Design and performant implementation of numerical methods for multiscale problems in plasma physics

Sever A. Hirstoaga

Contents

Résumé

Dans ce mémoire, nous présentons différentes stratégies pour approcher les solutions de modèles cinétiques faisant intervenir plusieurs échelles en temps. Plus précisément, nous nous intéressons à des systèmes de type Vlasov–Poisson pour décrire l'evolution dans l'espace des phases de particules chargées sous l'influence d'un champ électromagnétique. L'objectif de ce travail est d'étudier le comportement multi-échelle de ces équations à l'aide de l'analyse asymptotique et de proposer des méthodes numériques adaptées pour approcher leurs solutions.

Dans la première partie, nous proposons deux méthodes pour traiter la difficulté liée au caractère multi-échelle. La première est une méthode d'homogénéisation en temps, basée sur la notion de convergence à deux échelles. Dans cette approche, nous obtenons un modèle réduit approchant l'équation originelle de Vlasov. La deuxième méthode est un nouveau schéma en temps pour l'équation de Vlasov. Basé sur un intégrateur exponentiel, le schéma résout la petite échelle tout en utilisant des pas de temps macroscopiques. Des cas-tests issus de la physique des plasmas illustrent la précision et l'efficacité de la méthode.

Dans la deuxième partie, nous analysons la performance d'une implémentation "Particle-in-Cell" pour résoudre numériquement le système de Vlasov–Poisson. Le problème de la performance se pose lors de l'utilisation de schémas de discrétisation explicites, avec des paramètres numériques résolvant la petite échelle, ce qui entraîne un coût de calcul important. Nous développons des structures de données spécifiques pour optimiser les accès mémoire. Ce travail est complété par le développement d'une approche spécifique de répartition des calculs par parallélisme hybride, utilisant à la fois les mécanismes de mémoire partagée et de mémoire distribuée, que nous implémentons sur processeurs multi-cœurs. Des mesures classiques pour évaluer la performance des codes sont analysées.

Dans la troisième partie, nous développons un cadre de modélisation et de simulation pour résoudre des problèmes pertinents en physique des plasmas. Plus précisément, nous étudions *(i)* la dynamique d'un plasma d'ions et d'électrons suivant un "edge-localized mode", qui est une instabilité se produisant au bord d'un tokamak et *(ii)* un problème d'instabilité de diocotron dans un plasma. Pour modéliser ces phénomènes, nous proposons différents couplages faisant intervenir des équations de type Poisson, des équations fluides et des équations cinétiques de type Vlasov avec termes source et de collision. Leur résolution est effectuée avec des schémas numériques de type "asymptotic-preserving" permettant de traiter efficacement le caractère multi-échelle.

Mots-clés : physique des plasmas, équations Vlasov–Poisson, modèle multi-échelle, modèle réduit, schéma numérique, méthode particulaire, structures de données, processeurs multi-cœurs.

Introduction

This manuscript assembles my contributions in developing new mathematical and computational methods for analyzing the dynamics of charged particles, like electrons or ions, as a multiscale phenomenon. The underlying mechanisms of this general physical problem are described by systems of partial differential equations. The objective of this work is to study these equations and to implement various efficient numerical methods to approximate their solutions.

The aiming applications in this manuscript are in the domain of plasma physics and the physical problems are expressed by means of kinetic models. In such a model, each particle species in the plasma is characterized by a distribution function which corresponds to a statistical mean of the repartition of particles in phase space for a large number of realizations of the physical system. With regard to fluid models, a kinetic approach contains more information on the plasma, namely the distribution of the particle velocities at each position, which is an important information when the plasma is not at the thermodynamic equilibrium. More precisely, the mathematical models are based on the Vlasov equation, that describes the evolution of charged particles in an electromagnetic field which can either be self-consistent, *i.e.* generated by the particles themselves, or external, or both. The Vlasov equation is a transport equation in phase space which includes the physical space and the velocity space. The self-consistent electromagnetic field is calculated from Maxwell equations, the coupling being performed through the charge density and the current computed from the particles. In this work, however, we consider only the case of a self-consistent electric field, and thus, the coupling between the Vlasov equation and the Poisson equation.

The numerical treatment of the Vlasov–Poisson system is mainly performed in the present work by a Particle-in-Cell (PIC) method. Such a method needs a grid for the physical space only, but may exhibit the disadvantage of strong spurious oscillations when displaying the particle densities. This drawback is addressed in the manuscript by increasing the number of particles. Eulerian methods are also implemented for solving Vlasov equation, precisely finite volume schemes on uniform grids in the phase space. The accuracy of the latter method plays an important role in reproducing the complex physical phenomena we tackle. However, the main difficulty to handle in this case is to satisfy the CFL condition. In addition, going to higher dimensions in an Eulerian code is computationally intensive. As for the Poisson equation, standard numerical approaches are implemented, utilizing the Fourier transform or the finite difference method.

The contributions described in this manuscript are at the intersection between applied mathematics, computer science, and physics. As summarized below and detailed in the next chapters, we will discuss about **new mathematical models, the design and the development of new and existing numerical methods, their efficient implementation on machines with many cores, and some numerical results in specific plasma physics applications**. More precisely, the manuscript is split in three chapters, with the following objectives:

- (i) design of new mathematical and numerical methods for understanding the complex behaviours of solutions to Vlasov-like systems;
- (ii) development of efficient methods for an optimal use of computer resources when numerically solving these problems,
- (iii) development of computational models and of appropriate numerical methods for simulations of realistic problems in plasma physics.

The central problem, common to the three previous research directions, is the solving of transport equations involving several scales in time. The multiscale aspect makes the models difficult to tackle when we aim at avoiding a high computational cost. We can treat this issue in (at least) three manners and therefore, the overall content of the manuscript could also be read from the following perspective:

- (i) we solve asymptotic models, which are approximations of the full original model, but with a much lower numerical cost (Section 1.2 in Chapter 1);
- (ii) we solve the original model with a standard numerical method, and therefore a high performance computing approach is required (Chapter 2);
- (iii) we solve the original model by designing adapted numerical methods, which are not constrained by the smallest scale and thus, with a low computational cost (Section 1.3 in Chapter 1 and Chapter 3).

All the developments presented in the following would not have been possible without the interaction with colleagues from mathematics, physics, and computer science. Their names appear in my list of references. I will particularly mention in this introduction the contributions of PhD students and post-docs with whom I had the opportunity to collaborate and to develop joint publications.

The chapters are structured as follows:

Chapter 1 deals with **multiscale** Vlasov-like models issued from challenging problems in plasma physics. Even though these models are rather simplified and academic, they contain the difficulty of the presence of several time scales in the solutions, including high frequency oscillations. This is an outstanding question to be accounted for when solving realistic physical problems for plasmas. In order to cope with this issue, two strategies are proposed. First, we propose reduced models obtained by asymptotic analysis. These models are free of high frequency oscillations and can be solved numerically with standard schemes without demanding constraints on the numerical parameters. Second, we treat the full model with a numerical method based on exponential integrators, by which the high frequency oscillations are exactly solved whereas the slower process is treated in an approximate way. This numerical strategy is compared in terms of accuracy and efficiency to other existing methods that solve multiscale problems. Part of the contributions presented here benefits from the interactions during the PhD thesis of Mathieu Lutz (defended in October 2013 at Université de Strasbourg), advised by Éric Sonnendrücker (Max-Planck-Institut für Plasmaphysik, Garching) and Emmanuel Frénod (Université de Bretagne-Sud, Vannes). The results in [14] were obtained in the context of the TSAPPICC Cemracs project (`http://smai.emath.fr/cemracs/cemracs11/`) in 2011, at CIRM next to Marseille. The work in [10] was obtained during the post-doc of Xiaofei Zhao, supported by Inria through IPL FRATRES (`https://team.inria.fr/ipl-fratres/`) and co-advised with Nicolas Crouseilles (Inria Rennes). Finally, a part of the previous subjects was conceived in the framework of the Eurofusion Enabling Research Project "Verification and development of new algorithms for gyrokinetic codes" (2015–2017) whose principal investigator was Éric Sonnendrücker.

In Chapter 2, we focus on the **performance** of Particle-in-Cell (PIC) simulations for solving Vlasov–Poisson systems in six dimensional phase space. This is an important issue to be dealt with when solving problems processing a large amount of data. Despite significant recent advances, devising efficient strategies for utilizing modern supercomputing resources is still a challenging field. Mainly, we address specific data structures in order to optimize the memory accesses. Since a PIC simulation is memory bound, minimizing local data motion requires grouping data such that memory operations can be performed contiguously. The data structures are also useful to exploit efficiently parallelism patterns, like vectorization, multithreading, and multiprocessing. The sequential and parallel performances of our PIC codes are assessed with several classical measures. These contributions are concerned with the particle push, whereas the Poisson solver is not a central component in our optimization efforts. The starting point of this work was realized during the Cemracs project OPICV (`http://smai.emath.fr/cemracs/cemracs14/`) in 2014, that I proposed jointly with Edwin Chacon-Golcher (ELI Beamlines, Prague). The implementation was performed in the library SeLaLib [Sel]. The results which followed have been obtained in collaboration with Yann Barsamian (Université de Strasbourg) during his PhD thesis, co-advised with Éric Violard (Université de Strasbourg) and Michel Mehrenberger (Université de Strasbourg).

In Chapter 3 we develop a computational framework for **modelling and simulating** complex problems in plasma physics. More precisely, we study the problems of the diocotron instability in a non-neutral plasma and of the dynamics of two species of charged particles following an edge-localized mode (ELM) event in a tokamak's scrape-off layer. The partial realism of these physical problems leads to several mathematical and numerical difficulties. In this direction, we propose and solve different kinetic and fluid equations to treat the modelling questions, while for the numerical aspect, an asymptotic preserving strategy turns out to be fruitful. The subject concerning the study of ELM dynamics was mainly proposed by Giovanni Manfredi (Institut de Physique et Chimie des Matériaux de Strasbourg (IPCMS), Université de Strasbourg). The latest results were obtained in collaboration with David Coulette, during his post-doc. In order to develop this research direction, we benefited from the ANR program PEPPSI (2013–2017: `http://peppsi.u-strasbg.fr/`) whose principal investigator was Giovanni Manfredi. The study of the diocotron instability has carried out in a close collaboration with Michel Mehrenberger (Université de Strasbourg).

I started to work on the contributions exposed in this manuscript in 2009. My earlier research related to the PhD thesis (manuscript available on HAL at `https://hal.archives-ouvertes.fr/tel-00137228`) dealt with some questions arising in nonlinear functional analysis and its applications. One direction was to solve fixed point problems for nonexpansive operators and to consider an application of the proposed algorithms to image restoration problems. Another direction treated the abstract problem of finding zeros of multivalued maximal monotone operators in Hilbert spaces. This is a general frame entailing several well-known questions in variational inequalities, in convex analysis or more concretely, in optimization. A third direction was to study a yet more general problem (the equilibrium problem) that offers a unified framework for some of the first two previous subjects. The solutions to these three types of problems have been constructed in three different ways: (i) as limits of solutions to perturbed easier to solve problems, (ii) as limits of discrete well-behaved algorithms, and finally (iii) as the asymptotic behaviour of the solution to some associated continuous dynamical system. My publications related to this topic are [6; 7; 8; 18]. This work will not be detailed in the following.

Chapter 1

Numerical methods for multiscale Vlasov and Vlasov–Poisson models

The goal of this chapter is to expose our works on designing multiscale strategies for the numerical solution of problems exhibiting several time scales. The models considered in this chapter are linear Vlasov equations or Vlasov–Poisson equations, in a phase space of dimension two, four, or six, that we denote in the following by $1d \times 1d$, $2d \times 2v$, and $3d \times 3v$ respectively. As a consequence of specific physical modelling considerations, the role of the multiscale aspect is played in the equations by a small parameter. Standard explicit numerical schemes are therefore constrained by this parameter to prohibitive time computations.

In this framework, **the contribution of this chapter** is to propose **two alternative solutions** for treating with these problems: a technique of homogenization in time based on the two-scale convergence theory and a numerical method based on exponential integrators. The first one deals with **new two-scale asymptotic preserving models** separating terms which are free of multiple scales from terms depending in analytical way of the smallest scale. As for the second solution, we develop a **new explicit time-advancing scheme** which discretizes the full highly oscillating model and meanwhile uses large time steps with respect to the smallest scale.

The summary of the chapter is the following:

- Section 1.1 gives information at a glance about the kinetic multiscale models under consideration and their motivation.
- In Section 1.2 we develop the framework for building **new two-scale first order approximations** for Vlasov models, by means of classic tools from asymptotic analysis like the two-scale convergence. The presentation follows the references [12; 14].
- In Section 1.3 we design a **new time-stepping method** for a generic **multiscale** Vlasov model. This scheme enables an exact solving of the smallest scale and an approximation of the slower part, and is quite accurate when using very large time steps with respect to the smallest scale. The concerned references are, in chronological order, [16; 15; 17; 10].
- In Section 1.4 we conclude the chapter and some perspectives are drawn.

1.1 General context and equations of interest

The starting point of the works described in this chapter is the mathematical and numerical study of the dynamics of charged particles submitted to a large magnetic field. This problem is of great interest for describing strongly magnetized plasmas such as those encountered in a tokamak[1] device [HM03]. The general context is therefore that of the plasma magnetic confinement in the core of a tokamak. It is a weakly collisional regime [HM03], which justifies the use of the Vlasov equation as a first convenient approach for modelling the particles dynamics. In addition, the previous system should be coupled with Maxwell equations that describe the self-consistent electromagnetic field. However, we only consider in this chapter simplified models consisting in linear Vlasov equations and Vlasov–Poisson equations but containing a term corresponding to a large magnetic field.

The challenge of these problems is that the charged particles experience motions at several scales in time and space. For instance, **particles turn rapidly around the magnetic field line** and also drift in the perpendicular plane on a much slower time scale. Mathematically, the multiscale behaviour is given by the presence in the dimensionless equations of singular terms due to a small parameter $\varepsilon > 0$. An example is when ε is the ratio between the Larmor radius and the characteristic length, as in the regime of a strong magnetic field. Concretely, the difficulty in the problems treated in this chapter is that the small parameter generates high frequency oscillations in time in the solutions of the equations, that we aim at capturing numerically without a prohibitive computational cost.

We present in the following the models we tackle in the rest of the chapter. In a first step, we are interested in a linear Vlasov equation in $3d \times 3v$ with a strong magnetic field that reads:

$$\begin{cases} \dfrac{\partial f^\varepsilon}{\partial t} + \mathbf{v}\cdot\nabla_{\mathbf{x}} f^\varepsilon + \left(\mathbf{E} + \mathbf{v}\times\left(\mathbf{B} + \dfrac{\mathcal{M}}{\varepsilon}\right)\right)\cdot\nabla_{\mathbf{v}} f^\varepsilon = 0, \\[2ex] f^\varepsilon(t=0,\mathbf{x},\mathbf{v}) = f_0(\mathbf{x},\mathbf{v}), \end{cases} \tag{1.1}$$

where $f^\varepsilon = f^\varepsilon(t,\mathbf{x},\mathbf{v})$ is the distribution function of an ion gas submitted to external electric and magnetic field, $\mathbf{E} = \mathbf{E}(t,\mathbf{x})$ and $\mathbf{B} = \mathbf{B}(t,\mathbf{x})$ respectively, and where $t \in \mathbb{R}^+$, $\mathbf{x} \in \mathbb{R}^3$, $\mathbf{v} \in \mathbb{R}^3$ are the time, the position, and the velocity variable, respectively. The notation $\mathcal{M}$ stands for $2\pi\,\mathbf{e}_1$, where $\mathbf{e}_1$ is the first vector of the canonical base $(\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3)$ of $\mathbb{R}^3$ and f_0 denotes the given initial condition. The term $\mathcal{M}/\varepsilon$ entails the strong magnetic field, since $\varepsilon > 0$ is a small parameter. There is an ample literature treating models similar to (1.1). We refer for example to [FS98; GSR99; FS01; GSR03; Bos09; HK10] among the first works about different regimes, obtained by taking limits of Vlasov or Vlasov–Poisson equations corresponding to small parameters.

We next aim at solving a special $2d \times 2v$ instance of the previous model, without the $\mathbf{B}$ part. The equation reads:

$$\begin{cases} \dfrac{\partial f^\varepsilon}{\partial t} + \mathbf{v}\cdot\nabla_{\mathbf{x}} f^\varepsilon + \left(\mathbf{E} + \dfrac{1}{\varepsilon}\mathbf{v}^\perp\right)\cdot\nabla_{\mathbf{v}} f^\varepsilon = 0, \\[2ex] f^\varepsilon(t=0,\mathbf{x},\mathbf{v}) = f_0\left(\mathbf{x},\mathbf{v}\right), \end{cases} \tag{1.2}$$

where $\mathbf{x} = (x_1,x_2)$ stands for the position variable, $\mathbf{v} = (v_1,v_2)$ for the velocity variable, $\mathbf{v}^\perp$ for $(v_2,-v_1)$, $f^\varepsilon \equiv f^\varepsilon(t,\mathbf{x},\mathbf{v})$ is the distribution function, and $\mathbf{E} \equiv \mathbf{E}(t,\mathbf{x})$ corresponds to the electric

[1]a chamber with the shape of a torus, developed to produce controlled thermonuclear fusion power.

field. Such a model is the subject of works in [FS00; FRS01; Bos07]. In this direction, we first consider a Vlasov equation with a given external electric field and in a second stage, a coupling of the Vlasov equation to the Poisson equation, in the plane orthogonal to the magnetic field direction (see Section 2.1 of Chapter 2 for more details about the latter model).

Although the starting physical problem is linked to the plasma confinement, similar multiscale issues are encountered in models useful for particle beam plasma physics [Dav90]. Thus, we also consider a $1d \times 1v$ Vlasov equation of type

$$\begin{cases} \dfrac{\partial f^\varepsilon}{\partial t} + \dfrac{1}{\varepsilon} v \dfrac{\partial f^\varepsilon}{\partial r} + \Big(E^\varepsilon - \dfrac{r}{\varepsilon}\Big) \dfrac{\partial f^\varepsilon}{\partial v} = 0, \\ f^\varepsilon(t=0,r,v) = f_0(r,v), \end{cases} \tag{1.3}$$

where $f^\varepsilon \equiv f^\varepsilon(t,r,v)$, $t \in \mathbb{R}^+$, $r \in \mathbb{R}^+$, and $v \in \mathbb{R}$ and $E^\varepsilon \equiv E^\varepsilon(t,r)$ is assumed to have a given form, highly oscillating in time (see (1.22)).

In addition, in a numerical context we also tackle (1.3) when the electric field is given by the Poisson equation. The coupled problem reads

$$\begin{cases} \dfrac{\partial f^\varepsilon}{\partial t} + \dfrac{1}{\varepsilon} v \dfrac{\partial f^\varepsilon}{\partial r} + \Big(E^\varepsilon - \dfrac{r}{\varepsilon}\Big) \dfrac{\partial f^\varepsilon}{\partial v} = 0, \\ \dfrac{1}{r}\dfrac{\partial (r\, E^\varepsilon)}{\partial r} = \displaystyle\int_{\mathbb{R}} f^\varepsilon(t,r,v)\, dv, \\ f^\varepsilon(t=0,r,v) = f_0(r,v). \end{cases} \tag{1.4}$$

This system is the so-called paraxial model which turns out to be an approximation of the full Vlasov-Maxwell system for modelling focused particle beams dynamics (see [FS06; FSS09] and the references therein). In this case, even if the small parameter acts on the time variable, it is in fact the longitudinal variable of a thin beam (see [FSS09] for details and the physical meaning of the parameter).

1.2 New two-scale methods for Vlasov equations

Introduction. In the context of problems with oscillatory solutions, a hierarchy of three models is naturally present: the model parameterized by the oscillation period, the limit model and the two-scale model [Fré17]. Thus, if we aim at simulating a phenomenon where the oscillation period is small whereas the oscillation amplitude is not, it can be useful to have numerical methods based on an approximation of the two-scale model. In addition, if the oscillation period varies significantly over the domain of simulation, it is important to have numerical methods that approximate both the model parameterized by the oscillation period and the limit models. Phenomena occurring in a tokamak plasma enter this context: the strong magnetic field generates varying high frequency oscillations in the plasma.

Consider an equation whose solution has high frequency oscillations depending on a small parameter. In this framework, the concept of two-scale convergence is very useful, for the following reason: even if in some cases a reduced (or limit) system is enough to model a mean behaviour of the solution, in other cases it might be interesting to capture the oscillations, since they are acknowledged to give

contributions to the long time behaviour of the solution [Ari+09; Fré17]. Thus, while the weak limit with respect to the concerned parameter averages these oscillations[2], the two-scale limit captures them without entailing any oscillation when solving the two-scale model.
The theory of two-scale convergence was introduced at the end of the 80's in [Ngu89; Ngu90] and further developed in [All92]. Subsequently, this concept turned out to be useful for building numerical methods, not only in plasma physics but also for other applications (see [Ngu89; All92; Mou09; FSS09; Fré17] and the references therein).

The idea behind this section is to construct a **two-scale first order approximation** for the solution of a multiscale Vlasov equation. Roughly speaking, the meaning of this statement is that if $f^\varepsilon \equiv f^\varepsilon(t,\mathbf{z})$[3] is the solution to equation (1.1) or (1.3), then

$$\text{for small } \varepsilon > 0 \text{ we have } f^\varepsilon(t,\mathbf{z}) \approx F\Big(t, \frac{t}{\varepsilon}, \mathbf{z}\Big) + \varepsilon\, F_1\Big(t, \frac{t}{\varepsilon}, \mathbf{z}\Big),$$

where three points need to be specified: what is the meaning of the approximation, how are functions F and F_1 obtained, and what is precisely the dependence in t/ε of the functions F and F_1. We will see that F is the two-scale limit, derived from a function which does not contain high frequency oscillations. As for F_1, we will show that it can be written as the sum of two terms, one which is free of high frequency oscillations, whereas the second contains oscillations but is simply given by an explicit formula.

The reminder of this section is organized as follows. In the first part we build a **new two-scale macro-micro decomposition** for the equation in (1.1) on the basis of the previous approximation. In the second part, a **two-scale first order approximation** for equation (1.3) is proposed, together with **an algorithm** for finding its numerical solution, within a Particle-in-Cell (PIC) approach. I conclude this introduction by specifying that the basic ideas behind both parts of this section are mainly due to Emmanuel Frénod (Université de Bretagne-Sud, Vannes).

1.2.1 Two-scale asymptotic preserving model

The contribution of this part entails two points:

- a reformulation of the Vlasov problem in (1.1) with a macro-micro decomposition which separates the macro behaviour of the solution — without the fast rotation due to the magnetic field — from the micro behaviour.
- a convergence theorem, showing the asymptotic preserving property of this reformulation.

The aim in [12] is to make the first step towards the setting out of a new class of numerical methods: two-scale asymptotic preserving schemes. They are conceived to treat the following numerical difficulties: first, dealing efficiently on long time scales with solutions having high frequency oscillations (the *two-scale* approach) and second, an accurate and stable managing of the transition between different regimes, using a unified model (the *asymptotic preserving* approach). More precisely, we are interested in developing a model whose discretization is able to simulate both, the regime when the parameter is not small (*e.g.* the magnetic field is not large) and the limit regime obtained when the parameter is small. The discrete scheme automatically shifts from one regime to the other. In

[2]If f is a T-periodic function in a L^∞ space, then the sequence $(\phi_\varepsilon)_\varepsilon$ given by $\phi_\varepsilon = f(\cdot/\varepsilon)$ weak-$\star$ converges in L^∞ to the mean $\frac{1}{T}\int_0^T f$ when $\varepsilon \searrow 0$.
[3]variable $\mathbf{z}$ stands for both space and velocity variables.

addition, in the limit of the small parameter, the particle mean behaviour is efficiently described by the two-scale limit while the micro behaviour is expressed by some corrector. The ideas on which a **two-scale asymptotic preserving scheme** is based are illustrated in the diagram (1.5) below (see [12, Section 1] for the complete set of notations). Our problem (1.1) plays the role of the problem $E^\varepsilon\, u^\varepsilon = 0$ in the diagram.

$$\begin{tikzcd}
\begin{array}{c} u^\varepsilon \text{ solution of} \\ E^\varepsilon\, u^\varepsilon = 0 \end{array} \arrow[rrrr, "\varepsilon \to 0,\text{ weak-}*"] \arrow[drrr, "\varepsilon \to 0,\text{ two-scale}"] & & & & \begin{array}{c} u \text{ solution of} \\ E\, u = 0 \end{array} \\
& & & \begin{array}{c} U \text{ solution of} \\ \mathcal{E}\, U = 0 \end{array} \arrow[ur, "\frac{1}{|\mathcal{Z}|}\int_{\mathcal{Z}} d\zeta"'] & \\
& & \begin{array}{c} U^\varepsilon \text{ solution of} \\ \mathcal{E}^\varepsilon\, U^\varepsilon = 0 \end{array} \arrow[uull, "\zeta = \frac{z}{\varepsilon}"] \arrow[ur, "\varepsilon \to 0"'] & & \\
\begin{array}{c} u^\varepsilon_{\Delta z} \text{ solution of} \\ E^\varepsilon_{\Delta z}\, u^\varepsilon_{\Delta z} = 0 \end{array} \arrow[uuu, "\Delta z \to 0"] \arrow[rrrr, "\varepsilon \to 0"] & & & & \begin{array}{c} u_{\Delta z} \text{ solution of} \\ E_{\Delta z}\, u_{\Delta z} = 0 \end{array} \arrow[uuu, "\Delta z \to 0"'] \\
& & & \begin{array}{c} U_{\Delta z} \text{ solution of} \\ \mathcal{E}_{\Delta z}\, U_{\Delta z} = 0 \end{array} \arrow[uuu, "\Delta z \to 0"'] \arrow[ur, "\frac{1}{|\mathcal{Z}|}\int_{\mathcal{Z}}^{\boxed{\text{NUM}}} d\zeta"'] & \\
& & \begin{array}{c} U^\varepsilon_{\Delta z} \text{ solution of} \\ \mathcal{E}^\varepsilon_{\Delta z}\, U^\varepsilon_{\Delta z} = 0 \end{array} \arrow[uuu, "\Delta z \to 0"'] \arrow[uull, "\zeta = \frac{z}{\varepsilon}"] \arrow[ur, "\varepsilon \to 0"'] & &
\end{tikzcd} \tag{1.5}$$

There is a huge literature about asymptotic preserving schemes, with various applications (see for instance the review paper [Jin10]). We only cite here the classic papers [Kla99; Jin99] treating problems close to the ones we are interested in, and some works which build macro-micro decomposition methods [LM08; JS10; CL11]. However, these references did not consider the two-scale aspect.

The aim of the two-scale macro-micro decomposition is to build a **new model which is an equivalent reformulation of** (1.1) for all $\varepsilon > 0$ and which straightforwardly gives the two-scale limit model of (1.1) when $\varepsilon \to 0$. Before going further, we briefly recall existing results about the asymptotic behaviour of the solution of (1.1). Considering the initial condition and the electric and magnetic fields in appropriate spaces, the sequence $(f^\varepsilon)_\varepsilon$ two-scale converges to $F \equiv F(t, \tau, \mathbf{x}, \mathbf{v})$, a periodic function in τ[4] for every $(t, \mathbf{x}, \mathbf{v})$. It was shown [FS98] that the two-scale limit F expresses in terms of a function $G \equiv G(t, \mathbf{x}, \mathbf{u})$ by

$$F(t, \tau, \mathbf{x}, \mathbf{v}) = G\big(t, \mathbf{x}, \mathcal{R}(\tau)(\mathbf{v})\big), \tag{1.6}$$

[4]the additional rapid time variable

where $\mathcal{R}$ is the rotation matrix of angle τ around the magnetic field $\mathcal{M}$

$$\mathcal{R}(\tau) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(2\pi\tau) & -\sin(2\pi\tau) \\ 0 & \sin(2\pi\tau) & \cos(2\pi\tau) \end{pmatrix} \tag{1.7}$$

and G is the solution of the system

$$\begin{cases} \dfrac{\partial G}{\partial t} + \mathbf{u}_{||} \cdot \nabla_{\mathbf{x}} G + (\mathbf{E}_{||} + \mathbf{u} \times \mathbf{B}_{||}) \cdot \nabla_{\mathbf{u}} G = 0, \\ G(t=0, \mathbf{x}, \mathbf{u}) = f_0(\mathbf{x}, \mathbf{u}), \end{cases} \tag{1.8}$$

where we used the notation $\mathbf{u}_{||} = (\mathbf{u} \cdot \mathbf{e}_1)\, \mathbf{e}_1$.

Remark 1.2.1. *We remark that equation (1.8) for G is free of high frequency oscillations, and therefore explicit numerical schemes can be used without a high computational cost. However, the function F, approximating f^ε in the limit $\varepsilon \to 0$, still contains information about oscillations through the rotation matrix $\mathcal{R}$ (see (1.6)).*

Furthermore, under additional assumptions for the initial condition [FRS01], for small ε,

$$f^\varepsilon(t, \mathbf{x}, \mathbf{v}) \quad \text{is close to} \quad F\Big(t, \frac{t}{\varepsilon}, \mathbf{x}, \mathbf{v}\Big) \quad \text{in a strong sense,} \tag{1.9}$$

more precisely, if $[0, T]$ is a fixed interval of time,

$$\int_0^T \int_{\mathbb{R}^6} \Big| f^\varepsilon(t, \mathbf{x}, \mathbf{v}) - F\Big(t, \frac{t}{\varepsilon}, \mathbf{x}, \mathbf{v}\Big) \Big|^2 d\mathbf{x}\, d\mathbf{v}\, dt \to 0, \text{ when } \varepsilon \to 0.$$

In a similar way, it was shown in [FRS01] that the approximation in (1.9) may be improved to first order, claiming that for small ε,

$$\begin{aligned} f^\varepsilon(t, \mathbf{x}, \mathbf{v}) \quad \text{is close to} \qquad & G\Big(t, \mathbf{x}, \mathcal{R}\Big(\frac{t}{\varepsilon}\Big)(\mathbf{v})\Big) + \varepsilon\, G_1\Big(t, \mathbf{x}, \mathcal{R}\Big(\frac{t}{\varepsilon}\Big)(\mathbf{v})\Big) \\ & + \varepsilon\, l\Big(t, \frac{t}{\varepsilon}, \mathbf{x}, \mathbf{v}\Big). \end{aligned} \tag{1.10}$$

More precisely, the result asserts that

$$\frac{1}{\varepsilon}\bigg(f^\varepsilon(t, \mathbf{x}, \mathbf{v}) - G\Big(t, \mathbf{x}, \mathcal{R}\Big(\frac{t}{\varepsilon}\Big)(\mathbf{v})\Big)\bigg) \text{ two-scale converges to } F_1(t, \tau, \mathbf{x}, \mathbf{v}) \text{ as } \varepsilon \to 0, \tag{1.11}$$

where the function F_1 writes

$$F_1(t, \tau, \mathbf{x}, \mathbf{v}) = G_1\big(t, \mathbf{x}, \mathcal{R}(\tau)(\mathbf{v})\big) + l(t, \tau, \mathbf{x}, \mathbf{v}),$$

and where l is given by an explicit formula, while G_1 is the solution of an equation of the form

$$\frac{\partial G_1}{\partial t} + \mathbf{u}_{||} \cdot \nabla_{\mathbf{x}} G_1 + \big(\mathbf{E}_{||} + \mathbf{u} \times \mathbf{B}_{||}\big) \cdot \nabla_{\mathbf{u}} G_1 = RHS(t, \mathbf{x}, \mathbf{u}, \mathbf{E}, \mathbf{B}, G), \tag{1.12}$$

where the right-hand side can be explicitly computed as in Theorem 4.2 of [FRS01].

Remark 1.2.2. *We observe that the advection operator in equation (1.12) for G_1 is the same than that in equation (1.8) for G.*

Let us denote in the following, for any function $G \equiv G(t,\mathbf{x},\mathbf{u})$ and any τ

$$(G\circ\mathcal{R})(t,\tau,\mathbf{x},\mathbf{v}) = G\big(t,\mathbf{x},\mathcal{R}(\tau)(\mathbf{v})\big),$$

and for any function $\phi \equiv \phi(t,\tau,\mathbf{x},\mathbf{v})$

$$\begin{aligned}&(\phi\circ\mathcal{R})(t,\tau,\mathbf{x},\mathbf{v}) = \phi\big(t,\tau,\mathbf{x},\mathcal{R}(\tau)(\mathbf{v})\big) \text{ and}\\ &(\phi)^\varepsilon \equiv (\phi)^\varepsilon(t,\mathbf{x},\mathbf{v}) = \phi\Big(t,\frac{t}{\varepsilon},\mathbf{x},\mathbf{v}\Big).\end{aligned}$$

The two-scale macro-micro decomposition we proposed in [12] is inspired by (1.10) and writes

$$f^\varepsilon(t,\mathbf{x},\mathbf{v}) = G\Big(t,\mathbf{x},\mathcal{R}\Big(\frac{t}{\varepsilon}\Big)(\mathbf{v})\Big) + \varepsilon\, G_1^\varepsilon\Big(t,\mathbf{x},\mathcal{R}\Big(\frac{t}{\varepsilon}\Big)(\mathbf{v})\Big) + \varepsilon\, l\Big(t,\frac{t}{\varepsilon},\mathbf{x},\mathbf{v}\Big) + \varepsilon\, h^\varepsilon\Big(t,\frac{t}{\varepsilon},\mathbf{x},\mathbf{v}\Big), \tag{1.13}$$

where G_1^ε is intended to be close to G_1 when ε is small, and h^ε is the corrector to be taken into account when the order of magnitude of ε is 1. More precisely, we have obtained the following result for the decomposition of f^ε.

Theorem 1.2.3. *There exist the unique functions*

$$\begin{array}{rcccl} G_1^\varepsilon & \equiv & G_1^\varepsilon(t,\mathbf{x},\mathbf{u}) & \in & L^\infty\big(0,T;L^2(\mathbb{R}^6)\big) \text{ and}\\ h^\varepsilon & \equiv & h^\varepsilon(t,\tau,\mathbf{x},\mathbf{v}) & \in & L^\infty\big(0,T;L^\infty_{\#_1}(\mathbb{R}^+;L^2(\mathbb{R}^6))\big),\end{array} \tag{1.14}$$

such that for any τ

$$\boxed{f^\varepsilon(t,\mathbf{x},\mathbf{v}) = G\big(t,\mathbf{x},\mathcal{R}(\tau)(\mathbf{v})\big) + \varepsilon\, G_1^\varepsilon\big(t,\mathbf{x},\mathcal{R}(\tau)(\mathbf{v})\big) + \varepsilon\, l(t,\tau,\mathbf{x},\mathbf{v}) + \varepsilon h^\varepsilon(t,\tau,\mathbf{x},\mathbf{v}),} \tag{1.15}$$

where G *is the solution to equation* (1.8), l *is given by*

$$\begin{aligned} l(t,\tau,\mathbf{x},\mathbf{v}) \;=\;& \Big(\mathcal{R}(\tau+\frac14) - \mathcal{R}(\frac14)\Big)\,\mathbf{v}\cdot\nabla_{\mathbf{x}_\perp} G\big(t,\mathbf{x},\mathcal{R}(\tau)(\mathbf{v})\big)\\ &+ \Big[\Big(\mathcal{R}(\tau+\frac14) - \mathcal{R}(\frac14)\Big)\,\mathbf{E}(t,\mathbf{x}) + \mathcal{R}(\tau)(\mathbf{v})\times\Big(\big(\mathcal{R}(\tau+\frac14) - \mathcal{R}(\frac14)\big)\,\mathbf{B}(t,\mathbf{x})\Big)\Big]\\ &\quad\cdot\nabla_{\mathbf{u}_\perp} G\big(t,\mathbf{x},\mathcal{R}(\tau)(\mathbf{v})\big)^{5},\end{aligned} \tag{1.16}$$

and G_1^ε *and* h^ε *are such that*

$$\begin{aligned}(G\circ\mathcal{R}) + \varepsilon\,(G_1^\varepsilon\circ\mathcal{R}) &\in \operatorname{Ker}\Big(\frac{\partial}{\partial\tau} + (\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}\Big),\\ \varepsilon\, l + \varepsilon\, h^\varepsilon &\in \operatorname{Im}\Big(\frac{\partial}{\partial\tau} + (\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}\Big).\end{aligned} \tag{1.17}$$

In addition, there exists $k^\varepsilon \in L^\infty\big(0,T;L^\infty_{\#_1}(\mathbb{R}^+;L^2(\mathbb{R}^6))\big)$ *such that*

$$h^\varepsilon = \frac{\partial k^\varepsilon}{\partial\tau} + (\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}} k^\varepsilon \tag{1.18}$$

[5] We make use of the notation $\nabla_{\mathbf{x}_\perp} = \Big(0,\frac{\partial}{\partial x_2},\frac{\partial}{\partial x_3}\Big)^T$ and similarly for $\nabla_{\mathbf{u}_\perp}$.

and G_1^ε and k^ε are solutions to the following coupled variational problem: for any regular function γ, compactly supported in $[0,T)\times\mathbb{R}^3\times\mathbb{R}^3$ such that $\gamma\circ\mathcal{R}\in\mathrm{Ker}\Big(\dfrac{\partial}{\partial\tau}+(\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}\Big)$, G_1^ε satisfies

$$\begin{aligned}
&\int_0^T\int_{\mathbb{R}^6}G_1^\varepsilon\Big[\frac{\partial\gamma}{\partial t}+\big(\mathcal{R}(-\frac{t}{\varepsilon})\,\mathbf{u}\big)\cdot\nabla_{\mathbf{x}}\gamma+\Big[\mathcal{R}(\frac{t}{\varepsilon})\,\mathbf{E}+\mathbf{u}\times\big(\mathcal{R}(\frac{t}{\varepsilon})\,\mathbf{B}\big)\Big]\cdot\nabla_{\mathbf{u}}\gamma\Big]\,d\mathbf{x}\,d\mathbf{u}\,dt\\
&\quad-\int_0^T\int_{\mathbb{R}^6}\bigg(\frac{\partial(l\circ\mathcal{R}^{-1})}{\partial t}\bigg)^\varepsilon\gamma\,d\mathbf{x}\,d\mathbf{u}\,dt\\
&\quad+\int_0^T\int_{\mathbb{R}^6}\Big[\Big((\frac{\partial k^\varepsilon}{\partial\tau})\circ\mathcal{R}^{-1}\Big)^\varepsilon+\big(\mathcal{R}(-\frac{t}{\varepsilon})\,\mathcal{M}\big)\cdot\big((\nabla_{\mathbf{v}}k^\varepsilon)\circ\mathcal{R}^{-1}\big)^\varepsilon\Big]\,\frac{\partial\gamma}{\partial t}d\mathbf{x}\,d\mathbf{u}\,dt\\
&\quad+\int_0^T\int_{\mathbb{R}^6}\Big[(l\circ\mathcal{R}^{-1})^\varepsilon+\Big((\frac{\partial k^\varepsilon}{\partial\tau})\circ\mathcal{R}^{-1}\Big)^\varepsilon+\big(\mathcal{R}(-\frac{t}{\varepsilon})\,\mathcal{M}\big)\cdot\big((\nabla_{\mathbf{v}}k^\varepsilon)\circ\mathcal{R}^{-1}\big)^\varepsilon\Big]\\
&\qquad\qquad\cdot\Big[\big(\mathcal{R}(-\frac{t}{\varepsilon})\,\mathbf{u}\big)\cdot\nabla_{\mathbf{x}}\gamma+\big[\mathcal{R}(\frac{t}{\varepsilon})\,\mathbf{E}+\mathbf{u}\times\big(\mathcal{R}(\frac{t}{\varepsilon})\,\mathbf{B}\big)\big]\cdot\nabla_{\mathbf{u}}\gamma\Big]\,d\mathbf{x}\,d\mathbf{u}\,dt=0,^{6}
\end{aligned}\tag{1.19}$$

and for any regular function $\kappa\equiv\kappa(t,\tau,\mathbf{x},\mathbf{v})$ compactly supported in $[0,T)\times\mathbb{R}^3\times\mathbb{R}^3$ for every $\tau\in[0,1]$, k^ε satisfies

$$\begin{aligned}
&-\int_0^T\int_{\mathbb{R}^6}\bigg(\Big(\frac{\partial^2k^\varepsilon}{\partial t\,\partial\tau}\Big)^\varepsilon+(\mathbf{v}\times\mathcal{M})\cdot\Big(\frac{\partial\nabla_{\mathbf{v}}k^\varepsilon}{\partial t}\Big)^\varepsilon+\frac{1}{\varepsilon}\Big(\frac{\partial^2k^\varepsilon}{\partial\tau^2}\Big)^\varepsilon+\frac{1}{\varepsilon}(\mathbf{v}\times\mathcal{M})\cdot\Big(\frac{\partial\nabla_{\mathbf{v}}k^\varepsilon}{\partial\tau}\Big)^\varepsilon\bigg)\\
&\qquad\qquad\Big[\Big(\frac{\partial\kappa}{\partial\tau}\Big)^\varepsilon+(\mathbf{v}\times\mathcal{M})\cdot(\nabla_{\mathbf{v}}\kappa)^\varepsilon\Big]\,d\mathbf{x}\,d\mathbf{v}\,dt\\
&+\int_{\mathbb{R}^6}\Big[\frac{\partial k^\varepsilon}{\partial\tau}(0,0,\mathbf{x},\mathbf{v})+(\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}k^\varepsilon(0,0,\mathbf{x},\mathbf{v})\Big]\Big[\frac{\partial\kappa}{\partial\tau}(0,0,\mathbf{x},\mathbf{v})+(\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}\kappa(0,0,\mathbf{x},\mathbf{v})\Big]d\mathbf{x}d\mathbf{v}\\
&\quad+\int_0^T\int_{\mathbb{R}^6}\Big(\Big(\frac{\partial k^\varepsilon}{\partial\tau}\Big)^\varepsilon+(\mathbf{v}\times\mathcal{M})\cdot(\nabla_{\mathbf{v}}k^\varepsilon)^\varepsilon\Big)\Big[\mathbf{v}\cdot\Big(\frac{\partial\nabla_{\mathbf{x}}\kappa}{\partial\tau}\Big)^\varepsilon+(\mathbf{v}\times\mathcal{M})\cdot\big((\nabla_{\mathbf{x}}\nabla_{\mathbf{v}}\kappa)^\varepsilon\,\mathbf{v}\big)\\
&\qquad\qquad+(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\Big(\frac{\partial\nabla_{\mathbf{v}}\kappa}{\partial\tau}\Big)^\varepsilon+(\mathbf{E}\times\mathcal{M}+(\mathbf{v}\times\mathbf{B})\times\mathcal{M})\cdot(\nabla_{\mathbf{v}}\kappa)^\varepsilon\\
&\qquad\qquad+(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\big((\nabla_{\mathbf{v}}^2\kappa)^\varepsilon\,(\mathbf{v}\times\mathcal{M})\big)+\frac{1}{\varepsilon}(\mathbf{v}\times\mathcal{M})\cdot\Big(\frac{\partial\nabla_{\mathbf{x}}\kappa}{\partial\tau}\Big)^\varepsilon\\
&\qquad\qquad-\mathbf{v}\cdot(\nabla_{\mathbf{v}}\kappa)^\varepsilon+(\mathbf{v}\times\mathcal{M})\cdot\big((\nabla_{\mathbf{v}}^2\kappa)^\varepsilon(\mathbf{v}\times\mathcal{M})\big)\Big]\,d\mathbf{x}\,d\mathbf{v}\,dt\\
&\quad-\int_0^T\int_{\mathbb{R}^6}\Big(\frac{\partial(l\circ\mathcal{R}^{-1})}{\partial t}\Big)^\varepsilon\Big[\Big(\frac{\partial\kappa}{\partial\tau}\circ\mathcal{R}^{-1}\Big)^\varepsilon+\Big[\big(\mathcal{R}(-\frac{t}{\varepsilon})\,\mathbf{v}\big)\times\mathcal{M}\Big]\cdot\big((\nabla_{\mathbf{v}}\kappa)\circ\mathcal{R}^{-1}\big)^\varepsilon\Big]\,d\mathbf{x}\,d\mathbf{v}\,dt\\
&\quad+\int_0^T\int_{\mathbb{R}^6}(l)^\varepsilon\Big[\mathbf{v}\cdot\Big(\frac{\partial\nabla_{\mathbf{x}}\kappa}{\partial\tau}\Big)^\varepsilon+(\mathbf{v}\times\mathcal{M})\cdot\big((\nabla_{\mathbf{x}}\nabla_{\mathbf{v}}\kappa)^\varepsilon\,\mathbf{v}\big)+(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\Big(\frac{\partial\nabla_{\mathbf{v}}\kappa}{\partial\tau}\Big)^\varepsilon\\
&\qquad+(\mathbf{E}\times\mathcal{M}+(\mathbf{v}\times\mathbf{B})\times\mathcal{M})\cdot(\nabla_{\mathbf{v}}\kappa)^\varepsilon+(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\big((\nabla_{\mathbf{v}}^2\kappa)^\varepsilon\,(\mathbf{v}\times\mathcal{M})\big)\Big]\,d\mathbf{x}\,d\mathbf{v}\,dt\\
&\quad-\int_0^T\int_{\mathbb{R}^6}\Big(\frac{\partial F_1^\varepsilon}{\partial t}\Big)^\varepsilon\Big[\Big(\frac{\partial\kappa}{\partial\tau}\Big)^\varepsilon+(\mathbf{v}\times\mathcal{M})\cdot(\nabla_{\mathbf{v}}\kappa)^\varepsilon\Big]\,d\mathbf{x}\,d\mathbf{v}\,dt\\
&\quad+\int_{\mathbb{R}^6}F_1^\varepsilon(0,0,\mathbf{x},\mathbf{v})\Big[\frac{\partial\kappa}{\partial\tau}(0,0,\mathbf{x},\mathbf{v})+(\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}\kappa(0,0,\mathbf{x},\mathbf{v})\Big]\,d\mathbf{x}\,d\mathbf{v}
\end{aligned}$$

$$+\int_0^T\int_{\mathbb{R}^6}(F_1^\varepsilon)^\varepsilon\left[\mathbf{v}\cdot\left(\frac{\partial\nabla_{\mathbf{x}}\kappa}{\partial\tau}\right)^\varepsilon+(\mathbf{v}\times\mathcal{M})\cdot\left((\nabla_{\mathbf{x}}\nabla_{\mathbf{v}}\kappa)^\varepsilon\,\mathbf{v}\right)\right.$$
$$+(\mathbf{E}+\mathbf{v}\times\mathbf{B})\left(\frac{\partial\nabla_{\mathbf{v}}\kappa}{\partial\tau}\right)^\varepsilon+(\mathbf{E}\times\mathcal{M}+(\mathbf{v}\times\mathbf{B})\times\mathcal{M})\cdot(\nabla_{\mathbf{v}}\kappa)^\varepsilon \tag{1.20}$$
$$\left.+(\mathbf{E}+\mathbf{v}\times\mathcal{M})\cdot\left((\nabla_{\mathbf{v}}^2\kappa)^\varepsilon(\mathbf{v}\times\mathcal{M})\right)\right]d\mathbf{x}\,d\mathbf{v}\,dt=0,$$

where

$$F_1^\varepsilon(t,\tau,\mathbf{x},\mathbf{v})=\big(G_1^\varepsilon\circ\mathcal{R}\big)(t,\tau,\mathbf{x},\mathbf{v}).$$

The proof of the first part of Theorem 1.2.3 (see [12]) on the existence of functions G_1^ε and h^ε is based on the properties of the operator

$$\frac{\partial}{\partial\tau}+(\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}\,:\,L^\infty\big(0,T;L^\infty_{\#_1}(\mathbb{R}^+;L^2(\mathbb{R}^6))\big)\;\to\;L^\infty\big(0,T;L^\infty_{\#_1}(\mathbb{R}^+;L^2(\mathbb{R}^6))\big), \tag{1.21}$$

namely, that it is antisymmetric, non-bounded and with closed range and satisfies

$$\operatorname{Ker}\left(\frac{\partial}{\partial\tau}+(\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}\right)\oplus\operatorname{Im}\left(\frac{\partial}{\partial\tau}+(\mathbf{v}\times\mathcal{M})\cdot\nabla_{\mathbf{v}}\right)=L^\infty\big(0,T;L^\infty_{\#_1}(\mathbb{R}^+;L^2(\mathbb{R}^6))\big).$$

Then, in order to determine the equations for G_1^ε and h^ε, the strategy is the following: we put the decomposition (1.13) in a weak form of equation (1.1) with oscillating test functions. Then, using test functions in the kernel and in the range of the operator in (1.21), lead to the equations for G_1^ε and h^ε respectively.

Remark 1.2.4.

- *We are aware that the system (1.19)-(1.20) has not necessarily a unique solution.*
- *Under the hypotheses of uniqueness of the solutions to (1.19)-(1.20),* **the two-scale macro-micro problem**, *i.e., the system (1.8), (1.16), (1.19), (1.20) is equivalent to the original problem (1.1). This is due to the decomposition in (1.15).*
- *Problem (1.8), (1.16), (1.19), (1.20) plays the role of problem $\mathcal{E}^\epsilon U^\epsilon=0$ in diagram (1.5).*
- *The reformulation (1.8), (1.16), (1.19), (1.20) of the original problem has the advantage of decomposing the original solution f^ε into a macro part, $G\circ\mathcal{R}+\varepsilon\,G_1^\varepsilon\circ\mathcal{R}$, and a micro part, $\varepsilon l+\varepsilon h^\varepsilon$. Thus, the model is able to describe separately the evolution of oscillation at the macroscopic time scale (of $G\circ\mathcal{R}$ and $G_1^\varepsilon\circ\mathcal{R}$) which contains essential oscillation through $\mathcal{R}$, and the evolution of oscillation corrections (of l and h^ε) at the microscopic time scale.*

Eventually, the asymptotic behaviour of the model is studied. This is done by recalling the convergence result in (1.11), asserting that $G+\varepsilon G_1+\varepsilon l$ is the first-order approximation of f^ε. More precisely, on the base of Theorem 1.5 in [FRS01] we obtain the following two-scale convergence result.

Theorem 1.2.5. *We assume that $f_0\in L^2(\mathbb{R}^6)$, $\mathbf{E}\in W^{1,\infty}(\mathbb{R}^3)$, $\mathbf{B}\in W^{1,\infty}(\mathbb{R}^3)$, and $\partial l/\partial t$, $\nabla_{\mathbf{x},\mathbf{v}}l\in L^\infty\big(0,T;L^\infty_{\#_1}(\mathbb{R}^+;L^2(\mathbb{R}^6))\big)$. Then, when $\varepsilon\to0$, the solutions $G_1^\varepsilon\circ\mathcal{R}$ of (1.19) two-scale converge to $G_1\in L^\infty\big(0,T;L^2(\mathbb{R}^6)\big)$, the solution of (1.12). When $\varepsilon\to0$, the solutions k^ε of (1.20) two-scale converge to 0.*

[6]where $\mathcal{R}^{-1}(\tau)=\mathcal{R}(-\tau)$.

Theorem 1.2.5 allows us to conclude that our two-scale macro-micro model achieves the transition from the small ε regime to the one corresponding to $\varepsilon \sim 1$. In other words, the model satisfies the expected asymptotic behaviour symbolized by the arrows in the top layer of diagram (1.5).

1.2.2 First order two-scale method

The contribution of this part (see [14]) is to propose

- **a first order two-scale model** and
- **a numerical approach for it**,

in order to perform simulation of the long time evolution of f^ε given in equation (1.3) in the limit $\varepsilon \to 0$. In addition, the novelty with respect to existing mathematical literature is the consideration of an external focusing electric field of the form

$$E^\varepsilon(t,r) = E_0\Big(t, \frac{t}{\varepsilon}, r\Big) + \varepsilon E_1\Big(t, \frac{t}{\varepsilon}, r\Big) \quad \text{for small} \quad \varepsilon > 0, \tag{1.22}$$

where $E_{i=0,1} : (t,\tau,r) \mapsto E_{i=0,1}(t,\tau,r)$ are regular and periodic with respect to the rapid time variable τ. In a first step, we determine a two-scale first order approximation of f^ε of the type mentioned in the introduction of this section. In order to achieve this goal, we first recall the two-scale limit model obtained by means of the two-scale convergence in [FSS09]: $(f^\varepsilon)_\varepsilon$ two-scale converges when $\varepsilon \to 0$ to a function F which writes

$$F(t,\tau,r,v) = G(t, \mathcal{R}(\tau)(r,v)), \tag{1.23}$$

where the rotation operator $\mathcal{R}$ is defined similarly as before by

$$\mathcal{R}(\tau)(r,v) = \begin{pmatrix} \cos\tau & -\sin\tau \\ \sin\tau & \cos\tau \end{pmatrix} \begin{pmatrix} r \\ v \end{pmatrix}, \tag{1.24}$$

and $G \equiv G(t,q,u)$ is the solution to the problem

$$\begin{cases} \dfrac{\partial G}{\partial t} + \dfrac{1}{2\pi}\displaystyle\int_0^{2\pi} \mathcal{R}(\tau)\Big(0, E_0\big(t,\tau,\mathcal{R}(-\tau)_r(q,u)\big)\Big) d\tau \cdot \nabla_{q,u} G = 0, \\ G(t=0,q,u) = \dfrac{1}{2\pi} f_0(q,u). \end{cases} \tag{1.25}$$

Remark 1.2.6. *The equation for G is free of oscillations and contains, as expected, the average in the rapid time variable of the limit electric field E_0.*

Then, we build the first order approximation for f^ε under the form

$$f^\varepsilon(t,r,v) \approx G\Big(t, \mathcal{R}\Big(\frac{t}{\varepsilon}\Big)(r,v)\Big) + \varepsilon\left(G_1\Big(t, \mathcal{R}\Big(\frac{t}{\varepsilon}\Big)(r,v)\Big) + W\Big(t, \frac{t}{\varepsilon}, \mathcal{R}\Big(\frac{t}{\varepsilon}\Big)(r,v)\Big)\right), \tag{1.26}$$

for ε small, where $W \equiv W(t,\tau,r,v)$ is given by an explicit formula involving the field E_0 and the phase space derivatives of G, while $G_1 \equiv G_1(t,q,u)$ is the solution to a transport equation with the same advection operator as the one for G in (1.25) and a source term depending on E_0 and on phase space derivatives of G. More precisely, **the main result** of this section is the following.

Theorem 1.2.7. *Let $(f^\varepsilon)_\varepsilon$ be a sequence of solutions to Vlasov problem (1.3) with the electric field E^ε given by (1.22). There exist two functions $F : (t,\tau,r,v) \in [0,T) \times \mathbb{R} \times \mathbb{R}^2 \mapsto F(t,\tau,r,v)$ and $F_1 : (t,\tau,r,v) \in [0,T) \times \mathbb{R} \times \mathbb{R}^2 \mapsto F(t,\tau,r,v)$, 2π-periodic with respect to $\tau \in \mathbb{R}$, such that*

$$f^\varepsilon(t,r,v) \approx 2\pi\, F\left(t, \frac{t}{\varepsilon}, r, v\right) + \varepsilon\, 2\pi\, F_1\left(t, \frac{t}{\varepsilon}, r, v\right), \tag{1.27}$$

when ε is small. Moreover, regarding the zeroth order term, there exists a function $G : (t,q,u) \in [0,T) \times \mathbb{R}^2 \mapsto G(t,q,u)$ such that

$$F(t,\tau,r,v) = G(t, \mathcal{R}(\tau)(r,v)), \tag{1.28}$$

where $\mathcal{R}(\tau)$ is the rotation defined by (1.24) and G is solution to the problem (1.25). Regarding the first order term, there exist two functions $G_1 : (t,q,u) \in [0,T) \times \mathbb{R}^2 \mapsto G_1(t,q,u)$ and $W : (t,\tau,q,u) \in [0,T) \times \mathbb{R} \times \mathbb{R}^2 \mapsto W(t,\tau,q,u)$ such that

$$F_1(t,\tau,r,v) = G_1\big(t, \mathcal{R}(\tau)(r,v)\big) + W\big(t,\tau,\mathcal{R}(\tau)(r,v)\big), \tag{1.29}$$

where on the one hand, W is given by

$$\begin{aligned} W(t,\tau,q,u) \quad = \quad & \bigg[\frac{\tau}{2\pi}\int_0^{2\pi} \mathcal{R}(\sigma)\Big(0, E_0\big(t,\sigma,\mathcal{R}(-\sigma)_r(q,u)\big)\Big)\,d\sigma \\ & - \int_0^{\tau} \mathcal{R}(\sigma)\Big(0, E_0\big(t,\sigma,\mathcal{R}(-\sigma)_r(q,u)\big)\Big)\,d\sigma\bigg] \cdot \nabla_{q,u}G(t,q,u), \end{aligned} \tag{1.30}$$

and on the other hand, G_1 is solution to the problem

$$\begin{cases} \dfrac{\partial G_1}{\partial t}(t,q,u) + \dfrac{1}{2\pi}\displaystyle\int_0^{2\pi} \mathcal{R}(\tau)\Big(0, E_0\big(t,\tau,\mathcal{R}(-\tau)_r(q,u)\big)\Big)d\tau \cdot \nabla_{q,u}G_1(t,q,u) = \\ \quad \dfrac{1}{2\pi}\displaystyle\int_0^{2\pi}\bigg[\int_0^{\tau} \mathcal{R}(\sigma)\left(0, \frac{\partial E_0}{\partial t}(t,\sigma,\mathcal{R}(-\sigma)_r(q,u))\right)\,d\sigma \\ \qquad\qquad - \dfrac{\tau}{2\pi}\displaystyle\int_0^{2\pi} \mathcal{R}(\sigma)\left(0, \frac{\partial E_0}{\partial t}(t,\sigma,\mathcal{R}(-\sigma)_r(q,u))\right)\,d\sigma\bigg]d\tau \cdot \nabla_{q,u}G(t,q,u) \\ \quad + \bigg[\dfrac{1}{4\pi}\displaystyle\int_0^{2\pi} \nabla_{q,u}\mathcal{R}(\sigma)\Big(0, E_0\big(t,\sigma,\mathcal{R}(-\sigma)_r(q,u)\big)\Big)\,d\sigma \int_0^{2\pi} \mathcal{R}(\sigma)\Big(0, E_0\big(t,\sigma,\mathcal{R}(-\sigma)_r(q,u)\big)\Big)\,d\sigma \\ \qquad - \dfrac{1}{2\pi}\displaystyle\int_0^{2\pi}\Big(\nabla_{q,u}\mathcal{R}(\tau)\Big(0, E_0\big(t,\tau,\mathcal{R}(-\tau)_r(q,u)\big)\Big)\int_0^{\tau} \mathcal{R}(\sigma)\Big(0, E_0\big(t,\sigma,\mathcal{R}(-\sigma)_r(q,u)\big)\Big)\,d\sigma\Big)d\tau\bigg] \\ \qquad\qquad\qquad\qquad \cdot\nabla_{q,u}G(t,q,u)\,d\tau, \\ G_1(t=0,q,u) = 0. \end{cases} \tag{1.31}$$

Remark 1.2.8. *The rigorous meaning of the approximation in (1.27) is based on a two-scale convergence result of the sequence $\big((f^\varepsilon - F)/\varepsilon\big)_\varepsilon$ when $\varepsilon \to 0$ (see the proof of [14, Theorem 2.1]). It is important to emphasize that the expression for W and the equations for G and G_1 do not contain anymore high frequency oscillations in time (other that the oscillations through the rotation operator). Consequently, classic explicit numerical schemes for solving these equations are not constrained by small time step when solving the smallest scale, as for the original equation (1.3).*

In a second step, **we build numerical approximations** of the three functions in (1.26), by discretizing the equations (1.25), (1.30), and (1.31) on the base of a **particle method**. In this direction, the main steps of the algorithm we proposed are:

Algorithm 1.2.9.

(i) Compute at each time a particle approximation of G, by pushing in the phase space the numerical particles with respect to the advection operator in problem (1.25).

(ii) Compute an approximation of the gradient $\nabla_{q,u}G$ from the particle approximation of G which is compatible with the desired particle approximation of W.

(iii) Compute a particle approximation of W from equation (1.30).

(iv) Compute a particle approximation of the source term of the equation for G_1 and then a particle approximation of G_1 by solving problem (1.31) with the same advection operator as in problem (1.25).

The main difficulty in applying the previous steps of the algorithm consists in recovering approximations of the phase space derivatives of the unknowns from their particle approximation. Thus, we could utilize different convolution kernels as a regularization procedure, as in a Smoothed Particle Hydrodynamics (SPH) method [LL02; LL10]. This strategy is useful first, for approximating derivatives of the distribution function in any point of the space and second, for computing the approximated value of a function (like the electric field) in a particle by considering the contribution of the neighboring particles. Inspired by this idea, the solution we developed in [14] handles this issue in the following way.
First, we make use of the particle approximation of function G

$$G(t,q,u) = \sum_{k=1}^{N_p} \omega_k \, \delta\big(q - Q_k(t)\big)\, \delta\big(u - U_k(t)\big), \tag{1.32}$$

where δ is the Dirac mass, N_p is the number of macroparticles and $\big(Q_k(t), U_k(t)\big)$ is the position in phase space of macroparticle k which moves along a characteristic curve of the equation (1.25). Thus, the first step of the Algorithm 1.2.9 can easily be dealt with, as in [FSS09]. For the second step, however, we need to recover an approximation of the phase space derivatives of G. A regularization of the particle approximation (1.32) is thus needed. To this purpose, we introduce a regular function γ^α with support included in the interval $[0,\alpha]$, and such that the function $\gamma_k^\alpha : (q,u) \mapsto \gamma^\alpha\Big(\sqrt{(q-Q_k(t))^2 + (u-U_k(t))^2}\Big)$ has an integral with worth 1. A regularization of (1.32) is therefore given by

$$G(t,q,u) = \sum_{k=1}^{N_p} \omega_k \, \gamma^\alpha \left(\sqrt{(q-Q_k(t))^2 + (u-U_k(t))^2} \right). \tag{1.33}$$

Then we can compute numerical approximations of the space derivatives of G. Now we can handle the third step of the algorithm: first, we compute an approximation of the right-hand side in (1.30) which has the following shape

$$\sum_{k=1}^{N_p} \beta_k(t,\tau,q,u)\, (\gamma^\alpha)' \left(\sqrt{(q-Q_k(t))^2 + (u-U_k(t))^2} \right). \tag{1.34}$$

Then, we need to distribute the above approximation over the macroparticles in order to get W. Let the particle approximation of W be given by

$$W(t,\tau,q,u) = \sum_{k=1}^{N_p} \widetilde{\beta_k}(t,\tau)\, \delta\big(q - Q_k(t)\big)\, \delta\big(u - U_k(t)\big) \tag{1.35}$$

and regularized by

$$W(t,\tau,q,u) = \sum_{k=1}^{N_p} \widetilde{\beta_k}(t,\tau)\, \gamma^\alpha \left(\sqrt{(q - Q_k(t))^2 + (u - U_k(t))^2}\right). \tag{1.36}$$

Then we want that all the integrals of the sums in (1.34) and (1.36) over each subdomain of the phase space to be almost equal. From the numerical point of view, we want to verify this proximity only for the support C_i of functions γ_i^α that is

$$\forall i \in \{1,\dots,N_p\}, \qquad \int_{C_i} \sum_{k=1}^{N_p} \widetilde{\beta_k}(t,\tau)\, \gamma_k^\alpha(q,u)\, dq\, du = \int_{C_i} \sum_{k=1}^{N_p} \beta_k(t,\tau,q,u)\, \lambda_k^\alpha(q,u)\, dq\, du, \tag{1.37}$$

where $\lambda_k^\alpha : (q,u) \mapsto (\gamma^\alpha)' \left(\sqrt{(q - Q_k(t))^2 + (u - U_k(t))^2}\right)$. System (1.37) writes again

$$\forall i \in \{1,\dots,N_p\}, \qquad \sum_{k=1}^{N_p} \widetilde{\beta_k}(t,\tau) \int_{C_i} \gamma_k^\alpha(q,u)\, dq\, du = \int_{C_i} \sum_{k=1}^{N_p} \beta_k(t,\tau,q,u)\, \lambda_k^\alpha(q,u)\, dq\, du. \tag{1.38}$$

Therefore we can determine the coefficient $\widetilde{\beta_k}$ of the particle approximation of W up to the solution of the linear system (1.38). Finally, the last step of the algorithm is treated similarly as before in order to solve the equation for G_1.

The previous regularization approach can be quite heavy to implement as the size of the preceding systems to be solved is the number of particles. This number can be very large in order to achieve a given accuracy. More precisely, the terms in the right-hand side of the equations (1.30) and (1.31) might be small and need to be approximated with sufficient accuracy so that they actually bring improvement on the two-scale limit term on its own. In addition, in order to justify these efforts, an important point that needs to be dealt with in the future is to emphasize test cases where the two-scale limit model degenerates and thus, the first order approximation is of significant help.

As an alternative, I suggested to change the particle approach with a semi-Lagrangian method, in order to avoid the difficulties of regularization and thus benefit from a much simpler gradient calculation (see Section 1.4). This perspective could be further explored in the framework of a PhD thesis, for which I proposed a subject, in collaboration with Michel Mehrenberger (IRMA, Université de Strasbourg).

1.3 Design and development of large time step schemes for Vlasov and Vlasov–Poisson equations

Introduction. In this section we present a time-stepping method for dealing with highly oscillatory Vlasov and Vlasov–Poisson equations of the form (1.2), (1.3), and (1.4). We solve numerically these equations with a particle method, which necessitates the discretization of some ordinary differential equations, namely the characteristics of the Vlasov equations. Therefore, we are interested in ODEs of the following general type

$$\boldsymbol{y}'(t) = \frac{1}{\varepsilon} L\boldsymbol{y}(t) + F\big(t, \boldsymbol{y}(t)\big), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0, \tag{1.39}$$

where L is a linear operator and where F represents a nonlinear term. Numerous multiscale techniques have been designed to solve numerically such problems in different contexts, among which: a class of semi-implicit Runge–Kutta method proposed in [FR16], a two-scale formulation based integrator in [CLM13; Cro+17], Heterogeneous Multiscale Methods [E+07; Ari+09], and the stroboscopic averaging method [Cal+11]. The difficulty of equation (1.39) is its stiffness, namely that the solution evolves over two scales in time: rapid oscillations due to the linear term[7] and a slower evolution due to the nonlinear term. Our aim is to solve such equations with explicit schemes. Thus, any standard numerical scheme is limited to a very small time step, for solving the rapid period due the stiff linear term. The method we propose is based on an exponential integrator (see [HO10]). This approach consists in using the variation-of-constants formula

$$\boldsymbol{y}(t) = e^{\frac{t}{\varepsilon}L}\boldsymbol{y}_0 + \int_0^t e^{\frac{t-\tau}{\varepsilon}L} F\big(\tau, \boldsymbol{y}(\tau)\big)\, d\tau. \tag{1.40}$$

With the help of this formula we can first compute exactly the linear part. In all the examples we treat in the following, the operator e^L has an explicit form and thus, the stiff part can be computed analytically. Then, several methods may be built to derive approximations when integrating numerically the slower nonlinear term (see the exponential time differencing schemes in [CM02; HO10]). Nevertheless, such schemes do not allow a very large time step with respect to the oscillation in order to capture accurately the nonlinear term.

Alternatively, our algorithm proceeds with the numerical treatment of the integral part in (1.40) as follows: we solve the ODEs over one fast period using an explicit high order solver and then, **thanks to** (1.40), we compute **an approximation of the solution over a large whole number of periods**. This approximation holds under the assumption that, roughly speaking, the average of the fast behaviour of the particles does not change in long time. The second idea of the algorithm is to **find the right fast period of each particle** in order to achieve accuracy of the behaviour in long time. These ideas came to me from the work in [CM02] where thorough analysis of the accuracy of the numerical integrators for solutions on and off the slow manifold is performed.

In the following, I detail this method for the Vlasov models exposed above and I present the algorithm's **adaptations to two settings**. More precisely, if we admit that our basic algorithm has two main steps, a micro step and a macro step, the first setting's difficulty (in Section 1.3.1) asks for an improvement of the first step (*i.e.* of the motion at the smallest scale), while the second setting (in Section 1.3.2) for an improvement of the second step (*i.e.* of the motion at the greatest scale).

I conclude by mentioning that I implemented these schemes for all the following numerical tests in `Fortran`, in a standard PIC framework. Mainly, I developed particle codes for solving Vlasov and Vlasov–Poisson equations in $1d \times 1v$ and $2d \times 2v$ settings. The Poisson equations are solved by finite diference methods. I describe in Section 2.1 the steps I followed in order to implement a classic PIC code in the solving of a generic Vlasov–Poisson system.

[7]In our examples, the linear term entails rapid oscillation and not a rapid decay.

1.3.1 The method for test cases in $1d \times 1v$.

The contribution of this part is **to build a time-stepping scheme** for efficiently solving the Vlasov equations in (1.3) and (1.4): (i) in short times [16] and (ii) in long times [15] with an improved version. Before going into details, it is interesting to underline the multiscale aspect of the problem under consideration. When representing the solution of both equations by a beam of particles in the phase space, this beam evolves by rotating around the origin and in long times a bunch forms around the center of the beam from which filaments of particles are going out. The filaments are constituted of rapid particles whereas the particles in the center of the beam are slower. The challenge is to accurately capture these filaments with the numerical method, while keeping a reasonable computational cost. The scheme we propose is developed in a PIC framework for the Vlasov equations. Therefore we solve the corresponding characteristics

$$\begin{cases} \dfrac{\mathrm{d}R}{\mathrm{d}t} = \dfrac{1}{\varepsilon} V, \\ \dfrac{\mathrm{d}V}{\mathrm{d}t} = -\dfrac{1}{\varepsilon} R + E(t,R), \end{cases} \tag{1.41}$$

together with an initial condition, where the electric field is either given explicitly by $E(t,r) = -r^3$, or is computed on a grid from the Poisson equation. Even if the explicit Vlasov model is easy to solve, it is an important step in the validation of our scheme, since it allows to compute the error of the scheme alone, without the numerical errors of the Poisson equation. In the system above we should write $R^\varepsilon, V^\varepsilon$, and E^ε instead of R, V, and E respectively, because these quantities clearly depend on ε, but for simplicity, in the following we drop the index ε. Then, we remark that the dynamical system (1.41) can be rewritten as in (1.39) by denoting

$$\boldsymbol{y} = \begin{pmatrix} R \\ V \end{pmatrix}, \quad L = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \text{ and } F(t,\boldsymbol{y}) = \begin{pmatrix} 0 \\ E(t,R) \end{pmatrix}. \tag{1.42}$$

In this setting, the operator $e^{t\,L}$ entailing the stiff part is simply a rotation operator, like the one defined in (1.24). The aim of the exponential integrator introduced in [16] is **to solve** (1.41) **by using large time steps with respect to the typical period of oscillation**, which is about $2\pi\varepsilon$. More precisely, we fix a time step $\Delta t \gg 2\pi\varepsilon$ and determine the unique integer N and the unique real o such that

$$\begin{cases} \Delta t = N \cdot 2\pi\varepsilon + o, \\ 0 \le o < 2\pi\varepsilon. \end{cases} \tag{1.43}$$

Then, we apply the following algorithm to each particle (see Fig. 1.1 for an illustration).

Algorithm 1.3.1. Assume that $\boldsymbol{y}_n$ the solution of (1.41) at time t_n is given.

1. Compute $\boldsymbol{y}(t_n + 2\pi\varepsilon)$ by using a fine Runge–Kutta solver for (1.41) with initial condition $\boldsymbol{y}_n$.

2. Compute $\boldsymbol{y}\,(t_n + N \cdot 2\pi\varepsilon)$ by the rule

$$\boldsymbol{y}\,(t_n + N \cdot 2\pi\varepsilon) = \boldsymbol{y}_n + N\big(\boldsymbol{y}\,(t_n + 2\pi\varepsilon) - \boldsymbol{y}_n\big)$$

3. Compute $\boldsymbol{y}$ at time t_{n+1} by using a fine Runge–Kutta solver for (1.41) with initial condition $\boldsymbol{y}(t_n + N \cdot 2\pi\varepsilon)$.

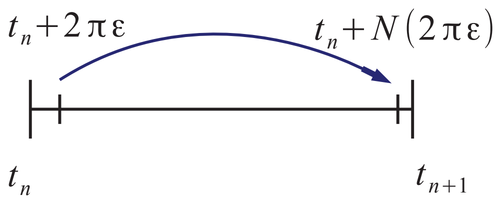


Figure 1.1: Illustration of Algorithm 1.3.1.

Using formula (1.40), this algorithm was formally proved in [16, Section 4] (see also Lemma 1.3.3) to provide an approximation of the solution under the assumptions that the time to make one rapid complete tour in the phase space is quasi constant and close to $2\pi\varepsilon$ and that the function $\tau \mapsto E(\tau, R(\tau))$ is about $2\pi\varepsilon$-periodic. Afterwards, the method was improved in order to be able to capture the small variations of the period. More precisely, **replacing $2\pi\varepsilon$ in Algorithm 1.3.1 by the mean of all particle times to make the first complete tour** resulted in more accurate results. This mean is computed by simply averaging the particles' periods (see Remark 1.3.4 below for explanations of how to compute numerically these periods). We give an account of the accuracy of the algorithm in Fig. 1.2, for several small values of ε, in two test-cases. In this figure we can remark the uniform convergence of the error with respect to ε.

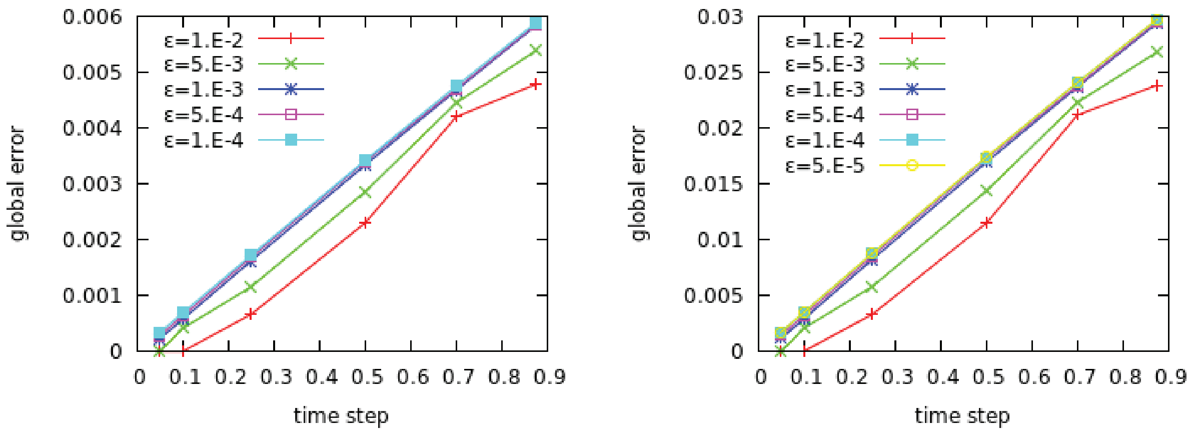


Figure 1.2: The Algorithm 1.3.1 with the particles' mean period: the global Euclidean error at time 3.5 for the Vlasov–Poisson case (left) and the case of (1.3) with $E(t, R) = -R^3$ (right).

However, we show in [15, Section 2] that this scheme still needs to be improved in order to perform accurate **long time simulations**. The idea is that using a more accurate period when pushing particles leads to better results. Precisely, the new scheme proposed in [15] differs from Algorithm 1.3.1 in that within the first step, instead of solving the ODE (1.41) during some fixed fast time for all the particles, **we push each particle with its own fast time that we need to compute**. Nevertheless, a naive implementation of this idea, following the steps of Algorithm 1.3.1, does not work in the Vlasov–Poisson case, basically because of the fact that all the particles are needed at some fixed time t for computing the self-consistent electric field. Therefore we adapt this algorithm by changing the steps so that the electric field can be computed. In order to introduce rigorously this idea, we denote for every $T > 0$, the T-periodic function

$$\boldsymbol{r}(\tau) = \begin{pmatrix} \cos\left(\frac{2\pi}{T}\tau\right) & \sin\left(\frac{2\pi}{T}\tau\right) \\ -\sin\left(\frac{2\pi}{T}\tau\right) & \cos\left(\frac{2\pi}{T}\tau\right) \end{pmatrix}. \tag{1.44}$$

The value T plays the role of a particle period at some fixed time. Then, after some easy calculations, we rewrite formula (1.40) giving the solution of the ODE under the following form: for any times s, t, such that $s < t$, we have

$$\boldsymbol{y}(t) = \boldsymbol{r}\Big(\frac{t-s}{\varepsilon}\Big)\boldsymbol{y}(s) + \boldsymbol{r}\Big(\frac{t-s}{\varepsilon}\Big)\int_s^t \boldsymbol{r}\Big(\frac{s-\tau}{\varepsilon}\Big)\boldsymbol{\beta}(\tau, \boldsymbol{y}(\tau))\,d\tau, \tag{1.45}$$

where we denoted

$$\boldsymbol{\beta}(\tau, \boldsymbol{y}) = F(\tau, \boldsymbol{y}) + \Big(1 - \frac{2\pi}{T}\Big)\frac{1}{\varepsilon} L\,\boldsymbol{y}.$$

Thus, if N_p is the number of particles, we compute for each $i \in \{1, \dots, N_p\}$, $\boldsymbol{y}_i^{n+1}$ from $\boldsymbol{y}_i^n$ as below (see Fig. 1.3).

Algorithm 1.3.2. Assume that for each $i \in \{1, \dots, N_p\}$, $\boldsymbol{y}_i^n$, the solution of (1.41) at time t_n is given.

1. Compute from $(\boldsymbol{y}_i^n)_{i\in\{1,\dots,N_p\}}$ the periods $(T_i^n)_{i\in\{1,\dots,N_p\}}$ at time t_n.

2. For each $i \in \{1, \dots, N_p\}$ compute the unique positive integers N_i^n and the unique reals $o_i^n \in [0, T_i^n)$ such that
$$\Delta t = N_i^n T_i^n + o_i^n.$$

3. For each $i \in \{1, \dots, N_p\}$ compute $\boldsymbol{y}_i(t_n + o_i^n)$ and $\boldsymbol{y}_i(t_n + o_i^n + T_i^n)$ by using a fine Runge–Kutta solver with initial condition $\boldsymbol{y}_i^n$.

4. For each $i \in \{1, \dots, N_p\}$ compute an approximation of $\boldsymbol{y}_i(t_{n+1})$ thanks to
$$\boldsymbol{y}_i^{n+1} = \boldsymbol{y}_i(t_n + o_i^n) + N_i^n\Big(\boldsymbol{y}_i(t_n + o_i^n + T_i^n) - \boldsymbol{y}_i(t_n + o_i^n)\Big).$$

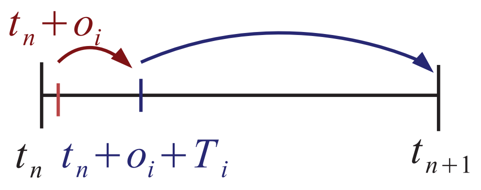


Figure 1.3: Illustration of Algorithm 1.3.2.

Algorithm 1.3.2 is justified formally by the following lemma.

Lemma 1.3.3. *Under the assumption that for every $i \in \{1, \dots, N_p\}$, $\boldsymbol{y}_i$ satisfies for every $n \in \mathbb{N}$*

$$\int_t^{t+N_i^n T_i^n} \boldsymbol{r}_i^n \left(\frac{t-\tau}{\varepsilon}\right) \boldsymbol{\beta}\big(\tau, \boldsymbol{y}_i(\tau)\big)\, d\tau \approx N_i^n \int_t^{t+T_i^n} \boldsymbol{r}_i^n \left(\frac{t-\tau}{\varepsilon}\right) \boldsymbol{\beta}\big(\tau, \boldsymbol{y}_i(\tau)\big)\, d\tau, \tag{1.46}$$

where $t = t_n + o_i^n$ and $\boldsymbol{r}_i^n$ corresponds to matrix in (1.44) with $T = T_i^n/\varepsilon$, we have for any time t_n the following formal approximation

$$\boldsymbol{y}_i(t_{n+1}) \approx \boldsymbol{y}_i(t_n + o_i^n) + N_i^n \Big(\boldsymbol{y}_i(t_n + o_i^n + T_i^n) - \boldsymbol{y}_i(t_n + o_i^n)\Big), \tag{1.47}$$

where $\boldsymbol{y}_i$ is the solution to equation (1.41) with a given initial condition $\boldsymbol{y}_i(t_n) = \boldsymbol{y}_i^n$.

This lemma is easily proved by succesive applications of formula (1.45).

Remark 1.3.4. *We give now some details about how to compute numerically the values of $(T_i^n)_{i\in\{1,\dots,N_p\}}$ and of the solution $\boldsymbol{y}_i$ at times like $t_n + o_i^n$.*

- *As explained in [16], the way to compute the $(T_i^n)_i$ from $(\boldsymbol{y}_i^n)_i$ is the following. First, we solve numerically the equation (1.41) with a time step $(\Delta t)_{\mathrm{RK}} = 2\pi\varepsilon/100$, for all initial conditions $\boldsymbol{y}_i^n$ until all the particles reach their trajectory's third extremum. The criterion for finding these extrema is the velocity's change of sign. We then state that each particle's period is the time interval between the first and the third extremum.*

- *The time step $(\Delta t)_{\mathrm{RK}}$ chosen in the third step of the algorithm is $2\pi\varepsilon/100$. It is easy to see that the reals $(t_n + o_i^n)_i$ and $(t_n + o_i^n + T_i^n)_i$ are located on a few cells of length $(\Delta t)_{\mathrm{RK}}$, since the periods of the particles vary a little. Thus, we approximate the values $\boldsymbol{y}_i(t_n + o_i^n)$ and $\boldsymbol{y}_i(t_n + o_i^n + T_i^n)$ by quadratic interpolations from their values at the multiples of $(\Delta t)_{\mathrm{RK}}$.*

Conclusion. We tested numerically this method in two cases: the Vlasov–Poisson equation and the case of (1.3) with $E(t,R) = -R^3$, see Fig. 1.4. Similar results were obtained for other larger values of ε. We note that the numerical solutions in the second case are very accurate since precise values for the periods are available and in addition they do not change in time. On the contrary, in the Vlasov–Poisson case, the periods are numerically approximated at several levels and therefore the scheme does not perform as well as for the other case. Nevertheless, the filaments of particles are accurately captured. In fact, our scheme simulates with the same accuracy both types of particles, rapid and slow. In addition, the advantage of this new method is to reproduce the stiff behaviour of the solution with a computational cost which is rather close to that of a reduced model.

1.3.2 The method for test cases in $2d \times 2v$

The contribution of [17] is to adapt the previously described time-stepping scheme to the equation in (1.2). The difficulty of the present model is that the dynamics of some particles entails a rapid circular motion coupled to a drift motion which is much slower; one can see this slower motion on the guiding center trajectory, illustrated in green in Fig. 1.5. There was no drift motion in the models in (1.3) or (1.4). Since we work with the PIC method, the ingredient is again the numerical solving of the characteristic equations

$$\begin{cases} \dfrac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathbf{V}, \\ \dfrac{\mathrm{d}\mathbf{V}}{\mathrm{d}t} = \dfrac{1}{\varepsilon}\mathbf{V}^\perp + \mathbf{E}(t, \mathbf{X}). \end{cases} \tag{1.48}$$

We first consider a given external electric field

$$\mathbf{E}(t,\mathbf{x}) = \begin{pmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{pmatrix}, \tag{1.49}$$

and in a second stage, a coupling with the Poisson equation. Even if the linear Vlasov model seems simple, it is an important step in the validation of our scheme, since it allows to compute the error with respect to an analytical solution. In addition, in the case of this model we can compute explicitly solutions which entail or not fast oscillations in time, allowing therefore to show the accuracy of our method with respect to **any** solution.

Remark 1.3.5. *In [17] we computed the analytic solution of the system (1.48) with (1.49) for a given initial condition. We have shown that the solution belongs to the vector space $\boldsymbol{F} = \mathcal{F}\times\mathcal{F}\times\mathcal{F}\times\mathcal{F}$, where*

$$\mathcal{F} = \operatorname{vect}\{\cos(a_\varepsilon t), \sin(a_\varepsilon t), \cos(b_\varepsilon t), \sin(b_\varepsilon t)\},$$

and where a_ε and b_ε are numbers depending of ε such that $a_\varepsilon \sim \sqrt{3}\,\varepsilon$ and $b_\varepsilon \sim 1/\varepsilon$. We have thus obtained that the solution to (1.48)-(1.49) contains fast oscillations in time of period $\frac{2\pi}{b_\varepsilon} \sim 2\pi\varepsilon$ and slow oscillations of period $\frac{2\pi}{a_\varepsilon} \sim \frac{2\pi}{\sqrt{3}\varepsilon}$. In this framework, we identified the solutions varying only at the slow time scale, the so-called slow manifold. It is the space $\boldsymbol{G} = \mathcal{G}\times\mathcal{G}\times\mathcal{G}\times\mathcal{G}$ where

$$\mathcal{G} = \operatorname{vect}\{\cos(a_\varepsilon t), \sin(a_\varepsilon t)\}.$$

Therefore, we are able to test the scheme we propose below for solutions having different oscillatory behaviours (see [CM02] for extensive numerical experiments in this direction). Thus, we identify the following three initial conditions leading to solutions which are respectively not oscillating, slowly oscillating, and highly oscillating:

$$\begin{aligned}
(\mathbf{x}_0^1, \mathbf{v}_0^1) &= \left(1, 0, \varepsilon, -\frac{2\varepsilon u_\varepsilon}{2-\varepsilon^2} + \frac{\varepsilon^3 u_\varepsilon}{2-\varepsilon^2}\right), \\
(\mathbf{x}_0^2, \mathbf{v}_0^2) &= \left(1, -\frac{u_\varepsilon}{w_\varepsilon}, \varepsilon\frac{w_\varepsilon}{u_\varepsilon}, -\frac{2\varepsilon u_\varepsilon}{2-\varepsilon^2} + \varepsilon\frac{w_\varepsilon}{u_\varepsilon} + \frac{\varepsilon^3 u_\varepsilon}{2-\varepsilon^2}\right), \\
(\mathbf{x}_0^3, \mathbf{v}_0^3) &= (1,1,1,1)\,.
\end{aligned} \tag{1.50}$$

The stiffness in (1.48) comes from the velocity equation and therefore we consider the exponential integrator only in velocity. Thus, the scheme is based, as in the previous section, on the variation-of-constants formula: if $s < t$ then we have

$$\mathbf{V}(t) = e^{\frac{t-s}{\varepsilon}L}\mathbf{V}(s) + e^{\frac{t-s}{\varepsilon}L}\int_s^t e^{\frac{s-\tau}{\varepsilon}L}\mathbf{E}(\tau, \mathbf{X}(\tau))\,d\tau,$$

while the position equation is simply given by

$$\mathbf{X}(t) = \mathbf{X}(s) + \int_s^t \mathbf{V}(\tau)\,d\tau.$$

In this setting, **the algorithm we propose in this section** is Algorithm 1.3.1 with $\boldsymbol{y} = \begin{pmatrix} \mathbf{X} \\ \mathbf{V} \end{pmatrix}$.

Remark 1.3.6. *We have discussed in Section 1.3.1 that using very accurate periods in the first step of the algorithm is an important issue in order to reduce the error of this step. Nevertheless, in this section, our tests revealed that computing each particle's period does not offer a significant improvement of the accuracy and thus, the use of $2\pi\varepsilon$ as the time period for each particle is enough*[8]*. It is for this reason that in this section, we use Algorithm 1.3.1 and not one like Algorithm 1.3.2.*

[8]The value $2\pi\varepsilon$ is the time period of the trajectory when $\mathbf{E} \equiv 0$ in (1.48).

Next, we formally justify that our algorithm provides approximation of the solution of (1.48). With the notations in (1.42) and (1.43) we have

Lemma 1.3.7. *Under the assumptions that for every t_n*

$$\int_{t_n}^{t_n+N\cdot(2\pi\varepsilon)} e^{\frac{t_n-\tau}{\varepsilon}L}\,\mathbf{E}\big(\tau,\mathbf{X}(\tau)\big)\,d\tau \approx N\cdot\int_{t_n}^{t_n+2\pi\varepsilon} e^{\frac{t_n-\tau}{\varepsilon}L}\,\mathbf{E}\big(\tau,\mathbf{X}(\tau)\big)\,d\tau \tag{1.51}$$

and

$$\int_{t_n}^{t_n+N\cdot(2\pi\varepsilon)} \mathbf{V}(\tau)\,d\tau \approx N\cdot\int_{t_n}^{t_n+2\pi\varepsilon} \mathbf{V}(\tau)\,d\tau, \tag{1.52}$$

we obtain the following formal approximation for the solution of (1.48) at time $t_n+N\cdot(2\pi\varepsilon)$ from its values at times t_n and $t_n+2\pi\varepsilon$:

$$\begin{pmatrix} \mathbf{X}\left(t_n+N\cdot(2\pi\varepsilon)\right) \\ \mathbf{V}\left(t_n+N\cdot(2\pi\varepsilon)\right) \end{pmatrix} \approx \begin{pmatrix} \mathbf{X}_n \\ \mathbf{V}_n \end{pmatrix} + N\cdot\begin{pmatrix} \mathbf{X}\left(t_n+2\pi\varepsilon\right)-\mathbf{X}_n \\ \mathbf{V}\left(t_n+2\pi\varepsilon\right)-\mathbf{V}_n \end{pmatrix}. \tag{1.53}$$

To show the reliability of the approximations in the hypotheses of Lemma 1.3.7, we compute the exact errors of these approximations in the case of the electric field given by (1.49). This is reported in [17, Section 3.3].

The method is validated through numerical experiments in three test cases: a linear Vlasov equation where the electric term in (1.2) is given by (1.49), a short time and respectively a long time Vlasov–Poisson problem. For the first case, we compute the errors of the scheme against the analytical solution in short and long times, while for the second case against a reference solution (*i.e.* a numerical solution computed with a very small time step), and for the third case we validate the scheme through an asymptotic result by comparing it to the guiding center solution (more precisely, we discuss in Section 2.1 about this asymptotic result linking the model in (2.3) to the limit guiding center model in (2.4)).

In the linear case the scheme is **uniformly accurate when ε vanishes** (see Fig. 1.6). We discuss hereafter the results concerning the behaviour of the global error for the Vlasov–Poisson problem, illustrated in Fig. 1.7 and Tables 1.1 & 1.2. We recall that N is the integral number of the rapid full tours appearing in the second step of Algorithm 1.3.1 (see (1.43)).

1. First, in Fig. 1.7 (left panel), we can see that for each fixed ε, the error decreases with decreasing time step, although N is changing. Thus, the scheme works for small time steps compared to the fast oscillation and is robust with respect to N. Second, we observe that, the smaller ε is, the smaller the error is, despite that N is significantly increasing when ε is smaller (see Table 1.1). This is an expected behaviour since the guiding center model becomes a better approximation of the long time Vlasov–Poisson model when ε goes to 0. Thus, the scheme works for large time steps with respect to the fast oscillation too.

2. In Table 1.2 and the right panel in Fig. 1.7, we detail the above comments by taking several values of ε. First, for similar reasons as before, when the time step is kept fixed, the error decreases with decreasing ε. Second, we notice once again the robustness of the scheme: the errors are stable when N is widely varying from 7 to 1 591.

3. In conclusion, the left panel in Fig. 1.7 shows that our scheme is convergent when $\Delta t\to 0$, uniformly in ε. Also, the right panel shows that the discretization of the solution to the long time Vlasov–Poisson system converges when $\varepsilon\to 0$ to a discretization of the guiding center model, independently of Δt. These facts underline the **asymptotic preserving behaviour of the scheme**.

	Δt=1E-3	Δt=3E-3	Δt=5E-3	Δt= 7E-3	Δt=9E-3	Δt=1E-2
$\varepsilon = 5.$E-3	6	19	31	44	57	63
$\varepsilon = 2.5$E-3	25	76	127	178	229	254
$\varepsilon = 1.$E-3	159	477	795	1 114	1 432	1 591
$\varepsilon = 1.$E-4	15 915	47 746	79 577	111 408	143 239	159 154

Table 1.1: The whole number of rapid full tours enclosed in a time step of the scheme, for several values of the time step and of ε; results corresponding to the left panel in Fig. 1.7.

	$\varepsilon = 1$E-3	$\varepsilon = 3$E-3	$\varepsilon = 5$E-3	$\varepsilon =$ 7E-3	$\varepsilon = 9$E-3	$\varepsilon = 1$E-2
Δt=1.E-2	1591	176	63	32	19	15
Δt=5.E-3	795	88	31	16	9	7

Table 1.2: The whole number of rapid full tours enclosed in a time step of the scheme, for several values of ε and of the time step; results corresponding to the right panel in Fig. 1.7.

Eventually, the algorithm described above was compared in [10] to three other existing multiscale techniques for solving the Vlasov–Poisson model corresponding to (1.2): an asymptotic preserving Runge–Kutta scheme [FR16], an integrator based on a two-scale formulation [Cro+17], and a stroboscopic averaging method [Cal+11]. We performed extensive numerical experiments in order to investigate and compare the accuracy, the computer time and the long time behaviour of these methods for a wide range of the parameter ε, from the value of 1 (the classical regime) to very small values (the limit regime). I do not develop these results here since there is not one numerical scheme giving the highest scores to all the questions precised above.

1.4 Conclusions and outlook

Conclusions. In this chapter, I developed two different strategies allowing to deal with Vlasov equations with several scales in time.

On one hand, I developed homogenization techniques like the two-scale convergence, in order to build new reduced models which approximate multiscale Vlasov equations. The result in Theorem 1.2.3 gives the form (and thus shows its complexity) of a unified model able to decompose highly oscillating parts of the solution from those which are slowly varying and thus to manage the transition from the regime $\varepsilon \ll 0$ to the regime $\varepsilon \sim 1$. Therefore, in Section 1.2.2, I turned over the consideration of a reduced model, whose advantage over the full Vlasov model is, clearly, that it is free of the smallest scale and thus has a lower computational cost. In addition, this framework displayed compelling theoretical properties (*e.g.* the result on the decomposition of the first order term in Theorem 1.2.7) that would be interesting to explore numerically in the future.

On the other hand, a second strategy consists in proposing a new method for advancing in time over the different scales: it uses macroscopic time steps but solves exactly the smallest scale at each macroscopic time step. I showed this scheme to provide very accurate numerical results with respect to the smallest scale, for the considered test cases. The strengths of the method are that it is very simple, robust, and efficient. However, it would be interesting to make the mathematical analysis of the method and (in order) to generalize it in view of its use to more wide application cases.

Perspectives. Concerning the results in Section 1.2.1, it seems difficult to develop and implement a numerical framework for the equations (1.19)-(1.20) in order to find the correctors G_1^ε and k^ε. However, the mathematical analysis developed in Theorem 1.2.3 is interesting for the reasons exposed above. Its novelty is mainly that it gives explicitly the form of the terms to be considered in the regime $\varepsilon \sim 1$. In addition, this result was at the root of the problem developed in Section 1.2.2. Thus, a first direction that I want to pursue in the future is to go further in the discretization and the implementation of the equations (1.25), (1.30), and (1.31), concerning the two-scale first order approximation in Theorem 1.2.7. More precisely, the challenge is to find and then to implement the appropriate numerical method which offers, with a not very high computational cost, the accuracy needed for capturing the (small) first-order corrector terms. In this way, there are two approaches that deserve to be developed (in the framework of a PhD thesis, for example):

- The **Particle-in-Cell** numerical approach. As already mentioned in Section 1.2, the difficulty is in finding the particular approximations for the derivatives of the distribution functions and the particular approximation for the function given by explicit formula, see Algorithm 1.2.9. To tackle this issue, a smoothed-particle hydrodynamics (SPH) method could be used [LL02; LL10]. In addition, it is known that SPH methods have advantages such as the numerical conservation of physical quantities, the ability to simulate phenomena occurring over several scales, and its resolution can easily be adapted with respect to variables like the particle density. One drawback is the large number of particles that might be needed for highly accurate simulations. Nevertheless, a high performance computing aspect (that I will develop for my PIC implementations in the next chapter) would play an important role and use of many core processors might be envisaged for dealing with this issue (for example, in the fluid dynamics community the SPH methods already ran over GPUs [HKK07]).

- A second approach for the two-scale models, would be to implement the Algorithm 1.2.9 in a **semi-Lagrangian** framework. In such case, gradients of functions are simply computed numerically, by approximating the derivative of the distribution function from its grid values in space or velocity. In this way, we would be able to obtain in a simple way the required accuracy on the numerical approximations in order to be able to capture the improvement of the first order (corrector) terms over the limit term. In this direction, we can be inspired by the work in [Mou09] where a backward semi-Lagrangian method (with an additional technique of a new phase space mesh) was implemented for the simulation of a two-scale limit model.

In addition, it is important to identify test cases, other than that in [14], where the use of the first order approximation might provide considerable additional knowledge to that of the two-scale limit. In this direction, I think that applying the two-scale approach to the $2d \times 2v$ model in (1.2) might be an interesting issue to explore.

As **an alternative** to the research direction above, it will be interesting to continue working on the subject developed in the second part of the chapter, especially for the model treated in Section 1.3.2. I first recall that the approximation of the **slow behaviour** in the Algorithm 1.3.1, namely step 2, assumes roughly that the guiding center of the particle motion evolves locally over a linear trajectory. Therefore, an interesting subject to pursue would be **to improve this approximation**. In this direction, I already tested the multiscale Backward-Forward Heterogeneous Multiscale Methods (BFHMM) in [Ari+13] on the highly oscillatory dynamical systems issued from the linear Vlasov equations, and the first outcomes were encouraging. Next, it is meaningful to consider the problem of dynamical systems where the stiff part is no more linear. This issue entails the case of a Vlasov equation with a strong magnetic field which is not homogeneous anymore. I implemented the algorithm in such a case, however the results were not satisfactory from the accuracy viewpoint, because the BFHMM scheme is able to follow only the slow variables of the motion, like the averaged motion of the guiding center, but not the guiding center. Therefore further refinements regarding knowledge of the rapid part are required and will be subject of my future research.

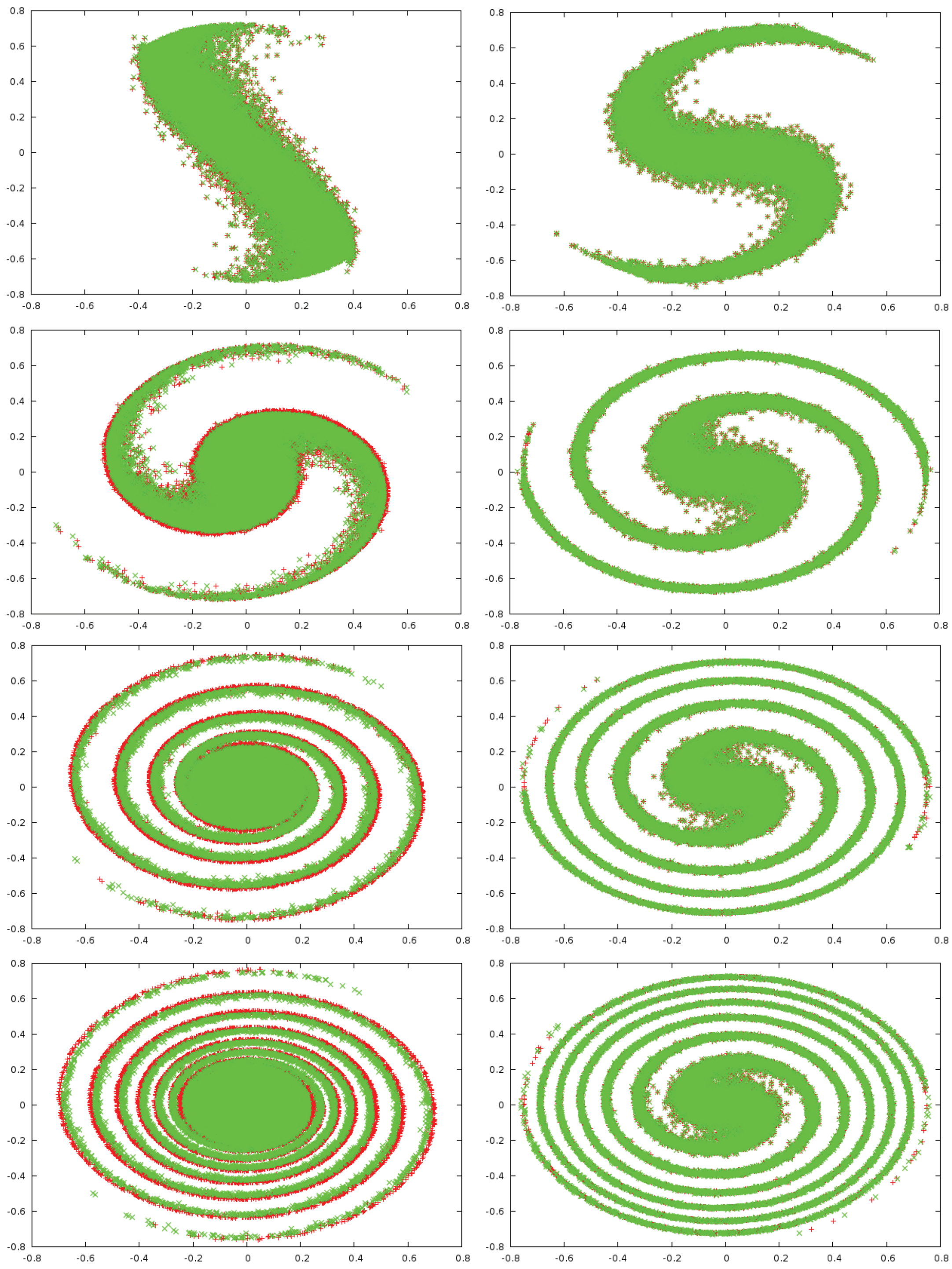


Figure 1.4: The cases of Vlasov–Poisson (left) and of Vlasov with $E(t, R) = -R^3$ (right), when $\varepsilon = 0.0001$. In red: the reference solution. In green: the solution obtained with Algorithm 1.3.2. From top to bottom the final times are $t = 10$, $t = 30$, $t = 60$, and $t = 90$. The time step for Algorithm 1.3.2 is $\Delta t = 0.5$.

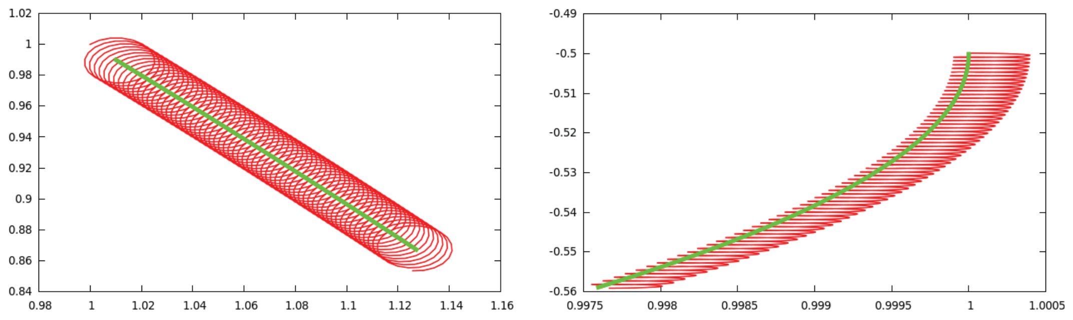


Figure 1.5: Solving (1.48)-(1.49) in the case when $\varepsilon = 0.01$. The initial positions and velocities $(\mathbf{x}_0^3, \mathbf{v}_0^3)$ (left) and $(\mathbf{x}_0^2, \mathbf{v}_0^2)$ (right) are introduced in (1.50). The evolution of the Guiding Center is in green and that of the position in red. The final time is $t = 4$.

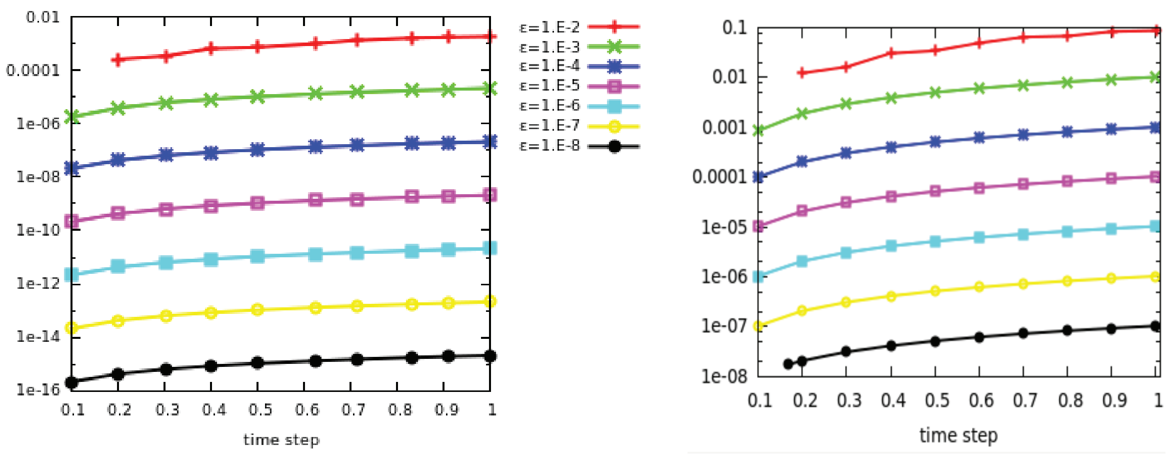


Figure 1.6: Global Euclidean error of the scheme at final time 10 (left) and at final time $1/\varepsilon$ (right), obtained with a particular initial condition in the linear Vlasov case, for several values of ε.

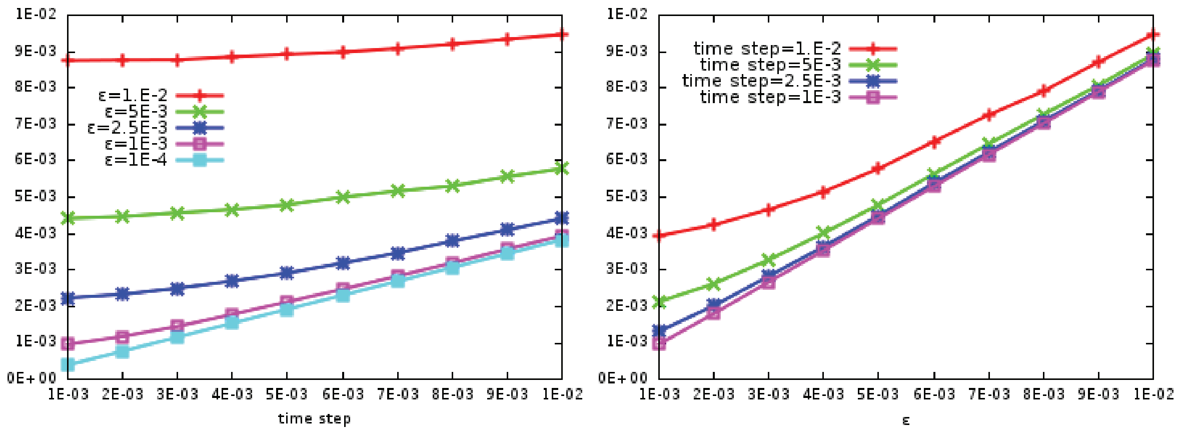


Figure 1.7: The difference between the solution computed with Algorithm 1.3.1 and the solution to the guiding center equation as a function of the time step (left) and of ε (right) at final time $t = 5$.

Chapter 2

Performance of Particle-in-Cell simulations for Vlasov–Poisson models

The goal of this chapter is to present our contributions in developing efficient methods for using advanced computing capabilities in order to perform simulations that exhibit computation and data intensive issues. For several mathematical or physical reasons, as for instance accurately simulating multiscale phenomena, gaining insights in understanding complex behaviours, or reproducing the realism of a physical problem, we are faced with the difficulty of processing a huge amount of data. Therefore, a first idea for dealing with this problem is to optimize the storage, the intrinsic communications, and the computations over these data. Afterwards, it is advantageous to use in parallel as many computational units as possible.

The models considered in this chapter are of Vlasov–Poisson type, for solving classic problems in plasma physics, in four and six dimensions of the phase space. The numerical approach is based on the Particle-in-Cell (PIC) method. Standard explicit numerical schemes are implemented leading to the constraint of resolving the smallest scale in space or time for the numerical parameters. It is important to stress that a naive implementation of these schemes and data involves memory bottlenecks in the PIC framework and thus requires large execution times when long runs are needed.

In this context, the contribution in this chapter is to address the problem of performance of a PIC code, by studying specific data structures to optimize memory access patterns and by exploiting three well-known types of parallelism on multi-core processors. This work was chronologically developed in the references [5], [3], [4], and [1] respectively. The summary is the following:

- First, in Section 2.1 we introduce the kinetic models under consideration and their computational difficulties. Then, for reproducibility concern, we describe the building blocks of the implementation of classical PIC method for solving numerically these models.
- In Section 2.2 we briefly describe the projects in which we performed our implementations and the motivation behind the search for performance.
- In Section 2.3 we analyze, compare, or propose new techniques and data structures and layouts for improving performance of PIC simulations in $2d \times 2v$ **over single core**.
- In Section 2.4 we go further with new optimization strategies, in terms of data structure and data layout, to gain more performance in $3d \times 3v$ **simulations over a multi-core shared memory** environment.

- In Section 2.5 we develop our choice of parallelism. We discuss the results of a **hybrid parallelism**: multi-threading (with OpenMP) and multi-processing (with MPI) by explaining the advantages and the limitations of our approach. The scalability of the parallelization is eventually assessed.
- In Section 2.6 we conclude the chapter and some perspectives are drawn.

2.1 The kinetic models and the PIC method

In this section, for the sake of completeness, I describe first, the different models that are implemented and the motivations behind their consideration. Then, in the second part, I introduce the numerical framework of a standard PIC method by giving details about the techniques and the numerical methods used within the implementation. This is an important fact for the reproducibility of the numerical and the performance results.
The history of our implementations starts with a `Fortran` code that I wrote in the library SeLaLib [Sel] for a $2d \times 2v$ PIC code. Then, during his PhD thesis [Bar18], Yann Barsamian ported and further developed this code in `C`, in a final $3d \times 3v$ version.

The kinetic models. We briefly present in this part the kinetic systems that we solve numerically in the following. They model classic test cases in plasma physics: Landau damping, drift-kinetic *vs.* guiding center simulations, and instability of a magnetized electron hole. The implemented numerical method will be described below, the difficulties being the multidimensional aspect of the problems and the high level of detail of their solutions. These numerical problems are quite expensive in terms of computer memory and execution time and consequently, the corresponding simulations need to be performed over a few multi-core processors.

• We start with the Vlasov–Poisson model for the Landau damping test case, a famous problem [MV11] which is discussed in any textbook of plasma physics. In $3d \times 3v$ the equations read

$$\begin{cases} \partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f - \mathbf{E}(\mathbf{x}, t) \cdot \nabla_{\mathbf{v}} f = 0, \\ \mathbf{E}\left(\mathbf{x}, t\right) = -\nabla_{\mathbf{x}} \phi(\mathbf{x}, t), \quad -\Delta_{\mathbf{x}} \phi(\mathbf{x}, t) = 1 - \displaystyle\int f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}, \\ f\left(\mathbf{x}, \mathbf{v}, t = 0\right) = f_0\left(\mathbf{x}, \mathbf{v}\right) \end{cases} \tag{2.1}$$

where f stands for the electron distribution function, the ions are supposed to be fixed, and $\mathbf{E}$ is the self-consistent electric field. One example of initial condition is given by

$$f_0\left(\mathbf{x}, \mathbf{v}\right) = \frac{1}{(2\pi v_{th}^2)^{3/2}} \left(1 + \alpha \cos(kx_1) \cos(kx_2) \cos(kx_3)\right) \exp\left(-\frac{|\mathbf{v}|^2}{2v_{th}^2}\right), \tag{2.2}$$

where the thermal velocity is set to $v_{th} = 1$. To introduce the first Fourier mode for the initial perturbation, we choose $\Omega_{\mathbf{x}} = [0; 2\pi/k]^3$ for the domain in the physical space. We use periodic boundary conditions in space. Several values for the amplitude α of the perturbation are considered. The conservation of the total energy of the Vlasov–Poisson model and the evolution of the electric energy obtained from the dispersion analysis were checked numerically for the verification of the code (see [5; 3; 4; 1] for several instances of the Landau damping concerning the dimension of the problem or the amplitude of the perturbation). This step is essential to make sure that a code correctly implements a given model [Son17]. Finally, we specify that the Landau damping simulations made up **the background to assess the performance** of our PIC codes.

• Next, we tackle the problem of modelling a strongly magnetized plasma constituted by electrons and a uniform and motionless background of ions. When tracking the electrons on a long enough time scale, a drift phenomenon occurs due to the self-consistent electric field in the plane orthogonal to the magnetic field. This is a multiscale behaviour since the electrons also execute a rapid circular motion around the magnetic field line. We assume in addition that the electric field created by the particles evolves only in the plane orthogonal to the magnetic field. The corresponding $2d \times 2v$ Vlasov–Poisson system is

$$\begin{cases} \partial_t f^\varepsilon + \dfrac{1}{\varepsilon}\left(\mathbf{v}\cdot\nabla_{\mathbf{x}} f^\varepsilon - \left(\mathbf{E}^\varepsilon(\mathbf{x},t) + \dfrac{1}{\varepsilon}\mathbf{v}^\perp\right)\cdot\nabla_{\mathbf{v}} f^\varepsilon\right) = 0, \\ \mathbf{E}^\varepsilon\left(\mathbf{x},t\right) = -\nabla_{\mathbf{x}}\phi^\varepsilon(\mathbf{x},t), \quad -\Delta_{\mathbf{x}}\phi^\varepsilon(\mathbf{x},t) = 1 - \displaystyle\int_{\mathbb{R}^2} f^\varepsilon(\mathbf{x},\mathbf{v},t)d\mathbf{v}, \\ f^\varepsilon\left(\mathbf{x},\mathbf{v},t=0\right) = f_0\left(\mathbf{x},\mathbf{v}\right), \end{cases} \tag{2.3}$$

where the vanishing parameter ε entails the strong magnetic field, supposed to be aligned with the x_3 direction: $\mathbf{B} = (0,0,1/\varepsilon)^T$. We denote by $\mathbf{x} = (x_1, x_2)$ the position variable, by $\mathbf{v} = (v_1, v_2)$ the velocity, and $\mathbf{v}^\perp = (v_2, -v_1)$. The term $(1/\varepsilon)\mathbf{v}^\perp$ corresponds to the projection of the magnetic force in the plane orthogonal to the magnetic field. The other singularity in the equation above is due to a time re-scaling that allows us to study the long-term behaviour of the electron distribution function and thus, to capture the drift phenomenon. The regime in which the Vlasov–Poisson system (2.3) is written corresponds to a time re-scaling of the classic drift-kinetic model. The scaling procedure leading to this regime is given in [FRS01] and the time re-scaling is carried out in [FS00] (see also Section 5.2.1 in [17]). Under appropriate assumptions on f_0, the limit model of (2.3) is derived in [FS00; Bos07] for example. Specifically, the authors established two types of convergence when ε goes to zero of the particle density towards the solution to the guiding center model

$$\begin{cases} \partial_t f_{GC} + \mathbf{E}^\perp\cdot\nabla_{\mathbf{x}} f_{GC} = 0, \\ \mathbf{E}\left(\mathbf{x},t\right) = -\nabla_{\mathbf{x}}\phi(\mathbf{x},t), \quad -\Delta_{\mathbf{x}}\phi(\mathbf{x},t) = 1 - f_{GC}, \\ f_{GC}\left(\mathbf{x},t=0\right) = \displaystyle\int_{\mathbb{R}^2} f_0\left(\mathbf{x},\mathbf{v}\right)d\mathbf{v}. \end{cases} \tag{2.4}$$

In this framework, in [5] we achieved simulations for a Kelvin–Helmholtz test case [Sho81], with an initial condition of type

$$f_0\left(\mathbf{x},\mathbf{v}\right) = \frac{1}{2\pi v_{th}^2}\left(1 + \sin(x_2) + \alpha\,\cos(kx_1)\right)\exp\left(-\frac{|\mathbf{v}|^2}{2v_{th}^2}\right), \tag{2.5}$$

defined in $\Omega_{\mathbf{x}} \times \mathbb{R}^2$, where $\Omega_{\mathbf{x}} = [0; 2\pi/k] \times [0; 2\pi]$, α is a small number as the amplitude of the perturbation, and the thermal velocity is set to $v_{th} = 1$. For the guiding center model, the initial condition consists in the $\mathbf{x}$ part only in (2.5), defined in $\Omega_{\mathbf{x}}$. Our aim was to illustrate numerically the convergence result alluded to above, by analyzing the relationship between the solutions of (2.3) and (2.4) (see Fig. 2.1 and [5] for more comparisons). **The computational effort** when solving equation (2.3) **stems from the stiff term** $1/\varepsilon^2$ which forces explicit numerical schemes to the use of very small time step. Thus, a rapid code is beneficial to achieve efficiently long time simulations.

• Finally, we present a test case of instability of a magnetized electron hole. As a nonlinear solution of a Vlasov–Poisson equation, an electron hole is a clump of electrons trapped in a phase space vortex. As above, ions are assumed to form a uniform and motionless background. The

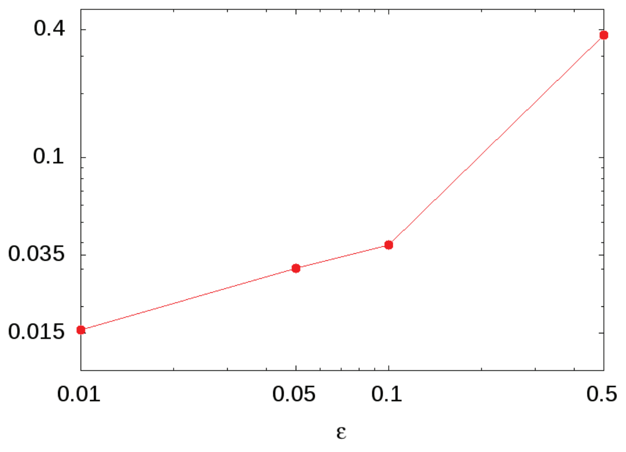


Figure 2.1: The global relative error at time $t = 5$ (in the L^2 norm) of the density of the Vlasov–Poisson solution of (2.3) with respect to the guiding center solution of (2.4), as function of ε.

literature in this field is abundant, see for instance [Hut17; Mus+00]. In a first step corresponding to a $1d \times 3v$ setting, we consider a starting potential of type

$$\phi(x) = \psi \ \exp\left(- 0.5(x - L)^2/\Delta_{\|}^2\right)$$

which generates the electron hole, where ψ, L, and $\Delta_{\|}$ are fixed parameters (see [1; Mus+00]). The initial distribution function is

$$f(x, \mathbf{v}) = F_1\big(v_1^2 - 2\phi(x)\big) \exp(-|\mathbf{v}|^2/2)$$

where $\mathbf{v} = (v_1, v_2, v_3)$, and

$$F_1(w) = \begin{cases} \frac{\sqrt{-w}}{\pi\Delta_{\|}^2}\Big(1 + 2\ln(\frac{\psi}{-2w})\Big) + \frac{6+(\sqrt{2}+\sqrt{-w})(1-w)\sqrt{-w}}{\pi(\sqrt{2}+\sqrt{-w})(4-2w+w^2)}, & \text{for } -2\psi \le w < 0, \\ \frac{6\sqrt{2}}{\pi(8+w^3)}, \text{ for } w > 0, & \end{cases}$$

contains the trapped particles for $w > 0$ and the passing particles for $w < 0$. In addition, we add an external magnetic field aligned with the x-direction. In this case, the evolution of the electron hole either rests intact, or its structure changes according to the gyro-to-bounce frequency ratio (see [Mus+00]). Thus, the trapped electrons both gyrate around the magnetic field line and bounce back and forth in the parallel direction.
In a second step, we performed simulations in a $2d\times3v$ setting, by perturbing the above equilibrium potential in the transverse direction, into

$$\phi(x_1, x_2) = \exp\left(- 0.5 \left((x_1 - L)/\Delta_{\|} - \alpha \cos(kx_2)\right)^2 \right),$$

where α is the measure of the perturbation and k its wave number. Our results are qualitatively similar to those from [Mus+00]. The challenge was **to capture the details in the small structures developed in the profiles of the density and the electric field in long time**, see Fig. 2.2. This asks for fine grids and therefore, fast code is needed in long runs.

The Particle-in-Cell method. The numerical treatment of the models above is done by means of the PIC method, which is a well-known approach to produce satisfactory results for Vlasov-like

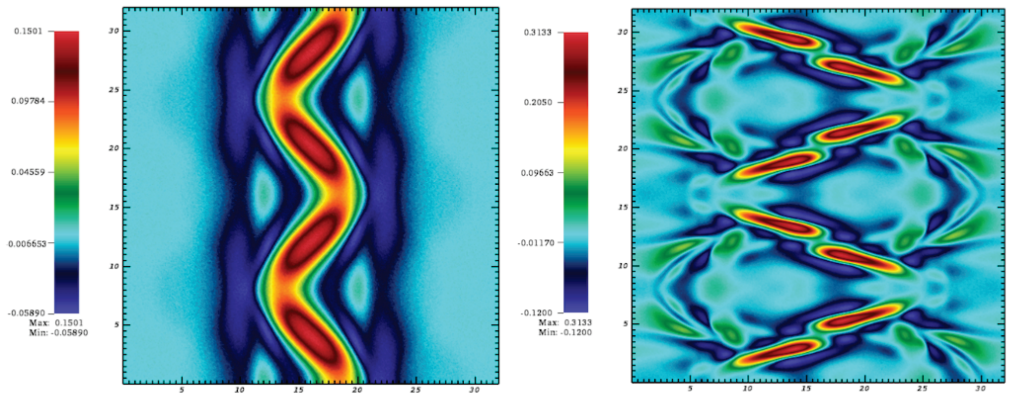


Figure 2.2: Electron hole simulation: profile of the density $\rho(x_1, x_2)$ at $t = 20$ (left) and $t = 40$ (right).

equations; see [BL85; HE88; CK00] for advanced discussions and for applications in plasma physics and in fluid dynamics.

Already in use as early as 1955 for solving "hydrodynamic problems" [Har55], the PIC method is very popular in the plasma physics community. The main advantages are its simplicity, its capability to be simply extended to higher dimension, and its ability to cope with the real particles dynamics at a relatively low cost. However, the particle method has an important drawback: it suffers from noise, namely strong spurious oscillations in the numerical results if the number of particles is not sufficiently high. It is known that the noise decreases like $1/\sqrt{N_p}$ with N_p the number of particles. The precise meaning of this statement [BS15, Th. 3.2.2] relies on a statistical framework in which the PIC method can fit. This is explained in the review [BS15], together with efficient noise-reduction strategies from the Monte Carlo literature which help improving the accuracy of PIC simulations. Moreover, we emphasize that there is an important literature devoted to deterministic error analysis of particle methods (see *e.g.* [BM82; Rav85; CK00]), but detailing this subject is beyond the scope of the present work. We mention the recent work [CP17] for new techniques to reconstruct accurate approximations of the particle density and their comparison to existing methods which reduce the oscillations in a particle method. We conclude this part by underlining that in order to guarantee the point-wise convergence of the numerical (particle) density towards the solution of the Vlasov equation[1], the number of particles per cell should increase significantly faster than the number of cells (see [Bar05, Section 5.3]; see also [CP17] and the references therein).

In the context of a Vlasov–Poisson model of the form (2.1), the PIC method consists in approximating the distribution function f by a finite number of numerical particles. Then, we compute the trajectories of these particles by solving the characteristic curves of the Vlasov equation, while the self-consistent field $\mathbf{E}$ is computed on a grid in the physical space. More precisely, we discretize the distribution function at every time step $t_n = n\Delta t$ by a collection of N_p particles with coordinates $((\mathbf{x}_k^n, \mathbf{v}_k^n))_{k\in\{1,\dots,N_p\}}$ in the phase space and then, we regularize it with a convolution kernel S

$$f_S(\mathbf{x}, \mathbf{v}, t_n) = \sum_{k=1}^{N_p} \omega_k S(\mathbf{x} - \mathbf{x}_k^n) S(\mathbf{v} - \mathbf{v}_k^n). \tag{2.6}$$

Typically, all the weights ω_k are of order $1/N_p$. As usual on Cartesian grids, we use B-splines as convolution kernel [Son17]. Although numerical tests were carried out with B-splines of degree 1

[1] or in other words, to reduce the noise of a PIC simulation.

and 3, we recall here only the form of a degree 1 B-spline, in a one dimensional physical space[2]

$$S(x) = \begin{cases} \dfrac{1}{\Delta x}\left(1 - \dfrac{|x|}{\Delta x}\right) & \text{if} \quad |x| < \Delta x \\ 0 & \qquad \text{otherwise,} \end{cases}$$

where Δx is the step of a fixed one dimensional grid. Every particle k moves along a characteristic curve of the Vlasov equation in (2.1)

$$\begin{cases} \dfrac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}(t), & \mathbf{x}(0) = \mathbf{x}_0, \\ \dfrac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\mathbf{E}\big(\mathbf{x}(t), t\big), & \mathbf{v}(0) = \mathbf{v}_0. \end{cases} \tag{2.7}$$

Therefore, the problem consists in finding the coordinates $(\mathbf{x}_k^{n+1}, \mathbf{v}_k^{n+1})$ at time t_{n+1} from their values at time t_n by using an ODE solver for (2.7) with initial condition $(\mathbf{x}_k^n, \mathbf{v}_k^n)$. The electric field term in (2.7) is computed in a particle position at every time step t_n as follows:

1. Compute in every grid point the density

$$\rho_S\left(\mathbf{x}, t_n\right) = \sum_{k=1}^{N_p} \omega_k S\left(\mathbf{x} - \mathbf{x}_k^n\right). \tag{2.8}$$

2. Solve the Poisson equation on the grid and deduce $\left(\mathbf{E}_j^n\right)_j$, the values of the grid electric field.

3. Interpolate the grid electric field with the same shape factor S yielding the density ρ_S[3], which gives the electric field at the particle position:

$$\mathbf{E}(\mathbf{x}, t_n) = \Delta\mathbf{x} \sum_j \mathbf{E}_j^n S(\mathbf{x} - \mathbf{y}_j), \tag{2.9}$$

   where $(\mathbf{y}_j)_j$ denote the grid nodes.

At this stage, three additional points should be clarified in order to achieve the description of the numerical approach and its implementation in our codes: how initialization is done and what are the ODE solver (or the particle pusher) and the field solver, respectively.

Concerning the first issue, we considered the Monte Carlo approach. Thus, a PIC simulation starts with the generation of a random sequence — actually pseudo-random sequence, since one cannot generate a truly random one on a computer — or quasi-random sequence following some given probability law, associated with the initial condition f_0. Such a sequence can be obtained from a uniform random sequence in $[0, 1]$ ([BFS87], see also [Son17, Section 8]). Indeed, starting from a uniform random sequence, there are different ways to draw values for any other given probability density function; for example, the Box–Muller algorithm for generating Gaussian densities or the rejection algorithm for generating densities like those in (2.2) or (2.5) (see [BFS87]). As for obtaining a uniform random sequence in $[0, 1]$ we generally used standard libraries included with the compilers or available software to generate pseudo-random sequences.
In our simulations, tests were performed also when initializing particles following sequences that

[2] a multidimensional B-spline is simply obtained by a tensor product of one dimensional B-splines.
[3] to get conservation of the total momentum in the periodic boundaries setting.

are not random, but quasi-random (like Hammersley or van der Corput sequences [DT97]). Such sequences are designed to fill the space as uniformly as possible, leading to a reduction of noise when compared to random sequences. This is the so-called quiet-start (see the references in [Bar05]) which leads to smaller errors in density and current at the beginning of the simulation, but not in long runs [Bar05, Section 5.3].

Second, we implemented several classic time stepping schemes with the aim of solving **explicitly** ODEs of the type given in (2.7). The leap-frog scheme (or equivalently the Störmer–Verlet scheme [HLW06]) was used when the characteristics have the form of the pendulum equation, as is the case of the model (2.1). This is a second order method and in addition, it is time-reversible and provides the energy conservation of the dynamical system. An explicit Runge–Kutta scheme was preferred to implementation when the Hamiltonian is not separable, like the guiding center equation in (2.4), since the Störmer–Verlet scheme is implicit in such a case. Finally, we used also the Boris method when the magnetic field came into play, namely for the third model. This scheme enjoys excellent long term accuracy, due to the property of the phase space volume conservation [BL85; Qin+13].

Third, we made use of the Fast Fourier Transform for solving the Poisson equation in a periodic box, since the problems we tackled involve periodic boundary conditions in a box-like physical domain. Thus, the time of calculation of the electric field is small in our simulations ($\sim 1\%$), being negligible when compared to the other steps of the PIC algorithm.

Conclusion. The multiscale behaviour in the models exposed in this section, requires, when solved with the numerical schemes above, **large numbers of particles** and **small discretization steps** in order to reduce the noise and to resolve the fine structures in the phase space. These features lead to **expensive computations** and **costly memory accesses**. Therefore, in order **to make simulations run in reasonable times**, code optimization and parallelism are needed. The rest of this chapter is devoted to this subject.

2.2 Framework and motivation of software contributions

Framework. I began working on code performance issues in the framework of the software library SeLaLib (the Semi-Lagrangian Library) [Sel]. SeLaLib is a modular library for kinetic and gyrokinetic simulations of plasma by, at the very beginning, semi-Lagrangian methods. However, I started to implement PIC methods in the library, with the help of Edwin Chacon-Golcher, one of its software engineer at that time.
As an application programming interface, SeLaLib provides the building blocks for parallel simulations of Vlasov-like and gyrokinetic equations. The library contains low-level utilities, input-output modules, and parallelization utilities. Developed by several researchers, it contains also a collection of simulations with various discretization methods for test-cases which are useful to tokamak plasma problems. My contribution was to write the modules for the first building blocks to be used for achieving performance in simulations with a standard PIC method in $2d \times 2v$. The theoretical tools behind these building blocks are developed in Section 2.1 and the optimization techniques for their efficient implementation are exposed in the following sections.

Then, I continued working on this subject in the frame of the PhD in computer science of Yann Barsamian. In a first time, he ported the blocks of the $2d \times 2v$ PIC code from SeLaLib to `C` and he further optimized it with the aim of reducing the number of memory transfers. In this setting I proposed the study of space-filling curves (see Section 2.3 and Section 2.4) as different data layout of the electric field and of the density, in order to use appropriate memory layouts for optimizing

the cache performance.
In a second time, he wrote the code called Pic-Vert [Bar18] in the $3d \times 3v$ setting. This was the ground for new findings in the direction of searching for new algorithms and data layouts, with the aim of improving the use of computer resources. The ideas of this second part are mainly due to Yann.

We will see in the following that our implementations and their performances depend on the architecture, meaning that some parameters are to be tuned manually with respect to the memory hierarchy. Thus, we need to specify the machines on which our simulations have run. We used computing nodes with 2 sockets each. Examples of sockets are:

(i) Intel Xeon E5-2670 @ 2.6 GHz (Sandy Bridge) with 32 GB of RAM, a maximum memory bandwidth of 51.2 GB/s, 4 memory channels and 8 cores, available on the computing center of the University of Strasbourg (*Mésocentre*);

(ii) Intel Xeon E5-2650v3 @ 2.3 GHz (Haswell) with 32 GB of RAM, a maximum memory bandwidth of 34 GB/s, 2 memory channels and 10 cores, on a local machine at the computer science department, University of Strasbourg;

(iii) Intel Xeon Platinum 8160 @ 2.1 GHz (Skylake) with 96 GB of RAM, a maximum memory bandwidth of 128 GB/s, 6 memory channels and 24 cores, available on the supercomputer Marconi at Cineca, a supercomputing centre for scientific research in Italy.

Motivation. Implementing a PIC method is an outstanding step in the field of computational science, when we aim at reducing the noise of the method by the use of very large number of particles. Numerous research efforts are recently devoted to more efficient PIC implementations with the aim of an optimal use of modern supercomputing resources (see for example [DS14; Ger+16; Vin+16; Joc+16]).

In the rest of the chapter, we describe **data structures and techniques** that we incorporated in our codes for **achieving performance**. Even if some of these procedures are known in theory, their implementation is subject to adaptation and adjustment to the specific problems to be solved, and further analysis and comparisons were carried out in order to obtain the best performance results. We develop also new optimization strategies to reinforce performance.

We start by briefly recalling the four steps followed at each time iteration of a PIC simulation: accumulate on the spatial grid the particle charge, solve the Poisson equation to obtain the grid electric field, interpolate this field to the particles, and finally push in time with this field the particle positions and velocities (see Fig. 2.3 for an instance of a PIC algorithm in $3d$). Therefore, two types of data are present in a PIC code: particles and grid quantities (the electric field and the charge density) that interact one with another. In addition, when increasing the size or the level of details of the problem under study, a simulation may involve billions of particles and hundreds of grid points per dimension and may need the execution of a large number of time steps. Thus, PIC simulations are quite expensive in terms of computer memory and execution time requirements, demanding large and powerful supercomputers. More precisely, in a PIC simulation "*moving data between and even within modern microprocessors is more time consuming than performing computations.*" [Bow+08] Consequently, a **minimization of the number of costly memory accesses results in large performance gains** in multi-core processing. These principles are accomplished through several key points, that we expose below. We conclude by specifying that all the performance results that follow are developed within Landau damping simulations in $2d \times 2v$ or $3d \times 3v$ with different numerical parameters.

Parameters

N: number of particles.

$ncx \times ncy \times ncz$: number of grid cells.

Δt: time step.

f_0: initial distribution function.

q and m: particle charge and mass.

Variables

$particles[N]$: set of particles, with position $\mathbf{x_p}$ and velocity $\mathbf{v_p}$.

$\rho[ncx][ncy][ncz]$: charge density.

$\mathbf{E}[ncx][ncy][ncz]$: electric field.

Algorithm

```
1  Randomly initialize $N$ particles following $f_0$
2  Compute initial $\rho$ and $\mathbf{E}$
3  For each time step                                  Leap-frog
4      If ($condition$), then
5          Sort the particles
6      Set all cells of $\rho$ to 0
7      For each particle
8          Interpolate $\mathbf{E}$ to $\mathbf{x_p}$          Stored in $\mathbf{E_p}$
9          Update $\mathbf{v_p}$                      $\mathbf{v_p}$ += $\frac{q}{m}\Delta t\, \mathbf{E_p}$
10         Update $\mathbf{x_p}$                      $\mathbf{x_p}$ += $\Delta t\, \mathbf{v_p}$
11         Accumulate charge from $\mathbf{x_p}$ on $\rho$
12     Compute $\mathbf{E}$ from $\rho$                 Poisson solver
```

Figure 2.3: Pseudo-code of a Particle-in-Cell (PIC) method.

2.3 Single core optimizations in $2d \times 2v$

In this section, we describe in detail the data structures for particles and for fields on a grid in the design of a PIC implementation in $2d \times 2v$, together with the optimizations and techniques we performed for reducing execution time. In a first step we develop the work we have done in SeLaLib [5], whose features are

- "cell index plus offset" for the particle representation,
- Array of Structures (AoS) for the particle data structure,
- redundant cell-based arrays for $\mathbf{E}$ and ρ,
- periodic particle sorting.

1. Particle representation. In a standard implementation, a particle is given by its position in the physical domain and its velocity ranging in some given bounds. While we keep the classic representation by two real numbers for the velocity, the position of particle is identified with a cell index i_{cell} and two normalized offsets within this cell. The advantages of this representation are well-known, see [Bow+08, Section III-E] for example. The main reasons are the reduced memory size for a particle and the reduced amount of arithmetical operations in the interpolation and the charge accumulation steps.

For a better understanding of this important point, let us enter into some details. Let $[x_{min}; x_{max}) \times [y_{min}; y_{max})$ denote the physical space and $\Delta x = (x_{max} - x_{min})/ncx$ and $\Delta y = (y_{max} - y_{min})/ncy$ the grid spacing, where ncx and ncy are the number of grid cells in the two directions. Thus, a particle positioned at $(x_{\text{physical}}, y_{\text{physical}})$ is mapped at the position $(x, y) \in [0; ncx) \times [0; ncy)$, where

$$x = \frac{x_{\text{physical}} - x_{min}}{\Delta x} \quad \text{and} \quad y = \frac{y_{\text{physical}} - y_{min}}{\Delta y}.$$

Then, we consider the integers[4]

$$i_x = \lfloor x \rfloor \quad \text{and} \quad i_y = \lfloor y \rfloor, \tag{2.10}$$

[4]For any real number x, we denote by $\lfloor x \rfloor$ the greatest integer less than or equal to x.

```
Update positions
1   Compute $(i_x, i_y)$ from $i_{\text{cell}}$.
2   Update $(x, y)$ using formula (2.11) and line 10 in Fig. 2.3.
3   Compute the new values of $(i_x, dx, i_y, dy)$ using formulas (2.10) and (2.11).
4   Compute $i_{\text{cell}}$ from $(i_x, i_y)$.
```

Figure 2.4: The update positions step, in the "cell index plus offset" particle representation.

and the normalized offsets (which are real numbers in $[0; 1)$)

$$dx = x - i_x \quad \text{and} \quad dy = y - i_y. \tag{2.11}$$

The cell index i_{cell} is defined to be a number in $\{0, 1, \ldots, ncx \cdot ncy - 1\}$, taken as the image of some one-to-one mapping depending on (i_x, i_y). For example, the commonly used row-major mapping is

$$\begin{aligned} (i_x, i_y) &\mapsto i_{\text{cell}} = i_x \cdot ncy + i_y \\ i_{\text{cell}} &\mapsto \begin{cases} i_x = \left\lfloor \frac{i_{\text{cell}}}{ncy} \right\rfloor \\ i_y = \text{mod}(i_{\text{cell}}, ncy). \end{cases} \end{aligned} \tag{2.12}$$

To conclude, a particle is stored in memory with one integer (i_{cell}), two numbers in single precision (dx and dy), and two numbers in double precision (vx and vy). It is important to stress that in this representation, the update particle position step (line 10 in Fig. 2.3) is to be accomplished now in four sub-steps, detailed in Fig. 2.4. From Fig. 2.4 we deduce also why we need the mapping and its inverse in (2.12).

2. Particle data structure. A second important point is the organization of the **particle list**, as

- a Structure of Arrays (SoA), where we store all the particle positions and velocities in the arrays $x(i), y(i), vx(i), vy(i)$ for $i \in \{1, 2, \ldots, N_p\}$, or
- an Array of Structures (AoS), where the particles are stored in a single array $p(i)$ for $i \in \{1, 2, \ldots, N_p\}$, where $p(i)$ contains the position and velocity of particle i.

This is a standard issue [Dec+96; Ver05; Bow+08]. In our first work [5] we implemented the AoS for listing the particles. The advantage of this listing is that all the particle information can be accessed in a single memory stream and thus no cache miss occurs when updating one particle's position and velocity. In addition, a single pass over the particle list should be performed for an efficient updating. However, we see below that this structure does not allow to vectorize the update position step.

3. Grid quantities data structure. The standard $2d$ representation of the electric field $\mathbf{E}$ and of the charge density ρ stores their values at the grid points: *e.g.* the grid values of $\mathbf{E}$ are stored in a 2d array $\mathbf{E}(i_x, i_y)$ where $i_x \in \{0, 1, \ldots, ncx - 1\}$ and $i_y \in \{0, 1, \ldots, ncy - 1\}$. In this case, the interpolation of the two dimensional arrays E_x and E_y requires access to memory locations that are not contiguous. A partial solution to avoid this issue was proposed in [Dec+96, Section IV]. However, the problem is solved by using a **redundant cell-based** one dimensional array to store field and charge density values [Bow03]. Precisely, all the field values at the corners of a cell are stored in a single data structure which is then laid out as a single one dimensional array: we store in $\widetilde{\mathbf{E}}(i_{\text{cell}})$ for $i_{\text{cell}} \in \{0, 1, \ldots, ncx \cdot ncy - 1\}$ the four values of $\mathbf{E}$ at the corners of cell i_{cell}. What makes these data structures redundant is the fact that the field values for a single grid point must be stored in multiple cell-based structures. Also, two additional necessary steps are to be considered: the conversion of the charge density $\widetilde{\rho}(i_{\text{cell}})$ to $\rho(i_x, i_y)$ which is needed for solving Poisson equation on the grid, and then, the conversion of $\mathbf{E}(i_x, i_y)$ issued from solving Poisson equation to $\widetilde{\mathbf{E}}(i_{\text{cell}})$. Even if such a data structure requires more memory and supplementary computations, we show below that this leads to an overall gain in performance

- through **cache hit improvements** (both for $\mathbf{E}$ and ρ) using space-filling curves, and
- through **efficient vectorization** of the accumulation step.

We will see that the redundant data structure gives much better execution times than the standard representation, especially in high dimensions (see $3d$ standard *vs.* Row-major in Table 2.4) and additionally, when the number of particles per cell is more than a hundred [4, Section 3.4].

4. Particle sorting. This technique consists in organizing in memory the particle list by cell index. Together with the particle and $\mathbf{E}/\rho$ data structures, the particle sorting allows to increase the number of the cache hits. More precisely, when two particles contiguous in memory are also in the same grid cell, they access the same $\mathbf{E}/\rho$ cell values in the interpolation/accumulation step. As the simulation evolves, the particles get unsorted, so this algorithm needs to be applied periodically. It was thus shown [Bow01; Ver05] that sorting periodically the particles results in significant performance improvements by reusing field and charge density data for many consecutive particles within a cell. We verified and illustrated this principle by plotting the time evolution of the memory bandwidth (MB) of the code, which is roughly the amount of data that is moved per unit time (high MB implies high performance). We thus obtained the **highest values of MB** at the time **when sorting was applied**: see Fig. 2.5 and also [5, Fig. 5] for observing the same behaviour when multithreading is in place.

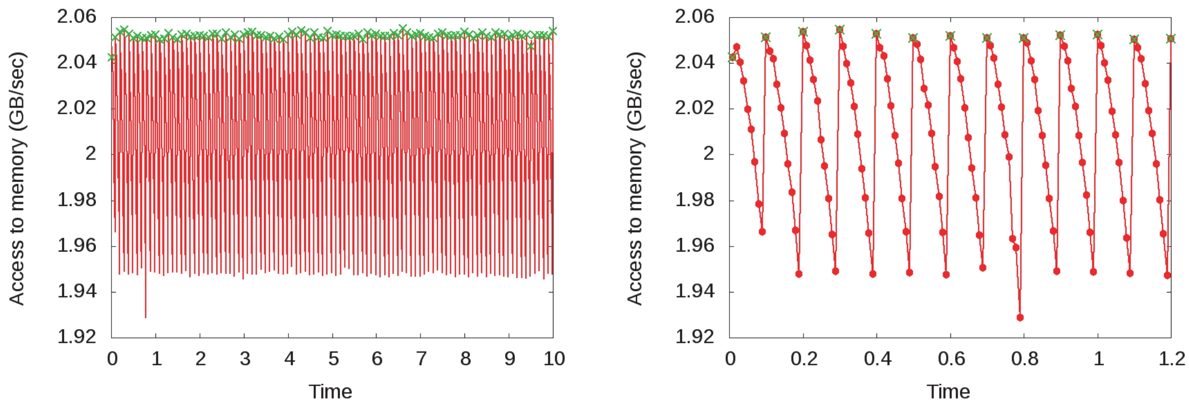


Figure 2.5: History of the memory bandwidth in GB per second. A run with 2 million particles on 1 core. At right a zoom at the beginning of the simulation. In green the steps where sorting is done. Architecture: Sandy Bridge.

We detail now our achievements implemented in the code Pic-Vert, in `C`, corresponding to reference [3]. At the end of this section, in Table 2.1, we report the accumulated gains with respect to a baseline code, of all the optimizations developed in the following.

5. Loop hoisting and loop fission. The loop hoisting consists in removing as much computation as possible from the particle loops which results in a significant improvement of the runtime performance. We implemented this for the velocities and the positions updates, by performing outside the particle loops multiplications by constants like $q\Delta t/m$ and $\Delta t/\Delta x$ respectively (see lines 9 and 10 in Fig. 2.3). The loop fission (see [Wol95]) means that the loop "**For each** particle" in Fig. 2.3 is broken into three parts: one loop to update-velocities, one to update-positions, and one loop to accumulate the charge. There are two main reasons to use three loops instead of one: (a) we can efficiently vectorize the update-positions as a stand-alone loop and (b) a separate processing of the arrays $\mathbf{E}$ and ρ in different loops leads to a better overall memory management.

6. Space-filling curves are different organizations in memory of the $\mathbf{E}$ and ρ cell values. We showed that using appropriate memory layouts for the redundant data structure optimizes further the cache performance.
The interest of using the space-filling curves can be explained as follows. In PIC codes memory accesses are a well-known major bottleneck. Each time that the code accesses a cell of $\mathbf{E}$ or ρ, a contiguous portion of the array is loaded into the cache: ideally, all the computations using these data cells should be performed while the information is still there, avoiding to reload them later from the main memory. We already saw that a periodic sorting of the particles needs to be performed to improve data locality. Nevertheless, sorting at every iteration would be computationally expensive and therefore **we have to find a memory layout of the cells such that the cache benefits from the sorting last as long as possible**. More precisely, we aim at constructing a mapping $(i_x, i_y) \mapsto i_{\text{cell}}$ such that, when a particle moves from a cell to another, the probability that its new cell index i_{cell} is close to the old one should be high.

The mapping of the row-major ordering in (2.12) was considered in the work in SeLaLib, within the point 3 above on the grid quantities data structure. We remark that this ordering has advantageous data locality when a particle moves along the y-axis (see equation (2.12) and Fig. 2.6): if i_y increases by one, the new cell index also increases by one (except for particles on the right edge of the grid), becoming exactly the index accessed by the following particles in the particle array. However, when a particle moves along the x-axis, this favorable behaviour is lost: if i_x increases by one, the cell-index changes by ncy which implies cache misses for the values of $\mathbf{E}$ and ρ.

We have thus implemented in [3] the L4D space-filling curve introduced in [Cha+99], but we proposed the following algorithm for its computation

$$\begin{aligned} &(i_x; i_y) \mapsto i_{\text{cell}} = \text{SIZE} \cdot i_x + \text{mod}(i_y, \text{SIZE}) + ncx \cdot \text{SIZE} \cdot \lfloor i_y/\text{SIZE} \rfloor \\ &i_{\text{cell}} \mapsto \begin{cases} i_x = \lfloor \text{mod}\big(i_{\text{cell}}, ncx \cdot \text{SIZE}\big)/\text{SIZE} \rfloor \\ i_y = \text{mod}(i_{\text{cell}}, \text{SIZE}) + \text{SIZE} \cdot \lfloor i_{\text{cell}}/\big(ncx \cdot \text{SIZE}\big) \rfloor , \end{cases} \end{aligned} \tag{2.13}$$

where SIZE is a parameter to be chosen carefully depending of the cache sizes. We recall that the mapping and its inverse in (2.13) are needed for two operations: first, to carry out the lines 1 and 4 in Fig. 2.4 in the updating the particle position and second, for the conversion of $\mathbf{E}$ and ρ values between the redundant structure and the grid structure.

In all, four different strategies for ordering the cells have been tested. We list them here from the least to the most computational-intensive, in terms of the computation of the mapping $(i_x, i_y) \mapsto i_{\text{cell}}$: the row-major (in Fig. 2.6), the L4D-order (in Fig. 2.11), the Morton-order (in Fig. 2.7), and the Hilbert-order. We refer to [3, Section IV.B] and the references therein for a detailed discussion. In conclusion, we compared these orderings and obtained **the best cache misses improvement with the L4D curve**, for the L2 and L3 cache levels (see Figs. 2.8 and 2.9). We reported also overall numbers on the L1, L2, and L3 cache levels in [3, Table II] in order to observe that the good

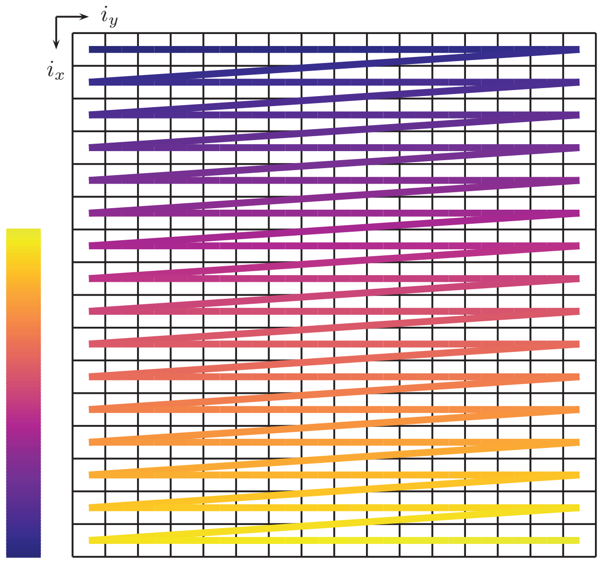


Figure 2.6: Row-major layout of a 16 x 16 matrix

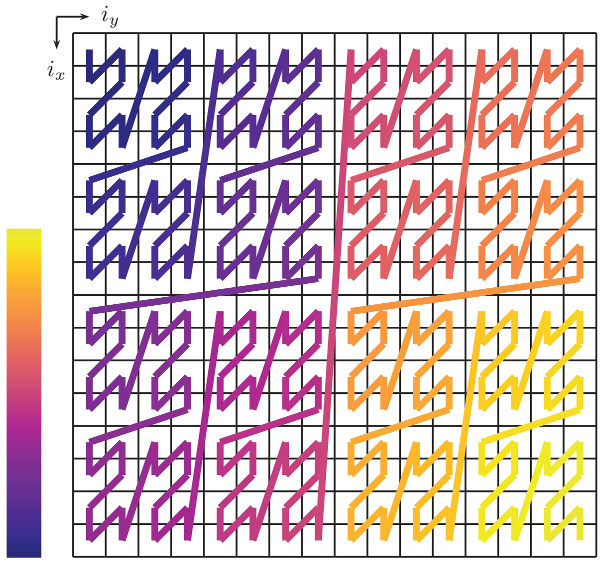


Figure 2.7: Morton layout of a 16 x 16 matrix.

data locality due to the sorting keeps longer in time for the three non-canonical curves than for the row-major.

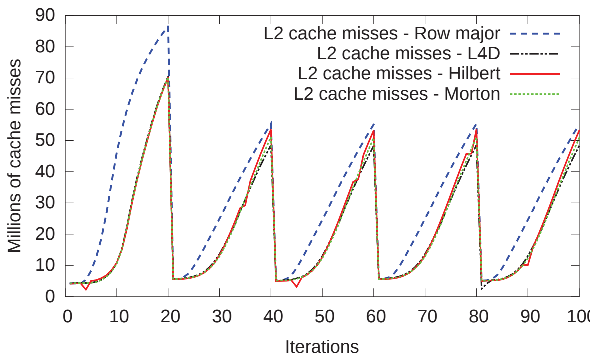


Figure 2.8: Millions of cache misses per iteration for the cache level 2 during the update-velocities and accumulate loops. Architecture: Haswell

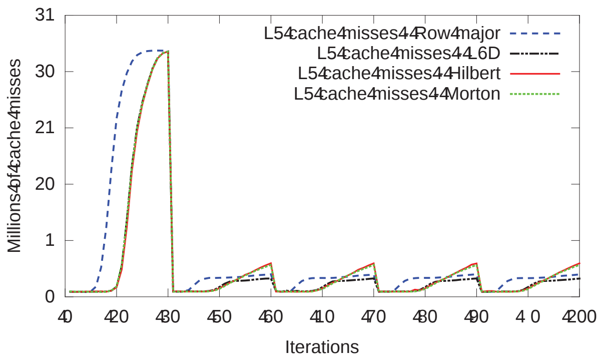


Figure 2.9: Millions of cache misses per iteration for the cache level 3 during the update-velocities and accumulate loops. Architecture: Haswell

7. Vectorization. This is a special instance of automatic parallelization. Modern[5] architectures can handle several operations at once, computing on vectors rather than on scalars. Precisely, the same operation (as that in line 10 in Fig. 2.3) may be computed in parallel for 4 particles, for example. Using a compilation flag is one possibility to automatically vectorize the code. However, to enable real vector performances, rewriting the code in addition to the use of an appropriate data structure is necessary. Indeed, to achieve the full power of vectorization requires that the Single Instruction operates on Multiple Data that are **contiguous** in memory. We used this technique in two ways:

- Vectorization of the update particle positions step. This was performed efficiently with the

[5]Single Instruction Multiple Data (SIMD)

SoA for the particle list. Indeed, using AoS led to unsatisfactory timings, since two data (*e.g.* positions) to be vectorized are not contiguous.

- The redundant structure described above for the charge density opened the possibility to vectorize the accumulation step like in [Vin+16, Section 4.1.2]. More precisely, the redundant writing of the charge density allows to put the accumulation operation in all the grid points of a cell in a single instruction format and thus to vectorize it, following the grid points: see Fig. 2.10 for a standard not-vectorized accumulation *vs.* a redundant vectorized one.

```
double rho[ncx][ncy]; // Standard 2d.
rho[i_x  ][i_y  ] += w * (1-dx[i]) * (1-dy[i]);
rho[i_x  ][i_y+1] += w * (1-dx[i]) * (  dy[i]);
rho[i_x+1][i_y  ] += w * (  dx[i]) * (1-dy[i]);
rho[i_x+1][i_y+1] += w * (  dx[i]) * (  dy[i]);

double rho_1d[ncx*ncy][4]; // Redundant.
float cx[4] = {  1.,  1.,  0.,  0.};
float sx[4] = { -1., -1.,  1.,  1.};
float cy[4] = {  1.,  0.,  1.,  0.};
float sy[4] = { -1.,  1., -1.,  1.};
for (corner = 0; corner < 4; corner++)
  rho_1d[i_cell[i]][corner] += w * (cx[corner] + sx[corner] * dx[i])
                               * (cy[corner] + sy[corner] * dy[i]);
```

Figure 2.10: The accumulate step: Standard *vs.* Redundant structures.

8. Optimized update-positions step. This technique is relevant when we have periodic boundary conditions in space. In such a setting, two operations are done when updating the particle positions: first, an `if`-test for verifying that the new position is inside the domain and second, function calls (like `modulo` or `floor`) for computing the i_{cell} and the offsets of a particle (see line 3 in Fig. 2.4). Using these operations as they are, either gives unsatisfactory results when using vectorization, either they are not vectorized at all (when using for example a GNU compiler). In this context, the proposed optimizations consist in rewriting the code conveniently, which is specific to `C`, such that the automatic vectorization be enabled for this step (see [3, Section IV.C]).

Conclusion. Overall gains. Our single-core optimizations are summarized in Table 2.1, where the baseline is a version of the code with the standard $2d$ data structure for $\mathbf{E}$ and ρ and the Array of Structures for the particles. We thus conclude that the **overall gain** in the execution time is of **42%** with respect to a standard code. The performance obtained in this way is compared in Table 2.2 to the $2d$ Vlasov-Poisson code presented in [DS14]. These results show the **importance of the architecture** in simulations. For example, when sorting, several runs showed that the optimal number of iterations between two sorting steps is 50 on Sandy Bridge architecture and 20 on Haswell architecture (the results in Table 2.2 are obtained for these sorting frequencies). The better results on the Haswell architecture are due to its bigger theoretical memory bandwidth (recall the comments in Section 2.2).

The strength of these data and procedures is reinforced when adding the strategies we propose in the following section, in a $3d \times 3v$ setting.

Table 2.1: Total execution time, gains and accumulated gains.

	Time(s)	Gains(%)	Acc. gains(%)
Baseline	120.4	0.0	0.0
+ Loop hoisting	113.4	5.8	5.8
+ Loop fission	97.9	13.7	18.7
+ Redundant arrays **E** and ρ (vectorized accumulation loop)	94.0	4.0	21.9
+ SoA for *particles* (vectorized update-positions loop)	76.0	19.1	36.9
+ The L4D space-filling curve for **E** and ρ	72.6	4.5	39.7
+ Optimized update-positions loop	68.8	5.2	42.8

Table 2.2: Time spent per particle per iteration, in nanoseconds.

	Decyk & Singh work [DS14] (on Nehalem)	Present work (on Sandy Bridge)	Present work (on Haswell)
Push	19.9	15.6	9.1
Accumulate	9.0	4.3	2.6
Reorder	0.3	–	–
Sorting	–	1.9	2.0
Total	29.2	21.8	13.7

2.4 Optimizations in $3d \times 3v$

In this section, we present several strategies that are proposed in references [4] and [1]. In a first step, in [4], we detail the extensions to $3d \times 3v$ of our preliminary results in $2d \times 2v$ in [3], by highlighting the difficulties and the solutions for these extensions. Specifically, our main contributions consist in proposing **a new space-filling curve in** $3d \times 3v$ (called L6D) to improve the cache reuse and **an adapted loop transformation (strip-mining)** to achieve efficient vectorization and to optimize memory transfers. Unlike the previous section where the analysis of the optimization strategies was performed on single core, in this section we work on a 24-core socket, with a shared-memory parallelism. The motivation of this point is our aim to port these strategies to simulations running on many multi-core processors. In addition, it is not clear if it is possible to extrapolate the single-core performance results to a multi-core socket. This is due to the different ratios between computational performance and memory bandwidth of the two configurations: a single core is assigned to a memory channel, while *e.g.* the 24 cores of a Skylake processor share 6 memory channels, leading to different memory accesses of the CPUs.

New space-filling curve. The framework detailed in $2d \times 2v$ in Section 2.3 at item 1. about the particle representation can be easily extended to the $3d \times 3v$ setting. Thus, a particle is stored in memory with one integer (i_{cell}), three numbers in single precision (dx, dy, and dz) and three numbers in double precision (vx, vy, and vz). Now, the cell index i_{cell} is a number in $\{0, 1, \ldots, ncx \cdot ncy \cdot ncz - 1\}$, taken as the image of some one-to-one mapping depending on (i_x, i_y, i_z). The commonly used row-major mapping becomes

$$
\begin{aligned}
&(i_x, i_y, i_z) \mapsto i_{\text{cell}} = (i_x \cdot ncy + i_y) \cdot ncz + i_z \\
&i_{\text{cell}} \mapsto \begin{cases} i_x = \left\lfloor \frac{i_{\text{cell}}}{ncz \cdot ncy} \right\rfloor \\ i_y = \operatorname{mod}\left(\left\lfloor \frac{i_{\text{cell}}}{ncz} \right\rfloor, ncy \right) \\ i_z = \operatorname{mod}(i_{\text{cell}}, ncz). \end{cases}
\end{aligned}
\tag{2.14}
$$

Such a mapping is used in the update positions step which has a similar form in $3d$ to that in Fig. 2.4. We have demonstrated in Section 2.3 at item 6. that different space-filling curves improve the cache performance with respect to the standard row-major curve. Similar comments about the data locality of the row-major ordering in $3d$ may be assessed: the mapping of the row-major ordering in (2.14) has advantageous data locality when a particle moves along the z-axis; if i_z increases by one, the new cell index also increases by one (except for particles on the top face of the grid), becoming exactly the index accessed by the following particles in the particle array. However, when a particle moves along the other axes, this favorable behaviour is lost: if i_y (resp. i_x) increases by one, the cell index changes by ncz (resp. $ncy \cdot ncz$) which implies cache misses for the values of $\mathbf{E}$ and ρ.

Therefore, going further in this direction, our contribution in [4] is to design the new L6D space-filling curve (see Fig. 2.12). This is a generalization to the $3d$ setting of the L4D curve in Fig. 2.11. We define the L6D bijection mapping by[6]

$$
\begin{aligned}
(i_x; i_y; i_z) \mapsto i_{\text{cell}} = {}& ncz \cdot \text{SIZE}^2 \cdot \big(\lfloor i_x/\text{SIZE} \rfloor + \lfloor i_y/\text{SIZE} \rfloor \cdot \lceil ncx/\text{SIZE} \rceil \big) \\
&+ i_z \cdot \text{SIZE}^2 + \operatorname{mod}(i_y, \text{SIZE}) \cdot \text{SIZE} + \operatorname{mod}(i_x, \text{SIZE})
\end{aligned}
$$

$$
i_{\text{cell}} \mapsto \begin{cases} i_x = \operatorname{mod}(i_{\text{cell}}, \text{SIZE}) + \text{SIZE} \cdot \left\lfloor \operatorname{mod}(i_{\text{cell}}, ncz \cdot \text{SIZE}^2 \cdot \lceil ncx/\text{SIZE} \rceil)/(ncz \cdot \text{SIZE}^2) \right\rfloor \\ i_y = \text{SIZE} \cdot \left\lfloor i_{\text{cell}}/\big(ncz \cdot \text{SIZE}^2 \cdot \lceil ncx/\text{SIZE} \rceil\big) \right\rfloor + \left\lfloor \operatorname{mod}(i_{\text{cell}}, \text{SIZE}^2)/\text{SIZE} \right\rfloor \\ i_z = \left\lfloor \operatorname{mod}(i_{\text{cell}}, ncz \cdot \text{SIZE}^2)/(\text{SIZE}^2) \right\rfloor, \end{cases}
$$

where SIZE is a parameter depending on the computer architecture, to be chosen such that data fit into the cache memory. As in Section 2.3, we compare the L6D ordering to three different strategies for ordering the cells: the $3d$ versions of the row-major, Morton, and Hilbert orderings. We present in Tables 2.3 and 2.4 the performance gains using these space-filling curves. We show that additional gains can be obtained with efficient computations of the bijection functions. In the tables, arrays means that we use additional arrays to store the indices i_x, i_y and i_z, otherwise we recompute them.

Comments. We focus on three meaningful comparisons in Tables 2.3 & 2.4. The first one is on the data structure: is it beneficial to use the redundant one for the $\mathbf{E}$ arrays? The only point where the code changes is in the update-velocities loop. We see in the tables that in $2d$, it is detrimental to use it if we stick to the row-major curve, but in $3d$ it is already beneficial with the canonical curve. We recall that we use many particles per grid cell, and that when using only a few particles per cell, the redundant data structure is not a good choice.
The second comparison concerns the data layout. Is it possible to obtain notable gains by changing the ordering of the grid cells? There are two places in the code where the changes might become significant: in the interpolation step (which is inside the Update $\mathbf{v}$ step in Tables 2.3 & 2.4) and in the accumulation step. This time, we can answer positively: taking another order than the canonical

[6]We recall that for any real number x, we denote by $\lfloor x \rfloor$ the greatest integer less than or equal to x. In addition, we denote by $\lceil x \rceil$ the least integer greater than or equal to x.

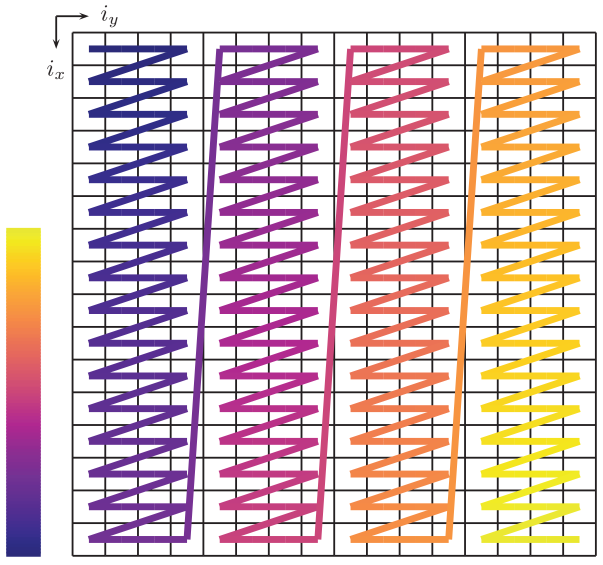


Figure 2.11: L4D layout of a 16 x 16 matrix

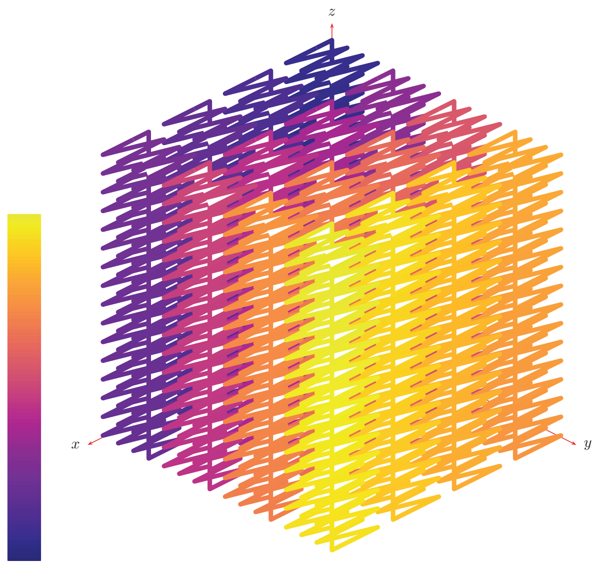


Figure 2.12: L6D layout of a 16 x 16 x 16 matrix

one, we can save time thanks to a reduction in the cache misses. In $2d$, the L4D and Morton curves seem to give similar and optimal timings, while in $3d$, the L6D curve allows additional gains and seems to be the best one.
The last comparison is on the particle data structure needed for the non-canonical orderings. We can either store the indices in additional arrays (here, arrays of `short int`), or re-compute them at each time step. When storing them, it requires more memory for the particles, therefore we need more time in the update-positions loop and in the sorting step.

We conclude by remarking that overall, the L6D curve led, in $3d$, to the most significant gain compared to the other existing orderings.

Strip-mining. In the $3d \times 3v$ setting too we apply the loop fission strategy, as described at item 5. in Section 2.3. However, the resulting code still needs to scan particle arrays three times, thus putting a lot of pressure on the memory bus. A loop transformation for avoiding this issue is the strip-mining [Wol95]. Instead of having three loops each scanning all the particles, we split the particle arrays in sub-arrays of size k (where k has to be chosen, depending on the architecture) and we have the three loops operate only on k particles. This transformation leads to the pseudo-code in Fig. 2.13 and speeds up the code by 22% in 2d (in our experiments choosing a strip-size k between 64 and 256 gives similar optimal results).
Unfortunately, this version of strip-mining does not improve performances in $3d$. This is explained by the fact that the cache is filled with too many values of $\mathbf{E}$, thus the expected gain in performance coming from the cache reuse of the particle arrays is out of reach. Thus, in $3d$, to be able to efficiently reuse the particle data, the strip-mining has to be done only on the two last loops. This transformation leads to the pseudo-code shown in Fig. 2.14 and speeds up the code by 12% (choosing a strip-size k between 32 and 128 gives similar optimal results).

Overall gains. The optimizations presented in this section are summarized in Table 2.5. The baseline is a version of the code with the standard $3d$ data structure for $\mathbf{E}$, the redundant one for ρ, and a loop hoisting as described in the previous section. The **overall gain** in the execution time is of **34%** with respect to the baseline code.

Table 2.3: $2d$ Space-filling Curves Timings

	Update $\mathbf{v}$	Update $\mathbf{x}$	Accumulate	Sort	Total
$2d$ standard	59.0	39.8	41.9	28.6	171.1
Row-major	63.6	39.7	42.8	28.6	176.8
L4D arrays	57.6	48.2	33.5	41.1	182.7
Morton arrays	60.2	48.0	29.4	40.7	180.7
Hilbert arrays	64.9	49.6	30.7	40.5	193.1
L4D	57.5	40.0	32.0	28.6	160.5
Morton	59.3	39.8	29.8	28.4	159.7
Hilbert	59.0	323.7	33.6	28.6	452.3

Time spent in the different loops (in seconds). Test case: $2d$ Landau damping on a $[0; 4\pi)^2$ grid decomposed in 512×512 cells, 1 billion particles, 100 iterations (sorting every 20 iterations), $\Delta t = 0.1$, Skylake 24 cores.

Table 2.4: $3d$ Space-filling Curves Timings

	Update $\mathbf{v}$	Update $\mathbf{x}$	Accumulate	Sort	Total
$3d$ standard	126.7	55.3	31.5	21.5	235.8
Row-major	92.6	55.3	31.5	21.4	201.7
L6D arrays	92.8	79.0	30.4	29.5	232.6
Morton arrays	96.5	79.0	30.3	27.5	234.2
Hilbert arrays	95.3	80.4	31.1	26.9	234.8
L6D	85.5	55.5	29.9	20.9	192.9
Morton	89.4	56.7	33.5	19.8	200.3
Hilbert	87.3	244.4	29.2	20.3	382.2

Time spent in the different loops (in seconds). Test case: $3d$ Landau damping as described in Section 2.1 with the initial condition in (2.2) with $\alpha = 0.01$ and $k = 0.5$, 1 billion particles, a grid of $64 \times 64 \times 64$ cells, 100 iterations (sorting every 10 time steps), $\Delta t = 0.05$, Skylake 24 cores.

Table 2.5: Gains of Different Optimizations

	Time (s)	Gains	Accumulated gains
Baseline	258.7	0.0%	0.0%
+ Loop Fission	235.8	8.9%	8.9%
+ Space-filling curve (L6D)	192.9	18.2%	25.4%
+ Strip-mining	169.5	12.1%	34.5%

Total execution time, gains and accumulated gains. Test case in Table 2.4.

```
**Foreach** subset of $k$ particles in $particles$,
    **Foreach** particle in this subset,
        Interpolate $\mathbf{E}$ to particle
        Update the velocity
    **Foreach** particle in this subset,
        Update the position
    **Foreach** particle in this subset,
        Accumulate particle charge to $\rho$
```

Figure 2.13: Particle-in-Cell pseudo-code with strip-mining.

```
**Foreach** particle in $particles$,
    Interpolate $\mathbf{E}$ to particle
    Update the velocity
**Foreach** subset of $k$ particles in $particles$,
    **Foreach** particle in this subset,
        Update the position
    **Foreach** particle in this subset,
        Accumulate particle charge to $\rho$
```

Figure 2.14: Particle-in-Cell pseudo-code with strip-mining on the two last loops only.

Table 2.6: Comparison between different codes

Code \ Nb. of particles	10^6	$16 \cdot 10^6$	10^9
Jocksch et al. work [Joc+16] (on 8 cores, 6.4 GB/s/core)	153.9	115.8	-
Our $3d \times 3v$ code in [4] (on 24 cores, 5.3 GB/s/core)	854.3	93.07	36.87

Time spent per particle per iteration per core, in nanoseconds. Test case in [Joc+16]: $[0; 2\pi)^2 \times [0; 1)$ decomposed in $512 \times 256 \times 1$ cells. Initial particle distribution uniform in space and velocity with $v_{max} = 1$ in each direction.

Eventually, maybe the most interesting measure of the performance of the whole simulation is **the total throughput of particles processed per unit time**. This number is $N_p \cdot N_{\text{iter}}/T$, where N_p is the number of particles, N_{iter} the number of iterations, and T the total execution time. Thus, in [4] we achieved simulations with 590 million particles processed per second, on 24 cores of Intel Skylake architecture, or equivalently, almost 25 million particles per second per core. This number was compared to that of another PIC code, presented in [Joc+16]. As an illustration we report in Table 2.6 the results in the $3d \times 3v$ setting. The simulations for three different particles number in Table 2.6 underline the following point: the redundant data structure for $\mathbf{E}$ is detrimental when using a low number of particles per cell (the case of 1 million particles), whereas as long as the particles per cell number is high enough, we obtain significant gain with respect to standard code.

Particles with chunk bags. We end this section by summarizing the contribution [1], where we propose an algorithm together with a **particle layout** whereby the particles are pushed **and** sorted on-the-fly. Thus, the integer i_{cell} is not stored anymore for each particle. In addition, these two points are efficiently implemented when exploiting vectorization and multithreading parallelism. These ideas come from the computer science literature and were mainly developed by Yann Barsamian and Arthur Charguéraud (Inria Nancy, CAMUS team) in the PIC framework and integrated in the code Pic-Vert. I will not enter into details, further comments on the topic can be found in [1] and the literature therein.

In this strategy, particles are stored cell by cell, in fixed-capacity arrays, called chunks. Several chunks might be needed to store all the particles contained in a same cell. Each cell is thus described by a linked list of chunks, a chunk bag. Actually two bags are associated with each cell,

corresponding to the future cell where a particle is pushed: a bag for slow particles (moving no more than one cell away) and a second bag for fast particles (moving more than 2 cells away). One reason for this bag separation is to preserve a high degree of multithreading parallelism when pushing the particles.
The algorithm minimizes the number of memory transfers: at each time step, each particle gets read from and is written to memory exactly once. The data structure does not involve any further move to additional buffers, nor reordering or shifting of data. This is a key feature for saving numerous memory operations, an important issue to fulfill since the PIC simulations are predominantly memory bound.
Finally, in [1] we assess the properties of the algorithm, as space usage, parallelization of critical loops, amount of memory transfers and we analyze its performance. Specifically, we compute the memory bandwidth (55% of the practical peak bandwidth is reached; see [1, Fig. 3]) and we study the impact of the fast moving particles: an efficient handling in parallel of such particles is achieved (see [1, Table 1]).

2.5 Parallelism results

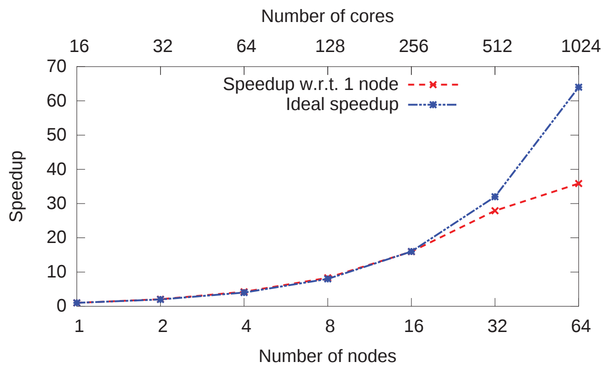
Our codes take advantage of a standard **hybrid parallelism**: the distributed memory paradigm (using multiple processes with MPI library) and the shared memory paradigm (using multiple threads with library OpenMP). These paradigms are utilized to parallelize the particle loops.

First, two technical points need to be emphasized. For the scaling measurements we used two supercomputers: *(i)* the Curie machine at the French GENCI-TGCC-CEA computing centre, where the typical used node was a dual socket Intel Xeon E5-2680 @ 2.7 GHz (Sandy Bridge), each socket having 64 GB of RAM, a maximum memory bandwidth of 51.2 GB/s, 4 memory channels and 8 cores, and *(ii)* the Marconi machine at the Italian supercomputing centre Cineca, with the Skylake architecture already mentioned in Section 2.2.
Then, while using MPI and OpenMP, care was taken to associate both processes and threads to the underlying hardware to reduce the requests to the remote memory. For this purpose, processes were placed in single processor sockets and threads in single cores. In this way, we attempted to minimize the negative impact of non-uniform memory access (NUMA) which can occur when a process uses cores placed in different sockets, for example. Thus we employed parallelization with multiple processes running on individual sockets and within these processes we parallelized with multiple threads. For example, on a Sandy Bridge node (having 2 sockets of 8 cores each), we used in the hybrid MPI + OpenMP approach, one MPI process per socket and 8 threads per process.

The state-of-the-art approach for parallelizing PIC simulations on distributed memory machines is to decompose the physical domain into smaller sub-domains and to assign the particles inside a sub-domain to a processor (among the wide literature, see *e.g.* [Bow+08; Ger+16]). In grid-based simulations, this strategy was shown to give good scaling results, as long as the work due to the communications through the sub-domain boundaries remains small compared to the computations inside the sub-domains. However, the main drawback of this technique is the difficulty of maintaining the load balance.

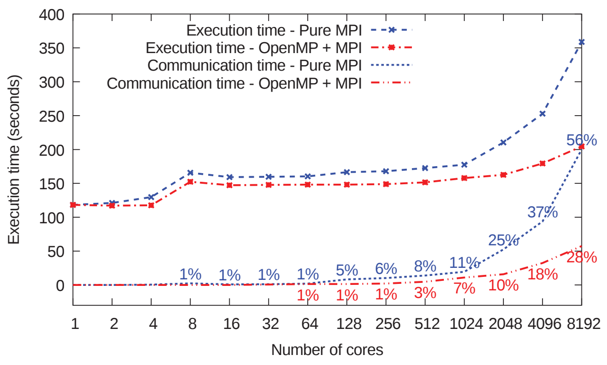
In our works [5; 3; 4] we handle the process-level parallelism by means of **particle decomposition** instead of domain decomposition: during the whole simulation, every process holds a fixed list of particles but it keeps track of all the grid quantities. Thus, at every iteration, every process accumulates the charge density associated with its particles and an `MPI_ALLREDUCE` operation gives the total charge density. The Poisson equation is then solved by every process over the whole grid.

Second, within each process, we add the multithreading parallelism for assigning different segments of a particle list to different threads.
The main advantage of this method is its simplicity: the only communication is via `MPI_ALLREDUCE` for the reduction of the charge density array and no particle has to move from one process to another during the simulation. Thus, all the computations are automatically work-balanced.



Test case: 256 x 256 cells, 800 million particles, 100 iterations simulation (sorting every 20 iterations). Architecture: Sandy Bridge.

Figure 2.15: Strong scaling on supercomputer Curie.

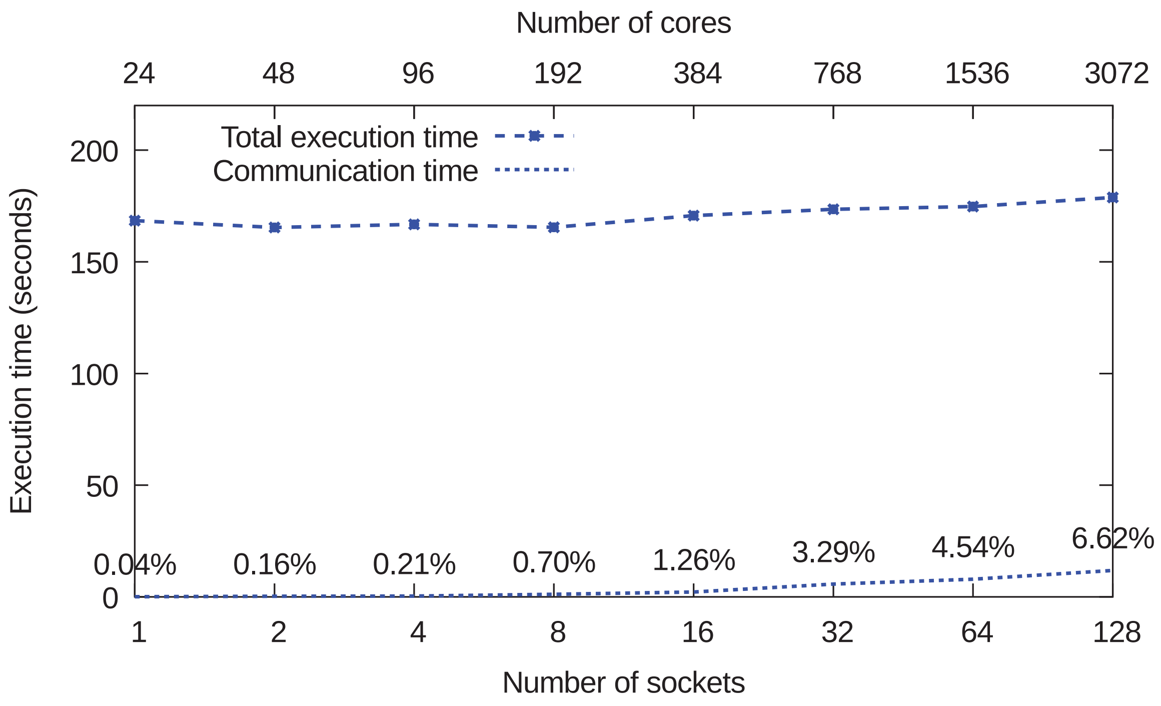


Test case: 128 x 128 cells, 50 million particles per core, 100 iterations (sorting every 50 iterations). Architecture: Sandy Bridge. Communication time is shown as percentage of the execution time.

Figure 2.16: Weak scaling on supercomputer Curie.

The bottleneck of this approach is that the scalability is highly limited by the global reduction step: when using a large number of processes, the communication becomes too costly. However, this issue may be controlled in practice, by using the full memory on each MPI process to put particles (see below for explanations). A second drawback of our approach is that in realistic simulations, requiring much larger grids than the $64 \times 64 \times 64$ grid used in our simulations, the very large number of cells makes the computations on one single process inefficient, due to the high number of cache misses involved. A solution to this problem may be the use of the domain decomposition method.

In order to assess the performance of the parallelization strategy we evaluated the strong and weak scaling. To compute the strong scaling, a problem with a fixed numerical size is ran over a growing number of processors. For drawing the weak scaling, we run simulations with a constant problem size by processor, for a growing number of processors. We need to specify that because we use particle decomposition and not domain decomposition, our weak scaling only increases the number of particles and not the grid size: see Fig. 2.16 for the $2d \times 2v$ case and Fig. 2.17 for the $3d \times 3v$ case.
First, we obtained the classic result of the advantage of using a hybrid parallelism instead of a pure multiprocess one, see Fig. 2.16.
Then, we demonstrated that as long as we use the full memory of every core, we achieve good weak scaling, with an acceptable communication time, up to 8 192 cores in a $2d \times 2v$ simulation and up to 3 072 cores in a $3d \times 3v$ simulation. More precisely, taking a look at the strong scaling in Fig. 2.15 we can see that the computation time per process decreases with the increasing processes number (since the number of particles per process decreases), while the communication time as percentage of the total time grows with the increasing processes number. Thus, in the case of the 64 nodes (with 128 processes), only 6.25 million particles are distributed per process and the MPI communications take 32% of the total time (Fig. 2.15). Nevertheless, from Fig. 2.17 we deduce



Test case: 64 x 64 x 64 grid, 1 billion particles per socket, 100 iterations simulation (sorting every 10 iterations). Architecture: Skylake. Communication time is shown as percentage of the execution time.

Figure 2.17: Weak scaling on supercomputer Marconi.

that using 400 million particles per process, the same number of processes (128) leads to far better results, since communications take in this case only 7% of the total time.

2.6 Conclusions and outlook

Conclusions. In this chapter, we exposed strategies to efficiently implement a Particle-in-Cell method for solving Vlasov–Poisson models in four dimensional and six dimensional phase spaces respectively. The focus of the optimization strategies was the particle push. The code organization mainly focused on optimization of the memory access, since a PIC implementation is memory bounded. We exploited also the vectorization capability, and the MPI and OpenMP parallelism for running simulations over a few thousands of cores. The code performance was assessed in Landau damping simulations. In addition, we accomplished code verification through other challenging test cases.

Perspectives. A first important issue is **to study and to adapt** the preceding procedures to **more complex problems**, with a high interest in physics, since only rather academic test cases have been treated in this chapter. In the long term, it would be interesting to perform simulations in order to understand challenging problems in laser systems, laboratory astrophysics, or other applications in plasma physics. In this direction, we make use of some of these optimization techniques in order to perform simulations of a two species plasma in $2d \times 2v$, in [2]. Part of this work is devoted to a mixing of the semi-Lagrangian and the PIC methods, since their cross-comparison can serve as further validation. Actually, we implemented in [2] both methods in the same code, as part of the SeLaLib library [Sel]. In this framework, we could consider the use of the PIC method for a species of particles and the semi-Lagrangian for other species, having thus the possibility to take advantage of the benefits of both methods.

A second important point to be done in the future is to study, to adapt, and to improve if necessary, the optimization techniques and algorithms developed in this chapter when one or several building blocks of the PIC method, developed in Section 2.1, are changed as follows.

- **Use of other time-stepping schemes.** The efficacy of the optimization strategies should be coupled with new innovative numerical schemes. More precisely, another perspective is to implement more accurate and stable time discretization schemes for solving multiscale problems, since in the present work, we only used standard explicit numerical methods. For example, implementing the time-stepping schemes developed in Section 1.3, in the high-performance computing framework may be an interesting problem which would lead firstly, to an improvement of the accuracy of the schemes themselves when applying the techniques developed in this chapter as they are, or secondly, to new optimization techniques adapted to the time-stepping scheme.
- **Use of higher degree convolution kernels.** It is important to integrate splines of higher degree (than 1) in the interpolation and accumulation steps. This fact requires more computations and most importantly, more data (in terms of $\mathbf{E}$ and ρ) to be moved from/to the memory. Thus, studying the behaviour of the data structures and layouts when higher degree convolution kernels are used is mandatory.
- **Use of other boundary conditions.** Another point to be developed is how to handle the optimization techniques of the updating positions step in the setting of other boundary conditions, not only periodic.

The **numerical solution of the Poisson equation** was not in the center of our optimization efforts. We already emphasized in the previous section that the solution method of the Poisson equation is not suitable for realistic simulations. Thus, more sophisticated solutions should be considered. Moreover, in space, we used simple Cartesian grids for modelling simple geometry and only periodic boundary conditions were implemented. If changing to curvilinear meshes and considering different boundary conditions are to be heeded, then it is essential to study the behaviour of the data structures in these configurations and adapt their implementation in view of performance. As already remarked, adding a layer of domain decomposition would equally help to deal with larger geometries.

Finally, it would be interesting to port and adapt all the techniques presented above on **other architectures** with a larger number of cores, such as Graphics Processing Unit (GPU) or Intel Many Integrated Core (MIC) architecture, which offer a massive thread parallelism. This is however not a straightforward task, since the complexity of some of the hardware which may require using different programming languages is challenging. We recall that we aim at focusing our research efforts in performing more realistic simulations and including more physical effects, together with the adaptation of the optimization techniques to these configurations. Therefore, addressing the challenges that the modern hardware architectures pose is essential in the search for the most efficient use of computer resources.

Chapter 3

Computational models for plasma physics problems

The steering vector of the work in this chapter is the understanding of some complex problems arising in plasma physics, by means of devising relevant mathematical models and reliable numerical methods for solving them. We thus present numerical results of kinetic-fluid simulations which are relevant for **the computational modelling of realistic problems** in plasma physics. The models are challenging because of the multiscale behaviour of their solutions. This difficulty comes from the realistic physical parameters (*e.g.* the species mass ratio, the Debye length) that need to be taken into account in order to follow some physical phenomena that are not well understood. Thus, the contribution of this chapter is to solve some complex physical problems with standard mathematical models and without requiring involved implementation of the numerical schemes. Therefore, the numerical schemes are not standard but chosen in order to adapt to the difficulties of the mathematical models.

The equations considered in this chapter are of Vlasov type and include source and collision terms to model specific phenomena in quasi-neutral plasma. The self-consistent electric field is obtained by coupling the previous system with a reformulated Poisson equation, for asymptotic preserving reasons. A guiding center model, coupling a continuity equation to standard Poisson equation is also studied in order to analyze the diocotron instability in a non-neutral plasma.

The presentation follows the references [21; 20; 22; 9; 11; 19; 13]. The summary of the chapter is the following:

- In Section 3.1 we study the problem of the dynamics of charged particles following an edge-localized mode (ELM) event. We model this problem in a kinetic setting for two species of particles with realistic mass ratio and then we perform Eulerian simulations in order to study the evolution of the particles and their interactions. Then, a hybrid (kinetic and fluid) model is considered in order to address more realistically the physical problem. The outcome of the numerical simulations allows to identify interesting complex physical phenomena that are eventually analyzed.

- In Section 3.2, we discuss the problem of the diocotron instability by means of the guiding center model in polar variables. We propose boundary conditions that are proved to lead to some conservation properties. Instability rates are also displayed. These questions are verified by simulation with a semi-Lagrangian method.

- In Section 3.3 we conclude the chapter and some perspectives are drawn.

3.1 The dynamics of edge-localized modes

Introduction. The mathematical context of this section is that of solving Vlasov–Poisson equations for modelling the parallel transport of ions and electrons in the scrape-off layer of a fusion plasma. This work is the fruit of several years of collaboration with Giovanni Manfredi (Institut de Physique et Chimie des Matériaux de Strasbourg (IPCMS), Université de Strasbourg). In this frame, the main mathematical contribution of this section entails two points.

In a first step, I proposed to use the **reformulated Poisson equation** in a coupling with Vlasov equations for both particle species. The reformulated equation is equivalent to the Poisson equation, but it allows to compute the self-consistent electric potential when the Debye length vanishes. In this way, one can perform simulation with realistic Debye length and with a grid step which has not to resolve this scale. Then, I implemented a finite difference scheme for the reformulated equation. Simulations of the coupled system showed the main features of an edge-localized mode and revealed interesting physical behaviour (like an early burst of suprathermal electrons).

In a second step, together with Giovanni Manfredi and David Coulette, we enriched the previous model with transport equations with source terms for both particle species (see (3.16)), in order to include effects due to the perpendicular dynamics. More precisely, the **new fluid equations for the perpendicular temperatures** entail a collision operator which models the isotropisation process, through which the parallel and the perpendicular dynamics balance. Then, I solved numerically these equations in a finite volume approach. More precisely, I implemented an upwind finite volume method improved with high-resolution corrections allowing to remove the spurious oscillations. Simulations of this model precisely identified the effect of the isotropisation process on different plasma properties.
I finally mention that our implementations are performed in a standard manner into a `Fortran` code (of less than 3000 lines), without involved physical considerations.

Before detailing these points, I develop first the physical problem we aim at solving and the context of our contributions.

The physical context. This section is concerned with the study of the dynamics of charged particles along a magnetic field line in a tokamak scrape-off layer (SOL) following an edge-localized mode (ELM) event. An ELM is a violent outburst of plasma in the SOL, which is the plasma region characterized by open magnetic field lines. Once the ELM-driven plasma pulse has crossed the magnetic separatrix[1], it travels mainly parallel to the magnetic field lines and ends up hitting the divertor plates. The resulting particle and energy fluxes on these components lead to a decrease of their lifetime, which is an outstanding issue in the operation of tokamak devices.

In our approach, we assume that the charged particles travel along the magnetic field lines, but not across them. Thus, we model the transport of two species of particles (ions and electrons) in the direction parallel to the magnetic field by Vlasov–Poisson equations, while in the perpendicular plane the plasma is supposed to be Maxwellian. We have first developed a $1d \times 1v$ Vlasov–Poisson code with open boundaries for ions and electrons [21]. The simulations reproduced with good accuracy the principal features of the plasma transport during ELMs, like the evolution in time of the particle and energy fluxes at the boundaries.

In a second stage, we compare in [20], the numerical results of our code with two different approaches: the $1d \times 3v$ Particle-in-Cell code `BIT1` (developed at the University of Innsbruck, see

[1] the boundary between closed and open field lines, separating the confined plasma region from the region where field lines connect to material surfaces.

[Tsk+08; Mou+13b]) and the one dimensional fluid code `SOLF1D` (developed at Culham, UK) which is based on a set of Braginskii-like equations [Hav+11]. These comparisons show a very satisfactory agreement with the PIC code when computing the energy fluxes of both species, whereas the fluid code overestimates these fluxes (see also [Hav+12]). We also note that our code and the PIC code yield similar results for the early burst of electrons, whereas the fluid code is not able to reproduce this effect, thus confirming its kinetic origin.

The numerical results in [21] are compared with analytical estimates based on a free-streaming model, with satisfactory general agreement. The free-streaming model (developed in [FP+06]) is the simplest kinetic description of the parallel transport, but neglects the plasma potential and the electron quantities. In [22] we proposed a set of modified and augmented (with electron quantities) free-streaming equations in order to overcome these two drawbacks. The new equations are benchmarked against (and justified by) numerical simulation of the Vlasov–Poisson equation in [21]. The agreement found between the two approaches justifies the applicability of the free-streaming model.

In the above models the parallel and perpendicular dynamics are completely decoupled for both particle species and the perpendicular velocity distributions are assumed to be Maxwellian with constant temperature. The purpose of [9] is to ascertain [Don+13] if the collision-driven relaxation between the parallel and perpendicular temperatures of each species during the ELM propagation can modify the shape of the distribution function and consequently the fluxes reaching the wall. Such a phenomenon was numerically shown by the outcomes of the PIC code `BIT1`, in [Mou+13b]. In order to examine this problem without developing a $1d \times 3v$ code, we extend in [9] the $1d \times 1v$ Vlasov–Poisson model by including a fluid equation for the evolution of the perpendicular temperature of both species. In the full hybrid[2] model, the coupling between the perpendicular and parallel quantities is performed through a collision operator.

Finally, in [11], the $1d \times 1v$ Vlasov–Poisson code was integrated to a $1d \times 3v$ setting. The aim of this novel framework is to study, not only the ELM dynamics, but also the plasma-wall interactions in the presence of a magnetic field. Thus, the particles are additionally subject to the action of a uniform external magnetic field, tilted with respect to the wall surface.

In the following, we detail the models and the numerical methods to solve these problems.

A $1d\times1v$ model for the parallel transport. Under the assumption that the charged particles travel along the magnetic field lines but not across them, we can adopt a one dimensional geometry along the parallel direction, while in the perpendicular plane the distribution function remains Maxwellian at all times. Therefore, in a $1d \times 3v$ phase space, the distribution (for each species j) reads (see [21]):

$$f_j(t,x,\mathbf{v}) = g_j(t,x,v_x)\, M_j(v_\perp), \tag{3.1}$$

where $\mathbf{v} = (v_x, v_\perp)$, $v_\perp = (v_y, v_z)$, $M_j(v_\perp) = 1/(2\pi\, v_{T_j}^2)\exp(-|v_\perp|^2/(2v_{T_j}^2))$, $v_{T_j} = \sqrt{T_j/m_j}$ is the thermal speed, and T_j and m_j stand for the temperature and the mass, respectively. The temperature is assumed to be the same for both species, $T_i = T_e = T_{\mathrm{ELM}}$. With this notation, the ion and electron evolution are described by Vlasov equations in the parallel phase space (x, v_x):

$$\partial_t g_j + v_x \partial_x g_j - \frac{q_j}{m_j}\partial_x\phi\, \partial_{v_x} g_j = S_j(t,x,v_x), \tag{3.2}$$

[2]a kinetic approach for the rapid parallel transport plus a fluid model for the slower perpendicular process.

where $q_j = \pm e$ is the charge and the electrostatic potential $\phi \equiv \phi(t,x)$ obeys the Poisson equation

$$\frac{\partial^2 \phi}{\partial x^2} = -\frac{e}{\epsilon_0}(n_i - n_e), \tag{3.3}$$

where n_j is the density of species j. The above equations are solved on an interval $x \in [-L, L]$, where L, the so-called parallel connection length, is the typical distance between the outer midplane and the divertor target. Also, $x = \pm L$ represent the locations of the divertor plates. The plates are assumed to be perfectly absorbing surfaces and are kept at constant electric potential, thus $\phi(\pm L) = 0$. The source terms S_j describe the growth of the **ELM event** and are written in the form

$$S_j(t,x,v_x) = s(t)\, n_{\mathrm{ELM}} \exp\left(-\frac{x^2}{2\sigma}\right) \frac{\exp(-v_x^2/(2v_{T_j}^2))}{\sqrt{2\pi} v_{T_j}},$$

where $s(t)$ models the pulse temporal profile. The parameter n_{ELM} denotes the peak density and $\sigma \sim L/10$. The values of the parameters were chosen to be typical for a realistic type-I ELM in the JET tokamak (see [FP+06]). Here, we are mainly interested in the plasma features on the plates, and in particular in the particle and energy fluxes (for both species) defined respectively by

$$\Gamma(t) = \int v_x f(t, \pm L, \mathbf{v})\, d\mathbf{v},$$

$$Q(t) = \int \frac{1}{2} v_x |\mathbf{v}|^2 f(t, \pm L, \mathbf{v})\, d\mathbf{v}.$$

The relevant physical regime is determined by **two dimensionless parameters**, the ion-to-electron mass ratio $\varepsilon = m_e/m_i = 1/1836$ and the ratio of the Debye length to the parallel connection length, $\lambda = \lambda_D/L \sim 10^{-6}$. After normalizing equations (3.2)-(3.3), we obtain the following scaled two species Vlasov–Poisson system

$$\partial_t g_i + v_x \partial_x g_i - \varepsilon\, \partial_x \phi\, \partial_{v_x} g_i = S_i(t,x,v_x), \tag{3.4}$$
$$\partial_t g_e + v_x \partial_x g_e + \ \partial_x \phi\, \partial_{v_x} g_e = S_e(t,x,v_x), \tag{3.5}$$
$$-\lambda^2 \frac{\partial^2 \phi}{\partial x^2} = n_i - n_e. \tag{3.6}$$

At the limit $\lambda \to 0$, called the quasi-neutral regime, the electric potential does not vanish but takes exactly the value required to enforce the quasi-neutrality constraint. However, the Poisson equation (3.6) becomes singular when $\lambda \to 0$ and therefore, it would be useful to replace it with an equivalent equation that can provide ϕ when $\lambda \to 0$. This can be achieved by employing the **"reformulated Poisson equation"** (see *e.g.* [CDV07] where the equation was used in the framework of an asymptotic preserving scheme for a fluid model in the quasi-neutral limit):

$$\lambda^2 \frac{\partial^2}{\partial t^2}\left(\frac{\partial^2 \phi}{\partial x^2}\right) + \frac{\partial}{\partial x}\left[(\varepsilon\, n_i + n_e)\frac{\partial \phi}{\partial x}\right] = -\frac{\partial^2}{\partial x^2}(R_i - R_e), \tag{3.7}$$

where $R_j = \int g_j v_x^2 dv_x$. The equation (3.7) is derived as follows. By taking the first two moments of the difference between (3.4) and (3.5), we obtain

$$\partial_t(n_i - n_e) + \partial_x(J_i - J_e) = 0, \tag{3.8}$$
$$\partial_t(J_i - J_e) + \partial_x(R_i - R_e) + (\varepsilon\, n_i + n_e)\partial_x \phi = 0, \tag{3.9}$$

where $J_j = \int g_j v_x dv_x$. We then subtract the spatial derivative of equation (3.9) from the time derivative of equation (3.8):

$$\frac{\partial^2}{\partial t^2}(n_i - n_e) - \frac{\partial}{\partial x}\left[(\varepsilon\, n_i + n_e)\frac{\partial \phi}{\partial x}\right] = \frac{\partial^2}{\partial x^2}(R_i - R_e).$$

Finally, using Poisson's equation we express $\partial^2_{tt}(n_i - n_e)$ in terms of ϕ and thus recover the reformulated Poisson equation (3.7) (see also [21, Appendix A] and the references therein for the justification of the equivalence with the standard Poisson equation). The specificity of equation (3.7) is that it is asymptotic preserving in the small parameter λ, leading to the possibility of using a grid spacing that exceeds the Debye length. In addition, when solving it implicitly in time, simulations remain stable with a time step larger than the inverse plasma frequency. This implies non-negligible gains in the computational time.

The numerical approach is classically used in the literature. The Vlasov equations are solved using an Eulerian method based on a uniform meshing of the parallel phase space (x, v_x). For the time-stepping, a classical second-order splitting scheme is used, which solves alternatively the advection in the physical space and the advection in the velocity space. Each advection step is performed using a third-order positive flux-conservative (PFC) method, with a slope corrector that prevents the distribution function from becoming negative [FSB01]. A typical simulation requires 1000 points both in physical space and in velocity space. The reformulated Poisson equation is solved with finite differences in time and in space, with an implicit discretization for the time-stepping, which allows to remove the standard numerical constraint imposed to the time step by explicit schemes for stability reasons [Deg+10]. More precisely, we first discretize in time equation (3.7) as follows

$$-\frac{\partial}{\partial x}\left(\lambda^2 \frac{\partial_x \phi^{m+1} - 2\partial_x \phi^m + \partial_x \phi^{m-1}}{\Delta t^2} + (\varepsilon n_i^m + n_e^m)\partial_x \phi^{m+1}\right) = \frac{\partial^2}{\partial x^2}(R_i^m - R_e^m),$$

where ϕ^m stands for the value of ϕ at time $t^m = m\,\Delta t$. Then, using Poisson's equation, we replace $\partial^2_{xx}\phi^m$ and $\partial^2_{xx}\phi^{m-1}$ by density terms at the corresponding time steps

$$-\frac{\partial}{\partial x}\left[\Big(\frac{\lambda^2}{\Delta t^2} + \varepsilon n_i^m + n_e^m\Big)\partial_x \phi^{m+1}\right] = \frac{\partial^2}{\partial x^2}(R_i^m - R_e^m) + \frac{1}{\Delta t^2}\Big(2(n_i^m - n_e^m) - (n_i^{m-1} - n_e^{m-1})\Big).$$

Next, we make use of a discrete form of equation (3.8) to replace $(n_i^m - n_e^m) - (n_i^{m-1} - n_e^{m-1})$ with $-\Delta t(\partial_x J_i^m - \partial_x J_e^m)$. We thus obtain the semi-discrete version of the reformulated Poisson equation

$$-\frac{\partial}{\partial x}\left[\Big(\lambda^2 + \Delta t^2(\varepsilon n_i^m + n_e^m)\Big)\partial_x \phi^{m+1}\right] = \Delta t^2 \frac{\partial^2}{\partial x^2}(R_i^m - R_e^m) + (n_i^m - n_e^m) - \Delta t(\partial_x J_i^m - \partial_x J_e^m),$$

which allows to compute ϕ^{m+1} from known quantities at time t^m. A finite difference method is then used to approximate the spatial derivatives in the equation, leading to a linear system to be solved. Thanks to the asymptotic preserving property of this numerical scheme for the reformulated Poisson equation, we employed in simulations a grid spacing that exceeds the Debye length and a time step larger than the inverse plasma frequency ω_{pe}^{-1}: we have typically used

$$\Delta x = 2\lambda_D{}^{3} \quad \text{and} \quad \Delta t = 4\omega_{pe}^{-1}.$$

Thus, we obtained a **reduced computational cost** with respect to standard explicit schemes which is quite important when performing long time simulations, of thousands of ω_{pe}^{-1}.

[3]in order to accurately resolve the sheath scale. The sheaths form at the boundary and have a thickness of a few Debye lengths.

Conclusion. The results of the numerical simulations performed in this framework draw several interesting physical questions, among which:

- We obtain with rather good accuracy some of the main features of an ELM signal, most notably its rapid rise followed by a much slower decay (see Fig. 3.1).
- The ion and electron particle fluxes are virtually identical, a consequence of quasi-neutrality. In contrast, the electron energy flux is smaller than that of the ions, the ratio depending of the temporal profile.
- For moderate connection lengths ($L \sim 1000\lambda_D$), the particles and energy fluxes are correctly described: by decreasing λ, the fluxes are virtually unchanged. Thus, even if simulation were performed for $\lambda \sim 10^{-3}$, the results for fluxes should remain the same for realistic $\lambda \sim 10^{-6}$.
- On the electron transit time scale, an early burst can be observed on the divertor plates, which corresponds to suprathermal electrons escaping the ions attraction due to their large kinetic energy. The remaining electrons are trapped in the potential well created by the ions (which now outnumber the electrons) and evolve in unison with them. On the ion timescale, the plasma is everywhere neutral, except for a positive net charge in front of the divertor plates (the sheaths).

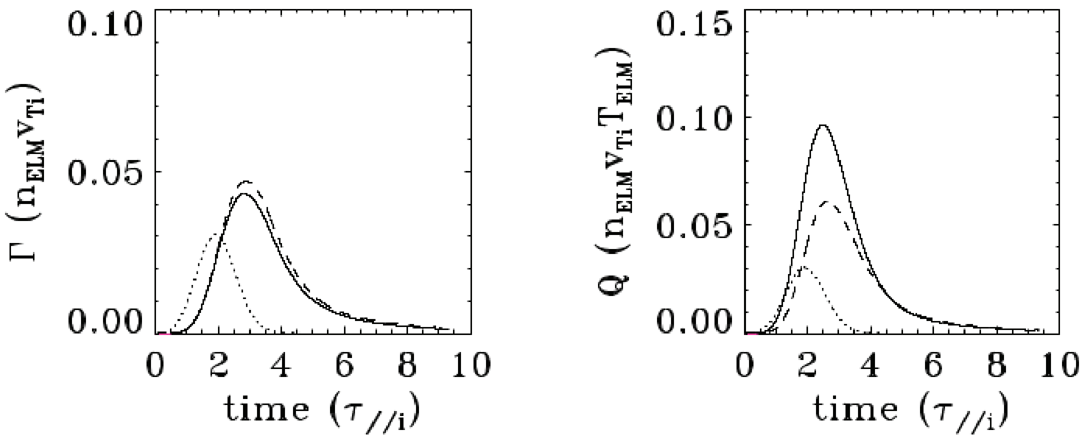


Figure 3.1: Evolution in time of the particles (left) and energy (right) fluxes, for the case with time-distributed source. Solid lines refer to the ions, dashed lines to the electrons. The dotted lines represent the source temporal profile s, in arbitrary units.

Parallel transport model with perpendicular temperature. The preceding model is extended in this part by assuming that the perpendicular temperature is no more constant, but depends on x and t. The aim of the approach in [9] is to study the influence of the relaxation between the parallel and perpendicular temperatures of each species on the fluxes at the divertor plates. This question is motivated by the work in [Mou+13b] which showed by means of a $1d \times 3v$ code that the transfer of electron thermal energy from the perpendicular plane to the parallel direction could indeed impact significantly the energy fluxes of both species. In the following we explain how to derive the fluid equation for the perpendicular temperature. Together with the

Vlasov–Poisson system in the parallel direction, the equation for the perpendicular temperature is able to capture the previous phenomenon, with an additional computational cost which is low in front of a $1d \times 3v$ kinetic code.

We broaden now the hypothesis in (3.1) by assuming that the distribution function can be written as

$$f_j(t,x,\mathbf{v}) = g_j(t,x,v_x)\,\frac{1}{2\pi T_{\perp,j}(t,x)}\exp\left(-\frac{m_j|v_\perp|^2}{2T_{\perp,j}(t,x)}\right), \tag{3.10}$$

where we recall that j stands for electrons or ions. The evolution of the full distribution function f_j is governed by a Vlasov equation with a BGK collision operator and a source term (see [9]):

$$\partial_t f_j + v_x\partial_x f_j - \frac{q_j}{m_j}\partial_x\phi\,\partial_{v_x} f_j = \mathcal{C}_j(f_j) + S_j, \tag{3.11}$$

with

$$\mathcal{C}_j(f_j) = \nu_j(f_{M_j} - f_j), \tag{3.12}$$

$$f_{M_j}(t,x,\mathbf{v}) = n_j\left(\frac{m_j}{2\pi T_j}\right)^{3/2}\exp\left(-\frac{m_j(v_x-u_{x,j})^2}{2T_j}\right)\exp\left(-\frac{m_j|v_\perp|^2}{2T_j}\right), \tag{3.13}$$

where in addition to the usual notation, ν_j is the relaxation rate for each species j, and n_j, $u_{x,j}$ and T_j are respectively the density, the mean velocity, and the temperature. These quantities are computed self-consistently from the distribution function f_j. Note that only drifts in the parallel direction are allowed in (3.13), *i.e.*, we assume that $\int f_j\, v_\perp d\mathbf{v} = 0$. The effect of the BGK term is to drive f_j towards the isotropic Maxwellian f_{M_j}. As in the previous part, the source term S_j models the ELM event:

$$S_j(t,x,\mathbf{v}) = s(t)\,N(x)\,G_j(v_x)\,H_j(v_\perp),$$

where $s(t)$ is a given temporal profile and

$$\begin{aligned} N(x) &= n_0\exp(-x^2/(2\sigma_0^2)) \\ G_j(v_x) &= \sqrt{m_j/(2\pi T_{||0})}\,\exp(-m_j v_x^2/(2T_{||0})) \\ H_j(v_\perp) &= m_j/(2\pi T_{\perp 0})\,\exp(-m_j|v_\perp|^2/(2T_{\perp 0})). \end{aligned}$$

We assume no temperature anisotropy for the growing ELM, *i.e.* $T_{||0} = T_{\perp 0} = T_0$. Injecting (3.10) into equation (3.11) and integrating over $v_\perp$, we otain the evolution equation for the parallel distribution g_j

$$\partial_t g_j + v_x\partial_x g_j - \frac{q_j}{m_j}\partial_x\phi\,\partial_{v_x} g_j = \nu_j(g_{M_j} - g_j) + s(t)\,N(x)\,G_j(v_x), \tag{3.14}$$

where

$$g_{M_j}(t,x,v_x) = n_j\left(\frac{m_j}{2\pi T_j}\right)^{1/2}\exp\left(-\frac{m_j(v_x-u_{x,j})^2}{2T_j}\right). \tag{3.15}$$

Then, taking second order moment in $v_\perp$ of equation (3.11), we derive **the evolution equation for $T_{\perp,j}$**

$$\partial_t T_{\perp,j} + u_{x,j}\,\partial_x T_{\perp,j} = \frac{\nu_j}{3}(T_{||,j} - T_{\perp,j}) + \frac{s\;N\,(T_{||0} - T_{\perp,j})}{n_j}, \tag{3.16}$$

where $T_{||,j}(t,x) = \dfrac{m_j}{n_j(t,x)}\displaystyle\int g_j(t,x,v_x)\,\big(v_x - u_{x,j}(t,x)\big)^2 dv_x$ and $T_j = \dfrac{1}{3}\big(T_{||,j} + 2T_{\perp,j}\big)$. The coupling between the parallel and perpendicular dynamics occurs in equation (3.14) through the total temperature in the parallel BGK operator, and in equation (3.16) through the quantities $n_j, u_{x,j}$, and $T_{||,j}$, which are moments of the distribution function g_j. Naturally, setting $\nu_j = 0$ in (3.14) we recover the collisionless model in (3.2). We underline the first term in the right-hand side of (3.16) which represents **the temperature isotropisation**. We also note that $T_{\perp,j}$ is transported by the parallel fluid velocity $u_{x,j}$. Eventually, the final model is the system formed by equations (3.14) and (3.16) for both ions and electrons, supplemented with the reformulated Poisson equation.

For the numerical approach, the same technique as in the previous part is adopted for the Vlasov and the reformulated Poisson equations. Next, we discuss the additional transport equation (3.16). Firstly, we use in time the classic Strang splitting, alternating in time the following steps:

$$\text{for } j \in \{i,e\} \text{ solve with } \Delta t/2 \text{ the equation} \quad \partial_t T_{\perp,j} = \frac{\nu_j}{3}\big(T_{||,j} - T_{\perp,j}\big) + \frac{s\ N\,(T_{||0} - T_{\perp,j})}{n_j}, \tag{3.17}$$

$$\text{for } j \in \{i,e\} \text{ solve with } \Delta t \text{ the equation} \quad \partial_t T_{\perp,j} + u_{x,j}\,\partial_x T_{\perp,j} = 0, \tag{3.18}$$

$$\text{for } j \in \{i,e\} \text{ solve with } \Delta t/2 \text{ the equation} \quad \partial_t T_{\perp,j} = \frac{\nu_j}{3}\big(T_{||,j} - T_{\perp,j}\big) + \frac{s\ N\,(T_{||0} - T_{\perp,j})}{n_j}.$$

Then, equation (3.17) is simply solved with an Euler scheme. Next, we detail the numerical approach for the advection equation in (3.18). Even though this equation is not conservative, it can be solved with an upwind finite-volume method improved by using high-resolution corrections [Lev02]. More precisely, following [Lev02], we denote by Q_i^n an approximation to the cell average of $T_{\perp,j}(t,x)$ over cell i at time t_n. Unlike a classical conservative scheme, the change in this cell average is not given by a flux difference, but it can be computed from the wave $\mathcal{W}_{i-1/2} = Q_i - Q_{i-1}$ and the speed $s_{i-1/2} = u_i$. Thus, the upwind method writes if $u > 0$

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x} s_{i-1/2}\mathcal{W}_{i-1/2},$$

or more generally, to be used also when $u < 0$,

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\Big(s^+_{i-1/2}\mathcal{W}_{i-1/2} + s^-_{i-1/2}\mathcal{W}_{i+1/2}\Big),$$

where $s^+ = \max(s,0)$ and $s^- = \min(s,0)$. This first order method can be improved by using high-resolution corrections, namely

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\Big(s^+_{i-1/2}\mathcal{W}_{i-1/2} + s^-_{i-1/2}\mathcal{W}_{i+1/2}\Big) - \frac{\Delta t}{\Delta x}\big(F_{i+1/2} - F_{i-1/2}\big), \tag{3.19}$$

where the left flux is defined by

$$F_{i-1/2} = \frac{1}{2}|s_{i-1/2}|\big(1 - \frac{\Delta t}{\Delta x}|s_{i-1/2}|\big)\widetilde{\mathcal{W}}_{i-1/2},$$

and where $\widetilde{\mathcal{W}}_{i-1/2}$ is obtained by applying a wave limiter

$$\widetilde{\mathcal{W}}_{i-1/2} = \mathcal{W}_{i-1/2} \cdot \ell\Big(\frac{\mathcal{W}_{I-1/2}}{\mathcal{W}_{i-1/2}}\Big)$$

with $I = i-1$ if $s_{i-1/2} > 0$ and $I = i+1$ if $s_{i-1/2} < 0$. We chose the minmod limiter

$$\ell(\theta) = \text{minmod}(1,\theta).$$

Remark 3.1.1. *In (3.19), when computing the fluxes, we took $s_{i-1/2} = s_{i+1/2} = u_i^n$. A better way would be to have u^n at $x_{i-1/2}$ and at $x_{i+1/2}$ and to use these values instead of u_i^n. Therefore, we make tests by taking a linear interpolation from u_i and u_{i-1} for computing $u_{i-1/2}$ and similarly for $u_{i+1/2}$ but the results are very close to those obtained with $s_{i-1/2} = s_{i+1/2} = u_i^n$.*
The scheme in (3.19) is not formally second-order accurate when applied to a smooth solution, even without limiter [Lev02]. However, in practice, thanks to the limiter, it works much better than the second-order Lax-Wendroff method which gives in our case spurious oscillations at the boundaries.

In the simulations, the time step is variable (between one-half and four times the inverse plasma frequency) in order to guarantee that the Courant-Friedrichs-Lewy (CFL) condition $|u_{x,e}|\,\Delta t < \Delta x$ is always satisfied. Details about the numerical parameters are given in [9].

Conclusion. As our main objective is to assess the impact of the parallel-perpendicular coupling on the fluxes reaching the target plates, we perform parametric scans in the collision rate ν_j. We draw in Figs. 3.2 & 3.3 particle and energy fluxes for both species, for several collision rates. The values of the ion and electron collision rates are not set independently but adjusted to be ELM-relevant [Don+13], namely $\nu_e/\nu_i = \sqrt{m_i/m_e}$. The outcomes of the simulations confirmed the following important result: while ion-ion collisions have an almost negligible effect, the impact of the electron-electron collisions can be quite significant on the various fluxes, both for electrons and ions. More precisely, the dominant effect is the electron-electron isotropisation, by which electron thermal perpendicular energy is transferred to the parallel motion. Part of this energy accelerates the electrons in the parallel direction, another part is transferred to the ions through acceleration by the self-consistent electric field. The net result is an increase (with increasing collision) of the peak values of the ion particle and energy fluxes at the target plate (see Fig. 3.3(a)&(d)), while the total electron energy flux decreases (see Fig. 3.2(d)).

3.2 The diocotron instability

Introduction. The contribution presented in this section, based on [19; 13], consists in semi-Lagrangian simulations for a $2d$ guiding center model in polar coordinates on an annulus. This model is used to investigate the evolution of the diocotron instability in a low density pure electron plasma, also called "slipping-stream" cf. [Dav90, Chapter 6]. The equation satisfied by the electron density $\rho \equiv \rho(t, r, \theta)$ is

$$\partial_t \rho - \frac{1}{r}\partial_\theta \Phi\, \partial_r \rho + \frac{1}{r}\partial_r \Phi\, \partial_\theta \rho = 0, \tag{3.20}$$

where the potential $\Phi \equiv \Phi(t, r, \theta)$ solves the Poisson equation

$$-\partial_{rr}^2 \Phi - \frac{1}{r}\partial_r \Phi - \frac{1}{r^2}\partial_{\theta\theta}^2 \Phi = \rho, \tag{3.21}$$

and where $t \in [0, T]$, $(r, \theta) \in \Omega = [r_{\min}, r_{\max}] \times [0, 2\pi]$. We assume periodic boundary conditions in θ, whereas different boundary conditions are discussed in the radial direction. In [19] we show that specific boundary conditions lead to conservation of the mass and of the electric energy. Then, in the framework of a standard stability analysis of an equilibrium, the dispersion relation is recalled from [Dav90] and discussed in tandem with the considered boundary conditions. Finally, the semi-Lagrangian code is validated in the linear phase against analytical growth rates given by the dispersion relation. Hereafter, we detail these lines.

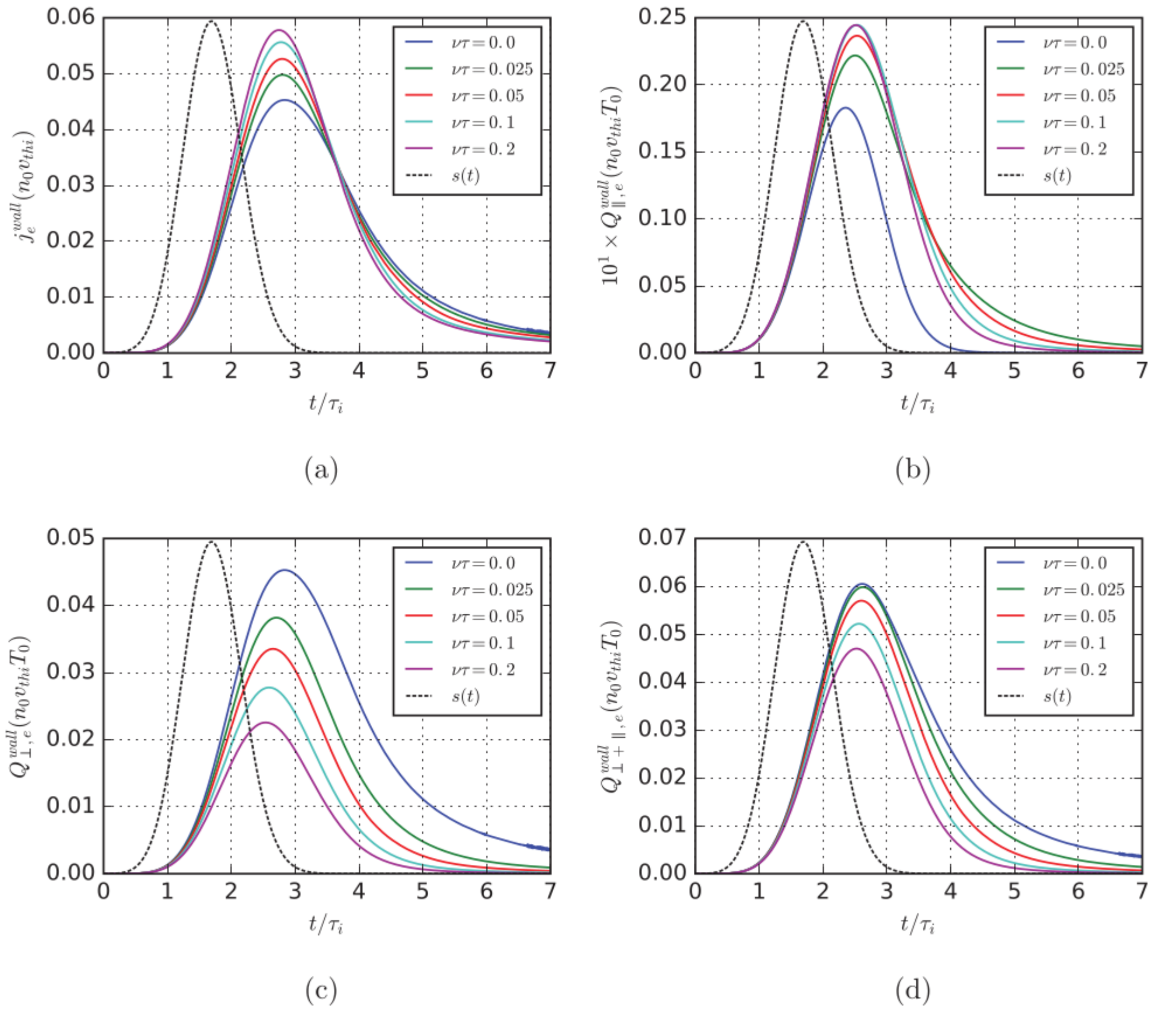


Figure 3.2: Electron fluxes at the target plate for the case with time-distributed source: (a) particle flux; (b) parallel energy flux; (c) perpendicular energy flux; (d) total energy flux. The dotted lines represent the source temporal profile s.

Following [Dav90; Pét09], the initial condition for the guiding center model is

$$\rho(t=0,r,\theta) = \begin{cases} 0, & r_{\min} \le r < r^-, \\ 1 + \varepsilon\cos(\ell\theta), & r^- \le r \le r^+, \\ 0, & r^+ < r \le r_{\max}, \end{cases} \tag{3.22}$$

where ε is a small parameter and ℓ is the mode number of the perturbation.

Radial boundary conditions for Φ. Different radial boundary conditions are proposed in [19] for the Poisson equation (3.21) and their influence on the conservation of the electric energy

$$\mathcal{E}(t) = \int_\Omega r|\partial_r\Phi|^2 + \frac{1}{r}|\partial_\theta\Phi|^2\, dr d\theta$$

and of the mass

$$\mathcal{M}(t) = \int_\Omega r\rho\, dr d\theta$$

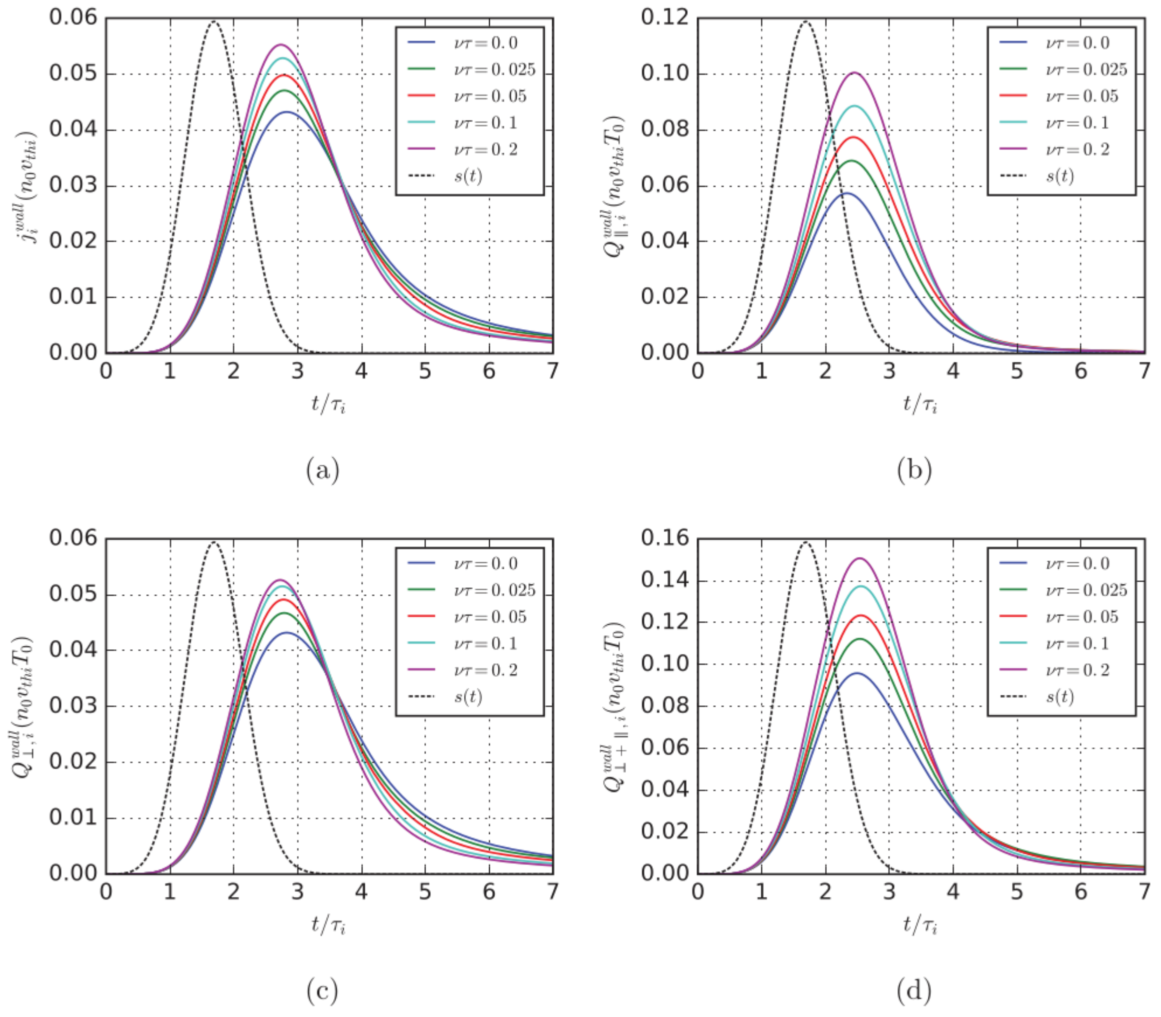


Figure 3.3: Ion fluxes at the target plate for the case with time-distributed source: (a) particle flux; (b) parallel energy flux; (c) perpendicular energy flux; (d) total energy flux. The dotted lines represent the source temporal profile s.

is discussed. First, Dirichlet boundary conditions are imposed at $r_{\min}$ and at $r_{\max}$:

$$\Phi(t, r_{\max}, \theta) = \Phi(t, r_{\max}, \theta) = 0, \qquad \forall t \geq 0, \, \forall \theta \in [0, 2\pi]. \tag{3.23}$$

Alternatively, the second following condition, named "Neumann mode 0" can be considered:

- Dirichlet boundary condition at $r_{\max}$,
- homogeneous Neumann boundary condition at $r_{\min}$ for the Fourier mode 0 in θ:

$$\int_0^{2\pi} \partial_r \Phi(t, r_{\min}, \theta) d\theta = 0,$$

- Dirichlet boundary condition at $r_{\min}$ for the other Fourier modes in θ:

$$\Phi(t, r_{\min}, \theta) = \frac{1}{2\pi} \int_0^{2\pi} \Phi(t, r_{\min}, \theta') d\theta'.$$

For these two choices of radial boundary conditions, we prove that the electric energy and the mass are preserved in time

$$\partial_t \mathcal{E} = \partial_t \mathcal{M} = 0.$$

This is an interesting property to retrieve numerically in order to validate the numerical methods. A last configuration has also been explored, in which Dirichlet boundary condition at $r_{\max}$ and homogeneous Neumann boundary condition at $r_{\min}$ are imposed. In this case, we do not know if the electric energy and the mass are preserved in time.

Instability rates. The equation allowing to obtain the growth rates of instability has been derived in [Dav90, Chapter 6]. We briefly recall this derivation here. To investigate the linear stability of the system (3.20)-(3.21), we assume small amplitude perturbations about a radial equilibrium $\big(n_0(r), \Phi_0(r)\big)$, solution to (3.20)-(3.21). We then search an approximate solution in the form

$$\begin{aligned}\rho(t,r,\theta) &\simeq n_0(r) + \varepsilon\, \widehat{n}_{1,\ell}(r)\exp(\mathrm{i}\ell\theta - \mathrm{i}\omega t),\\ \Phi(t,r,\theta) &\simeq \Phi_0(r) + \varepsilon\, \widehat{\Phi}_{1,\ell}(r)\exp(\mathrm{i}\ell\theta - \mathrm{i}\omega t),\end{aligned}$$

where ω is the complex oscillation frequency, with $\mathrm{Im}(\omega) > 0$ corresponding to instability. Injecting this particular solution in (3.20)-(3.21) leads to the following system

$$\begin{cases} -\mathrm{i}\omega\widehat{n}_{1,\ell} - \mathrm{i}\ell\dfrac{\partial_r n_0}{r}\widehat{\Phi}_{1,\ell} + \mathrm{i}\ell\dfrac{\partial_r \Phi_0}{r}\widehat{n}_{1,\ell} \;=\; 0, \\[2ex] -\partial_{rr}^2\widehat{\Phi}_{1,\ell} - \dfrac{\partial_r\widehat{\Phi}_{1,\ell}}{r} + \dfrac{\ell^2}{r^2}\widehat{\Phi}_{1,\ell} = \widehat{n}_{1,\ell}. \end{cases}$$

Next, we express $\widehat{n}_{1,\ell}$ in terms of $\widehat{\Phi}_{1,\ell}$ to get an equation on $\widehat{\Phi}_{1,\ell}$:

$$\left(-\mathrm{i}\omega + \mathrm{i}\ell\frac{\partial_r\Phi_0}{r}\right)\left(-\partial_{rr}^2\widehat{\Phi}_{1,\ell} - \frac{\partial_r\widehat{\Phi}_{1,\ell}}{r} + \frac{\ell^2}{r^2}\widehat{\Phi}_{1,\ell}\right) = \mathrm{i}\ell\frac{\partial_r n_0}{r}\widehat{\Phi}_{1,\ell}. \tag{3.24}$$

This problem can be approximated numerically by making a finite difference discretization in the radial variable. Thus, equation (3.24) is approximated by a linear system of type $M\phi = c\phi$, where $c = \omega/\ell$ and M is a matrix of size the number of grid points in the radial direction. The problem is then reduced to find the eigenvalues c of the problem $M\phi = c\phi$. Afterwards, for each ℓ, we look for the eigenvalue c which has the greatest strictly positive imaginary part.
In the specific case of the initial condition in (3.22), analytical computations can be performed and thus, an explicit solution to equation (3.24) is found in [19, Section 3] for the three previous sets of boundary conditions. The analytical growth rates $\mathrm{Im}(\omega)$ of the **linear** phase, obtained for $\varepsilon = 10^{-6}$, are then compared to the results of the numerical simulation based on (3.20)-(3.21).

Unlike [Pét09], where a PIC method was used, we implemented a standard backward semi-Lagrangian (BSL) method for solving (3.20) and finite differences in r plus a Fourier method in θ for solving (3.21). For different perturbation modes ℓ (as in [Pét09]) and for the boundary conditions in (3.23), we report in Table 3.1 the values of the growth rates obtained with the simulation, in comparison with the values given by the analytical dispersion relation. Similar results for the other sets of boundary conditions are reported in [19].

In the **nonlinear** case, we consider the initial condition (3.22) with $\varepsilon = 0.5$ and we study the influence of the radial boundary conditions. We obtain numerically the convergence of the electric

ℓ	r^-	r^+	$\text{Im}(\omega)$	analytical $\text{Im}(\omega)$
2	4	5	$0.2875,\ t \in [26, 55]$	0.288739227554270
3	4	5	$0.3667,\ t \in [26, 72]$	0.367315895142460
4	4	5	$0.3852,\ t \in [26, 51]$	0.384081542249742
7	6	7	$0.3424,\ t \in [26, 61]$	0.337573424025866

Table 3.1: Growth rate obtained by the semi-Lagrangian method together with the time interval of validity and the growth rate given by the linearized analysis with Dirichlet boundary conditions.

energy and the mass conservation for Dirichlet and for “Neumann mode 0” boundary conditions. On the contrary, for the third condition, namely homogeneous Neumann at $r_{\min}$, these quantities are no more conserved in long time.

A new $2d$ conservative semi-Lagrangian (CSL) method was developed in the PhD thesis of Pierre Glanc (Université de Strasbourg)[4], see also [13, Section 2]. This method improves the lack of mass conservation of the standard backward semi-Lagrangian. Indeed, the numerical experiments shown that the new CSL method preserves the total mass better than the standard BSL method (see Figs. 3, 5 in [13]). In addition the same conclusion was drawn for the conservation of the electric energy (see Figs. 2, 4 in [13]). These results were obtained for the “Neumann mode 0” boundary condition. The third boundary condition is more challenging for the numerical validation – the CSL method does not preserve the energy and the mass – and deserves a deeper analysis (for example, comparison with Particle-in-Cell simulation can be envisaged).
We end this part by highlighting also that the CSL method was tested and validated in [13] in the context of the more challenging $4d$ drift-kinetic model [Gra+06] but this is not reported here.

3.3 Conclusions and outlook

Conclusions. In this chapter two meaningful physical problems have been analyzed by means of modelling with partial differential equations and their numerical simulation: the dynamics of ELMs and the diocotron instability.

In the first section, we constructed a $1d \times 1v$ hybrid model (kinetic in the parallel direction and fluid in the perpendicular plane) that treats the parallel transport of ions and electrons in the scrape-off layer, including the effect of temperature anisotropy. The model constitutes a useful tool to study energy deposition on the divertor plates following an ELM event.

In the second section, a guiding center model in polar coordinates was used to investigate the evolution of the diocotron instability. Several boundary conditions in the radial direction are discussed and some of them are shown to obey conservation properties, that we retrieved numerically with standard schemes. A linear stability analysis was also performed and instability rates were obtained allowing to validate the numerical method in the linear phase.

Perspectives. Concerning the subject of ELMs, further improvements on the present model can be envisaged. First, a more sophisticated collision operator (like Fokker-Planck *etc.*) could be used in place of the BGK term. Electron-ion collisions, which were neglected in our work, should also be included. Second, perpendicular drifts should be taken into account. This would lead to a complete

[4]advisors: N. Crouseilles (Inria Rennes), E. Frénod (Univ. de Bretagne-Sud, Vannes), and M. Mehrenberger (Univ. de Strasbourg)

set of fluid equations (continuity, momentum, and energy) for the perpendicular dynamics. Finally, for going further in the realism of the physical phenomenon, neutral particle dynamics near the divertor could also be included in an extended model.

In the diocotron instability part, we performed first tests on the basis of the guiding center model, where the external magnetic field in the orthogonal plane was assumed uniform. Thus, a decreasing with the radius magnetic field (as in [Pét09]) can be interesting to consider. Also, the test case with the particle injection in [Pét09] seems to be challenging for the semi-Lagrangian method in terms of the boundary conditions for the density. In addition, cross-comparison of the semi-Lagrangian code with the PIC code can be fruitful for understanding the nonlinear phase, where theoretical results are not available.

The semi-Lagrangian methods have become a good choice for performing accurate simulations of Vlasov equation. Even though they need a grid for the phase space which is computationally intensive in high dimensions, a high level of parallelism with the required state-of-the-art high performance computing can tackle now this difficulty (as for example the `Gysela` code in a $5d$ phase space [Gra+16]). Thus, it would be interesting to go further in the understanding of the behaviour of the backward semi-Lagrangian method in the case of the third set of boundary conditions (Neumann at $r_{\min}$). Also, a deeper analysis is needed in order to clarify if the conservation properties are verified for these boundary conditions.

General outlook

I already gave some details at the end of each chapter about the research that I intend to pursue. I conclude now with its general lines from a different perspective.

From the **applied mathematics and scientific computing point of view**, I will continue through the following lines.

First, developing new mathematical models in order to tackle the complex behaviour of the solutions for Vlasov-type or similar equations is an important point. Going further with the homogenization techniques, the three-scale convergence for example ([AB96]), seems to me an interesting way to reveal different aspects of the multiscale behaviour in the solutions. Another point which can be developed is to free the two-scale approach from the periodic setting, going to the more general ergodic setting (as in [Dal08]). The latter setting would be more adapted to the applications, since the rapid motion of the particles is clearly not periodic in time.

Second, for the numerical approach, I will continue with the development of the PIC method. This is a powerful method for several reasons. A first argument is its simplicity of implementation in any dimension. In addition, the parallelism, at least at the particles level, is tractable when going to higher dimension. Secondly, it is straightforward to simulate complex motions with a PIC scheme, which is due to the mesh-free aspect during the particle push step. I think an opportunity that should be seized is to go further, in using physical grid-free methods (see [Chr+06]). This approach can be very useful to the implementation of a first order two-scale model, as already discussed.
However, I envisage working with semi-Lagrangian schemes also (recall Sections 1.4 and 3.2). As a mix between PIC and Eulerian methods, a semi-Lagrangian method has the accuracy of an Eulerian scheme but is free of CFL condition. However, what seems more difficult with respect to a PIC scheme, going to higher dimension in a semi-Lagrangian code is computationally intensive and the parallelism needs to be tackled carefully.

From the **applications point of view**, a general target of my future work is to go to realistic kinetic two-species simulations arising in some plasma physics problems. By realistic I understand the following issues:
(i) the needed dimensionality for the phase space (typically $3d \times 3v$),
(ii) the different scales between the ions and the electrons, and
(iii) the geometry adapted to the physical problem in view.
In order to achieve these objectives, using appropriate mathematical methods with high performance computing is mandatory.

Thus, concerning the first point above, use of several levels of parallelism is necessary to handle efficiently large meshes and/or large numbers of numerical particles. Investigation of new choices of parallelism and of new algorithms is also an interesting point to be developed. I dealt with some of these ideas in Chapter 2, but without taking into account the two other points. One future direction of research is thus to adapt the work in Chapter 2 to frameworks considering points *(ii)* and *(iii)*.

Regarding the second point, the problem of the multiple scales, I worked in two directions. First, in Chapter 3, I performed simulations involving ions and much faster electrons by solving numerically the smallest scale, but this was done in a reduced setting $(1d\times 1v)$. Second, in Chapter 1, I developed adapted time schemes in order to avoid the numerical resolution of all the scales. In the future, I intend to apply such schemes to perform two-species simulations; for example to use a standard time-stepping for the ions and an exponential integrator in time for the electrons, able to use the same ion time step. Validating the outcomes of such a scheme against a reference solution, computed on the base of the frameworks developed in Chapter 2, is clearly an important fact to be achieved.

As for the last point, I need to deal with this problem entirely, all the configurations considered in this manuscript being simple physical domains.

References of the author

[1] Y. Barsamian, A. Charguéraud, S. A. Hirstoaga, and M. Mehrenberger. "Efficient Strict-Binning Particle-in-Cell Algorithm for Multi-Core SIMD Processors". In: *24th International Conference on Parallel and Distributed Computing (Euro-Par)*. Vol. 11014. Lecture Notes in Computer Science. Springer, Cham, 2018, pp. 749–763.

[2] Y. Barsamian, J. Bernier, S. A. Hirstoaga, and M. Mehrenberger. "Verification of 2Dx2D and two-species Vlasov-Poisson solvers". In: *ESAIM: Proceedings* 63 (2018), pp. 78–108.

[3] Y. Barsamian, S. A. Hirstoaga, and E. Violard. "Efficient Data Structures for a Hybrid Parallel and Vectorized Particle-in-Cell Code". In: *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*. IEEE. 2017, pp. 1168–1177.

[4] Y. Barsamian, S. A. Hirstoaga, and E. Violard. "Efficient data layouts for a three-dimensional electrostatic Particle-in-Cell code". In: *Journal of Computational Science* 27 (2018), pp. 345–356.

[5] E. Chacon-Golcher, S. A. Hirstoaga, and M. Lutz. "Optimization of Particle-In-Cell simulations for Vlasov-Poisson system with strong magnetic field". In: *ESAIM: Proceedings* 53 (2016), pp. 177–190.

[6] P. L. Combettes and S. A. Hirstoaga. "Equilibrium programming in Hilbert spaces". In: *Journal of Nonlinear and Convex Analysis* 6.1 (2005), pp. 117–136.

[7] P. L. Combettes and S. A. Hirstoaga. "Approximating curves for nonexpansive and monotone operators". In: *Journal of Convex Analysis* 13.3/4 (2006), p. 633.

[8] P. L. Combettes and S. A. Hirstoaga. "Visco-penalization of the sum of two monotone operators". In: *Nonlinear Analysis: Theory, Methods & Applications* 69.2 (2008), pp. 579–591.

[9] D. Coulette, S. Hirstoaga, and G. Manfredi. "Effect of collisional temperature isotropisation on ELM parallel transport in a tokamak scrape-off layer". In: *Plasma Physics and Controlled Fusion* 58.8 (2016), p. 085004.

[10] N. Crouseilles, S. A. Hirstoaga, and X. Zhao. "Multiscale Particle-in-Cell methods and comparisons for the long-time two-dimensional Vlasov-Poisson equation with strong magnetic field". In: *Computer Physics Communications* 222 (2018), pp. 136–151.

[11] D. Coulette, G. Manfredi, and S. Hirstoaga. "Kinetic modeling and numerical simulation of plasma-wall interactions in magnetic fusion devices". In: *Proceedings of the 43rd European Physical Society Conference on Plasma Physics*. 2016, p. 05.127.

[12] N. Crouseilles, E. Frénod, S. A. Hirstoaga, and A. Mouton. "Two-Scale Macro-Micro decomposition of the Vlasov equation with a strong magnetic field". In: *Mathematical Models and Methods in Applied Sciences* 23.8 (2013), pp. 1527–1559.

[13] N. Crouseilles, P. Glanc, S. A. Hirstoaga, E. Madaule, M. Mehrenberger, and J. Pétri. "A new fully two-dimensional conservative semi-Lagrangian method: applications on polar grids, from diocotron instability to ITG turbulence". In: *The European Physical Journal D* 68.9 (2014), p. 252.

[14] E. Frénod, M. Gutnic, and S. A. Hirstoaga. "First order Two-Scale Particle-In-Cell numerical method for the Vlasov equation". In: *ESAIM: Proceedings* 38 (2012), pp. 348–360.

[15] E. Frénod, S. A. Hirstoaga, and M. Lutz. "Long-time simulation of a highly oscillatory Vlasov equation with an exponential integrator". In: *Comptes Rendus Mécanique* 342.10-11 (2014), pp. 595–609.

[16] E. Frénod, S. A. Hirstoaga, and E. Sonnendrücker. "An exponential integrator for a highly oscillatory Vlasov equation". In: *Discrete & Continuous Dynamical Systems - S* 8.1 (2015), pp. 169–183.

[17] E. Frénod, S. A. Hirstoaga, M. Lutz, and E. Sonnendrücker. "Long-time behaviour of an exponential integrator for a Vlasov-Poisson system with strong magnetic field". In: *Communications in Computational Physics* 18.2 (2015), pp. 263–296.

[18] S. A. Hirstoaga. "Iterative selection methods for common fixed point problems". In: *Journal of Mathematical Analysis and Applications* 324.2 (2006), pp. 1020–1035.

[19] E. Madaule, S. A. Hirstoaga, M. Mehrenberger, and J. Pétri. "Semi-Lagrangian simulations of the diocotron instability". Report: https://hal.inria.fr/hal-00841504/. 2013.

[20] G. Manfredi, S. Hirstoaga, S. Devaux, E. Havlickova, and D. Tskhakaya. "Parallel transport in a tokamak scrape-off layer". In: *Proceedings of the 38th European Physical Society Conference on Plasma Physics*. 2011, P4.063.

[21] G. Manfredi, S. Hirstoaga, and S. Devaux. "Vlasov modelling of parallel transport in a tokamak scrape-off layer". In: *Plasma Physics and Controlled Fusion* 53.1 (2011), p. 015012.

[22] D. Moulton, W. Fundamenski, G. Manfredi, S. Hirstoaga, and D. Tskhakaya. "Comparison of free-streaming ELM formulae to a Vlasov simulation". In: *Journal of Nuclear Materials* 438 (2013), S633–S637.

Other references

[AB96] G. Allaire and M. Briane. "Multiscale convergence and reiterated homogenisation". In: *Proc. Roy. Soc. Edinburgh: Sect. A Mathematics* 126.2 (1996), pp. 297–342.

[All92] G. Allaire. "Homogenization and Two-scale Convergence". In: *SIAM J. Math. Anal.* 23.6 (1992), pp. 1482–1518.

[Ari+09] G. Ariel, B. Engquist, H.-O. Kreiss, and R. Tsai. "Multiscale Computations for Highly Oscillatory Problems". In: *Multiscale Modeling and Simulation in Science, LNCSE, vol. 66.* Ed. by B. Engquist, P. Lötstedt, and O. Runborg. Springer, Berlin, Heidelberg, 2009, pp. 237–287.

[Ari+13] G. Ariel, B. Engquist, S. Kim, Y. Lee, and R. Tsai. "A multiscale method for highly oscillatory dynamical systems using a Poincaré map type technique". In: *Journal of Scientific Computing* 54.2-3 (2013), pp. 247–268.

[Bar05] R. Barthelmé. "Le problème de conservation de la charge dans le couplage des équations de Vlasov et de Maxwell". Thesis. Université de Strasbourg, 2005. URL: http://scd-theses.u-strasbg.fr/998/01/barthelme.pdf.

[Bar18] Y. Barsamian. "Pic-Vert: A Particle-in-Cell Implementation for Multi-Core Architectures". Thesis. Université de Strasbourg, 2018. URL: http://www.barsamian.am/Pic-Vert/.

[BFS87] P. Bratley, B. L. Fox, and L. E. Schrage. *A Guide to simulation.* Springer, 1987.

[BL85] C. K. Birdsall and A. B. Langdon. *Plasma Physics via Computer Simulation.* McGraw-Hill, New York, 1985.

[BM82] J. T. Beale and A. Majda. "Vortex Methods. II. Higher Order Accuracy in Two and Three Dimensions". In: *Mathematics of Computation* 39.159 (1982), pp. 29–52.

[Bos07] M. Bostan. "The Vlasov-Maxwell System with Strong Initial Magnetic Field: Guiding-Center Approximation". In: *Multiscale Model. Simul.* 6.3 (2007), pp. 1026–1058.

[Bos09] M. Bostan. "The Vlasov-Poisson system with strong external magnetic field. Finite Larmor radius regime". In: *Asymptotic Analysis* 61.2 (2009), pp. 91–123.

[Bow+08] K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, and T. J. T. Kwan. "Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation". In: *Physics of Plasmas* 15.5 (2008), p. 055703.

[Bow01] K. J. Bowers. "Accelerating a Particle-in-Cell Simulation Using a Hybrid Counting Sort". In: *Journal of Computational Physics* 173.2 (2001), pp. 393–411.

[Bow03] K. J. Bowers. "Speed optimal implementation of a fully relativistic particle push with charge conserving current accumulation on modern processors". In: *Proceedings of the 18th Int. Conf. Numerical Simulation of Plasmas (ICNSP).* 2003, pp. 383–386. URL: http://web.mit.edu/ned/ICNSP/ICNSP_BookofAbstracts.pdf.

[BS15] A. Bottino and E. Sonnendrücker. "Monte Carlo particle-in-cell methods for the simulation of the Vlasov-Maxwell gyrokinetic equations". In: *J. Plasma Phys.* 81.5 (2015), p. 435810501.

[Cal+11] M. P. Calvo, P. Chartier, A. Murua, and J. M. Sanz-Serna. "A stroboscopic numerical method for highly oscillatory problems". In: *Lect. Notes Comput. Sci. Eng.* 82 (2011), pp. 73–87.

[CDV07] P. Crispel, P. Degond, and M.-H. Vignal. "An asymptotic preserving scheme for the two-fluid Euler–Poisson model in the quasineutral limit". In: *Journal of Computational Physics* 223.1 (2007), pp. 208–234.

[Cha+99] S. Chatterjee, V. V. Jain, A. R. Lebeck, S. Mundhra, and M. Thottethodi. "Nonlinear Array Layouts for Hierarchical Memory Systems". In: *Proceedings of the 13th International Conference on Supercomputing.* ACM, 1999, pp. 444–453.

[Chr+06] A. J. Christlieb, R. Krasny, J. P. Verboncoeur, J. W. Emhoff, and I. D. Boyd. "Grid-free plasma simulation techniques". In: *IEEE Transactions on Plasma Science* 34.2 (2006), pp. 149–165.

[CK00] G.-H. Cottet and P. D. Koumoutsakos. *Vortex Methods: Theory and Practice.* Cambridge University Press, 2000.

[CL11] N. Crouseilles and M. Lemou. "An asymptotic preserving scheme based on a micro-macro decomposition for collisional Vlasov equations: diffusion and high-field scaling limits". In: *Kinet. Relat. Models* 4 (2011), pp. 441–477.

[CLM13] N. Crouseilles, M. Lemou, and F. Méhats. "Asymptotic preserving schemes for highly oscillatory Vlasov-Poisson equations". In: *Journal of Computational Physics* 248 (2013), pp. 287–308.

[CM02] S. M. Cox and P. C. Matthews. "Exponential time differencing for stiff systems". In: *Journal of Computational Physics* 176.2 (2002), pp. 430–455.

[CP17] M. Campos Pinto. "Analysis and design of numerical methods for problems arising in plasma physics". Habilitation Thesis. Université Pierre et Marie Curie, 2017.

[Cro+17] N. Crouseilles, M. Lemou, F. Méhats, and X. Zhao. "Uniformly accurate Particle-in-Cell method for the long time solution of the two-dimensional Vlasov-Poisson equation with uniform strong magnetic field". In: *Journal of Computational Physics* 346 (2017), pp. 172–190.

[Dal08] A.-L. Dalibard. "Homogenization of linear transport equations in a stationary ergodic setting". In: *Communications in Partial Differential Equations* 33.5 (2008), pp. 881–921.

[Dav90] R. C. Davidson. *Physics of Nonneutral Plasmas.* Addison-Wesley (Reading, Massachusetts), 1990.

[Dec+96] V. K. Decyk, S. R. Karmesin, A. de Boer, and P. C. Liewer. "Optimization of particle-in-cell codes on reduced instruction set computer processors". In: *Computers in Physics* 10.3 (1996), pp. 290–298.

[Deg+10] P. Degond, F. Deluzet, L. Navoret, A.-B. Sun, and M.-H. Vignal. "Asymptotic-preserving particle-in-cell method for the Vlasov–Poisson system near quasineutrality". In: *Journal of Computational Physics* 229.16 (2010), pp. 5630–5652.

[Don+13] C. Dong, H. Ren, H. Cai, and D. Li. "Temperature relaxation in a magnetized plasma". In: *Physics of Plasmas* 20.10 (2013), p. 102518.

[DS14] V. K. Decyk and T. V. Singh. “Particle-in-Cell algorithms for emerging computer architectures”. In: *Computer Physics Communications* 185.3 (2014), pp. 708–719.

[DT97] M. Drmota and R. F. Tichy. *Sequences, discrepancies and applications.* Springer, 1997.

[E+07] W. E, B. Engquist, X. Li, W. Ren, and E. Vanden-Eijnden. “Heterogeneous multiscale methods: a review”. In: *Commun. Comput. Phys.* 2 (2007), pp. 367–450.

[FP+06] W. Fundamenski, R. Pitts, et al. “A model of ELM filament energy evolution due to parallel losses”. In: *Plasma Physics and Controlled Fusion* 48.1 (2006), p. 109.

[FR16] F. Filbet and L. M. Rodrigues. “Asymptotically stable particle-in-cell methods for the Vlasov–Poisson system with a strong external magnetic field”. In: *SIAM Journal on Numerical Analysis* 54.2 (2016), pp. 1120–1146.

[FRS01] E. Frénod, P.-A. Raviart, and E. Sonnendrücker. “Two-scale expansion of a singularly perturbed convection equation”. In: *J. Math. Pures Appl.* 80.8 (2001), pp. 815–843.

[Fré17] E. Frénod. *Two-Scale Approach to Oscillatory Singularly Perturbed Transport Equations.* Springer, 2017.

[FS00] E. Frénod and E. Sonnendrücker. “Long time behavior of the two dimensionnal Vlasov equation with a strong external magnetic field”. In: *Math. Models Methods Appl. Sci.* 10.4 (2000), pp. 539–553.

[FS01] E. Frénod and E. Sonnendrücker. “The finite Larmor radius approximation”. In: *SIAM J. Math. Anal.* 32.6 (2001), pp. 1227–1247.

[FS06] F. Filbet and E. Sonnendrücker. “Modeling and numerical simulation of space charge dominated beams in the paraxial approximation”. In: *Math. Models Methods Appl. Sci.* 16.5 (2006), pp. 763–791.

[FS98] E. Frénod and E. Sonnendrücker. “Homogenization of the Vlasov equation and of the Vlasov-Poisson system with a strong external magnetic field”. In: *Asymptotic Analysis* 18.3-4 (1998), pp. 193–214.

[FSB01] F. Filbet, E. Sonnendrücker, and P. Bertrand. “Conservative numerical schemes for the Vlasov equation”. In: *Journal of Computational Physics* 172.1 (2001), pp. 166–187.

[FSS09] E. Frénod, F. Salvarani, and E. Sonnendrücker. “Long time simulation of a beam in a periodic focusing channel via a two-scale PIC-method”. In: *Math. Models Methods Appl. Sci.* 19.2 (2009), pp. 175–197.

[Ger+16] K. Germaschewski, W. Fox, S. Abbott, N. Ahmadi, K. Maynard, L. Wang, H. Ruhl, and A. Bhattacharjee. “The Plasma Simulation Code: A modern particle-in-cell code with patch-based load-balancing”. In: *Journal of Computational Physics* 318 (2016), pp. 305–326.

[Gra+06] V. Grandgirard, M. Brunetti, P. Bertrand, N. Besse, X. Garbet, P. Ghendrih, G. Manfredi, Y. Sarazin, O. Sauter, E. Sonnendrücker, J. Vaclavik, and L. Villard. “A drift-kinetic Semi-Lagrangian 4D code for ion turbulence simulation”. In: *Journal of Computational Physics* 217.2 (2006), pp. 395–423.

[Gra+16] V. Grandgirard, J. Abiteboul, J. Bigot, T. Cartier-Michaud, N. Crouseilles, G. Dif-Pradalier, C. Ehrlacher, D. Esteve, X. Garbet, P. Ghendrih, et al. “A 5D gyrokinetic full-f global semi-lagrangian code for flux-driven ion turbulence simulations”. In: *Computer Physics Communications* 207 (2016), pp. 35–68.

[GSR03] F. Golse and L. Saint-Raymond. “The Vlasov-Poisson system with strong magnetic field in quasineutral regime”. In: *Math. Models Methods Appl. Sci.* 13 (2003), pp. 661–714.

[GSR99] F. Golse and L. Saint-Raymond. “The Vlasov-Poisson system with strong magnetic field”. In: *J. Math. Pures Appl.* 78 (1999), pp. 791–817.

[Har55] F. Harlow. *A Machine Calculation Method for Hydrodynamic Problems.* Tech. rep. LAMS-1956. Los Alamos Scientific Laboratory, 1955.

[Hav+11] E. Havlíčková, W. Fundamenski, V. Naulin, A. H. Nielsen, R. Zagorski, J. Seidl, and J. Horáček. “Steady-state and time-dependent modelling of parallel transport in the scrape-off layer”. In: *Plasma Physics and Controlled Fusion* 53.6 (2011), p. 065004.

[Hav+12] E. Havlíčková, W. Fundamenski, D. Tskhakaya, G. Manfredi, and D. Moulton. “Comparison of fluid and kinetic models of target energy fluxes during edge localized modes”. In: *Plasma Physics and Controlled Fusion* 54 (2012), p. 045002.

[HE88] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles.* Institute of Physics, Philadelphia, 1988.

[HK10] D. Han-Kwan. “The three-dimensional finite Larmor radius approximation”. In: *Asymptotic Analysis* 66.1 (2010), pp. 9–33.

[HKK07] T. Harada, S. Koshizuka, and Y. Kawaguchi. “Smoothed particle hydrodynamics on GPUs”. In: *Computer Graphics International.* Vol. 40. 2007, pp. 63–70.

[HLW06] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations.* 2nd ed. Springer Series in Computational Mathematics, 2006.

[HM03] R. D. Hazeltine and J. D. Meiss. *Plasma confinement.* Courier Corporation, 2003.

[HO10] M. Hochbrück and A. Ostermann. “Exponential integrators”. In: *Acta Numer.* 19 (2010), pp. 209–286.

[Hut17] I. H. Hutchinson. “Electron holes in phase space: What they are and why they matter”. In: *Physics of Plasmas* 24 (2017), p. 055601.

[Jin10] S. Jin. “Asymptotic preserving (AP) schemes for multiscale kinetic and hyperbolic equations: a review”. In: *Lecture notes for summer school on methods and models of kinetic theory (M&MKT), Porto Ercole (Grosseto, Italy)* (2010), pp. 177–216.

[Jin99] S. Jin. “Efficient asymptotic-preserving (AP) schemes for some multiscale kinetic equations”. In: *SIAM J. Sci. Comp.* 21 (1999), pp. 441–454.

[Joc+16] A. Jocksch, F. Hariri, T.-M. Tran, S. Brunner, C. Gheller, and L. Villard. “A Bucket Sort Algorithm for the Particle-In-Cell Method on Manycore Architectures”. In: *Parallel Processing and Applied Mathematics: 11th Intl. Conf. (PPAM).* Springer Intl. Publishing, 2016, pp. 43–52.

[JS10] S. Jin and Y. Shi. “A Micro-Macro Decomposition-Based Asymptotic-Preserving Scheme for the Multispecies Boltzmann Equation”. In: *SIAM J. Sci. Comp.* 31 (2010), pp. 4580–4606.

[Kla99] A. Klar. “A Numerical Method for Kinetic Semiconductor Equations in the Drift Diffusion Limit”. In: *SIAM J. Sci. Comp.* 20 (1999), pp. 1696–1712.

[Lev02] R. J. Leveque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge University Press, 2002.

[LL02] S. Li and W. K. Liu. “Meshfree and particle methods and their applications”. In: *Applied Mechanics Reviews* 55.1 (2002), pp. 1–34.

[LL10] M. Liu and G. Liu. “Smoothed particle hydrodynamics (SPH): an overview and recent developments”. In: *Archives of computational methods in engineering* 17.1 (2010), pp. 25–76.

[LM08] M. Lemou and L. Mieussens. “A New Asymptotic Preserving Scheme Based on Micro-Macro Formulation for Linear Kinetic Equations in the Diffusion Limit”. In: *SIAM J. Sci. Comp.* 31 (2008), pp. 334–368.

[Mou+13b] D. Moulton, P. Ghendrih, W. Fundamenski, G. Manfredi, and D. Tskhakaya. “Quasineutral plasma expansion into infinite vacuum as a model for parallel ELM transport”. In: *Plasma Physics and Controlled Fusion* 55.8 (2013), p. 085003.

[Mou09] A. Mouton. “Two-scale semi-lagrangian simulation of a charged particles beam in a periodic focusing channel”. In: *Kinet. Relat. Models* 2.2 (2009), pp. 251–274.

[Mus+00] L. Muschietti, I. Roth, C. W. Carlson, and R. E. Ergun. “Transverse Instability of Magnetized Electron Holes”. In: *Phys. Rev. Lett.* 85.1 (2000), pp. 94–97.

[MV11] C. Mouhot and C. Villani. “On Landau damping”. In: *Acta Mathematica* 207.1 (2011), pp. 29–201.

[Ngu89] G. Nguetseng. “A General Convergence Result for a Functional Related to the Theory of Homogenization”. In: *SIAM J. Math. Anal.* 20.3 (1989), pp. 608–623.

[Ngu90] G. Nguetseng. “Asymptotic Analysis for a Stiff Variational Problem Arising in Mechanics”. In: *SIAM J. Math. Anal.* 21.6 (1990), pp. 1394–1414.

[Pét09] J. Pétri. “Non-linear evolution of the diocotron instability in a pulsar electrosphere: two-dimensional particle-in-cell simulations”. In: *Astronomy & Astrophysics* 503.1 (2009), pp. 1–12.

[Qin+13] H. Qin, S. Zhang, J. Xiao, J. Liu, Y. Sun, and W. M. Tang. “Why is the Boris algorithm so good?” In: *Physics of Plasmas* 20.8 (2013), p. 084503.

[Rav85] P.-A. Raviart. “An analysis of particle methods”. In: *Numerical methods in fluid dynamics (Como, 1983), Lecture Notes in Mathematics.* Springer, 1985, pp. 243–324.

[Sel] *SeLaLib.* `http://selalib.gforge.inria.fr/`.

[Sho81] M. Shoucri. “A two-level implicit scheme for the numerical solution of the linearized vorticity equation”. In: *Internat. J. Numer. Methods Engrg.* 17 (1981), pp. 1525–1538.

[Son17] E. Sonnendrücker. *Numerical Methods for the Vlasov-Maxwell equations.* Springer, 2017.

[Tsk+08] D. Tskhakaya, S. Kuhn, Y. Tomita, K. Matyash, R. Schneider, and F. Taccogna. “Self-Consistent Simulations of the Plasma-Wall Transition Layer”. In: *Contributions to Plasma Physics* 48.1-3 (2008), pp. 121–125.

[Ver05] J. P. Verboncoeur. “Particle simulation of plasmas: review and advances”. In: *Plasma Physics and Controlled Fusion* 47.5A (2005), p. A231.

[Vin+16] H. Vincenti, M. Lobet, R. Lehe, R. Sasanka, and J.-L. Vay. “An efficient and portable SIMD algorithm for charge/current deposition in Particle-In-Cell codes”. In: *Computer Physics Communications* 210 (2016), pp. 145–154.

[Wol95] M. J. Wolfe. *High Performance Compilers for Parallel Computing.* Addison-Wesley Longman Publishing Co., Inc., 1995.