Hermite spline interpolation on patches for a parallel solving of the Vlasov-Poisson equation

N. Crouseilles* G. Latu† E. Sonnendrücker‡

Abstract

This work is devoted to the numerical simulation of the Vlasov equation using a phase space grid. In contrast with Particle In Cell (PIC) methods which are known to be noisy, we propose a semi-Lagrangian type method to discretize the Vlasov equation in the two dimensional phase space. As this kind of method requires a huge computational effort, one has to carry out the simulations on parallel machines. To this purpose, we present a method using patches decomposing the phase domain, each patch being devoted to a processor. Some Hermite boundary conditions allow to reconstruct a good approximation of the global solution. Several numerical results show the accuracy and the good scalability of the method with up to 64 processors. This work is a part of the CALVI Project.

Contents

*INRIA Lorraine (CALVI Project), IECN Nancy and IRMA Strasbourg, France. crouseil@math.u-strasbg.fr

†LSIIT Strasbourg and INRIA Lorraine (CALVI Project), France. latu@icps.u-strasbg.fr

‡IRMA Strasbourg and INRIA Lorraine (CALVI Project), France. sonnen@math.u-strasbg.fr

1 Introduction

The Vlasov-Poisson equation describes the evolution of a system of charged particles under the effects of a self-consistent electric field. The unknown f is a distribution function of particles in the phase space which depends on the time $t \geq 0$, the physical space $x \in \mathbb{R}^d$ and the velocity $v \in \mathbb{R}^d$, where d is the dimension $d = 1, 2, 3$. This kind of model can be used for the study of beam propagation, collisionless or gyrokinetic plasmas.

The numerical resolution of Vlasov type equations, the solution of which depends at least on 6 variables plus time, is performed most of the time using particle methods (Particle In Cell methods) where the plasma is approached by a finite number of macro-particles. The trajectories of these particles are computed using the characteristic curves given by the Vlasov equation, whereas the self-consistent electric field is computed on a fixed grid [3]. Even if these methods give satisfying results with relatively few particles, for some applications however (in particular when particles in the tail of the distribution function play an important physical role, or when one wants to study the influence of density fluctuations which are at the origin of instabilities), it is well known that the numerical noise inherent to the particle methods becomes too significant. Consequently, methods which discretize the Vlasov equation on a phase space grid have been proposed (see [9, 10, 11, 14, 15, 22, 24] for plasma physics and [1, 25] for others applications). Among these Eulerian methods, the semi-Lagrangian method consists in computing directly the distribution function on a Cartesian grid of the phase space. The computation is done by integrating the characteristic curves backward at each time step and interpolating the value at the feet of the characteristics by some interpolation techniques (Lagrange, Hermite or cubic splines for example). We refer the reader to [24] for more details on the semi-Lagrangian method and to [11] for a comparison of Eulerian solvers dedicated to the Vlasov equation.

Eulerian methods have proven their efficiency on uniform meshes in two dimensional phase space, but when the dimensionality increases, the number of points on a uniform grid becomes very important which makes numerical simulations challenging. Two kinds of strategy have been recently developed to simulate four dimensional problems. Some adaptive methods decrease computational cost by keeping only a subset of all grid points. Such methods use moving distribution function grids well suited to manage data locality. For more details, we refer the reader to [6, 17, 23]. On the other side, some parallelized version of codes have been implemented to simulate high dimensional problems (see [8, 12]). Generally, the numerical scheme is based on a time splitting scheme which can be parallelized very efficiently on a moderate number of processors using a global transposition between each split step. Apart from this transposition that can be overlapped with computations, there is no communication between the processors. However, when heterogeneous grids and several hundreds or more processors are targeted (see [16, 19]), a global transposition involves a huge amount of data beeing transfered and this can become very inefficient. For these reasons, we develop in this paper a local spline interpolation technique that avoids any global transposition.

This work is devoted to the parallel implementation of the semi-Lagrangian method by using

the cubic spline interpolation operator. In order to check the method, we have designed the parallel software LOSS (LOcal Splines Simulator). Even if cubic spline interpolation seems to be a good compromise between accuracy (small diffusivity) and simplicity, it does not provide the locality of the reconstruction since all the values of the distribution function are used for the reconstruction in each cell. To overcome this problem of global dependency, we decompose the phase space domain into patches, each patch being devoted to one processor. One patch computes its own local cubic spline coefficients by solving reduced linear systems; Hermite boundary conditions are imposed at the boundary of the patches to reconstruct a global $\mathcal{C}^1$ numerical solution.

In fact, our strategy consists in getting a parallel version of the code, the results of which are as close as possible to the results of the sequential version. Even if the methodology remains slightly different from the sequential case (essentially due to the local resolution of the cubic spline coefficients *versus* the global resolution), our main efforts consist in recovering in the best possible way the global resolution. Thanks to an adapted treatment of the Hermite boundary conditions, the obtained numerical results are then in good agreement with those obtained with the sequential version of the code. Moreover, some communications between processors have to be managed in a suitable way; indeed, as particles can leave the subdomain, their information must be forwarded to the appropriate processor that controls the subdomain in which the particles now reside. Such interprocessor communications would involve a relatively large amount of data exchange, but a condition on the time step allows us to control the shifts so that the communications are only done between adjacent processors. Hence, this communication scheme enables us to obtain competitive results from a scalability point of view. Let us mention that even if a uniform grid is used here, the methodology could be extended to sets of lines which are not equally spaced (adaptive meshes for example).

This work contributes to the improvment of a five dimensional semi-Lagrangian gyrokinetic code which simulates the turbulent transport in magnetized fusion plasma. This high dimensional problem is very demanding in terms of numerics hence, the code is devoted to be massively parallelized; a time-splitting algorithm allows to reduce the problem into a sequence of one dimensional and two dimensional advections. Our method enables to accurately solve this advection part using parallel computations (see [16] for more details).

The paper is organized as follows. First, we draw up some basic properties of the Vlasov-Poisson model. Then, we recall the main steps of the semi-Lagrangian method. Next, we propose the Hermite spline interpolation on patches before illustrating the efficiency of the method by presenting several numerical results.

2 The Vlasov-Poisson model

The evolution of the distribution function of particles $f(t,x,v)$ in phase space $(x,v) \in \mathbb{R}^d \times \mathbb{R}^d$ with $d = 1, 2, 3$ and t the time, is given by the dimensionless Vlasov equation

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f + F(t,x,v) \cdot \nabla_v f = 0, \tag{2.1}$$

where the force field $F(t,x,v)$ can be coupled to the distribution function f, t is the scaled time. For the Vlasov-Poisson system, this coupling is done through the macroscopic density ρ

$$\rho(t,x) = \int_{\mathbf{R}^d} f(t,x,v)dv.$$

The force field which only depends on t and x makes the system nonlinear and is given by

$$F(t,x,v) = E(t,x), \quad E(t,x) = -\nabla_x \phi(t,x), \quad -\Delta_x \phi(t,x) = \rho(t,x) - 1, \tag{2.2}$$

where E is the electric field and ϕ the electric potential. These two quantities depend on the total charge in the plasma where the ions form a fix and uniform background. In the sequel, we briefly recall some classical estimates on the Vlasov-Poisson system (2.1)-(2.2). First of all, mass and momentum are preserved with time,

$$\frac{d}{dt}\int_{\mathbf{R}^d\times\mathbf{R}^d} f(t,x,v)\begin{pmatrix}1\\ v\end{pmatrix}dxdv = 0, \qquad t \in \mathbb{R}^+.$$

Next, multiplying the Vlasov equation (2.1) by $|v|^2$ and performing an integration by parts, we find the conservation of energy for the (2.1)-(2.2) system

$$\frac{d}{dt}\left(\int_{\mathbf{R}^d\times\mathbf{R}^d} f(t,x,v)\,\frac{|v|^2}{2}dxdv + \frac{1}{2}\int_{\mathbf{R}^d}|E(t,x)|^2dx\right) = 0, \qquad t \in \mathbb{R}^+.$$

Finally, the Vlasov-Poisson equation (2.1)-(2.2) conserves the kinetic entropy

$$H(t) = \int_{\mathbf{R}^d\times\mathbf{R}^d} f(t)\,\log(f(t))dxdv = H(0).$$

On the other hand, we can define the characteristic curves of the Vlasov-Poisson equation (2.1)-(2.2) as the solutions of the following first order differential system

$$\left\{\begin{array}{rcl} \dfrac{dX}{dt}(t;s,x,v) &=& V(t;s,x,v),\\[2ex] \dfrac{dV}{dt}(t;s,x,v) &=& E(t,X(t;s,x,v)),\end{array}\right. \tag{2.3}$$

with the initial conditions

$$X(s;s,x,v) = x, \quad V(s;s,x,v) = v.$$

We denote by $(X(t;s,x,v),V(t;s,x,v))$ the position in phase space at the time t, of a particle which was in (x,v) at time s. Let say that $t \to (X(t;s,x,v),V(t;s,x,v))$ is the characteristic curves solution of (2.3). Then, the solution of the Vlasov-Poisson equation (2.1)-(2.2) is given by

$$\begin{aligned} f(t,x,v) &= f(s,X(s;t,x,v),V(s;t,x,v)) &(2.4)\\ &= f_0(X(0;t,x,v),V(0;t,x,v)), \quad (x,v)\in\mathbb{R}^d\times\mathbb{R}^d, \quad t\geq 0, &(2.5)\end{aligned}$$

where f_0 is a given initial condition of the Vlasov-Poisson equation. This equality means that the distribution function f is constant along the characteristic curves which is the basis of the numerical method we present in the next section. For more details, we refer the reader to [4, 13].

3 The semi-Lagrangian method

In this section, we will recall the principles of the semi-Lagrangian method for the Vlasov-Poisson equation (see [24] for details) in two dimensions of the phase space.

First of all, we introduce a finite set of mesh points $(x_i, v_j), i = 0, ..., N_x$ and $j = 0, ..., N_v$ to discretize the computational domain. Then, given the value of the distribution function f at the mesh points at any given time step t^n, we obtain the new value at mesh points (x_i, v_j) at t^{n+1} using

$$f(t^n + \Delta t, x_i, v_j) = f(t^n, X^n, V^n),$$

where the notations $(X^n, V^n) = (X(t^n; t^n + \Delta t, x_i, v_j), V(t^n; t^n + \Delta t, x_i, v_j)$ are used for the solutions of (2.3), and Δt stands for the time step. For each mesh point (x_i, v_j), the distribution function f is then computed at t^{n+1} in two steps

1. Find the starting point of the characteristic ending at (x_i, v_j), *i.e.* X^n and V^n.

2. Compute $f(t^n, X^n, V^n)$ by interpolation, f being known only at mesh points at time t^n.

In order to deal with the first step, we have to introduce a time discretization of (2.3). To remain second order accurate in time, we use the Ampère equation to get an approximation of the electric field at time $t^n + \Delta t$

$$\frac{\partial E(t,x)}{\partial t} = -J(t,x), \tag{3.1}$$

where $J = J(t,x)$ is the current given by

$$J(t,x) = \int_{\mathbf{R}} f(t,x,v) v dv.$$

The equation (3.1) is discretized as follows

$$E_i^{n+1/2} = E_i^n - \frac{\Delta t}{2} J_i^n,$$

where Δt is the time step, E_i^n is the electric field evaluated at $t = t^n$ in $x = x_i$ and J_i^n is the current evaluated at time t^n in x_i, and is given by

$$J_i^n = \sum_{j=0}^{N_v} f(t^n, x_i, v_j) v_j \Delta v, \tag{3.2}$$

with Δv the velocity step. Then, we solve (2.3) at the second order accurate in time thanks to a predictor-corrector scheme. The semi-Lagrangian method then reduces to the following algorithm:

Let us suppose that $f(t^n, x_i, v_j), E_i^n$ are known on the mesh points

Step 1. Computation of a prediction of $E_i^{n+1/2}$, called $\tilde{E}_i^{n+1/2}$, by solving the Ampère equation

$$\tilde{E}_i^{n+1/2} = E_i^n - \frac{\Delta t}{2} J_i^n,$$

where J_i^n is computed *via* (3.2).

Step 2. Resolution of (2.3)

• Backward advection of $\Delta t/2$ in the spatial direction:

$$X^{n+1/2} = X^{n+1} - \frac{\Delta t}{2} V^{n+1}.$$

• Backward advection of Δt in the velocity direction:

$$V^n = V^{n+1} - \Delta t\, \tilde{E}^{n+1/2}(X^{n+1/2}).$$

• Backward advection of $\Delta t/2$ in the spatial direction:

$$X^n = X^{n+1/2} - \frac{\Delta t}{2} V^n.$$

Step 3. Interpolation of $f(t^n, X^n, V^n)$ and updating of the distribution function thanks to the following equality

$$f(t^{n+1}, X^{n+1}, V^{n+1}) = f(t^n, X^n, V^n),$$

and computation of the density $\rho^{n+1}(X^{n+1})$

$$\rho^{n+1}(X^{n+1}) = \int_{\mathrm{R}} f(t^{n+1}, X^{n+1}, v) dv.$$

Step 4. Correction step: computation of the electric field by solving the Poisson equation at time t^{n+1}

$$\frac{\partial E^{n+1}}{\partial x} = \rho^{n+1} - 1.$$

Hence, the step 2 allows to compute the starting point of the characteristic (X^n, V^n) thanks to the knowledge of (X^{n+1}, V^{n+1}). Once we have followed the characteristics curves backward, we have to evaluate the distribution function at the end points of the characteristic curves which do not generally coincide with the mesh where f is known (step 3). The last step is a correction step since the predicted electric field $\tilde{E}^{n+1}$ is replaced by the true electric field E^{n+1} at time t^{n+1}, solution to the Poisson equation at time t^{n+1}. Let us remark that the evaluation of the electric field at time $t^{n+1/2}$ in $X^{n+1/2}$ that does not belong necessarily to the mesh is performed thanks to a linear approximation.

This algorithm may be iterated so that the predicted electric field $\tilde{E}^{n+1}$ becomes sufficiently close to the true electric field E^{n+1} at time t^{n+1}. In practice, one iteration of this algorithm already gives enough accuracy.

4 The local spline interpolation

In this section, we present our interpolation technique based on a cubic spline method (see [5, 18, 24]). Even if the cubic spline approach is quite standard for solving Vlasov equations,

it remains a global method since it requires the values of the distribution function on all the domain, which is an inconvenient from a parallelization point of view. Our approach avoids this globality. Indeed, we decompose the phase space into several patches, each patch being devoted to one processor. The strategy is based on adapted boundary conditions which allow a $\mathcal{C}^1$ reconstructed solution on the global phase space domain even on the patches boundaries.

We first present the interpolation on one patch in an unidimensional context before focusing on the two dimensional case.

4.1 The local spline interpolation in one dimension

Let us consider a function f which is defined on a global domain $[x_{\min}, x_{\max}] \subset \mathbb{R}$. This domain is decomposed into several subdomain called generically $[x_0, x_N]$; each subdomain will be devoted to a processor. In the following, we will use the notation $x_i = x_0 + ih$, where h is the mesh size: $h = (x_N - x_0)/(N+1)$.

Let us now restrict the study of $f : x \to f(x)$ on an interval $[x_0, x_N]$, $N \in \mathbb{N}$, where x_0 and x_N are to be chosen, according to the decomposition domain. The projection s of f onto the cubic spline basis reads

$$f(x) \simeq s(x) = \sum_{\nu=-1}^{N+1} \eta_\nu B_\nu(x),$$

where B_ν is the cubic B-spline

$$B_\nu(x) = \frac{1}{6h^3} \begin{cases} (x - x_{\nu-2})^3 & \text{if } x_{\nu-2} \le x \le x_{\nu-1}, \\ h^3 + 3h^2(x - x_{\nu-1}) + 3h(x - x_{\nu-1})^2 & \\ \quad -3(x - x_{\nu-1})^3 & \text{if } x_{\nu-1} \le x \le x_\nu, \\ h^3 + 3h^2(x_{\nu+1} - x) + 3h(x_{\nu+1} - x)^2 & \\ \quad -3(x_{\nu+1} - x)^3 & \text{if } x_\nu \le x \le x_{\nu+1}, \\ (x_{\nu+2} - x)^3 & \text{if } x_{\nu+1} \le x \le x_{\nu+2}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

The interpolating spline s is uniquely determined by $(N+1)$ interpolating conditions

$$f(x_i) = s(x_i), \quad \forall i = 0, ..., N, \tag{4.2}$$

and the Hermite boundary conditions at both ends of the interval in order to obtain a $\mathcal{C}^1$ global approximation

$$f'(x_0) \simeq s'(x_0), \quad f'(x_N) \simeq s'(x_N). \tag{4.3}$$

The only cubic B-spline not vanishing at point x_i are $B_{i\pm1}(x_i) = 1/6$ and $B_i(x_i) = 2/3$. Hence (4.2) yields

$$f(x_i) = \frac{1}{6}\,\eta_{i-1} + \frac{2}{3}\,\eta_i + \frac{1}{6}\,\eta_{i+1}, \quad i = 0, ..., N. \tag{4.4}$$

On the other hand, we have $B'_{i\pm1}(x_i) = \pm 1/(2h)$, and $B'(x_i) = 0$. Thus the Hermite boundary conditions (4.3) become

$$f'(x_0) \simeq s'(x_0) = -1/(2h)\,\eta_{-1} + 1/(2h)\,\eta_1, \tag{4.5}$$

and

$$f'(x_N) \simeq s'(x_N) = -1/(2h)\ \eta_{N-1} + 1/(2h)\ \eta_{N+1}.$$

Finally, $\eta = (\eta_{-1}, ...\eta_{N+1})^T$ is the solution of the $(N+3) \times (N+3)$ system $A\eta = F$, where F is the following vector and

$$F = \left[f'(x_0), f(x_0), ..., f(x_N), f'(x_N)\right]^T. \tag{4.6}$$

and A denotes the following matrix

$$A = \frac{1}{6}\begin{pmatrix} -3/h & 0 & 3/h & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & & \vdots \\ 0 & 1 & 4 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & -3/h & 0 & 3/h \end{pmatrix}. \tag{4.7}$$

Let us precise that f' is a notation, and from a numerical point of view, the derivative of f has to be approximated in a good sense. We will focus on this in the sequel of the paper.

Resolution of the linear system $A\eta = F$

The matrix A of the linear system has a special structure. It's LU decomposition is of the following form

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ -h/3 & 1 & 0 & \ddots & & \vdots \\ 0 & l_1 & 1 & \ddots & & \vdots \\ 0 & 0 & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & l_N & 1 & 0 \\ 0 & \cdots & 0 & -(3l_N)/h & (3l_{N+1})/h & 1 \end{pmatrix},$$

and

$$U = \frac{1}{6}\begin{pmatrix} -3/h & 0 & 3/h & 0 & \cdots & 0 \\ 0 & d_1 & 2 & 0 & & \vdots \\ 0 & 0 & d_2 & 1 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & d_{N+1} & 1 \\ 0 & \cdots & 0 & 0 & 0 & (3d_{N+2})/h \end{pmatrix},$$

where l_i and d_i can be computed from the following relations

$$d_1 = 4,\ l_1 = 1/4,\ d_2 = 4 - 2l_1 = 7/2,$$

from $i = 2, N$

$$l_i = 1/d_i,$$
$$d_{i+1} = 4 - l_i,$$

and

$$l_{N+1} = \frac{1}{d_N d_{N+1}},$$
$$d_{N+2} = 1 - l_{N+1}.$$

The LU decomposition of A can then be computed once for all. At each time step, a spline interpolant needs to be computed solving $LU\eta = F$ into two steps: the resolution of $L\varphi = F$, and then, the resolution of $L\eta = \varphi$.

4.2 The local spline interpolation in two dimensions

In a two dimensional space, f is projected on a cubic spline basis $\forall (x,y) \in [x_0, x_{N_x}] \times [y_0, y_{N_y}]$ as follows

$$f(x,y) \simeq s(x,y) = \sum_{\nu=-1}^{N_x+1} \sum_{\beta=-1}^{N_y+1} \eta_{\nu,\beta} B_\eta(x) B_\beta(y). \tag{4.8}$$

The same notations used in the previous section are performed and we have to compute the coefficients $\eta_{\nu\beta}$. To that purpose, we first solve the $N_y + 1$ following systems

$$s(x, y_j) = \sum_{\nu=-1}^{N_x+1} \gamma_\nu(y_j) B_\nu(x), \quad \forall j = 0, ..., N_y, \tag{4.9}$$

where

$$\gamma_\nu(y_j) = \left[\gamma_{-1}(y_j), \gamma_0(y_j), ..., \gamma_\nu(y_j), ..., \gamma_{N_x}(y_j), \gamma_{N_x+1}(y_j)\right]^T.$$

Each of the $N_y + 1$ systems (4.9) satisfies the $N_x + 1$ interpolation conditions (at fixed j)

$$f(x_i, y_j) = s(x_i, y_j), \quad i = 0, ..., N_x,$$

and the Hermite boundary conditions in the x-direction

$$\frac{\partial f}{\partial x}(x_0, y_j) \simeq \frac{\partial s}{\partial x}(x_0, y_j), \quad \frac{\partial f}{\partial x}(x_{N_x}, y_j) \simeq \frac{\partial s}{\partial x}(x_{N_x}, y_j).$$

We have denoted by

$$\gamma_\nu(y_j) = \sum_{\beta=-1}^{N_y+1} \eta_{\nu,\beta} B_\beta(y_j). \tag{4.10}$$

We have been brought to solve $N_y + 1$ linear systems $A\gamma_\nu(y_j) = F(y_j)$, one for each value of j, involving the $(N_x+3)\times(N_x+3)$ matrix (4.7) and a (N_x+3) vector similar to (4.6) evaluated in y_j. Following the same procedure used previously (*via* the LU decomposition), we then obtain the $(N_x + 3)$ vector of unknown $\gamma_\nu(y_j)$, for $j = 0, ..., N_y$

$$\gamma_\nu(y_j) = \left[\gamma_{-1}(y_j), \gamma_0(y_j), ..., \gamma_\nu(y_j), ..., \gamma_{N_x}(y_j), \gamma_{N_x+1}(y_j)\right]^T.$$

The second step consists in the resolution for each $\nu = -1, .., N_x + 1$ of a one dimensional problem given by (4.10). However, the left hand side of this system is only known for values of y_j, $j = 0, ..., N_y$ (*i.e.* it is a vector of $N_y + 1$ components) whereas the right hand side is a (N_y+3) vector. Some boundary conditions are necessary to close the system. Hermite boundary conditions are imposed for the first and last component of the vector (which corresponds to $j = -1$ and $j = N_y$), that is to say, we have to compute $\gamma'_\nu(y_0)$ and $\gamma'_\nu(y_{N_y})$, $\forall \nu = -1, ..., N_x+1$. To achieve this task, we solve two systems: we first derive (4.10) with respect to the y variable, and then evaluate it in $y_j = y_0$ and $y_j = y_{N_y}$. The Hermite boundary conditions have to be adapted to this particular case. Consequently, we have to solve the two following systems (associated to $j = 0$ and $j = N_y$): $A\gamma'_\nu(y_j) = \partial_y f(x, y_j)$, where A is the matrix (4.7),

$$\gamma'_\nu(y_j) = \left[\gamma'_{-1}(y_j), ..., \gamma'_\nu(y_j), ..., \gamma'_{N_x+1}(y_j)\right]^T,$$

and the right hand side writes

$$\partial_y f(x, y_j) = [\partial_{xy} f(x_0, y_j), \partial_y f(x_0, y_j), ..., \partial_y f(x_i, y_j), ..., \partial_y f(x_{N_x}, y_j), \partial_{xy} f(x_{N_x}, y_j)]^T.$$

Once we computed $\gamma'_\nu(y_0)$ and $\gamma'_\nu(y_{N_y})$, for all $\nu = -1, ..., N_x + 1$, we solve the system (4.10) which writes here $A\eta_{\nu,\beta} = \Gamma_{\nu\beta}$. The matrix A is given by (4.7) and the right hand side $\Gamma_{\nu\beta}$ reads

$$\Gamma_{\nu\beta} = \left[\gamma'_\nu(y_0), \gamma_\nu(y_0, ..., \gamma_\nu(y_{N_y}), \gamma'_\nu(y_{N_y})\right]^T,$$

for each value of $\nu = -1, ..., N_x + 1$.

Once the spline coefficients $\eta_{\nu,\beta}$ have been computed for all ν and β, the value of f at the origin of the characteristics (X^n, V^n) (determined following the algorithm of section 3) is taken to be the value of the spline $s(X^n, V^n)$. If (X^n, V^n) belongs to $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$, the approximation of the function $f(X^n, V^n)$ is given by

$$s(X^n, V^n) = \sum_{\nu=i-1}^{i+2} \left(\sum_{\beta=j-1}^{j+2} \eta_{\nu,\beta} B_\nu(X^n) B_\beta(V^n) \right),$$

where B_ν and B_β are given by (4.1). To compute $s(X^n, V^n)$ for all mesh points requires $\mathcal{O}(N_x N_y)$ floating points operations.

Remark 4.1 *In summary, we have to solve*

$$(N_y + 1) \text{ systems of size } (N_x + 3) \times (N_x + 3) \quad (\text{to get } \gamma_\nu(y_j), \quad \forall j = 0, ..., N_y),$$

$$2 \text{ systems of size } (N_x + 3) \times (N_x + 3) \quad (\text{to get } \gamma'_\nu(y_0) \quad \text{and} \quad \gamma'_\nu(y_{N_y})),$$

$$(N_x + 3) \text{ systems of size } (N_y + 3) \times (N_y + 3) \quad (\text{to get } \eta_{\nu,\beta}).$$

From a computational cost point of view, the resolution of a linear system of size N_x using the LU decomposition needs $\mathcal{O}(N_x)$ operations. Such a resolution has to be done N_y times for the x-direction; the same is true for the y-direction. Finally, the two dimensional interpolation leads to $\mathcal{O}(N_x N_y)$ operations.

4.3 Towards an accurate parallelization

In order to get accurate numerical simulations, one has to take care of boundary conditions for each local LU solve. Indeed, our strategy consists in being as closer as possible to the corresponding sequential version. Hence, from a decomposition of the global domain into several patches, each processor being devoted to a patch, one wants that our local resolution of cubic spline coefficients recovers in the best way an usual resolution on the global domain. To that purpose, some efforts have to be done to approximate the derivatives of f in a particular way with respect to x and y. The points where derivatives must be computed are shared between two processors since x_0 and x_N are both beginning and end of subdomains (x_N of the target processor corresponds to x_0 of the adjacent processor). Hence, these derivatives of f join adjacent subdomains and play an important role in the quality of the numerical results (see figure 5 in section 6).

Different ways have been explored to obtain the derivatives: finite differences of different orders, cubic spline approximation ... In order to reconstruct a smooth approximation (let say $\mathcal{C}^1$ on the global domain), the cubic spline approximation has been chosen. Indeed, we remark that even in regions where f is smooth enough, a finite differences approximation remains quite different from a cubic spline approximation given by (4.5). Hence, as we want to reconstruct the distribution function *via* a cubic spline approximation, the first line of the linear system the matrix of which is given by (4.7) can introduce some numerical errors which can be propagated in the rest of the system; in the numerical experimentations we have drived, the final results are damaged, especially when one observes the mass conservation. Indeed, the finite differences approximation leads to some variations of the mass conservation which is an inconvenient for the long time behaviour of the numerical solution. On the contrary, the approximation of the derivatives using cubic splines enables us to obtain a robust code with a relatively small number of discrete points.

By constructing an approximation of the derivatives using the cubic spline coefficients and the equalities (4.4) and (4.5), we manage to overcome this kind of error (see figure 6 in section 6). Moreover, the final global reconstructed numerical solution is consistent with a numerical solution which is computed through a sequential resolution. Let us explain it in the following in the one dimensional case (the multi-dimensional case can be deduced straightforwardly). First, relations (4.5) and (4.4) enable us to write

$$\begin{aligned} s'(x_i) &= \frac{1}{2h}(\eta_{i+1} - \eta_{i-1}), \\ &= \frac{1}{2h}\left(\frac{3}{2}f_{i+1} - \frac{1}{4}\eta_i - \frac{1}{4}\eta_{i+2} - \frac{3}{2}f_{i-1} + \frac{1}{4}\eta_{i-2} + \frac{1}{4}\eta_i\right), \\ &= \frac{3}{4h}(f_{i+1} - f_{i-1}) + \frac{1}{8h}(\eta_{i-2} - \eta_i + \eta_i - \eta_{i+2}), \end{aligned} \tag{4.11}$$

to obtain the following equality

$$s'(x_i) = \frac{3}{4h}(f_{i+1} - f_{i-1}) - \frac{1}{4}(s'(x_{i-1}) + s'(x_{i+1})). \tag{4.12}$$

If we inject (4.11) in (4.12) to compute $s'(x_{i\pm1})$, we obtain

$$\begin{aligned} s'(x_i) &= \frac{3}{4h}(f_{i+1}-f_{i-1}) - \frac{1}{4}\left(\frac{3}{4h}(f_{i+2}-f_{i-2}) + \frac{1}{8h}(\eta_{i-3}-\eta_{i+1}+\eta_{i-1}-\eta_{i+3})\right), \\ &= \frac{3}{4h}(f_{i+1}-f_{i-1}) - \frac{1}{4}\left(\frac{3}{4h}(f_{i+2}-f_{i-2})\right) - \frac{1}{16}(2s'(x_i)+s'(x_{i-2})+s'(x_{i+2})), \end{aligned}$$

then we get another expression for the derivative of s

$$s'(x_i) = \frac{6}{7h}(f_{i+1}-f_{i-1}) - \frac{3}{14h}(f_{i+2}-f_{i-2}) + \frac{1}{14}(s'(x_{i+2})-s'(x_{i-2})). \tag{4.13}$$

Thanks to (4.13), the evaluation of $s'(x_{i+2})$ and $s'(x_{i-2})$ leads to the following new approximation of $s'(x_i)$

$$\begin{aligned} \alpha s'(x_i) &= \frac{6}{7h}(f_{i+1}-f_{i-1}) - \frac{3}{14h}(f_{i+2}-f_{i-2}) + \frac{6}{98h}(f_{i+3}-f_{i+1}+f_{i-1}-f_{i-3}) \\ &\quad -\frac{3}{14^2h}(f_{i+4}-f_{i-4}) + \frac{1}{14^2}(s'(x_{i+4})-s'(x_{i-4})). \end{aligned} \tag{4.14}$$

with $\alpha = (1-2/14^2)$. A last iteration allows us to obtain a high order approximation of the derivative of s

$$\alpha s'(x_i) = \sum_{j=-8}^{j=8} \omega_j f_{i+j} + \frac{1}{\alpha 14^2}(s'(x_{i+8})+s'(x_{i-8})) + \frac{2}{\alpha 14^2}s'(x_i),$$

to obtain

$$\left(1 - \frac{2}{14^2} - \frac{2}{(1-2/14^2)14^2}\right) s'(x_i) = \sum_{j=-8}^{j=8} \omega_j f_{i+j} + \frac{1}{\alpha 14^2}(s'(x_{i+8})+s'(x_{i-8})), \tag{4.15}$$

where the derivatives $s'(x_{i+8})$ and $s'(x_{i-8})$ are evaluated thanks to a finite differences approximation of order 4. For example, $s'(x_{i+8})$ is approximated by

$$s'(x_{i+8}) = \left(-f(x_{i+10}) + 8f(x_{i+9}) - 8f(x_{i+7}) + f(x_{i+6})\right)/(12h)$$

where h is the step discretization. Even if this choice may introduce some noise in the final evaluation of $s'(x_i)$, the commited errors remains quite negligeable since the use of the finite differences is now sufficiently far from the junction points. The final approximation of $s'(x_i)$ then reads

$$\begin{aligned} s'(x_i) &= \sum_{j=-10}^{j=10} \tilde{\omega}_j f_{i+j}, \\ &= \sum_{j=-10}^{j=-1} \tilde{\omega}_j^- f_{i+j} + \sum_{j=1}^{j=10} \tilde{\omega}_j^+ f_{i+j}, \end{aligned} \tag{4.16}$$

since the coefficient $\tilde{\omega}_0$ is null here. Let us note that ω_j^- and ω_j^+ are computed once for all. Other iterations can also be done, but formula (4.16) gives satisfying results.

The coefficients $\tilde{\omega}_j$, $j = -10, ..., 10$ are summarized in the table 1. The $\tilde{\omega}_j^+$ coefficients are given from the following relation: $\tilde{\omega}_j^+ = -\tilde{\omega}_j^-$.

$\tilde{\omega}^-_{-10}$	$\tilde{\omega}^-_{-9}$	$\tilde{\omega}^-_{-8}$	$\tilde{\omega}^-_{-7}$	$\tilde{\omega}^-_{-6}$
0.2214309755E-5	-1.771447804E-5	7.971515119E-5	-3.011461267E-4	1.113797807E-3
$\tilde{\omega}^-_{-5}$	$\tilde{\omega}^-_{-4}$	$\tilde{\omega}^-_{-3}$	$\tilde{\omega}^-_{-2}$	$\tilde{\omega}^-_{-1}$
-4.145187862E-3	0.01546473933	-0.05771376946	0.2153903385	-0.8038475846

Table 1: *Coefficients for the approximation of the derivatives.*

5 Parallelization of computations

In order to perform a parallelization of the interpolation step, data and computation have to be distributed onto processors. A classical technique of domain decomposition is used here to split the phase space in subdomains. Thus, a single processor works on local data and shares information located on the borders of its subdomain with adjacent processors. The set of values exchanged with the eight processors in the neighborhood of a given processor is named the *ghost* area. This area is needed because each processor has to know information belonging to others, in order to build the right hand side matrix of section 4.2. Values of function f and some kind of derivatives are stored in the ghost zone in order to manage this step. From a parallel performance point of view, the number of values transmitted between processors must be minimal. So the ghost zone should be chosen as small as possible.

Indeed, on the patch, only points (x_i, y_j) for $i = 0, ..., N_x - 1$ and $j = 0, ..., N_y - 1$ are known, and the interpolation step requires the knowledge of values on the patches borders; moreover we have to evaluate the derivative in (x_0, y_j) and (x_{N_x}, y_j), for all j, (x_i, y_0) and (x_i, y_{N_y}) for all i, which requires (see the previous section) a linear combination of 21 points.

The knowledge of these points enables us to build and solve the LU systems, and to interpolate on $[x_0, x_{N_x}] \times [y_0, y_{N_y}]$. But we have to take into account the advected points that come out the targeted patch. As mentionned in the introduction, we impose a restriction on the time step to enforce the displacement to be lower to the cell size. Hence, the interpolation area becomes $[x_0 - \Delta x, x_{N_x} + \Delta x] \times [y_0 - \Delta y, y_{N_y} + \Delta y]$ (where Δx and Δy denote the steps discretization) and additional cubic spline coefficients have to be computed.

To that purpose, the resolution of the linear systems described in section 4.2 takes into account the following augmented right hand side matrix (the derivatives are approximated thanks to (4.16))

$$\begin{pmatrix} f(-1,-1) & \partial_y f(-1,0) & f(-1,0) & \cdots & \partial_y f(-1,N_y) & f(-1,N_y+1) \\ \partial_x f(0,-1) & \partial^2_{xy} f(0,0) & \partial_x f(0,0) & \cdots & \partial^2_{xy} f(0,Ny) & \partial_x f(0,N_y+1) \\ f(0,-1) & \partial_y f(0,0) & f(0,0) & \cdots & \partial_y f(0,N_y) & f(0,N_y+1) \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ f(N_x,-1) & \partial_y f(N_x,0) & f(N_x,0) & \cdots & \partial_y f(N_x,Ny) & f(N_x,N_y+1) \\ \partial_x f(N_x,-1) & \partial^2_{xy} f(N_x,0) & \partial_x f(N_x,0) & \cdots & \partial^2_{xy} f(N_x,N_y) & \partial_x f(N_x,N_y+1) \\ f(N_x+1,-1) & \partial_y f(N_x+1,0) & f(N_x+1,0) & \cdots & \partial_y f(N_x+1,N_y) & f(N_x+1,N_y+1) \end{pmatrix}. \tag{5.17}$$

The resolution in the x-direction is done for all j, which gives us the temporary spline

coefficients $\gamma_\nu(y_j)$, $\nu = -1, .., N_x$ and $j = -2, ..., N_y + 1$. The coefficients corresponding to $\nu = -2$ and $\nu = N_x + 1$ are deduced from (4.4) applied for $i = -1$ and $i = N_x$.

In the same way, the resolution in the y-direction is done for all $\nu = -2, ..., N_x + 1$, and gives the coefficients $\eta_{\nu,\beta}$ for $\beta = -1, ..., N_y + 1$; the boundary values $\eta_{\nu,-2}$ and η_{ν,N_y+1} are obtained from (4.4)

The target processor has to gather all points needed to compose the matrix (5.17). To that purpose, as the values of the distribution function are known at (x_i, y_j) for $i = 0, ..., N_x - 1$ and $j = 0, ..., N_y - 1$. The local ghost zone received from others processors

- $f(-1, j)$ for $j = 0, ..., N_y - 1$,
- $f(i, -1)$ for $i = 0, ..., N_x - 1$,
- $f(N_x : N_x + 1, j)$ for $j = 0, ..., N_y - 1$,
- $f(i, N_y : N_y + 1)$ for $i = 0, ..., N_x - 1$,
- $f(-1, -1)$,
- $f(-1, N_y : N_y + 1)$,
- $f(N_x : N_x + 1, -1)$,
- $f(N_x : N_x + 1, N_y : N_y + 1)$,
- some ponderative summations of 10 points which are computed on the neighboring processors to evaluate all derivatives.

6 Numerical simulations

In this section, some numerical results obtained with the methodology we exposed above are presented. We compare sequential and parallel simulations for two problems that occur in plasma physics: the Landau damping and the two stream instability test cases.

6.1 Landau damping

In this section, we propose to validate the method against the standard test case of the Landau damping. We study the evolution of electrons whose distribution function is initially an isotropic Maxwellian of density and temperature equal to one. The plasma is then perturbed and a periodic damped wave is then created. The purpose of this numerical test is the study of the evolution of this damped wave. To achieve this task, we consider the distribution function of electrons which is solution to the Vlasov-Poisson system (2.1)-(2.2).

Linear Landau damping

The initial condition associated to the scaled Vlasov-Poisson equation (2.1)-(2.2) has the following form

$$f_0(x, v) = \frac{1}{\sqrt{2\pi}} \exp(-v^2/2)(1 + \alpha \cos(kx)), \quad (x, v) \in [0, 2\pi/k] \times \mathbb{R}, \tag{6.18}$$

where k is the wave number and $\alpha = 0.001$ is the amplitude of the perturbation, so that we consider linear regimes here. To capture the Landau damping, the size of the velocity domain

must be chosen greater than the phase velocity v_ϕ. The phase velocity is equal to ω/k, where ω is the frequency related to k and is approximated by

$$\omega^2 \simeq 1 + 3k^2. \tag{6.19}$$

Then, we set $v_{max} = 6$ where the velocity domain extend from $-v_{max}$ to v_{max}. We use a number of cells $N_v = 32$ or 64 for the velocity domain and $N_x = 32$ or 64 in the spatial direction. The boundary conditions for the distribution function are periodic in the space variable and compact in the velocity direction. Finally, the wave number is taken equal to $k = 0.3$ or 0.5. The final time is $T = 60\,\omega_p^{-1}$, with ω_p the plasma frequency.

In this test, we are interested in the evolution of the square root of the electric energy approximated by

$$\mathcal{E}_h(t) = \left(\sum_i E_i^2(t) \Delta x \right)^{1/2}, \tag{6.20}$$

where Δx is the space step. According of the Landau theory, the amplitude of $\mathcal{E}_h(t)$ is expected to be exponentially decreasing with a frequency ω.

On figure 1, we represent the evolution of $\log(\mathcal{E}_h(t))$ in the sequential case; two different values of the wave number are shown: $k = 0.3$ and $k = 0.5$. The number of cells is equal to $N_x = N_v = 32$ or $N_x = N_v = 64$. We observe that $\mathcal{E}_h(t)$ is always exponentially decreasing, and the damping rate becomes larger when k increases, as predicted by the Landau theory. The damping rate obtained are given by $\gamma = 0.0127$ for $k = 0.3$, and $\gamma = 0.154$ for $k = 0.5$, which are very similar to the predicted values available in the literature (see [20, 2, 10]).

We also observe the "recurrence effect" (see [21]): on figure 1 (b) for example, with N_x=N_v=32 points, the amplitude of the electric energy increases at $t \simeq 31\ \omega_p^{-1}$ which appears in a quite good agreement with the theoritical time $T_R \approx 2\pi/(k\Delta v)$. This time is predicted from the free streaming case (Δv is the velocity step). This phenomenon is pushed away by taking more points in velocity (see figure 1 with $N_x = N_v = 64$ points).

On figure 2, the same results are shown in the parallel case. The phase space domain is decomposed into 4 patches of the same size 32×32 points, so that the global domain involves 64×64 points; moreover, Hermite boundary conditions are imposed at the boundary of each patch using the approximation (4.16). We can observe that the results are very similar to the sequential case since the damping rate are the same and the recurrence effect occurs at the correct time.

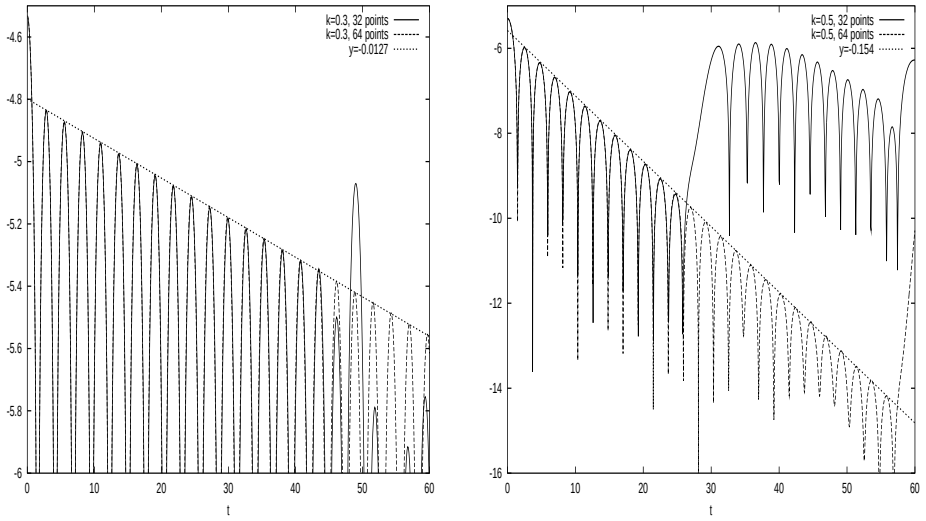
Strong Landau damping

The initial datum is the following

$$f(0,x,v) = \frac{1}{\sqrt{2\pi}} \exp(-v^2/2)(1 + \alpha \cos(kx)), \quad (x,v) \in [0, 2\pi/k] \times \mathbb{R},$$

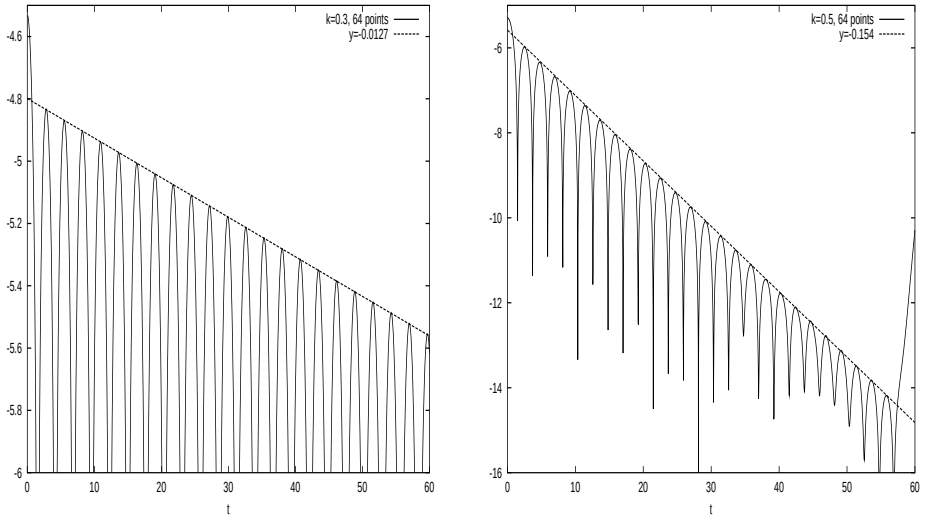
where the amplitude of the initial perturbation of the density is taken equal to $\alpha = 0.5$. Moreover, the wave number $k = 0.5$, whereas v_{max} is equal to 6.5 in order to take into account nonlinear effects. The number of cells will be equal to $N_x = N_v = 128$.

We are also interested in the evolution of $\log(\mathcal{E}_h(t))$ (where $\mathcal{E}_h(t)$ is given by (6.20)) as a function of the time. The linear theory of the previous test can not be applied in this case since



(a) (b)

Figure 1: *Electric energy as a function of time for the linear Landau damping in the sequential case.* (*a*) k=0.3 *with* N_x=N_v=32 *points and* N_x=N_v=64 *points.* (*b*) k=0.5 *with* N_x=N_v=32 *points and* N_x=N_v=64 *points.*



(a) (b)

Figure 2: *Electric energy as a function of time for the linear Landau damping in the parallel case.* (*a*) k=0.3 *with* N_x=N_v=64 *points.* (*b*) k=0.5 *with* N_x=N_v=64 *points.*

the nonlinear effects have to be taken into account. Nevertheless, this test has been studied by several authors and comparisons can be made with numerical results available in the literature (see [2, 10, 11, 21]).

On figure 3, we compare the evolution of the logarithm of electric energy between the sequential case and the parallel case. We notice that the amplitude of the electric energy is first exponentially decreasing in time, and oscillates around a constant for larger times for the two simulations. As in the linear case, the sequential and the parallel case present quite good results compared to results available in the literature. Moreover, until large times, the two cases are very similar, and only at $t \simeq 50\,\omega_p^{-1}$, the two results become slightly different. On figure 3, we also plot a reference solution computed using 512×1024 points.

Moreover, we can see on figure 4 the projection of the distribution function as a function of the velocity. We plot the following quantity

$$F(v) = \int_0^{2\pi/k} f(x,v)dx,$$

as a function of the velocity for different times, in the parallel case. We observe that particles whose kinetic energy is smaller than the potential energy are trapped by electrostatic waves around the phase velocity $v_\phi = \omega/k$, where small bumps appear preceded by small holes. Only the parallel case is presented, since the sequential results are too close to discuss differences. We can first notice that the projection is symmetric with respect to the origin. This a consequence of centered approximation of the derivatives (see section 4.3). Indeed, uncentered approximation using finite differences formulas introduces some unsymmetry in the distribution function leading to a loss of accuracy. Moreover, we observe a good junction (similar to the sequential case) of the global reconstructed distribution function in $v = 0$ where is located a decomposition point of our parallelization; the patches are joined to each other by the Hermite boundary conditions which preserves an accurate global numerical approximation, even when the distribution function is very unstable (see on figure 4 at $t = 35\,\omega_p^{-1}$ for example).

Moreover, to emphasize the influence of the approximation of the derivative on the results, we plot on figures 5 and 6 the evolution in time of the total relative mass. Comparisons between the parallel and sequential case are presented in two different contexts; on figure 5, all the derivatives are approximated through the following 4 order finite differences operator

$$s'(x_i) \simeq \frac{-f(x_{i-2}) + 8f(x_{i-1}) - 8f(x_{i+1}) + f(x_{i+2})}{12h},$$

where h is the step corresponding to the derivative direction. On figure 6, the derivatives are replaced by the formula (4.16). We can observe that the finite differences approximation is not well suited for the parallel implementation since the total mass presents some important oscillations from $t \simeq 20\ \omega_p^{-1}$, these fluctuations becoming greater when time increases. On the contrary, the use of cubic spline approximation with 21 points leads to a mass conservation which is very similar to the mass conservation occuring in the sequential case. Let us remark that the use of finite differences or cubic spline approximation does not affect the mass conservation in the sequential case.

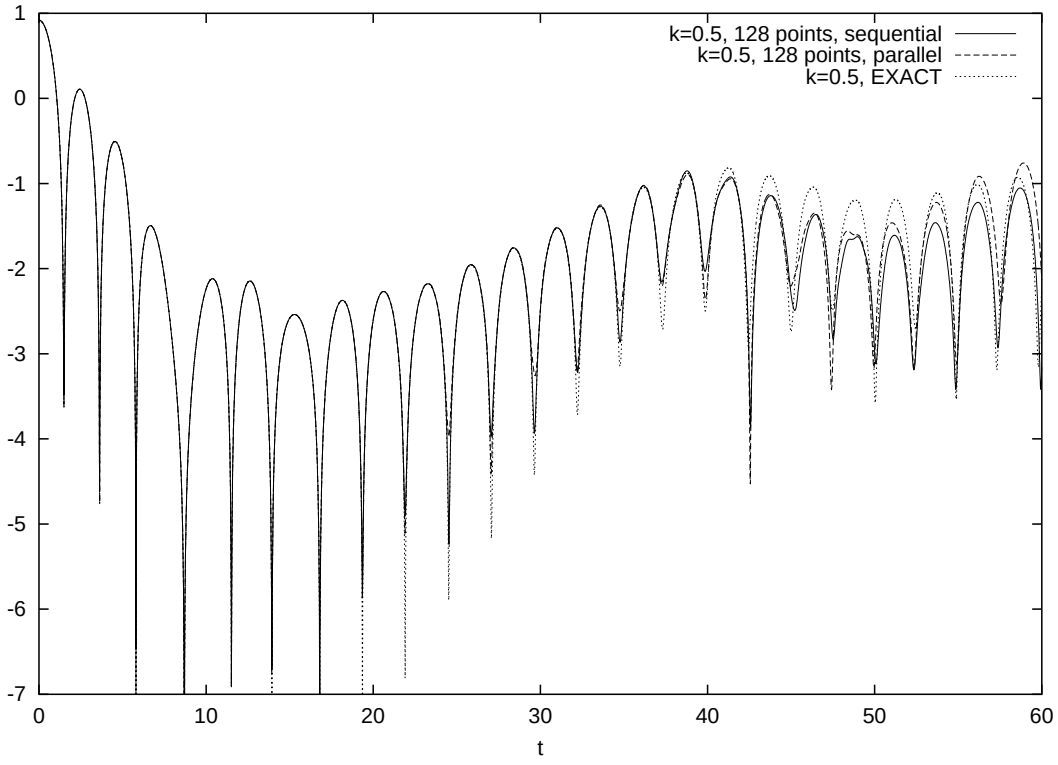


Figure 3: *Electric energy as a function of time for the strong Landau damping. Comparison between the sequential and the parallel case.* k=0.5 *and* N_x=N_v=128. *An almost "exact" solution (*512×1024 *points) is plotted for comparison.*

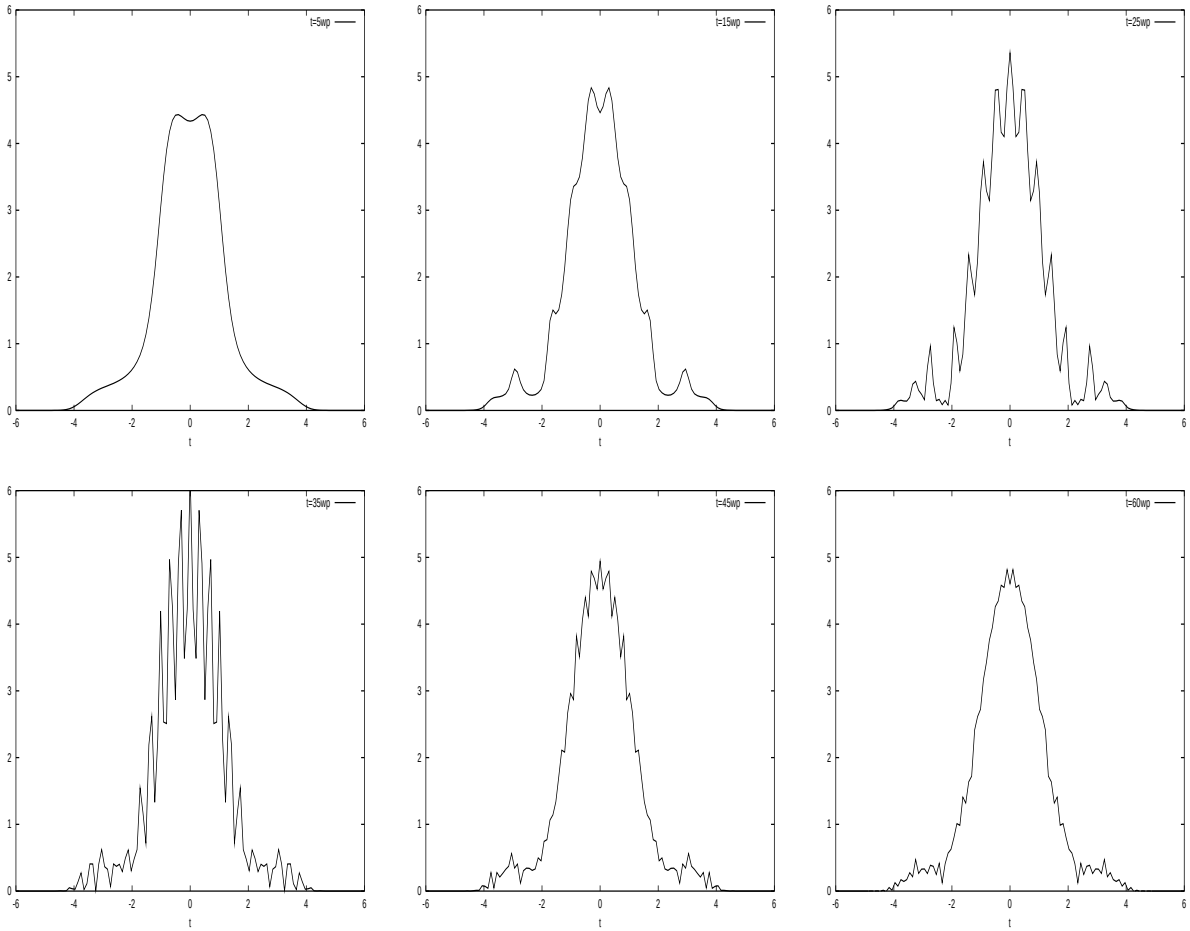


Figure 4: *Time development of the spatially integrated distribution function for the strong Landau damping. Parallel case.* $N_x = N_v = 128$.

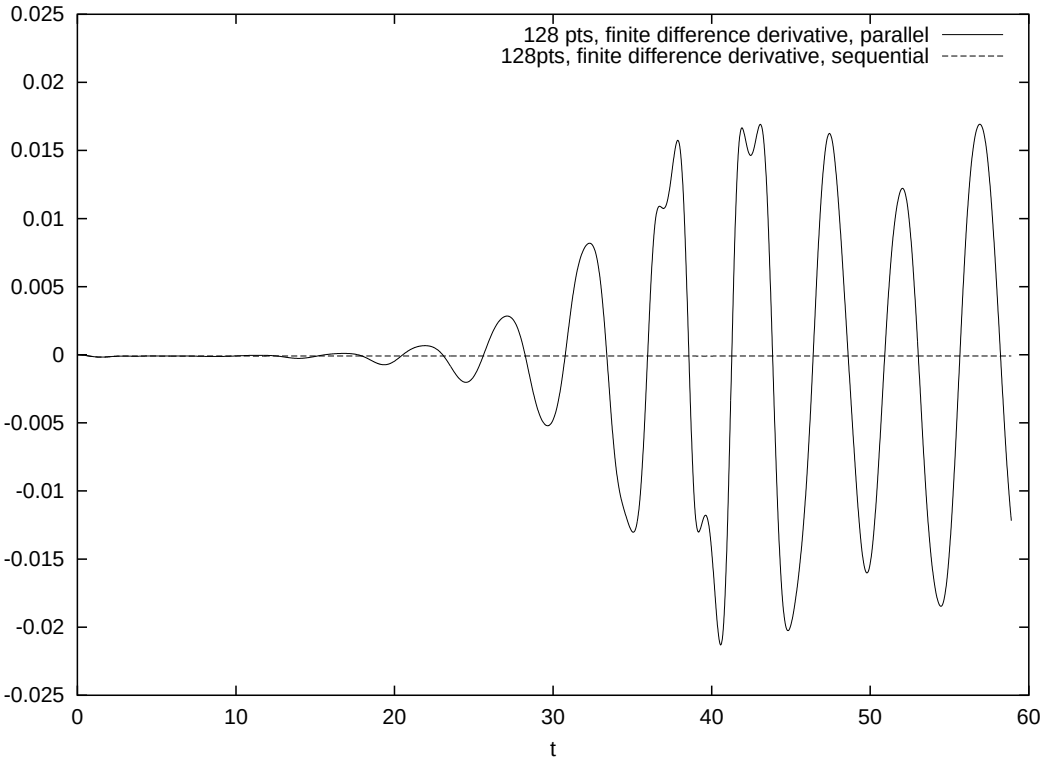


Figure 5: *Comparison between the sequential and parallel case for the total relative mass conservation as a function of time, for the strong Landau damping. The finite differences approximation of order* 4 *is used.*

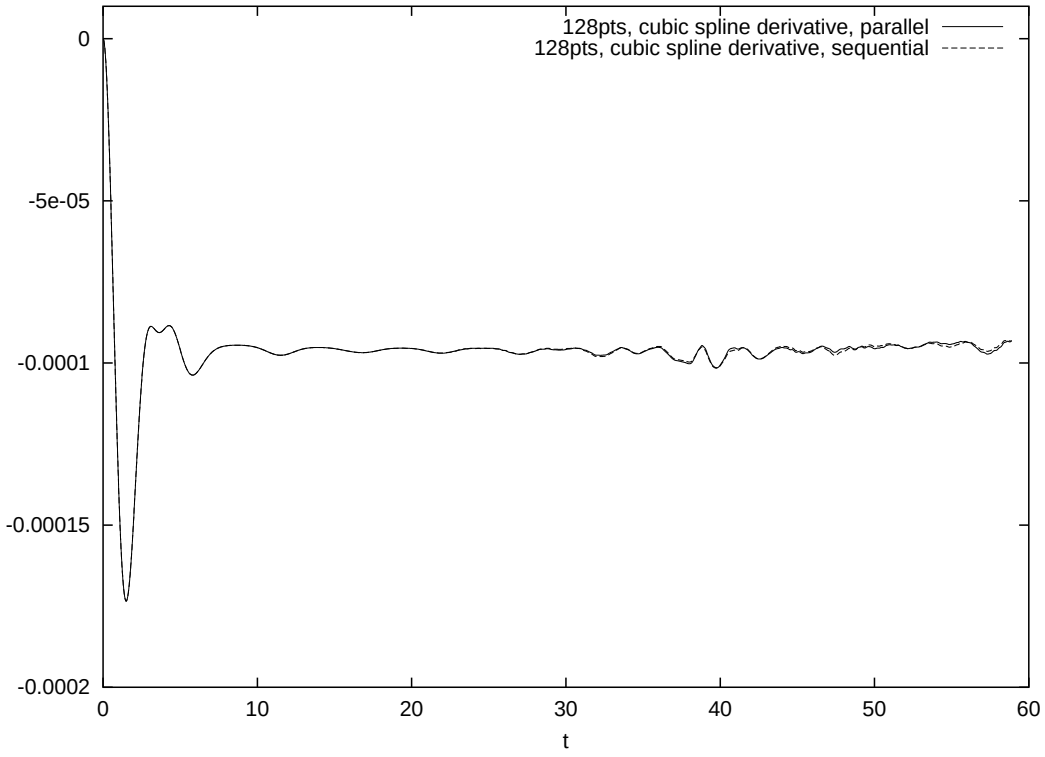


Figure 6: *Comparison between the sequential and parallel case for the total relative mass conservation as a function of time, for the strong Landau damping. The cubic spline approximation with* 21 *points is used.*

6.2 Two stream instability

In this subsection, we solve the Vlasov-Poisson equation considering the following initial datum

$$f(0,x,v) = \frac{1}{\sqrt{2\pi}} v^2 \exp(-v^2/2)(1+\alpha\cos(kx)), \quad (x,v) \in [0, 2\pi/k] \times \mathbb{R},$$

where the amplitude of the perturbation is $\alpha = 0.05$, the number of wave is $k = 0.5$, and v_{max} is taken equal to 9. The number of mesh points is $N_x = 128$ in space and to $N_v = 128$ in velocity to get a good accuracy. The final time is $T = 500\ \omega_p^{-1}$.

In this model, two streams of charged particles encounter each other in the physical space with opposite velocities (see [3] for more details). When evolving in time, a perturbation occurs and grows rapidly. In the phase space, this perturbation corresponds to a vortex creation at the center of the computational domain.

On figures 7 and 8, we plot the time evolution of the distribution function in the phase space in the parallel case (4 patches equally decompose the phase space domain). We observe at time $t \simeq 10\ \omega_p^{-1}$ (where ω_p is the plasma frequency) vortex creation which is associated to trapped particles. From $t \simeq 10\ \omega_p^{-1}$ until $t \simeq 20\ \omega_p^{-1}$, the instability grows rapidly and a hole appears. After $t \simeq 20\ \omega_p^{-1}$, the trapped particles oscillate in the electrostatic potential and the vortex rotates periodically. These remarks are in good agreement with the results available in [2, 11].

This simulation is quite interesting since the hole has to stay at the middle of the computational domain during all the simulation; a displacement of this centered vortex can occur due to a failing numerical resolution for large times. Here again, the good mass conservation depends on the derivative approximation, as explained previously; in particular, we remarked that finite differences approximations (even centered ones) lead to uncorrected results for large times (the hole comes out early). Moreover, the inherent precision of the cubic spline interpolation allows to follow thin filaments developed by the solution; even if the methodology which enables the parallelization is slightly different from the sequential version, we observe that the parallelization does not affect the precision due to the spline interpolation.

Finally, several tests have been implemented to evaluate the influence of the number of processors on the numerical results and on the time simulation. The number of points in each patch has to be important enough (for small patches, the overhead in computations to estimate the derivatives becomes too large). The performance of the parallel algorithm is summarized in the tables 2 and 3, for the two stream instability implemented with $N_x = N_v = 512$ points. The experiments were conducted on two parallel machines: a cluster[1] of 11 IBM nodes (16-way Power5 processors at 1.9 Ghz), a shared memory SGI machine[2] of 512 processors (Origin 3800 with 500 Mhz processors). Let us mention that the results presented in tables 2 and 3 do not take into account the diagnostics computations.

Our speedup is quite good since it takes into account the numerical solution of the Poisson equation. Indeed, after each two dimensional advection in the phase space, the Poisson equation has to be globally solved (corrector part of the algorithm, see section 3). This step is time consuming when the number of processor increases. Nevertheless, our methodology focuses on

[1]The IBM machine belongs to the M3PEC group, Bordeaux 1 University

[2]The SGI machine is located at Montpellier, France, at the computing center CINES `http://www.cines.fr`

Numbers of processors	1	4	8	16	32	64
Time (in s.)	40	9.4	4.7	2.4	1.2	0.7
Speedup	1	4.25	8.51	16.6	33.3	57.14

Table 2: *Speedup for the two stream instability on a shared memory SGI machine. The results corresponds to* 512×512 *points in the phase space; the simulation is stopped after* 300 *iterations.*

Numbers of processors	1	4	8	16	32	64
Time (in s.)	89.2	19.4	9.7	4.9	2.6	1.6
Speedup	1	4.59	9.19	18.2	34.3	55.75

Table 3: *Speedup for the two stream instability on a cluster of* 11 *IBM nodes. The results corresponds to* 512×512 *points in the phase space; the simulation is stopped after* 300 *iterations.*

the interpolation step. In particular, we are confident that higher dimensional problems will improve the speedup since we will perform communication-computation overlap easily in the four dimensional cases. Let us remark that the decomposition of the global domain does not affect the numerical results neither the length of the simulation.

7 Conclusion

In this paper, we introduced a local semi-Lagrangian method which has been applied to the Vlasov-Poisson equation. The methodology seems to present a good behaviour when it is tested on standard plasma configurations. Indeed, the numerical results show the good efficiency of our code for the two dimensional case, and its good scalability shown with up to 64 processors.

Using the method we introduced here, we developed a fortran 90 module to locally interpolate any advected function on a two dimensional domain. This module will then enable us to deal with many problems occuring in plasma physics using the semi-Lagrangian methodology. Future extensions will be devoted to the paraxial Vlasov model. Moreover, coupling this methodology with the moving grid strategy can also be envisaged. Finally, a new algorithm that overcomes the restriction on the time step has to be developed later on.

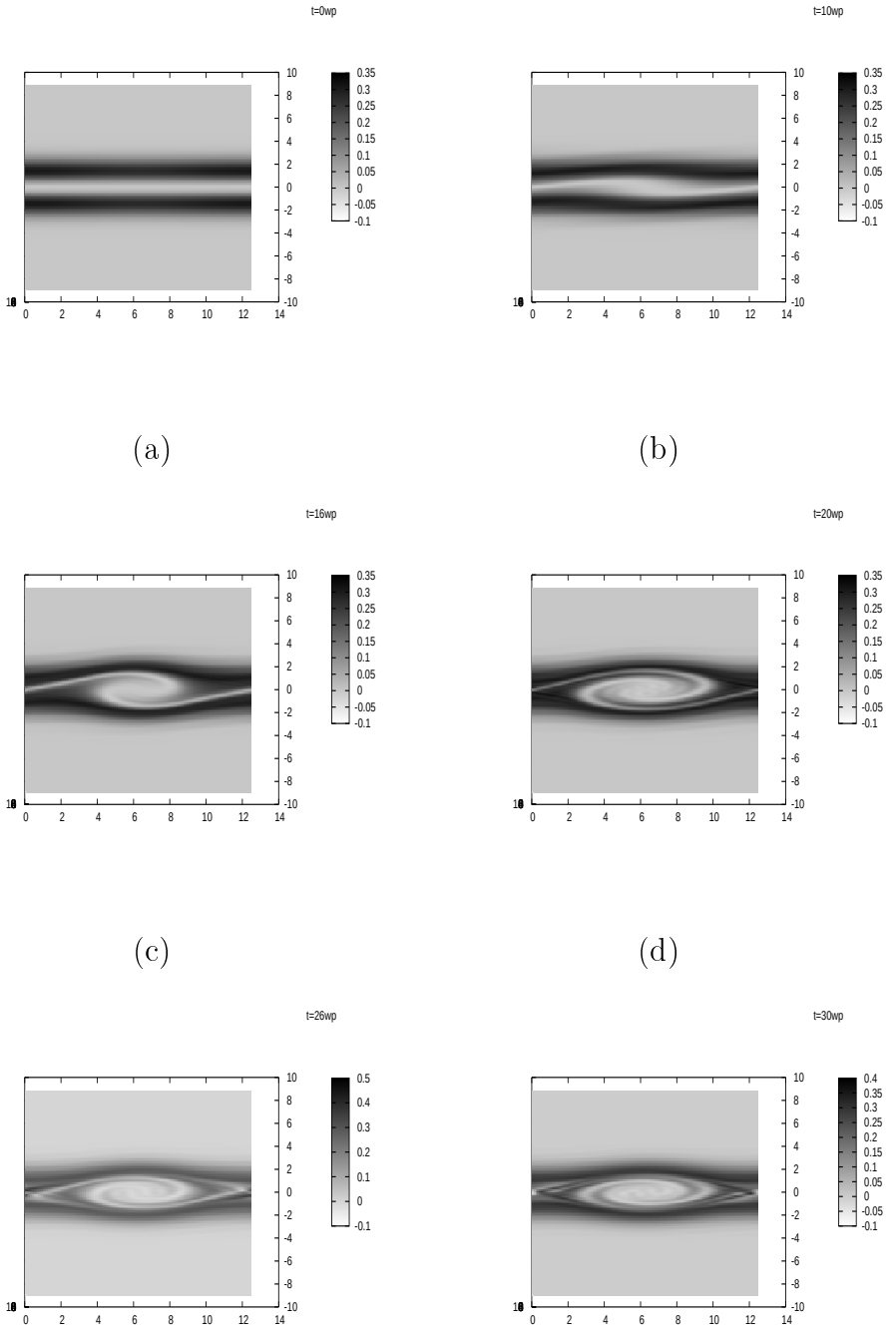
References

[1] R. Bermejo, *Analysis of an algorithm for the Galerkin-characteristic method*, Numer. Math., **60**, pp. 163-194, (1991).

[2] N. Besse, E. Sonnendrücker, *Semi-Lagrangian schemes for the Vlasov equation on an unstructured mesh of phase space*, J. Comput. Phys., **191**, pp. 341-376, (2003).

[3] C.K. Birdsall, A.B. Langdon, *Plasma Physics via Computer Simulation*, Institute of Physics Publishing, Bristol and Philadelphia, 1991.

[4] F. Bouchut, F. Golse, M. Pulvirenti, *Kinetic Equations and Asymptotic Theory*, Series in Applied Mathematics, P.G. Ciarlet and P.L. Lions ed., Gauthier-Villars, Paris, 2000.

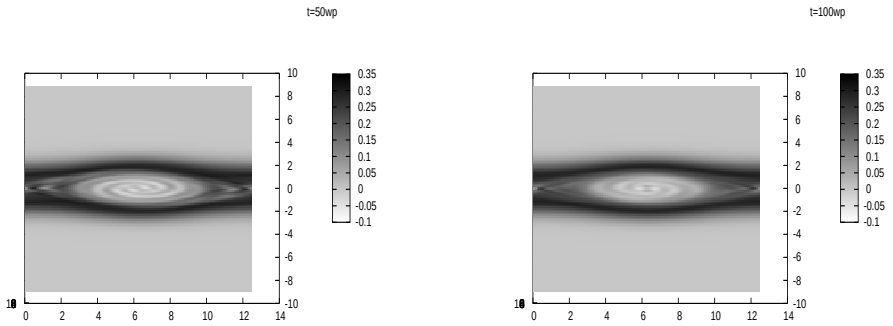


(a) (b)

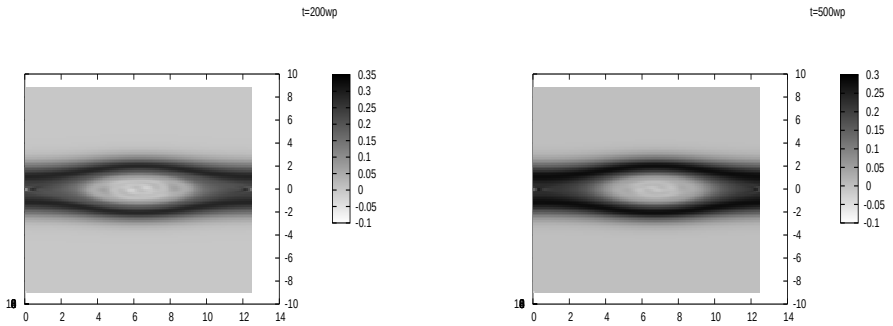
(c) (d)

(e) (f)

Figure 7: *Evolution of the distribution function f in the phase space, with $N_x = 128$ and $N_v = 128$ in the parallel case (4 processors are used), for the two stream instability.* (a) $t = 0\,\omega_p^{-1}$, (b) $t = 10\,\omega_p^{-1}$, (c) $t = 16\,\omega_p^{-1}$, (d) $t = 20\,\omega_p^{-1}$, (e) $t = 26\,\omega_p^{-1}$, (f) $t = 30\,\omega_p^{-1}$.



(g) (h)



(i) (j)

Figure 8: *Evolution of the distribution function f in the phase space, with $N_x = 128$ and $N_v = 128$ in the parallel case (4 processors are used), for the two stream instability.* (g) $t = 50\,\omega_p^{-1}$, (h) $t = 100\,\omega_p^{-1}$, (i) $t = 200\,\omega_p^{-1}$, (j) $t = 500\,\omega_p^{-1}$.

[5] C. DeBoor, *A practical guide to splines*, Springer-Verlag New-York, 1978.

[6] M. Campos-Pinto, M. Merhenberger, *Adaptive numerical resolution of the Vlasov equation*, Numerical Methods for Hyperbolic and Kinetic Equation, (2004).

[7] C.Z. Cheng, G. Knorr, *The integration of the Vlasov equation in configuration space*, J. Comput. Phys., **22**, p. 330, (1976).

[8] O. Coulaud, E. Sonnendrücker, E. Dillon, P. Bertrand, A. Ghizzo, *Parallelization of semi-Lagrangian Vlasov codes*, J. Plasma Phys., **61**, pp. 435-448, (1999).

[9] M.R. Feix, P. Bertrand, A. Ghizzo, Series on Advances in Mathematics for Applied Science: Kinetic Theory and Computing, B. Perthame ed., 1994.

[10] F. Filbet, E. Sonnendrücker, P. Bertrand, *Conservative numerical schemes for the Vlasov equation*, J. Comput. Phys., **172**, pp. 166-187, (2001).

[11] F. Filbet, E. Sonnendrücker, *Comparison of Eulerian Vlasov solvers*, Comput. Phys. Comm., **151**, pp. 247-266, (2003).

[12] F. Filbet, E. Violard, *Parallelization of a Vlasov Solver by Communication Overlapping*, Proceedings PDPTA 2002, (2002).

[13] R.T. Glassey, *The Cauchy Problem in Kinetic Theory*, SIAM, Philadelphia, PA, 1996.

[14] A. Ghizzo, P. Bertrand, M.L. Begue, T.W. Johnston, M. Shoucri, *A Hilbert-Vlasov code for the study of high-frequency plasma beatwave accelerator*, IEEE Transaction on Plasma Science, **24**, p. 370, (1996).

[15] A. Ghizzo, P. Bertrand, M. Shoucri, T.W. Johnston, E. Filjakow, M. R. Feix, *A Vlasov code for the numerical simulation of stimulated Raman scattering*, J. Comput. Phys., **90**, pp. 431-457, (1990).

[16] V. Grandgirard, M. Brunetti, P. Bertrand, N. Besse, X. Garbet, P. Ghendrih, G. Manfredi, Y. Sarrazin, O. Sauter, E. Sonnendrücker, J. Vaclavik, L. Villard, *A drift-kinetic semi-Lagrangian 4D code for ion turbulence simulation*, J. Comput. Phys., to appear.

[17] M. Gutnic, M. Haefele, I. Paun, E. Sonnendrücker, *Vlasov simulation on an adaptive phase space grid*, Comput. Phys. Comm., **164**, pp. 214-219, (2004).

[18] G. Hammerlin, K.H. Hoffmann, Numerical Mathematics, Springer-Verlag New-York, 1991.

[19] C.C. Kim, S.E. Parker, *Massively parallel three-dimensional toroïdal gyrokinetic flux-tube turbulence simulation*, J. Comput. Phys., **161**, pp. 589-604, (2000).

[20] C.J. McKinstrie, R.E. Giacone, E.A. Startsev, *Accurate formulas for the Landau damping rates of electrostatic waves*, Phys. of Plasmas, **6**, pp. 463-466, (1999).

[21] G. Manfredi, *Long time behaviour of strong linear Landau damping*, Phys. Rev. Letters, **79**, (1997).

[22] M. Shoucri, G. Knorr, *Numerical integration of the Vlasov equation*, J. Comput. Phys., **14**, pp. 84-92, (1974).

[23] E. Sonnendrücker, F. Filbet, A. Friedman, E. Oudet, J.L. Vay, *Vlasov simulation of beams on a moving phase space grid*, Comput. Phys. Comm., **164**, pp. 390-395, (2004).

[24] E. Sonnendrücker, J. Roche, P. Bertrand, A. Ghizzo, *The semi-Lagrangian method for the numerical resolution of the Vlasov equations*, J. Comput. Phys., **149**, pp. 201-220, (1999).

[25] A. Staniforth, J. Coté, *Semi-Lagrangian integration schemes for atmospheric models – a review*, Monthly Weather Review, **119**, pp. 2206-2223, (1991).