Optimization of Particle-In-Cell simulations for Vlasov-Poisson system with strong magnetic field

Edwin Chacon-Golcher, Sever Adrian Hirstoaga, Mathieu Lutz

Optimization of Particle-In-Cell simulations for Vlasov-Poisson system with strong magnetic field

Edwin Chacon-Golcher [*] Sever A. Hirstoaga [†] Mathieu Lutz [‡]

Abstract

We study the dynamics of charged particles under the influence of a strong magnetic field by solving numerically the Vlasov-Poisson and the guiding center models. By using appropriate data structures, we implement efficiently a Particle-In-Cell method in order to perform simulations with a large number of particles. We present numerical results for classical one-dimensional Landau damping and two-dimensional Kelvin-Helmholtz test cases. The implementation follows a standard hybrid MPI/OpenMP parallelization. Code performance is assessed by the observed speedup and attained memory bandwidth.

Introduction

This work focuses on the numerical solution of a multiscale, four-dimensional Vlasov-Poisson equation which statistically describes the evolution in time of charged particles. This system models 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. In order to study this multiscale behaviour we consider the particular configuration where the strong magnetic field is aligned with the x_3 direction: $\mathbf{B} = (0, 0, 1/\varepsilon)^T$, where ε is a vanishing parameter and where we assume that the electric field created by the particles evolves only in the plane orthogonal to the magnetic field. It is well-known (see [2, 9]) that when ε goes to zero, this Vlasov model may be approximated by a reduced model: the guiding center equation. However, when solving the Vlasov equation, the small parameter induces fast time oscillations in the solution, thus making classical numerical schemes very expensive.

In this context, the aim of this work is twofold. First, we attempt to efficiently implement classical numerical methods in order to produce simulations that execute in reasonable time. Second, our purpose is to compute the numerical solution of the four-dimensional Vlasov-Poisson system and therefore we need to validate the code. Thus, the first aim deals with a software implementation problem (using appropriate data structures and assessing their performance). Such part of the work deals with code optimization and parallelization. The second aim of this work relates with the need to ensure that the code gives a good approximation of the mathematical model in question. To this effect we use known analytic approximations or the

[*]Institute of Physics of the ASCR, ELI-Beamlines, 18221 Prague, Czech Republic

[†]Inria Nancy-Grand Est, TONUS Project & IRMA (UMR CNRS 7501), Université de Strasbourg, France

[‡]IRMA (UMR CNRS 7501), Université de Strasbourg, France & IMT (UMR CNRS 5219), Université de Toulouse, France

conservation of some quantities to verify correctness. In addition, we notice that, with the further aim to improve the recent time scheme in [8] and to extend it to more general problems than the aforementioned multiscale model, we need to have at our disposal a fast code yielding the reference solution.

Since a Vlasov model has the structure of a transport equation in phase space, most of the numerical methods are based on the study of its characteristics. In this paper we perform the numerical approach by particle methods (see [1, 11]). We are thus led to advance in time a number of macro-particles following these characteristics. Although particle methods are known as noisy (see [1]), methods exist to improve their precision, as the linearly-transformed particle scheme recently studied in [5] for a generic transport problem. An alternative approach are the semi-Lagrangian methods (see [11, 7]) which show to be very accurate when solving Vlasov-like equations. Lastly, another powerful method to address noise in Particle-In-Cell simulations is to increase the number of particles in the simulation, but this is only practical when the implementation is efficient. We pursue this method here. The result is that in a high-dimension setting, a particle method is adequate to study this problem with the amount of computational resources at our disposal.

In this work we follow a standard Particle-In-Cell (PIC) framework (see [1]) for solving a Vlasov-Poisson system. In order that phenomena occurring at small scales can be captured, one needs a very large number of macro-particles and fine enough grids in time and space. This yields a very computationally demanding problem. To make the resulting simulations execute in an acceptable time, we choose standard methods of parallelization (using the well-known multiprocess and multithreading libraries MPI and OpenMP) and less widely employed methods of code design using specialized data structures to optimize memory access patterns, see the approach developed in [3] and [4]. We thus performed simulations able to process, at their best, 265 million particles per second in average, on one node with 16 cores. The implementation was done in the library SeLaLib [1]. For validation of the code we have performed simulations of the well-known Landau damping problem. Additionally, we simulated the Kelvin-Helmholtz instability test case. Finally we compare the numerical solution of the multiscale four-dimensional Vlasov-Poisson system against the one for the guiding center model by illustrating the convergence result that was alluded to at the begining of this Introduction.

The rest of the paper is organized as follows. In Section 1 we write the equations to be solved and we briefly describe the PIC method for the numerical solution. In Section 2 we detail the PIC implementation, discuss about its efficiency, and show the numerical results concerning the Vlasov-Poisson and the guiding center models. We conclude in Section 3 and draw some lines of future work.

[1] http://selalib.gforge.inria.fr

1 The numerical approach for the Vlasov-Poisson model

1.1 The Vlasov-Poisson and the guiding center models

The Vlasov-Poisson system in which we are interested in this paper is

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

where the unknown $f^\varepsilon = f^\varepsilon(\mathbf{x},\mathbf{v},t)$ represents the distribution of electrons in phase space (position and velocity coordinates) and in time in the plane orthogonal to the magnetic field. We denote by $\mathbf{x} = (x_1, x_2)$ the position variable, $\mathbf{v} = (v_1, v_2)$ the velocity, $\mathbf{v}^\perp = (v_2, -v_1)$, and by $\mathbf{E}^\varepsilon$ the electric field induced by the particles. The electric field is obtained by solving a Poisson equation coupled with the particle's equation through the source terms: n_i denotes the constant ion charge density and

$$\rho^\varepsilon(\mathbf{x},t) = -\int_{\mathbb{R}^2} f^\varepsilon(\mathbf{x},\mathbf{v},t)d\mathbf{v} \tag{2}$$

stands for the electron charge density. 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 rescaling 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 (1) is written corresponds to a time rescaling of the classical drift-kinetic model. The scaling procedure leading to this regime is given in [10] and the time rescaling is done in [9] (see also Section 5.2.1 in [8]).

The limit model of (1) is derived in [2, 9], under some hypotheses for f_0. The authors have established that the particle density (2) converges in some ways, when ε goes to zero, towards the unique solution to the guiding center model:

$$\begin{cases} \partial_t f_{GC} + \mathbf{E}^\perp\cdot\nabla_{\mathbf{x}} f_{GC} = 0, \\ \mathbf{E}(\mathbf{x},t) = -\nabla_{\mathbf{x}}\phi(\mathbf{x},t), \quad -\Delta_{\mathbf{x}}\phi(\mathbf{x},t) = n_i - f_{GC}, \\ f_{GC}(\mathbf{x},t=0) = \displaystyle\int_{\mathbb{R}^2} f_0(\mathbf{x},\mathbf{v})\,d\mathbf{v}. \end{cases} \tag{3}$$

As already mentioned, the numerical solution of (1) is based on the characteristics of the Vlasov equation:

$$\begin{cases} \dfrac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = \dfrac{1}{\varepsilon}\mathbf{V}(t), \\ \dfrac{\mathrm{d}\mathbf{V}(t)}{\mathrm{d}t} = -\dfrac{1}{\varepsilon}\mathbf{E}^\varepsilon(\mathbf{X}(t),t) - \dfrac{1}{\varepsilon^2}\mathbf{V}^\perp(t). \end{cases} \tag{4}$$

The main difficulty when solving these ODEs is in the stiff term $1/\varepsilon^2$ which requires that the explicit numerical schemes use a very small time step. This drives the interest to the asymptotic

model in (3) whose solution does not contain high oscillations in time. However, the aim of this work is to produce a reference solution of the Vlasov-Poisson equation (1) for several values of ε. We notice that this asks for long time simulations since the final time is of order 1 while the numerical schemes need to solve the $1/\varepsilon^2$ term.

1.2 The Particle-In-Cell method

We solve the Vlasov-Poisson system (1) on a doubly periodic physical domain by a standard PIC method (see [1]). This consists in discretizing the distribution function at every time step $t_n = n\Delta t$ by a collection of N_p macro-particles with coordinates $(\mathbf{x}_k^n, \mathbf{v}_k^n)$ and then in a regularizing step with a convolution kernel S

$$f_S^\varepsilon(\mathbf{x}, \mathbf{v}, t_n) = \frac{1}{N_p} \sum_{k=1}^{N_p} S(\mathbf{x} - \mathbf{x}_k^n) \, S(\mathbf{v} - \mathbf{v}_k^n) \,. \tag{5}$$

As usual on cartesian meshes, we used B-splines as convolution kernel (see [11]). More precisely, tests were carried out with B-splines of degree 1 and 3, but all the simulations in this paper were mainly performed with a degree 1 B-spline, which in one dimension of the physical space writes

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

After a randomized initialization for positions and velocities, the macro-particles are advanced in time with a leap-frog discretization of the characteristics in (4)

$$\begin{cases} \mathbf{x}_k^{n+1} = \mathbf{x}_k^n + \dfrac{\Delta t}{\varepsilon} \mathbf{v}_k^{n+\frac{1}{2}} \\ \mathbf{v}_k^{n+\frac{1}{2}} = \mathbf{v}_k^{n-\frac{1}{2}} - \dfrac{\Delta t}{\varepsilon} \left(\mathbf{E}^n(\mathbf{x}_k^n) + \dfrac{1}{\varepsilon} \left(\dfrac{\mathbf{v}_k^{n+\frac{1}{2}} + \mathbf{v}_k^{n-\frac{1}{2}}}{2} \right)^\perp \right) . \end{cases} \tag{6}$$

The electric field is computed by solving the Poisson equation on a uniform cartesian grid by a Fourier method. To this aim, at every time step the particle charge density is computed on the grid by

$$\rho_S^\varepsilon(\mathbf{x}, t_n) = \frac{1}{N_p} \sum_{k=1}^{N_p} S(\mathbf{x} - \mathbf{x}_k^n) \,. \tag{7}$$

In order to get conservation of the total momentum, we use the same shape factor S in order to compute the field at the particle positions:

$$\mathbf{E}^n(\mathbf{x}, t) = \Delta x_1 \Delta x_2 \sum_j \mathbf{E}_j^n \, S(\mathbf{x} - \mathbf{x}_j) \,, \tag{8}$$

where $\left(\mathbf{x}_j\right)_j$ stand for the grid nodes and $\left(\mathbf{E}_j^n\right)_j$ denote the grid values of the electric field at time t_n.

2 PIC implementation and simulations

The multiscale behaviour in the model above requires large numbers of particles and small time steps in order to resolve the fine structures in phase space, which makes this an expensive computation. In order to make simulations run in reasonable times, code optimization and parallelization are needed. We test the performance of the PIC implementation only on a standard Landau damping test case, briefly recalled in section 2.1. Speedup due to parallelization and access to memory curves of the code are given in section 2.2. Finally, we show four-dimensional simulations with a comparison between the Vlasov-Poisson equation (1) and the guiding center model (3).

All the simulations in this paper were done at the computing center of the Université de Strasbourg (*Mésocentre*), on Intel(R) Xeon(R) E5-2670 @ 2.54 GHz processors. The typical node used at the *Mésocentre* contains 2 sockets, each with 8 cores, 32 GB of local memory and an L3 cache size of 20 MB.

The PIC code uses standard parallelization libraries for both, the distributed memory paradigm (using multiple software processes with MPI libraries), and the shared memory paradigm (multiple threads with OpenMP). While using these libraries, care was taken to associate both processes and threads to the underlying hardware to reduce the requests to remote memory. For this purpose, software processes were placed in single processor sockets (the hardware unit that groups a set of cores) and software threads in single cores. In this way, we attempted to minimize the negative impact of non-uniform memory access (NUMA) which can happen when a software process uses cores placed in different sockets, for example. Thus we employ parallelization with multiple processes running on individual sockets and within these processes we parallelize with multiple threads. Details of what is parallelized in the code with MPI and with OpenMP and other optimization techniques are discussed below.

2.1 Landau damping simulations

The purpose of this section is to solve numerically the Landau damping test case in order to validate the code (see [7, 11] and the references therein). The relevant equations are

$$\begin{cases} \partial_t f + \mathbf{v}\cdot\nabla_{\mathbf{x}} f - \mathbf{E}(\mathbf{x},t)\cdot\nabla_{\mathbf{v}} f = 0, \\ \mathbf{E}\,(\mathbf{x},t) = -\nabla_{\mathbf{x}}\phi(\mathbf{x},t), \quad -\Delta_{\mathbf{x}}\phi(\mathbf{x},t) = n_i - \displaystyle\int_{\mathbb{R}^2} f(\mathbf{x},\mathbf{v},t)d\mathbf{v}, \\ f\,(\mathbf{x},\mathbf{v},t=0) = f_0\,(\mathbf{x},\mathbf{v}) \end{cases} \tag{9}$$

and the initial condition is given by

$$f_0\,(\mathbf{x},\mathbf{v}) = \frac{1}{2\pi v_{th}^2}\big(1+\alpha\cos\,(kx_1)\,\big)\exp\left(-\frac{v_1^2+v_2^2}{2v_{th}^2}\right), \tag{10}$$

where $v_{th}=1$, $k=0.5$, and

$$\mathbf{\Omega}_{\mathbf{x}} = \left[0;\frac{2\pi}{k}\right]\times[0;1] \tag{11}$$

is the domain in the physical space. We use classical periodic boundary conditions for the physical domain. Therefore we take $n_i = 1$ such that the Poisson equation has solution. Even if

the data stand in four-dimensional phase space, this test case is mainly two-dimensional since the perturbation α occurs only in the x_1 direction of the physical space.

Before showing the implementation and performance issues, we discuss the numerical results. It is known that the total energy of the Vlasov-Poisson model is constant in time:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{\mathbf{x}}}\int_{\mathbb{R}^2} f(\mathbf{x},\mathbf{v},t)\frac{|\mathbf{v}^2|}{2}\,d\mathbf{v}d\mathbf{x} + \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{\mathbf{x}}} |\mathbf{E}\,(\mathbf{x},t)\,|^2\,d\mathbf{x} = 0. \tag{12}$$

The numerical conservation of this quantity is important for the validation of the simulation. In Fig. 1 we plot the time history of the total energy for different values of the perturbation α. These runs used 2 million particles, a grid of 256 x 2 cells and a time step of 0.1. We observe a good behaviour of the time evolution of the total energy which is preserved up to 0.05% when $\alpha = 0.5$ (this is comparable to the results in [7]). Clearly, smaller is the perturbation α smaller is the error in the total energy.

Then, we are interested in the time evolution of the logarithm of the L^2-norm of $\mathbf{E}$. When α is small, by doing a linear approximation to the Vlasov-Poisson equation one may compute an analytic solution approximating the electric field in (9) (see [11]). More precisely, after a short time, we have a very good approximation of E_1 :

$$E_1(x_1,t) \approx 4\,\alpha\,0.3677\,e^{-0.1533\,t}\sin(kx_1)\cos(1.4156\,t - 0.536245).$$

Thus, we can compare the numerical results with the analytic approximation by computing the logarithm of the electric field's L^2-norm. We have done tests with $\alpha = 0.1$ and $\alpha = 0.005$. We tested that increasing the number of particles per cell makes the computed electric field energy match on longer runs the analytic linear approximation. The smaller value of α is challenging for the PIC simulation, asking for a large number of particles in order to obtain a noiseless electric field energy in long run. In Fig. 2 (at right) we illustrate the log of the electric field energy for a simulation of more than 200 million particles over a grid with 512 x 2 cells.

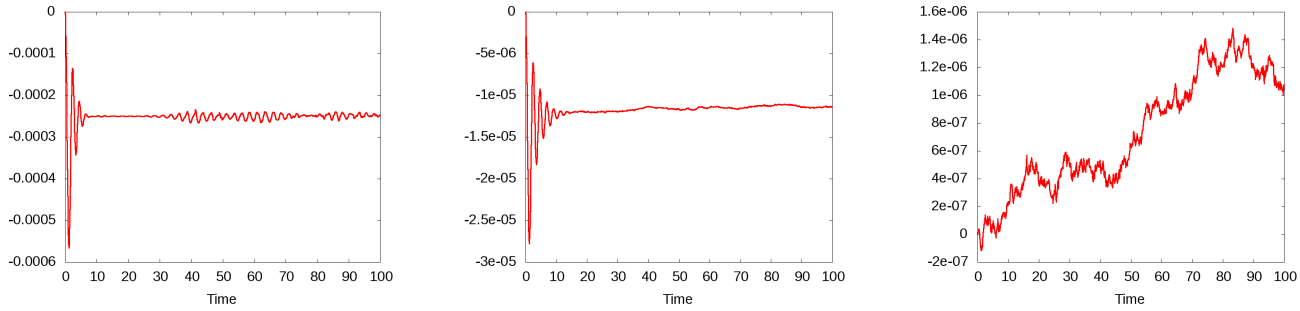

Figure 1: Time evolution of the relative error on the total energy for three values of the perturbations. At the left $\alpha = 0.5$, in the middle $\alpha = 0.1$, and at the right $\alpha = 0.005$.

2.2 Code organization

For the sequential implementation of this code we have focused on the same general type of data structures and algorithms as introduced in the work by Bowers (see [3] and [4]) and also

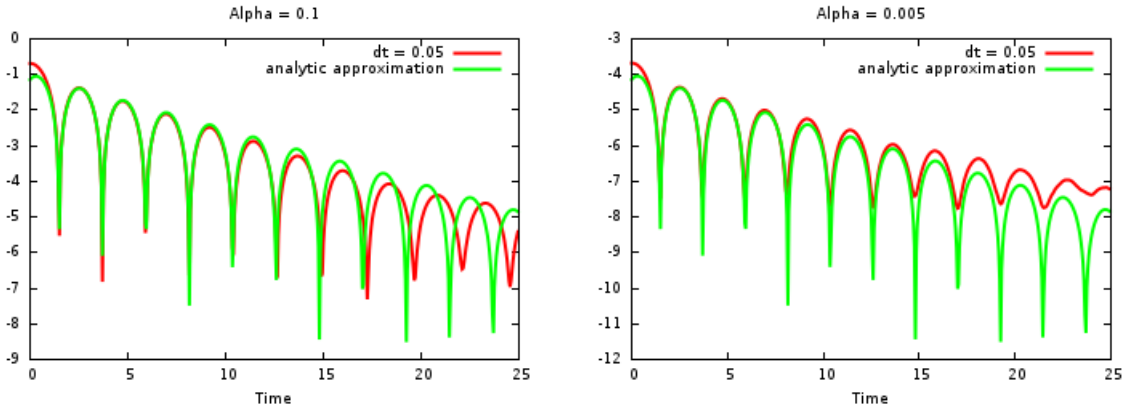

Figure 2: Time evolution of the log of the electric field energy for two perturbations. At left, $\alpha = 0.1$, a run with 16E4 particles per cell and 256 x 2 cells. At right, $\alpha = 0.005$, a run with 2E5 particles per cell and 512 x 2 cells.

implemented in other codes (see [12]). This approach is tailored to the modern commodity CPU architecture as it attempts to minimize and order the memory access as its guiding principle. This acknowledges that memory access is relatively expensive compared with computations. Thus, a minimization of the number of memory accesses and furthermore, an increase of the amount of sequential memory access results in very large performance gains. These principles are accomplished through several key design points:

- particle lists organized as arrays of structures: the array of structures obeys the principle of "data that are used together should be stored together"; all the particle information needed to represent a particle can be thus accessed in a single memory stream. Contrast this with an approach based on a structure of arrays in which the information of position, velocity, etc. are stored independently in their own arrays. The explicit representation of the particle's cell enables us to implement an efficient particle loop, but this can only be seen when considering all of the following items together.

- reduced memory footprint of the particle structure: the choice of representing the particle coordinates as a combination of the cell index plus the local (normalized) offset within the cell, as opposed to using the particle's global coordinates permits the reduction of the memory needed for each particle. Furthermore this representation reduces the amount of arithmetical operations needed during the interpolation and charge deposition procedures. There are other advantages to this approach related to the flexibility of the meshes that can be modeled, but these advantages were not exploited in this work.

- redundant, cell-based data structures to store field and deposited charge values: the field quantities that need to be accessed during the particle push are stored on a cell-basis (all the field values at the corners of the cell are stored in a single data structure which is then laid out as a single array). Together with the data structure used to represent the particle (see below), this maximizes the amount of memory that is accessed sequentially. What makes these data structures *redundant* is the fact that the field values for a single point must be stored in multiple cell-based structures, but the cost of accessing more memory is offset by ensuring that this access is done sequentially. Furthermore, the execution time is dominated by the size of the

particle arrays, not the field arrays.

- use of particle sorting techniques to maximize sequential memory access: With the previously described items, all the information needed to push the particles is stored in three different arrays in memory (one for particles, two for fields), the field information is indexed by cell number and the particle's cell is explicitly represented. If only the particles are sorted by cell number, this enables us to stream through these arrays in sequential form. Hence the use of a particle sorting algorithm. The push therefore looks like this: take a particle, read its cell, summon the necessary field data and complete the push. Read the next particle, read its cell (most likely the same as the previous particle), summon the field data (most likely already in the cache due to the previous particle, etc. As the simulation evolves, the particles get unsorted, so this algorithm needs to be applied periodically, but far from every time step.

- integrated inner loop: the main loop is responsible for advancing the particles, checking if the particle is out of bounds or needs further post-processing, and depositing the charge on the grid. All of these operations are condensed in a single pass over the particle list.

One necessary step, the conversion of the charge data structue into a two-dimensional array usable by the Poisson solver, is done outside of the inner loop. At least one more memory stream is needed in the more general version of this approach, to store references to particles which may need postprocessing outside of the main loop.

The above general features are enough to implement an optimal PIC loop from the perspective of memory access patterns. As mentioned above, during the natural progression of the simulation, the particles fall out of their cell-based ordering, thus yielding some performance degradation over time. For this reason some periodic re-sorting of the particle lists needs to be applied. In the tests reported here, the sorting frequency has been chosen to maximize the overall particle throughput.

Now, we describe the parallelization issues. First, by means of a distributed memory paradigm we have followed a simple strategy that permits scaling up to a few nodes: parallelization of the particle list processing while using interprocess communications to gather the charge density information and then redundantly computing the electric fields over the whole domain. No domain decomposition is involved. In other words, each process takes responsibility of the whole life-cycle of a particle list with particles defined over the whole region: from creation, initialization and time-stepping. Only at the time where the electric fields are needed the processes share their individual contributions to the total charge distribution. Second, within each process, we use the multithreading for assigning different segments of the particle list to the different threads. More precisely, we parallelize the loops over particles for the pusher in time, the charge accumulation, and the interpolation of the electric field. Then, we take into account the thread/core affinity control features of OpenMP to maximize the chances that a given core will access the memory that is closest to it. Thus, management of thread/core affinity through Intel's `KMP_AFFINITY` environment variable (= `compact`) yielded clear benefits. In addition, this setting yielded uniform run times on multiple runs on 16 cores.

We run 2 MPI processes, each on a different socket and with a number of threads from 1 to 8. This leads to simulations using from 1 to 16 threads in all, but distributed over 2 processes as described above. In order to assess the efficiency of our code we performed simulations with three data sizes: 2 million, 20 million, and 200 million particles. Thus, the data (particles and fields) size is bigger than the L3 cache size (thus the memory access speed is determined by the access speed to the main memory). The reference grid was set at 512 x 16 cells, the time step

to $\Delta t = 0.01$ and the iterations to 1000, when $\alpha = 0.1$.

The results of using thread affinity are illustrated in Fig. 3 and Table 1 where the best run times among several runs are given. The best speedup of 9.43 was reached on 16 cores for the biggest data size. As expected, Fig. 3 (at right) shows the linear growth of the run time with respect to the increase of the data size. This occurs when using one or several threads.

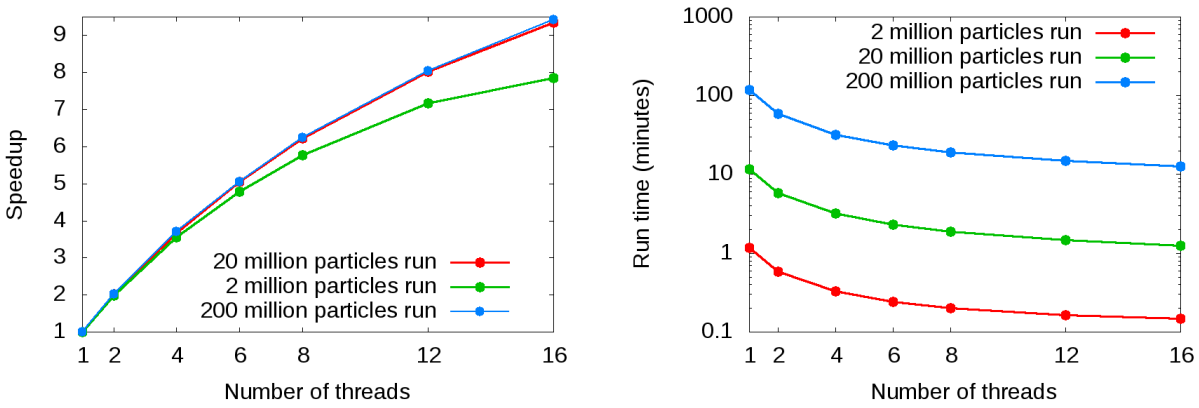

Figure 3: Speedup and best times by thread number for simulations with 3 data sizes

Threads	Time (seconds)	Speedup
1	703.2	1.0
2	348.2	2.02
4	191.5	3.67
6	139.2	5.05
8	112.9	6.23
12	87.7	8.02
16	75.3	9.34

Table 1: Best speedup and best times by thread number. The simulation with 20 million particles, 1000 iterations and a grid of 512 x 16 cells.

It is interesting to get an idea of the performance of the memory subsystem in these simulations. We have computed at each time step the obtained memory bandwidth during the particle push. This number is $N \cdot f/T$, where N is the number of particle loads and stores, f the memory-size of one particle (in our case, 32 bytes), and T the time to process the particle list. In Fig. 4 we illustrate the time evolution of this measure in GB per second when running on a single CPU. We can see that the highest memory bandwidth speeds are obtained when particles are sorted, as expected. In Fig. 5 we observe that, for two data sizes, similar values were obtained, when using one or several cores.

Maybe the most interesting measure of the performance of the whole simulation (not only the particle push) is the number $N_p \cdot N_{\text{iter}}/T$, where N_p is the number of particles, N_{iter} the number of iterations and T the total execution time. This number is reported in Table 2 for the three considered data sizes. Thus, we achieved a simulation with 265 million particles advanced

per second on a 2-socket node with 16 cores.

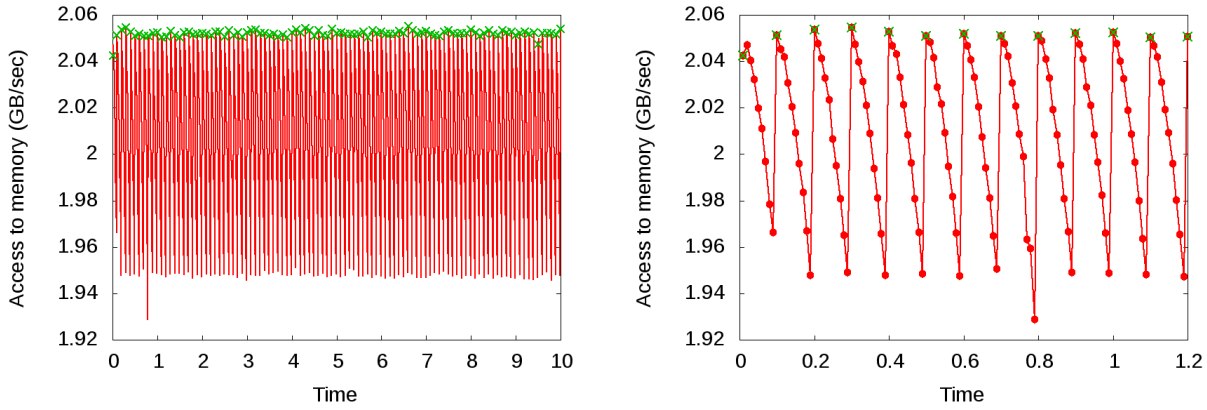

Figure 4: Time evolution of the access to memory in GB per second for the 2 million particles run on 1 core. At right a zoom at the begining of the simulation. In green the steps where sorting is done.

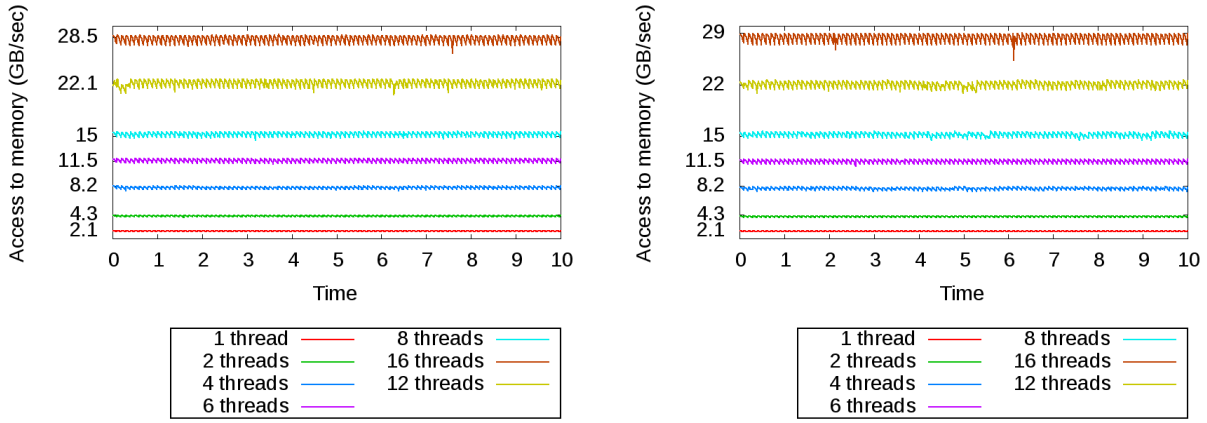

Figure 5: Time evolution of the accesses to memory in GB per second for the 2 million (left) and 20 million (right) particles runs, with multithreading.

Simulation\Threads	1	2	4	8	16
2 million particles	28.5	56.8	101.7	165.2	224.3
20 million particles	28.4	57.4	104.4	177.1	265.6
200 million particles	28.0	56.9	103.5	174.4	263.8

Table 2: Million of particles processed per second for 3 runs on a single or several cores of 1000 iterations with a grid of 512 x 16 cells.

2.3 Vlasov-Poisson vs. guiding center simulations

The aim of this section is to compare a reference solution of the Vlasov-Poisson system in (1) when ε decreases to a reference solution of the guiding center model in (3). For both systems we implement standard PIC methods as described previously. Next we discuss the numerical parameters and the results.

We consider the initial condition for a Kelvin-Helmholtz instability (see [11])

$$f_0(\mathbf{x},\mathbf{v}) = \frac{1}{2\pi}\exp\left(-\frac{v_1^2+v_2^2}{2}\right)\big(\sin(x_2)+0.05\,\cos(kx_1)\big), \tag{13}$$

defined in $\mathbf{\Omega_x}\times\mathbb{R}^2$, where $\mathbf{\Omega_x}=[0;2\pi/k]\times[0;2\pi]$ and $k=0.5$. For the guiding center model, the initial condition consists obviously in the $\mathbf{x}$ part in (13), defined in $\mathbf{\Omega_x}$.

First, for the numerical solution of the guiding center system, we push the particles in time with an explicit second order Runge-Kutta scheme. The time step is $\Delta t = 0.01$. We put 10 million particles and a cartesian grid with 256 x 128 cells. The energy (L^2-norm of $\mathbf{E}$) and the enstrophy (L^2-norm of f_{GC}) are known to be conserved quantities in time. Therefore, in order to validate the code, we compute these values. In Fig. 6 we plot in time $\big(||\mathbf{E}(t)||_{L^2}-||\mathbf{E}(0)||_{L^2}\big)/||\mathbf{E}(0)||_{L^2}$ and the same quantity for the enstrophy. We observe a good conservation of these values (this is comparable to the results in [6, 7]). Then, some contour plots of the guiding center density at different times are shown in Fig. 7. We let the simulation run until $t=100$ and the last obtained contour is similar to that at $t=64$.

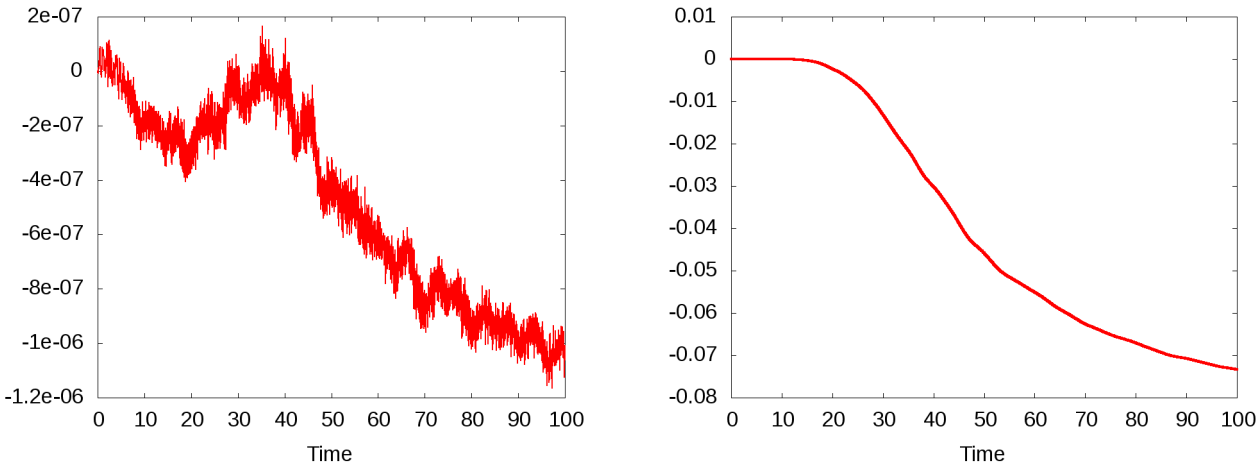

Figure 6: The time evolutions of the electric field energy (left) and the enstrophy (right) in the Kelvin-Helmholtz instability.

Second, we illustrate the convergence result mentioned in Section 1.1. Thus, we compute a reference solution for the Vlasov-Poisson model by using the leap-frog scheme in (6) with the time step $dt = 2\pi\varepsilon^2/80$, in order to solve the time period of the solution. The same numerical parameters as for the guiding center simulation are used. We recall that periodic boundary conditions are considered in the physical space. Now, for a quantitative verification of the convergence result we compute some norm in space of the densities. Thus, we first plot in Fig.

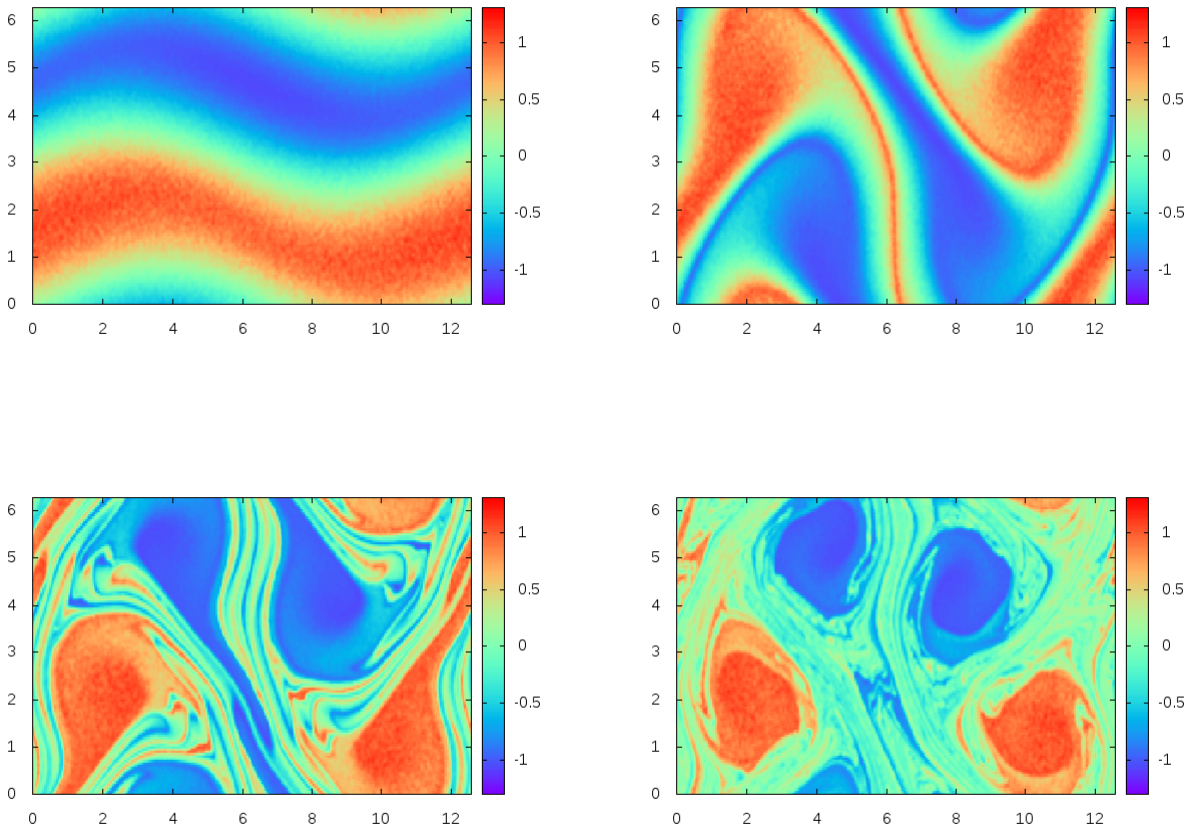

Figure 7: The Kelvin-Helmholtz instability: contour plots of the density given by the guiding center model, at times 6, 14, 32, and 64 (from left to right and from top to bottom).

8, as a function of ε, the maximum in time until $t = 5$ of the L^2 norm of the difference between f_{GC} in (3) and the density in (2) of the Vlasov-Poisson model (the value obtained for $\varepsilon = 0.5$ is of order 4). We notice again that this represents a long time simulation since the fastest scale of the solution of the system (1) is of order ε^2. Then, going further, in Fig. 9 we represent countours of the densities for $\varepsilon = 0.1$ and $\varepsilon = 0.5$, at two different final times, $t = 12$ and $t = 22$. We also performed simulations for $\varepsilon = 0.05$ (not shown here) and the Vlasov-Poisson reference solution qualitatively matches the guiding center solution. We thus observe that smaller values of ε make the Vlasov-Poisson solution closer to the guiding center solution. A more precise view of this convergence result is illustrated in Fig. 10, where cuts of the densities are done at 3 values of the x_1 direction, at $x_1 = 0$, $x_1 = \pi$ and $x_1 = 2\pi$. We finally remark that the guiding center numerical solution is a good approximation of the Vlasov-Poisson numerical solution even if the value of ε is not very small, namely $\varepsilon = 0.1$.

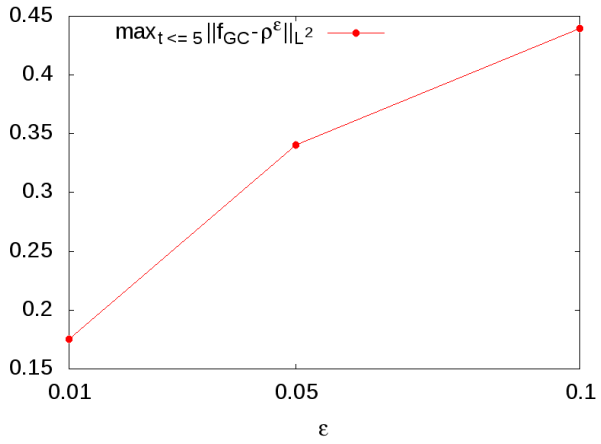

Figure 8: The global error at time $t = 5$ of the density of the Vlasov-Poisson solution with respect to the guiding center solution

3 Conclusion and outlook

The aim of this paper was to efficiently implement a Particle-In-Cell method for solving some nonlinear Vlasov-like models in a four-dimensional phase space. The organization of the code follows an existing work ([3]) and mainly focuses on optimization of the memory access. We demonstrated the performance of our code within some numerical experiments on the Landau damping test-case. We also did verification of the code on simulation of Kelvin-Helmholtz instability. In addition, this test-case allowed us to numerically illustrate a convergence result beetwen a Vlasov-Poisson model with strong magnetic field and the guiding center model.

In the following, we intent to implement the time stepping scheme introduced in [8]. This method allows to solve multiscale Vlasov-Poisson models by treating the stiff terms as an exponential integrator. In the framework of an optimized PIC implementation, this scheme potentially provides a solution to perform efficient simulations of highly oscillating six-dimensional models without solving the smallest scale.

References

[1] C. K. Birdsall, A. B. Langdon, Plasma physics via computer simulation, Institute of Physics, Bristol (1991).

[2] M. Bostan, *The Vlasov-Maxwell System with Strong Initial Magnetic Field: Guiding-Center Approximation*, Multiscale Model. Simul., 6(3), pp 1026–1058 (2007).

[3] K. J. Bowers, *Accelerating a Particle-in-Cell Simulation Using a Hybrid Counting Sort*, J. Comput. Phys. 173, pp 393–411 (2001).

[4] K. J. Bowers, B. J. Albrigth, L. Yin, B. Bergen, and T.J.T. Kwan, *Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation*, Phys. Plasmas 15, 055703 (2008).

[5] M. Campos-Pinto, *Towards smooth particle methods without smoothing*, to appear in J. of Scientific Computing.

[6] A. Christlieb, W. Guo, M. Morton, and J.-M. Qiu, *A High Order Time Splitting Method Based on Integral Deferred Correction for Semi-Lagrangian Vlasov Simulations*, J. Comput. Phys. 267, pp 7–27, (2014).

[7] N. Crouseilles, M. Mehrenberger, and E. Sonnendrücker, *Conservative semi-Lagrangian schemes for Vlasov equations*, J. Comput. Phys. 229(6), pp 1927–1953 (2010).

[8] 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*, Commun. Comput. Phys., 18, pp 263–296 (2015).

[9] E. Frénod, E. Sonnendrücker, *Long time behavior of the two dimensionnal Vlasov equation with a strong external magnetic field*, Math. Models Methods Appl. Sci., 10(4), pp 539–553 (2000).

[10] E. Frénod, P.-A. Raviart, and E. Sonnendrücker, *Two-scale expansion of a singularly perturbed convection equation*, J. Math. Pures Appl., 80(8), pp 815–843 (2001).

[11] E. Sonnendrücker, *Approximation numérique des équations de Vlasov-Maxwell*, notes de cours de M2, Université de Strasbourg, (2010).

[12] D. Tskhakaya, R. Schneider, *Optimization of PIC codes by improved memory management*, J. Comput. Phys., 225, pp 829–839 (2007).

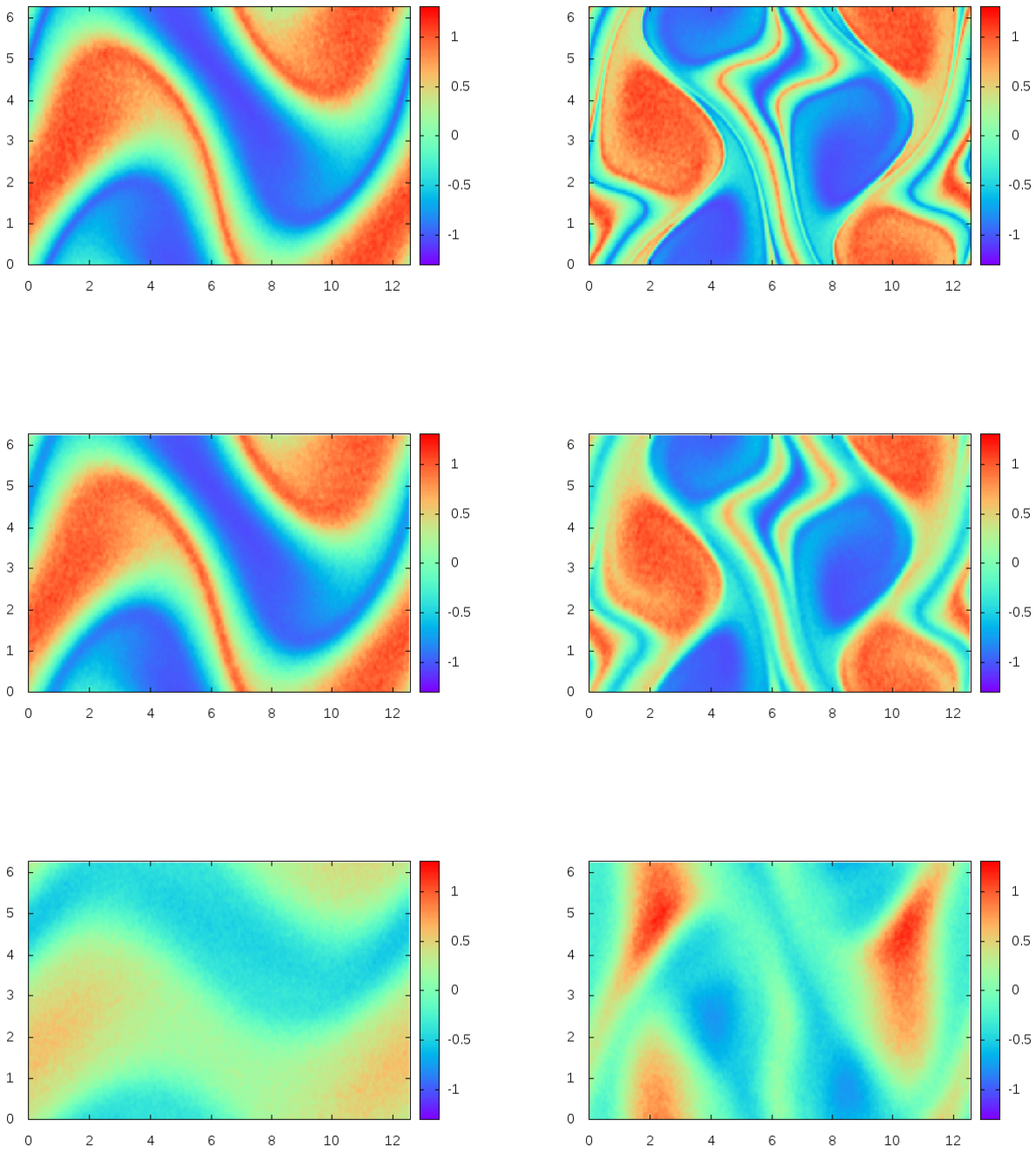

Figure 9: The Kelvin-Helmholtz instability: contour plots of the densities given by the guiding center model (top row), by the Vlasov-Poisson model with $\varepsilon = 0.1$ (middle row) and $\varepsilon = 0.5$ (bottom row), at times 12 (left) and 22 (right).

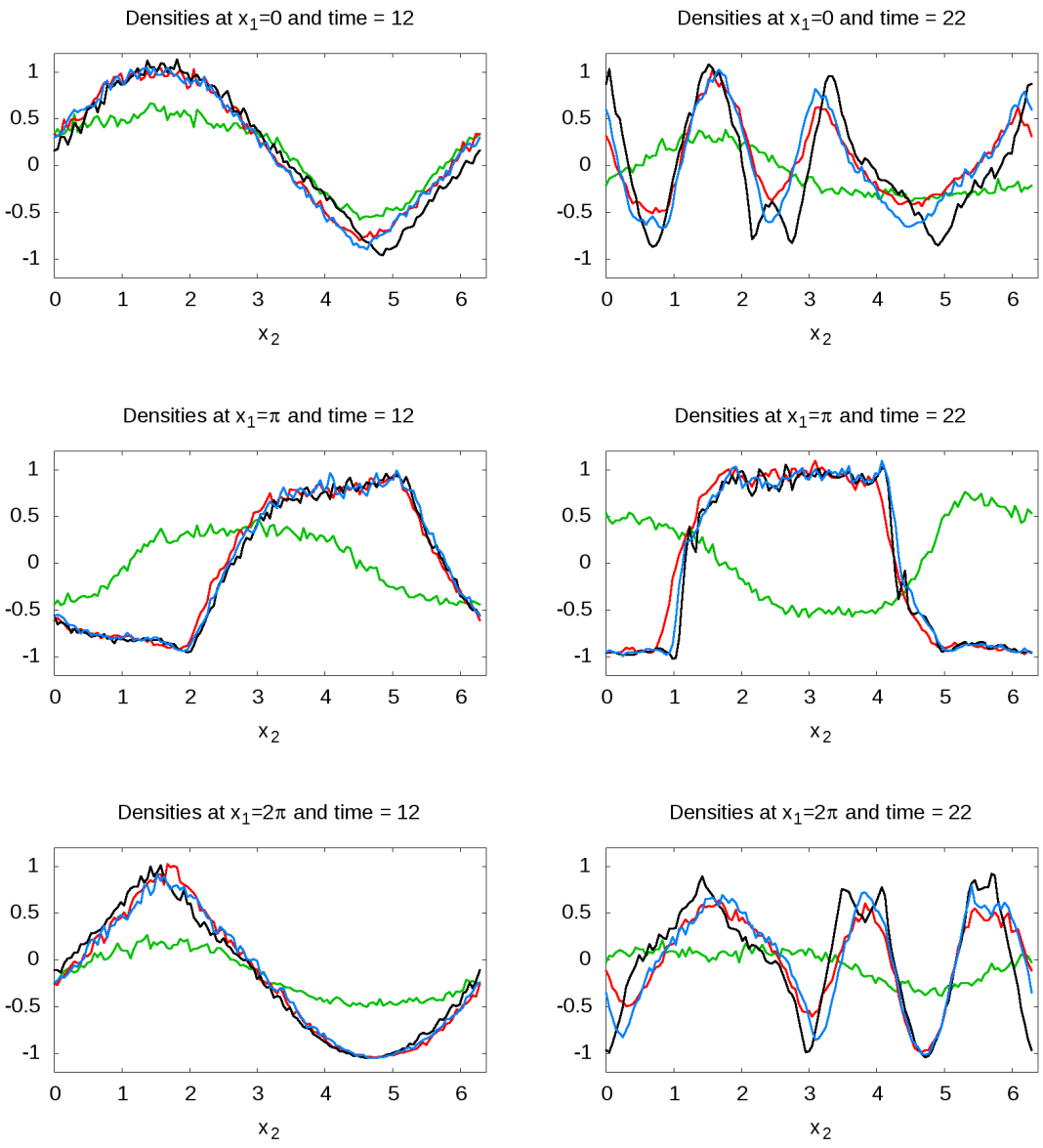

Figure 10: The Kelvin-Helmholtz instability: Cuts in different x_1 directions of the densities of the guiding center model (in black), the Vlasov-Poisson model with $\varepsilon = 0.05$ (in blue), with $\varepsilon = 0.1$ (in red), and with $\varepsilon = 0.5$ (in green). The final times are 12 (at left) and 22 (at right).