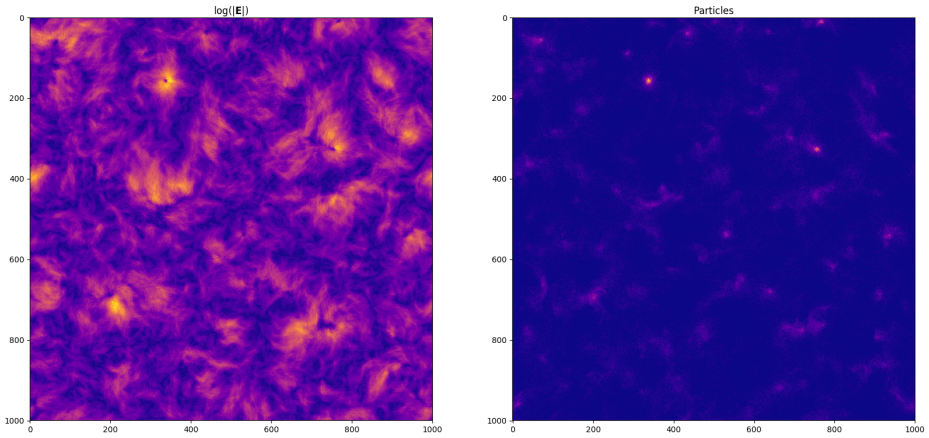


FIGURE 5.10: Halfway point of the simulation of figure 5.8. The clusters are now less numerous and larger (u. arb.).

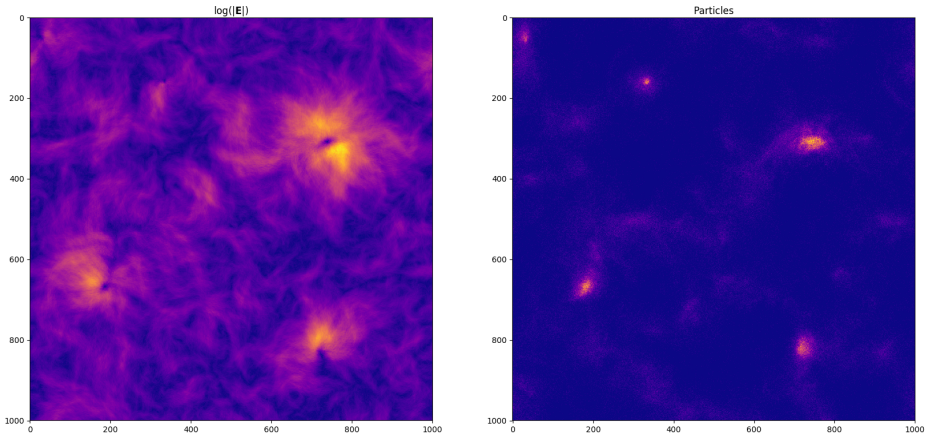


FIGURE 5.11: Final state of the simulation of figure 5.8. Only a few large clusters are observable, and some low density clouds in the intermediate space (u. arb.).

their interaction with the electromagnetic field, we use the common approximation of limiting the state of the atom to a small number of possible electronic states. This model is a good description of the atom when the field is tuned to the energy gaps between the electronic states under consideration, or other transitions are forbidden by selection rules. The quantum model of the electronic state of each atom is given by the Hamiltonian

$$\hat{H} = \frac{\hat{\mathbf{p}}^2}{2m_e} + e\hat{\mathbf{r}} \cdot \mathbf{E}\left(\mathbf{r}_p\right),$$

where $\hat{\mathbf{r}}$ and $\hat{\mathbf{p}}$ are the position and momentum operators of the optical electron. This is only valid when the wavelength of the field is much larger than the size of the atom (dipolar approximation). The state of a large collection of atoms is then described in terms of the density matrix ρ_{ij}, which evolves according to the Bloch equations[57,58]

$$\dot{\rho}_{ij} = \sum_{ij} \left(\mathrm{i}\Omega_{ij}\rho_{ij} + \mathrm{i}\omega_{ij}\rho_{ij} + \Gamma_{ij}\rho_{ij} \right),$$

where $i, j \in \{1, 2, ..., n\}$ and ω_{ij} and Γ_{ij} correspond to the transition frequencies and decay rates between states i and j, respectively. The quantities Ω_{ij} are the instantaneous Rabi frequencies defined as

$$\Omega_{ij} = \frac{\boldsymbol{\mu}_{ij} \cdot \mathbf{E}}{\hbar},$$

where the $\boldsymbol{\mu}_{ij} = e \langle i | \hat{\mathbf{r}} | j \rangle$ represent the dipole moment between states i and j.

The vector $\boldsymbol{\mu}_{ij}$ depends on the orientation of the atom in space that can be affected by te field. This reorientation can be modelled using the unit vector $\mathbf{e}$ as $\boldsymbol{\mu}_{ij} = \mu_{ij}\mathbf{e}$ and adapting the Landau-Lifshitz equation[59]

$$\dot{\mathbf{e}} = \alpha \left(\mathbf{e} \times \mathbf{E} \right) - \beta \left(\mathbf{e} \times \left(\mathbf{e} \times \mathbf{E} \right) \right),$$

where the first and second terms correspond to the precession and damping, respectively. The parameters α and β are obtained empirically, while the μ_{ij} can be measured by spectroscopic techniques.

The field also produces a dipolar force given by[60]

$$\mathbf{F} = \langle \hat{\boldsymbol{\mu}} \cdot \nabla \mathbf{E} \left(\hat{\mathbf{r}}_a \right) \rangle = -\nabla \left(\sum_{ij} \rho_{ij} \left(\boldsymbol{\mu}_{ij} \cdot \mathbf{E} \left(\mathbf{r} \right) \right) \right),$$

which affects the motion of the atoms. The atoms may also affect the field by producing a local polarization field

$$\mathbf{P} \left(\mathbf{r} \right) = \eta \left(\mathbf{r} \right) \sum_{ij} \boldsymbol{\mu}_{ij} \rho_{ij},$$

where $\eta \left(\mathbf{r} \right)$ is the particle density.

The Bloch and Landau-Lifshitz equations are treated numerically using a second order Runge-Kutta method. The `Bloch` class that handles this physical system was developed by my colleague João Costa. The modular construction of the entire code allows the `Bloch`, `EMF`, and `Particles` classes to interact with each other seamlessly. In the simulation, a gas of 2-level atoms is uniformly distributed on a 1-dimensional domain of 1 µm. Two counter-propagating plane waves are then placed to form a stationary wave

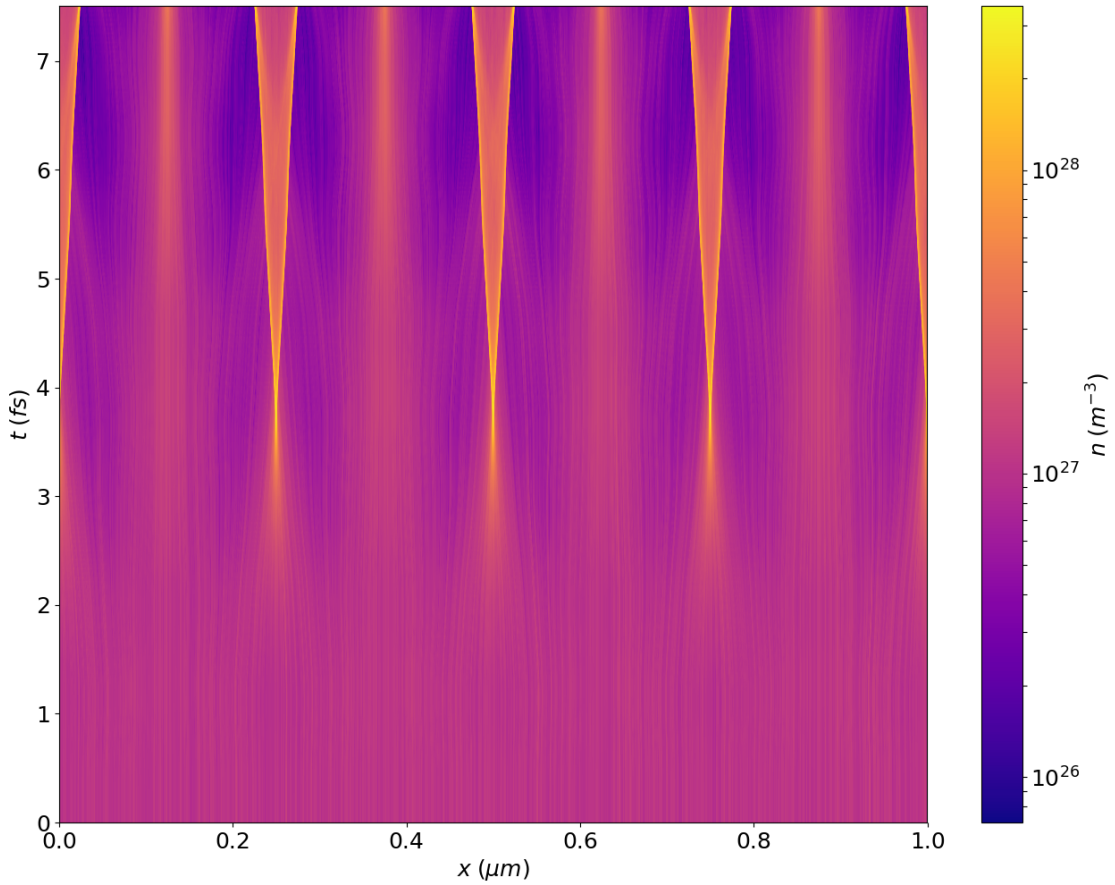


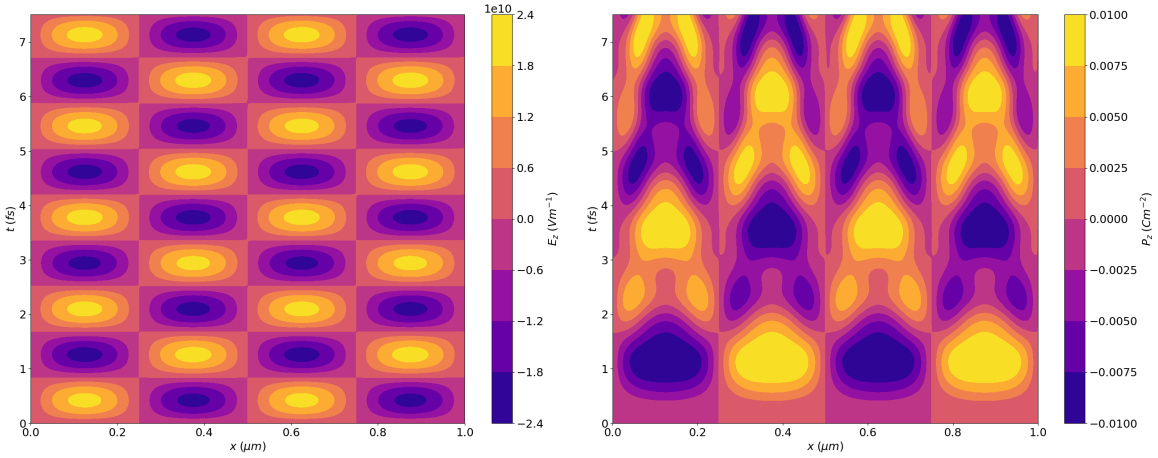
FIGURE 5.12: Temporal evolution of the positions of the particles for the case where $\omega_{12} = 0.5\frac{2\pi c}{\lambda}$. the particles can be seen drifting towards the maxima of the standing wave.

of $\lambda = 500\,\text{nm}$. The transition frequency ω_{12} of the particles was tested for two different conditions $\omega_{12} = 0.5\frac{2\pi c}{\lambda}$ and $\omega_{12} = 1.5\frac{2\pi c}{\lambda}$, corresponding to figures 5.12 and 5.14, respectively. Furthermore, figures 5.13 and 5.15 represent the evolution of the $\hat{\mathbf{z}}$ component of the electric and polarization fields for the two simulations, respectively.

We can see on the figures that the particles gather at the maxima of the stationary waves, or the nodes. For the case where $\omega_{12} = 0.5\frac{2\pi c}{\lambda}$, the field is blue-shifted in relation to the transition frequency, and the atoms flock to the nodes of the stationary wave, while for the case where $\omega_{12} = 1.5\frac{2\pi c}{\lambda}$, the field is red-shifted and the particles gather at the maxima of the wave. This is in accordance with the existing literature[16,17].

5.4 Conclusions

This chapter illustrated the versatility of our simulation software. The first example considered the interaction between an ultra-short laser pulse and a plasma cloud, where it



(A) (B)

FIGURE 5.13: Temporal evolution of the $\hat{\mathbf{z}}$ component of the electric (A) and polarization fields (B) for the case where $\omega_{12} = 0.5\frac{2\pi c}{\lambda}$.

was possible to illustrate the transition between two particle acceleration regimes. In the second example, we have shown how it is possible to model gravitational attraction in a gas cloud using this code. Finally, in the third example, we simulated the optical trapping of 2-level atoms in a stationary electromagnetic field, reproducing the dependence on the detuning observed in experiments[16,17].

Naturally, this is but a small sample of the potential applications of this simulations software. By combining it with other modules or by making small adaptations in the existing ones it is possible to go much further.

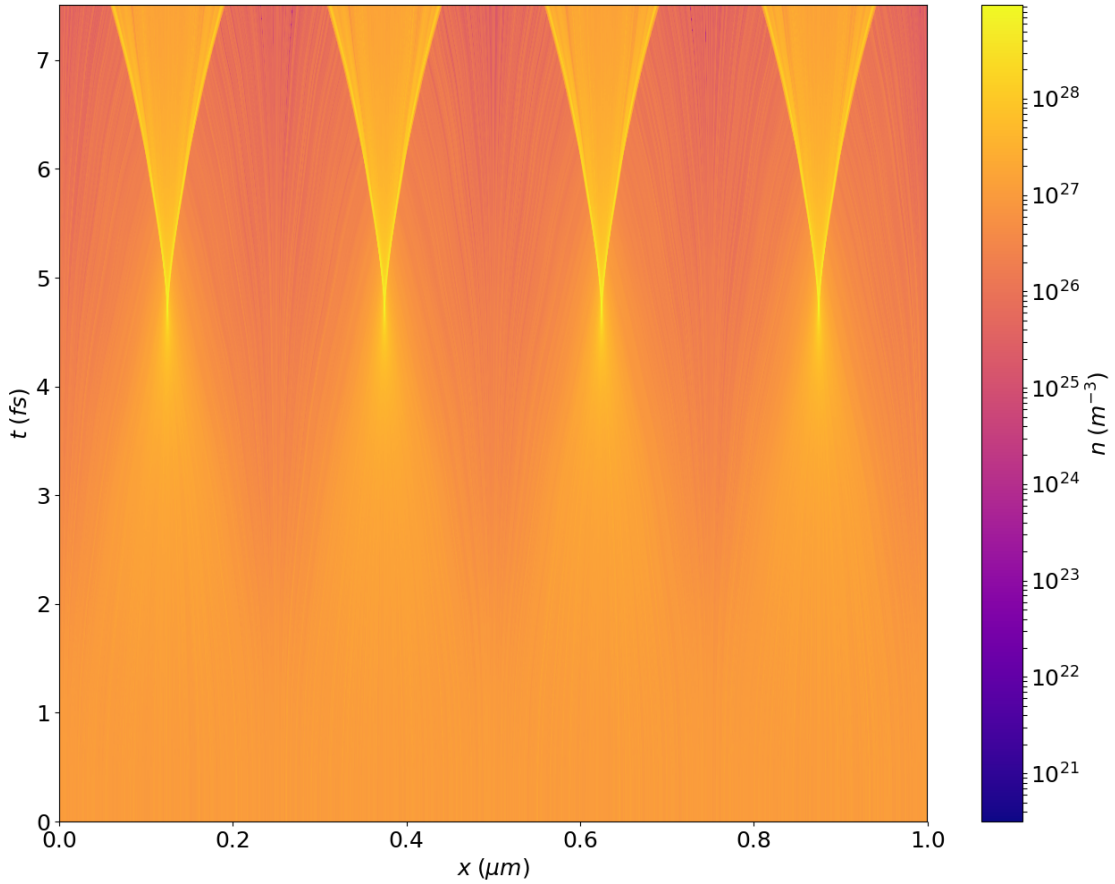
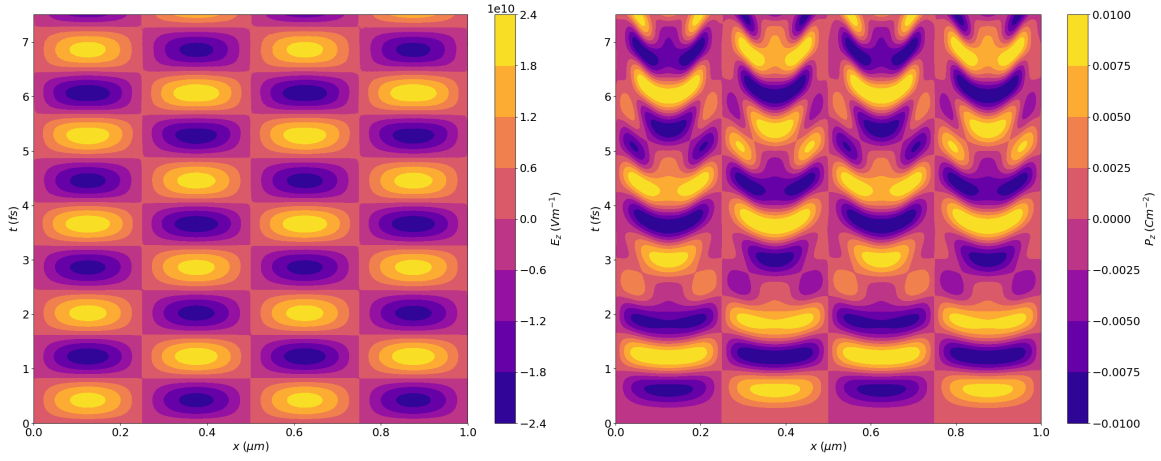


FIGURE 5.14: Temporal evolution of the positions of the particles for the case where $\omega_{12} = 1.5\frac{2\pi c}{\lambda}$. the particles can be seen drifting towards the nodes of the standing wave.



(A) (B)

FIGURE 5.15: Temporal evolution of the $\hat{\mathbf{z}}$ component of the electric (A) and polarization fields (B) for the case where $\omega_{12} = 1.5\frac{2\pi c}{\lambda}$.

Chapter 6

Concluding remarks and future work

During the course of this work, we aimed to study many-body systems composed of atoms, ions, and electrons interacting with the electromagnetic field. Concretely, the objective was to study the analytical and numerical models available, and develop computational tools to model these systems. To achieve this, we focused on the PIC method from plasma Physics.

The code was developed in C++ with the *ArrayFire* library for the CPU and GPU backends. It contains the following numerical methods: FDTD method, Vay particle pusher, interpolation methods, Esirkepov current deposition method, and Poisson equation solver, all available in three dimensions.

The prominent features of our software are its modularity, evidenced by the distinct physical nature of each test case, and the performance enhancement brought by GPU computing. Specifically, when we compared an high-end GPU (GTX Titan) and an high-end CPU (i7-4930K in single thread), we measured a top speedup of 108. Moreover, the code includes a new type of ABC for the FDTD method, which we named MABC, that shows promising results.

In chapter 5, some of the possible applications of the code were shown and include: a direct application of the PIC code for a problem of plasma-based particle acceleration, transport phenomena in gravitational systems described by the gravitoelectromagnetic equations, and transport phenomena in quantum dipolar gases. While the first two examples are almost direct applications of a standard PIC code, the third uses some of the features of the code, such as the FDTD methods, the Vay pusher and the interpolation methods, and other components developed by my colleague João Costa to simulate the

internal electronic states of the atoms. The end result is a numerical model capable of displaying quantum-optomechanical effects.

There are other physical systems that could be studied with our software such as magnetic effects using the hybrid PIC-Bloch scheme of the last test case, and meta-materials[61].

Some other physical systems are out of reach for the current state of the code. Nonetheless, the modular construction of the code allows the inclusion of other physical processes such as collisions, useful to model neutral fluids according to the DSMC method. The other limitation of the code comes from the fact that it can only run on a single GPU. Extending it to multiple GPUs would be necessary to permit larger simulation domains and faster run-times. Finally, another essential tool would be better 3-dimensional visualization routines not only in real-time, while the simulation runs, but also for posterior analysis of the results.

To conclude, this work opens perspectives in the study of quantum many body systems out of equilibrium. In the approach presented here, the quantum nature of the system is restricted to the internal degrees of freedom of the atoms, while motion of the atoms and the field remain classical. However, this is sufficient to study a wide range of systems and phenomena, starting from coherent optical effects, and transport phenomena, to nonlinear effects, and many others.

During the development of this thesis I was fortunate enough to work with an extremely dedicated and friendly team. I was also able to work in computational physics, an area I intend to continue pursuing, and to see on a computer screen many physical systems and effects usually hidden from the naked eye.

Publications

- Development of a quantum particle in cell algorithm in GPU for solving Maxwell-Bloch equations. - M. Gomes, J. C. Costa, R. A. Alves, Nuno A. Silva, A. Guerreiro; AOP17 Proceedings; 2017.

- Solver of the Einstein equations using GPUs under the gravitoelectromagnetic approximation. - M. Gomes, J. C. Costa, R. A. Alves, Nuno A. Silva, A. Guerreiro; AOP17 Proceedings; 2017.

- The analogue quantum mechanical of plasmonic atoms. - R. A. Alves, J. C. Costa, Miguel Gomes, Nuno A. Silva, A. Guerreiro; AOP17 Proceedings; 2017.

- Solving the multi-level Maxwell-Bloch equations using GPGPU computing for the simulation of nonlinear optics in atomic gases. - J. C. Costa, M. Gomes, R. A. Alves, Nuno A. Silva, A. Guerreiro; AOP17 Proceedings; 2017.

- Doppler broadening effects in plasmonic quantum dots. - R. A. Alves, J. C. Costa, M. Gomes, Nuno A. Silva, A. Guerreiro; AOP17 Proceedings; 2017.

- Fast physical ray-tracing method for gravitational lensing using heterogeneous supercomputing in GPGPU. - J. C. Costa, M. Gomes, R. A. Alves, Nuno A. Silva, A. Guerreiro; AOP17 Proceedings; 2017.

- Tunable light fluids using quantum atomic optical systems. - Nuno A. Silva, R. A. Alves, J. C. Costa, M. Gomes, A. Guerreiro; AOP17 Proceedings; 2017.

- Pinching optical potentials for spatial nonlinearity management in Bose Einstein condensates. - Nuno A. Silva, R. A. Alves, J. C. Costa, M. Gomes, A. Guerreiro; AOP17 Proceedings; 2017.

- Dissipative solitons in 4-level atomic optical systems. - Nuno A. Silva, R. A. Alves, J. C. Costa, M. Gomes, A. Guerreiro; AOP17 Proceedings; 2017.

- Physical ray-tracing method for anisotropic optical media in GPGPU. - A. Guerreiro, R. A. Alves, J. C. Costa, M. Gomes, Nuno A. Silva; AOP17 Proceedings; 2017.

- Quantum wires as sensors of the electric field: A model into quantum plasmonics. - R. A. Alves, J. C. Costa, M. Gomes, Nuno A. Silva, A. Guerreiro; OFS-25 Proceedings; 2016.

Appendix A

The Pseudospectral Time-Domain method

As described in section 3.2, PSTD is similar to FDTD but the spatial derivatives are evaluated using spectral methods. For example, the spatial derivatives in the Maxwell equations can be expressed using Fourier transforms as

$$\dot{\mathbf{B}} = \mathcal{F}^{-1}\left\{\begin{bmatrix} 0 & \mathrm{i}k_z & -\mathrm{i}k_y \\ -\mathrm{i}k_z & 0 & \mathrm{i}k_x \\ \mathrm{i}k_y & -\mathrm{i}k_x & 0 \end{bmatrix} \mathcal{F}\{\mathbf{E}\}\right\}$$

$$\dot{\mathbf{D}} = \mathcal{F}^{-1}\left\{\begin{bmatrix} 0 & -\mathrm{i}k_z & \mathrm{i}k_y \\ \mathrm{i}k_z & 0 & -\mathrm{i}k_x \\ -\mathrm{i}k_y & \mathrm{i}k_x & 0 \end{bmatrix} \mathcal{F}\{\mathbf{H}\}\right\} - \mathbf{J}_f$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transform, and the inverse Fourier transform, respectively. Next, we apply the temporal discretization and the **D** and **E** fields are again sampled at different instants from the **B** and **H** fields, according to the *leapfrog* scheme

$$D_x^{t+\Delta_t} = D_x^t + \Delta_t\left(\mathcal{F}^{-1}\left\{ik_z\mathcal{F}\left\{H_y^{t+\frac{\Delta_t}{2}}\right\} - ik_y\mathcal{F}\left\{H_z^{t+\frac{\Delta_t}{2}}\right\}\right\} - J_x^{t+\frac{\Delta_t}{2}}\right)$$

$$D_y^{t+\Delta_t} = D_y^t + \Delta_t\left(\mathcal{F}^{-1}\left\{ik_x\mathcal{F}\left\{H_z^{t+\frac{\Delta_t}{2}}\right\} - ik_z\mathcal{F}\left\{H_x^{t+\frac{\Delta_t}{2}}\right\}\right\} - J_y^{t+\frac{\Delta_t}{2}}\right)$$

$$D_z^{t+\Delta_t} = D_z^t + \Delta_t\left(\mathcal{F}^{-1}\left\{ik_y\mathcal{F}\left\{H_x^{t+\frac{\Delta_t}{2}}\right\} - ik_x\mathcal{F}\left\{H_y^{t+\frac{\Delta_t}{2}}\right\}\right\} - J_z^{t+\frac{\Delta_t}{2}}\right)$$

$$
\begin{aligned}
B_x^{t+\frac{3\Delta_t}{2}} &= B_x^{t+\frac{\Delta_t}{2}} + \Delta_t \mathcal{F}^{-1}\left\{ik_y \mathcal{F}\left\{E_z^{t+1}\right\} - ik_z \mathcal{F}\left\{E_y^{t+1}\right\}\right\} \\
B_y^{t+\frac{3\Delta_t}{2}} &= B_y^{t+\frac{\Delta_t}{2}} + \Delta_t \mathcal{F}^{-1}\left\{ik_z \mathcal{F}\left\{E_x^{t+1}\right\} - ik_x \mathcal{F}\left\{E_z^{t+1}\right\}\right\} \\
B_z^{t+\frac{3\Delta_t}{2}} &= B_z^{t+\frac{\Delta_t}{2}} + \Delta_t \mathcal{F}^{-1}\left\{ik_x \mathcal{F}\left\{E_y^{t+1}\right\} - ik_y \mathcal{F}\left\{E_x^{t+1}\right\}\right\}
\end{aligned}
$$

With PSTD the Yee cell is not necessary but, since the Fourier transform will be calculated with the FFT algorithm, the sampling should still be on a regular mesh. Despite being able to solve the problem of numerical dispersion, the time complexity of this algorithm is $\mathcal{O}(n \log n)$ while the FDTD has complexity $\mathcal{O}(n)$, where n is the number of cells.

Appendix B

From the Vlasov equation to PIC

The derivation of each equations of motion for the super-particles in a PIC code is shown here in detail. The section is divided for the moment along the position and momentum.

B.1 First moment along position

We take the first moment of the CBE along the position. The first resulting term will be

$$w_p \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i\partial_t \left(\prod_j S_j\left(j-j_p\right)\delta\left(p_j-p_{j,p}\right)\right) d^3\mathbf{r}\, d^3\mathbf{p} =$$

$$= w_p\partial_t \left[\int_{-\infty}^{+\infty} iS_i\left(i-i_p\right) di \prod_k \left(\int_{-\infty}^{+\infty} S_k\left(k-k_p\right) dk\right) \prod_j \left(\int_{-\infty}^{+\infty} \delta\left(p_j-p_{j,p}\right) dp_j\right)\right] =$$

$$= w_p\partial_t i_p,$$

where $i,\, j \in \{x,\, y,\, z\}$, $k \in \{x,\, y,\, z\}_{\setminus\{i\}}$, and for the last equality the first integral yields the average of S_i while the others evaluate to 1.

For the second term we have

$$w_p \sum_j \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} iv_j\partial_j \left(\prod_k S_k\left(k-k_p\right)\delta\left(p_k-p_{k,p}\right)\right) d^3\mathbf{r}\, d^3\mathbf{p} =$$

$$= w_p \sum_j \int_{-\infty}^{+\infty} i\partial_j \left(\prod_k S_k\left(k-k_p\right)\right) d^3\mathbf{r} \int_{-\infty}^{+\infty} v_j \left(\prod_k \delta\left(p_k-p_{k,p}\right)\right) d^3\mathbf{p} =$$

$$= w_p \sum_j \int_{-\infty}^{+\infty} i\partial_j \left(\prod_k S_k\left(k-k_p\right)\right) d^3\mathbf{r} \int_{-\infty}^{+\infty} \frac{p_j}{m}\left(1+\frac{|\mathbf{p}|}{mc^2}\right)^{-\frac{1}{2}} \delta^3\left(\mathbf{p}_k-\mathbf{p}_{k,p}\right) d^3\mathbf{p} =$$

$$= w_p \sum_j \int_{-\infty}^{+\infty} i\partial_j \left(\prod_k S_k \left(k - k_p \right) \right) d^3\mathbf{r}\, v_{j,p} =$$

$$= w_p \left[\int_{-\infty}^{+\infty} i\partial_i \left(\prod_k S_k \left(k - k_p \right) \right) d^3\mathbf{r}\, v_{i,p} + \underbrace{\sum_l \int_{-\infty}^{+\infty} i\partial_l \left(\prod_k S_k \left(k - k_p \right) \right) d^3\mathbf{r}\, v_{l,p}}_{A=0} \right]$$

$$= w_p \left[\left(\underbrace{\left[iS_i \left(i - i_p \right) \right]_{-\infty}^{+\infty}}_{=0} - \underbrace{\int_{-\infty}^{+\infty} S_i \left(i - i_p \right) di}_{=1} \right) \underbrace{\prod_l \left(\int_{-\infty}^{+\infty} S_l \left(l - l_p \right) d^3\mathbf{r} \right)}_{=1} v_{i,p} \right] =$$

$$= -w_p v_{i,p}$$

where $i, j, k \in \{x, y, z\}$, $l \in \{x, y, z\}_{\setminus\{i\}}$, $A = 0$ because for all j one of the integrals is $\int_{-\infty}^{+\infty} \partial_i S_j \left(i - i_p \right) di = 0$, and $v_{i,p}$ relates to $p_{i,p}$ as defined by equation (2.7). It is important to note here that for relativistic motion only the choice of δ^3 for the momentum *shape factor* will yield this result.

Finally, the third term yields

$$w_p \sum_j \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} iF_j \partial_{p_j} \left(\prod_k S_k \left(k - k_p \right) \delta \left(p_k - p_{k,p} \right) \right) d^3\mathbf{r}\, d^3\mathbf{p} =$$

$$= w_p \sum_j \int_{-\infty}^{+\infty} iF_j \left(\prod_k S_k \left(k - k_p \right) \right) d^3\mathbf{r} \underbrace{\int_{-\infty}^{+\infty} \partial_{p_j} \left(\prod_k \delta \left(p_k - p_{k,p} \right) \right) d^3\mathbf{p}}_{=0} = 0$$

where $i, j, k \in \{x, y, z\}$ and the last equality comes from integrating the derivative of $\delta \left(p_j - p_{j,p} \right)$ along j.

B.2 First moment along momentum

The next equation comes from taking the first moment along the momentum p_i. The first term will be similar to the first term in appendix B.1:

$$w_p \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p_i \partial_t \left(\prod_j S_j \left(j - j_p \right) \delta \left(p_j - p_{j,p} \right) \right) d^3\mathbf{r}\, d^3\mathbf{p} =$$

$$= w_p \partial_t \left[\prod_j \left(\int_{-\infty}^{+\infty} S_i \left(j - j_p \right) dj \right) \int_{-\infty}^{+\infty} p_i \delta \left(p_i - p_{i,p} \right) dp_i \prod_k \left(\int_{-\infty}^{+\infty} \delta \left(p_k - p_{k,p} \right) dk \right) \right] =$$

$$= w_p \partial_t p_{i,p},$$

where, again, $i, j \in \{x, y, z\}$ and $k \in \{x, y, z\}_{\setminus\{i\}}$.

The second term is

$$w_p \sum_j \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p_i v_j \partial_j \left(\prod_k S_k \left(k - k_p\right) \delta \left(p_k - p_{k,p}\right) \right) d^3\mathbf{r}\, d^3\mathbf{p} = 0$$

$$w_p \sum_j \underbrace{\int_{-\infty}^{+\infty} \partial_j S_j \left(j - j_p\right) dj}_{=0} \prod_l \left(\int_{-\infty}^{+\infty} S_l \left(l - l_p\right) dl \right) \int_{-\infty}^{+\infty} p_i v_j \left(\prod_k \delta \left(p_k - p_{k,p}\right) \right) d^3\mathbf{p} = 0$$

where $i, j, k \in \{x, y, z\}$ and $l \in \{x, y, z\}_{\setminus\{i\}}$.

At last, the third term evaluates to

$$w_p \sum_j \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p_i F_j \partial_{p_j} \left(\prod_k S_k \left(k - k_p\right) \delta \left(p_k - p_{k,p}\right) \right) d^3\mathbf{r}\, d^3\mathbf{p} =$$

$$= w_p \sum_j F_j \underbrace{\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p_i \partial_{p_j} \left(\prod_k S_k \left(k - k_p\right) \delta \left(p_k - p_{k,p}\right) \right) d^3\mathbf{r}\, d^3\mathbf{p}}_{A=-\delta_{ij}} = -w_i F_i,$$

where $A = -\delta_{ij}$ results from, essentially, a repetition of the rearrangement done for the second term in appendix B.1.

Bibliography

[1] D. A. Gurnett and A. Bhattacharjee, *Space Weather: The Physics Behind a Slogan (Lecture Notes in Physics)* (Springer, 2005) p. 138.

[2] A. R. Choudhuri, *The Physics of Fluids and Plasmas: An Introduction for Astrophysicists*, edited by A. R. Choudhuri (Cambridge University Press, 1998).

[3] J. T. Mendonca and H. Tercas, *Physics of Ultra-Cold Matter: Atomic Clouds, Bose Einstein Condensates and Rydberg Plasmas (Springer Series on Atomic, Optical, and Plasma Physics)* (Not Avail, 2014).

[4] J. T. Mendonça, A. M. Martins, and A. Guerreiro, "Field quantization in a plasma: Photon mass and charge," Physical Review E **62**, 2989 (2000).

[5] J. Vieira, R. Trines, E. Alves, R. Fonseca, *et al.*, "High Orbital Angular Momentum Harmonic Generation," Physical Review Letters **117** (2016), 10.1103/physrevlett.117.265001.

[6] A. Anders, *Handbook of Plasma Immersion Ion Implantation and Deposition* (Wiley-VCH, 2000).

[7] A. Goers, G. Hine, L. Feder, B. Miao, F. Salehi, J. Wahlstrand, and H. Milchberg, "Multi-MeV Electron Acceleration by Subterawatt Laser Pulses," Physical Review Letters **115** (2015), 10.1103/physrevlett.115.194802.

[8] R. Andreani and M. Gasparotto, "Overview of fusion nuclear technology in Europe," Fusion Engineering and Design **61-62**, 27 (2002).

[9] Y. Strebkov and V. Belyakov, "Overview of fusion nuclear technology in Russia," Fusion Engineering and Design **61-62**, 47 (2002).

[10] N. Fleurot, C. Cavailler, and J. Bourgade, "The Laser Mégajoule (LMJ) Project dedicated to inertial confinement fusion: Development and construction status," Fusion Engineering and Design **74**, 147 (2005).

[11] I. Eames and J. B. Flor, "New developments in understanding interfacial processes in turbulent flows," Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **369**, 702 (2011).

[12] M. Fleischhauer, A. Imamoglu, and J. P. Marangos, "Electromagnetically induced transparency: Optics in coherent media," Reviews of Modern Physics **77**, 633 (2005).

[13] G. Huang, L. Deng, and M. G. Payne, "Dynamics of ultraslow optical solitons in a cold three-state atomic system," Physical Review E **72** (2005), 10.1103/physreve.72.016617.

[14] M. Fox, *Optical Properties of Solids (Oxford Master Series in Physics)* (Oxford University Press, 2002).

[15] J. ming Liu, *Photonic Devices* (Cambridge University Press, 2005).

[16] B. L. Schmittberger and D. J. Gauthier, "Transverse optical and atomic pattern formation," Journal of the Optical Society of America B **33**, 1543 (2016).

[17] H. W. Benjamin Bederson, ed., *Advances in Atomic, Molecular, and Optical Physics*, Advances in Atomic, Molecular, and Optical Physics, Vol. 42 (Academic Press, 1999).

[18] J. Eisert, M. Friesdorf, and C. Gogolin, "Quantum many-body systems out of equilibrium," Nature Physics **11**, 124 (2015).

[19] A. Vlasov, *Many-particle Theory and Its Application to Plasma* (Gordon & Breach Science Publishers Ltd, 1961).

[20] K. T. McDonald, "Limits on the applicability of classical electromagnetic fields as inferred from the radiation reaction," (2000), arXiv:physics/0003062 .

[21] A. Taflove, S. C. Hagness, and M. Piket-May, in *The Electrical Engineering Handbook*, edited by A. House (Elsevier BV, 2005) pp. 629–670.

[22] K. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," IEEE Transactions on Antennas and Propagation **14**, 302 (1966).

[23] H. Zhao, S. Crozier, and F. Liu, "A high definition, finite difference time domain method," Applied Mathematical Modelling **27**, 409 (2003).

[24] D. J. Griffiths, *Introduction to Electrodynamics (4th Edition)* (Pearson, 2012).

[25] F. H. Harlow and J. E. Welch, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface," Physics of Fluids **8**, 2182 (1965).

[26] F. Harlow., *The particle-in-cell method for hydrodynamic calculations*, Tech. Rep. (Los Alamos scientific laboratory of the university of california, 1957).

[27] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, "The affine particle-in-cell method," ACM Transactions on Graphics **34**, 511 (2015).

[28] J. Brackbill, "FLIP MHD: A particle-in-cell method for magnetohydrodynamics," Journal of Computational Physics **96**, 163 (1991).

[29] J.-G. Liu and W.-C. Wang, "An Energy-Preserving MAC–Yee Scheme for the Incompressible MHD Equation," Journal of Computational Physics **174**, 12 (2001).

[30] K. Thorne, "The particle kinetics of plasma," (2003).

[31] C. K. Birdsall and A. B. Langdon, *Plasma Physics Via Computer Simulation* (Taylor & Francis Ltd, 2004).

[32] J. M. Dawson, "Particle simulation of plasmas," Reviews of Modern Physics **55**, 403 (1983).

[33] J. P. Boris, *Proceeding of the 4th Conference on Numerical Simulation of Plasmas* (Naval Res. Lab., 1970).

[34] A. V. Higuera and J. R. Cary, "Structure-preserving second-order integration of relativistic charged particle trajectories in electromagnetic fields," Physics of Plasmas **24**, 052104 (2017).

[35] J.-L. Vay, "Simulation of beams or plasmas crossing at relativistic velocity," Physics of Plasmas **15**, 056701 (2008).

[36] A. B. Langdon, "On enforcing Gauss' law in electromagnetic particle-in-cell codes," Computer Physics Communications **70**, 447 (1992).

[37] J. Villasenor and O. Buneman, "Rigorous charge conservation for local electromagnetic field solvers," Computer Physics Communications **69**, 306 (1992).

[38] T. Esirkepov, "Exact charge conservation scheme for Particle-in-Cell simulation with an arbitrary form-factor," Computer Physics Communications **135**, 144 (2001).

[39] T. Umeda, Y. Omura, T. Tominaga, and H. Matsumoto, "A new charge conservation method in electromagnetic particle-in-cell simulations," Computer Physics Communications **156**, 73 (2003).

[40] B. Marder, "A method for incorporating Gauss' law into electromagnetic PIC codes," Journal of Computational Physics **68**, 48 (1987).

[41] Smilei, "Smilei," `http://www.maisondelasimulation.fr/smilei/algorithms.html`, accessed: 2017-08-27.

[42] P. T. de Abreu, *Multiscale High-Performance Computing in Plasma Physics*, Ph.D. thesis, Instituto Superior Técnico de Lisboa (2010).

[43] PIConGPU, "PIConGPU," `https://github.com/ComputationalRadiationPhysics/picongpu`, accessed: 2017-08-27.

[44] T. Umeda, Y. Omura, and H. Matsumoto, in *Proceedings of ISSS-7* (2005).

[45] J. Yu, X. Jin, W. Zhou, B. Li, and Y. Gu, "High-Order Interpolation Algorithms for Charge Conservation in Particle-in-Cell Simulations," Communications in Computational Physics **13**, 1134 (2013).

[46] D. Luebke, in *2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (Institute of Electrical and Electronics Engineers (IEEE), 2008).

[47] M. J. Flynn, "Some Computer Organizations and Their Effectiveness," IEEE Transactions on Computers **C-21**, 948 (1972).

[48] N. Corporation, "Cuda toolkit documentation v8.0," `http://docs.nvidia.com/cuda/` (2016), accesed on 2017-01-26.

[49] R. Banger and B. Bhattacharyya, *OpenCL Programming by Example* (Packt Publishing, 2013).

[50] J. Sanders and E. Kandrot, *CUDA by Example* (Addison Wesley, 2010).

[51] P. Yalamanchili, U. Arshad, Z. Mohammed, P. Garigipati, *et al.*, "ArrayFire - A high performance software library for parallel computing with an easy-to-use API," (2015).

[52] M. Eloy, A. Guerreiro, J. T. Mendonça, and R. Bingham, "Hamiltonian formulation of direct laser acceleration in vacuum," Journal of Plasma Physics **73** (2007), 10.1017/s0022377806006131.

[53] M. Capitelli, A. Casavola, G. Colonna, and A. D. Giacomo, "Laser-induced plasma expansion: theoretical and experimental aspects," Spectrochimica Acta Part B: Atomic Spectroscopy **59**, 271 (2004).

[54] B. Mashhoon, F. Gronwald, and H. I. Lichtenegger, in *Gyros, Clocks, Interferometers...: Testing Relativistic Graviy in Space* (Springer Berlin Heidelberg, 1999) pp. 83–108.

[55] B. Mashhoon, F. W. Hehl, and D. S. Theiss, "On the gravitational effects of rotating masses: The Thirring-Lense papers," General Relativity and Gravitation **16**, 711 (1984).

[56] V. Majernik, "Field approach to gravitation and its significance in astrophysics," Astrophysics and Space Science **14**, 265 (1971).

[57] C. Gerry and P. Knight, *Introductory Quantum Optics* (Cambridge University Press, 2005).

[58] W. H. Louisell, *Quantum Statistical Properties of Radiation (Pure & Applied Optics)* (John Wiley & Sons Inc, 1973).

[59] L. Landau and E. Lifshitz, "On the theory of the dispersion of magnetic permeability in ferromagnetic bodies," Phys. Z. Sowjetunion **8**, 101 (1935).

[60] A. K. Sarma and P. Kumar, "High and uniform coherence creation in Doppler-broadened double lambda-like atomic system by a train of femtosecond optical pulses," Laser Phys. **25**, 056001 (2015).

[61] A. Lopes, *Radiation Tension in nonlinear plasma metamaterials*, Master's thesis, Instituto Superior Técnico (2015).