Int. J. Appl. Math. Comput. Sci., *2007, Vol. 17, No. 3, 335–349*
DOI: 10.2478/v10006-007-0028-x

HERMITE SPLINE INTERPOLATION ON PATCHES FOR PARALLELLY SOLVING THE VLASOV-POISSON EQUATION

NICOLAS CROUSEILLES [*], GUILLAUME LATU [**], ERIC SONNENDRÜCKER [***]

[*] INRIA Lorraine, CALVI
e-mail: crouseil@math.u-strasbg.fr

[**] INRIA Futurs, Scalapplix
e-mail: latu@labri.fr

[***] IRMA Strasbourg and INRIA Lorraine, CALVI
e-mail: sonnen@math.u-strasbg.fr

This work is devoted to the numerical simulation of the Vlasov equation using a phase space grid. In contrast to 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. For this purpose, we present a method using patches decomposing the phase domain, each patch being devoted to a processor. Some Hermite boundary conditions allow for the reconstruction of a good approximation of the global solution. Several numerical results demonstrate the accuracy and the good scalability of the method with up to 64 processors. This work is a part of the CALVI project.

Keywords: Vlasov-Poisson equation, semi-Lagrangian method, parallelism

1. Introduction

The Vlasov-Poisson equation describes the evolution of a system of charged particles subjected to the effects of a self-consistent electric field. The unknown f is a distribution function of particles in the phase space which depends on 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 solution of Vlasov-type equations, which depend at least on 6 variables plus time, is most often performed 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 (Birdsall and Langdon, 1991). Even though these methods produce satisfactory results with relatively few particles, for some applications (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 in the particle methods becomes too significant. Consequently, methods which discretize the Vlasov equation on a phase space grid have been proposed (see (Feix *et al.*, 1994; Filbet *et al.*, 2001; Filbet and Sonnendrücker, 2003; Ghizzo *et al.*, 1996; Ghizzo *et al.*, 1990; Shoucri and Knorr, 1974; Sonnendrücker E. *et al.*, 1999) for plasma physics and (Bermejo, 1991; Staniforth and Coté, 1991) for other 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 beginning of the characteristics by some interpolation techniques (e.g., Lagrange, Hermite or cubic splines). We refer the reader to (Sonnendrücker E. *et al.*, 1999) for more details on the semi-Lagrangian method and to (Filbet and Sonnendrücker, 2003) for a comparison of Eulerian solvers dedicated to the Vlasov equation.

Eulerian methods have proven their efficiency on uniform meshes in the 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 the 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 (Campos-Pinto and Merhenberger, 2004; Gutnic *et al.*, 2004; Sonnendrücker *et al.*, 2004). On the other hand, some parallelized versions of codes were implemented to simulate high-dimensional problems (Coulaud *et al.*, 1999; Filbet and Violard, 2002). Generally, the numerical schemes are based on time-splitting schemes 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 (Grandgirard *et al.*, 2006; Kim and Parker, 2000), a global transposition involves a huge amount of data beeing transferred and this may become very inefficient. For these reasons, in this paper we develop 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 though 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 dependence, 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 determination of the cubic spline coefficients *versus* the global solution), our main efforts consist in recovering the global resolution in the best possible way. 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 huge amount of data exchange, but a condition on the time step allows us to control the shifts so that the communications are performed only between adjacent processors. Hence, this communication scheme enables us to obtain competitive results from a scalability point of view. Let us mention that even though a uniform grid is used here, the methodology could be extended to sets of lines which are not equally spaced (e.g., adaptive meshes).

This work contributes to the improvement 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 and, therefore, the code is devoted to be massively parallelized. A time-splitting algorithm allows us to reduce the problem into a sequence of one-dimensional and two-dimensional advections. Our method enables us to accurately solve this advection part using parallel computations (see (Grandgirard *et al.*, 2006) 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. 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 denoting time is given by the dimensionless Vlasov equation

$$\frac{\partial f}{\partial t}+v\cdot\nabla_x f+F(t,x,v)\cdot\nabla_v f=0, \qquad (1)$$

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

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

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

$$F(t,x,v)=E(t,x),\quad \nabla_x\cdot E(t,x)=\rho(t,x)-1, \qquad (2)$$

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 fixed and uniform background. In the sequel, we briefly recall some classical estimates regarding the Vlasov-Poisson system (1), (2). First of all, mass and momentum are preserved in time,

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

Next, multiplying the Vlasov equation (1) by $|v|^2$ and integrating the result by parts, we express the conservation of the energy for the system (1)–(2):

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left[\int_{\mathbb{R}^d\times\mathbb{R}^d} f(t,x,v)|v|^2\,\mathrm{d}x\,\mathrm{d}v+\int_{\mathbb{R}^d}|E(t,x)|^2\,\mathrm{d}x\right]=0,$$

for all $t\in\mathbb{R}^+$. Finally, the Vlasov-Poisson system (1)–(2) conserves the kinetic entropy

$$H(t)=\int_{\mathbb{R}^d\times\mathbb{R}^d} f(t)\,\log(f(t))\,\mathrm{d}x\,\mathrm{d}v = H(0).$$

On the other hand, we can define the characteristic curves of the Vlasov-Poisson system (1)–(2) as the solutions to the following first-order differential system:

$$\begin{cases}\dfrac{\mathrm{d}X}{\mathrm{d}t}(t;s,x,v) = V(t;s,x,v),\\[2ex] \dfrac{\mathrm{d}V}{\mathrm{d}t}(t;s,x,v) = E(t,X(t;s,x,v)),\end{cases}\tag{3}$$

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 time t, of a particle which was at (x,v) at time s. Let $t\to(X(t;s,x,v),V(t;s,x,v))$ be the characteristic curve solution to (3). Then the solution of the Vlasov-Poisson equation (1)–(2) is

$$\begin{aligned} f(t,x,v) &= f\Big(s,X(s;t,x,v),V(s;t,x,v)\Big)\\ &= f_0\Big(X(0;t,x,v),V(0;t,x,v)\Big), \end{aligned}\tag{4}$$

for all $(x,v)\in\mathbb{R}^d\times\mathbb{R}^d, t\ge 0$, where f_0 is a given initial condition of the Vlasov-Poisson equation. This 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 (Bouchut *et al.*, 2000; Glassey, 1996).

3. Semi–Lagrangian Method

In this section, we recall the principles of the Semi-Lagrangian method for the Vlasov-Poisson equation (see (Sonnendrücker E. *et al.*, 1999) for details) in a two-dimensional phase space.

First of all, we introduce a finite set of mesh points $(x_i,v_j), i=0,\dots,N_x$ and $j=0,\dots,N_v$ to discretize the computational domain. Then, given the values 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 notation

$$(X^n,V^n)=(X(t^n;t^n{+}\Delta t,x_i,v_j),V(t^n;t^n{+}\Delta t,x_i,v_j)$$

is used for the solutions of (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 (3). To guarantee second-order accuracy in time, we use the Ampère equation to get an approximation of the electric field at time $t^n+\Delta t$,

$$\frac{\partial E(t,x)}{\partial t}=-J(t,x),\tag{5}$$

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

$$J(t,x)=\int_{\mathbb{R}} f(t,x,v)v\,\mathrm{d}v.$$

Equation (5) is discretized as

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

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

$$J_i^n=\sum_{j=0}^{N_v} f(t^n,x_i,v_j)v_j\Delta v,\tag{6}$$

Δv being the velocity step. Then we solve (3) with the second-order accuracy 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 at the mesh points.

Step 1. Computation of a prediction of $E_i^{n+1/2}$, denoted by $\tilde{E}_i^{n+1/2}$, through 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 (6).

Step 2. Solution of (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 $f(t^n, X^n, V^n)$, update of the distribution function owing to

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

and computation of the density

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

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, Step 2 allows for the computation of 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}, the 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 necessarily belong 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. Local Spline Interpolation

In this section, we present our interpolation technique based on a cubic spline method (DeBoor, 1978; Hammerlin and Hoffmann, 1991; Sonnendrücker E. *et al.*, 1999). 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 in the whole domain, which is inconvenient from a parallelization point of view. Our approach avoids this globality. Indeed, we decompose the phase space into several patches, each being assigned to one processor. The strategy is based on adapted boundary conditions which yield a $\mathcal{C}^1$ reconstructed solution on the global phase space domain even on the patch boundaries.

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

4.1. Local Spline Interpolation in One Dimension.

Consider a function f which is defined on a global domain $[x_{\min}, x_{\max}] \subset \mathbb{R}$. This domain is decomposed into several subdomains denoted generically by $[x_0, x_N]$. Each subdomain will be assigned 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 $f : x \to f(x)$ to 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 is

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

where the cubic B-spline B_ν is defined by

$$B_\nu(x) = \frac{1}{6h^3} \begin{cases} (x - x_{\nu-2})^3 \text{ if } x_{\nu-2} \le x \le x_{\nu-1}, \\ h^3 + 3h^2(x - x_{\nu-1}) + 3h(x - x_{\nu-1})^2 \\ \quad -3(x - x_{\nu-1})^3 \text{ if } x_{\nu-1} \le x \le x_\nu, \\ h^3 + 3h^2(x_{\nu+1} - x) + 3h(x_{\nu+1} - x)^2 \\ \quad -3(x_{\nu+1} - x)^3 \text{ if } x_\nu \le x \le x_{\nu+1}, \\ (x_{\nu+2} - x)^3 \text{ if } x_{\nu+1} \le x \le x_{\nu+2}, \\ 0 \text{ otherwise.} \end{cases} \tag{7}$$

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

$$f(x_i) = s(x_i), \quad \forall i = 0, \dots, N \tag{8}$$

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

$$f'(x_0) \simeq s'(x_0), \quad f'(x_N) \simeq s'(x_N). \tag{9}$$

The only cubic B-splines not vanishing at point x_i are $B_{i\pm1}(x_i) = 1/6$ and $B_i(x_i) = 2/3$. Hence, (8) yields

$$f(x_i) = \frac{1}{6}\,\eta_{i-1} + \frac{2}{3}\,\eta_i + \frac{1}{6}\,\eta_{i+1}, \quad i = 0, \dots, N. \tag{10}$$

On the other hand, we have $B'_{i\pm1}(x_i) = \pm 1/(2h)$, and $B'(x_i) = 0$. Thus, the Hermite boundary conditions (9) become

$$f'(x_0) \simeq s'(x_0) = -1/(2h)\,\eta_{-1} + 1/(2h)\,\eta_1, \tag{11}$$

and

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

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

$$F = [f'(x_0), f(x_0), \dots, f(x_N), f'(x_N)]^T \tag{12}$$

and

$$A = \frac{1}{6}\begin{pmatrix} -3/h & 0 & 3/h & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & & \vdots \\ 0 & 1 & 4 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & -3/h & 0 & 3/h \end{pmatrix}. \tag{13}$$

Bear in mind that f' is a notation, and from a numerical point of view, the derivative of f has to be approximated in a well-defined sense. We will focus on this in the sequel of the paper.

Solving the linear system $\boldsymbol{A\eta = F}$. The matrix A of the linear system has a special structure. Its 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:

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

for each $i = 2, \dots, N$ we have

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

and

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

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

4.2. Local Spline Interpolation in Two Dimensions.

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

$$f(x,y) \simeq s(x,y) = \sum_{\nu=-1}^{N_x+1}\sum_{\beta=-1}^{N_y+1} \eta_{\nu,\beta} B_\eta(x) B_\beta(y). \tag{14}$$

The same notation as that used in the previous section is employed and we have to compute the coefficients $\eta_{\nu\beta}$. For that purpose, we first solve $N_y + 1$ systems

$$s(x, y_j) = \sum_{\nu=-1}^{N_x+1} \gamma_\nu(y_j) B_\nu(x), \quad \forall j = 0, \dots, N_y, \tag{15}$$

where

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

Each of the $N_y + 1$ systems (15) satisfies $N_x + 1$ interpolation conditions (at fixed j)

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

and the Hermite boundary conditions in the x-direction

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

We have

$$\gamma_\nu(y_j) = \sum_{\beta=-1}^{N_y+1} \eta_{\nu,\beta} B_\beta(y_j). \tag{16}$$

We have thus arrived at solving $N_y + 1$ 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 (13) and an (N_x+3) vector similar to (12) evaluated at y_j. Following the procedure used previously (via the LU decomposition), we then obtain the (N_x+3) vector of unknown $\gamma_\nu(y_j)$, for $j = 0, \dots, N_y$,

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

The second step consists in solving a one dimensional problem given by (16) for each $\nu = -1, \dots, N_x + 1$. However, the left-hand side of this system is only known for values of y_j, $j = 0, \dots, N_y$ (i.e., it is a vector of $N_y + 1$ components) whereas the right-hand side is an $(N_y + 3)$ vector. Some boundary conditions are necessary to complete the system. Hermite boundary conditions are imposed for the first and last components of the vector (which correspond to $j = -1$ and $j = N_y$, respectively), that is to say, we have to compute $\gamma'_\nu(y_0)$ and $\gamma'_\nu(y_{N_y})$, $\forall \nu = -1, \dots, N_x + 1$. To achieve this task, we solve two systems: we first differentiate (16) with respect to y, and then evaluate the result at $y_j = y_0$ and $y_j = y_{N_y}$. The Hermite boundary conditions have to be adapted to this particular case. Consequently, we have to solve the following two systems (associated with $j = 0$ and $j = N_y$): $A\gamma'_\nu(y_j) = \partial_y f(x, y_j)$, where A is the matrix (13),

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

and the right-hand side is

$$\begin{aligned}\partial_y f(x, y_j) \;=\; & [\partial_{xy} f(x_0, y_j), \partial_y f(x_0, y_j), \\ & \dots, \partial_y f(x_i, y_j), \dots, \partial_{xy} f(x_{N_x}, y_j)]^T.\end{aligned}$$

Once we have computed $\gamma'_\nu(y_0)$ and $\gamma'_\nu(y_{N_y})$, for all $\nu = -1, \dots, N_x + 1$, we solve the system (16), which here takes the form $A\eta_{\nu,\beta} = \Gamma_{\nu\beta}$. The matrix A is given by (13) and the right-hand side is

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

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

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 (7). Computing $s(X^n, V^n)$ for all mesh points requires $\mathcal{O}(N_x N_y)$ floating points operations.

Remark 1. In summary, we have to solve

- $N_y + 1$ systems of size $(N_x + 3) \times (N_x + 3)$ (to get $\gamma_\nu(y_j), \forall j = 0, \dots, N_y$),
- 2 systems of size $(N_x + 3) \times (N_x + 3)$ (to get $\gamma'_\nu(y_0)$ and $\gamma'_\nu(y_{N_y})$),
- $N_x + 3$ systems of size $(N_y + 3) \times (N_y + 3)$ (to get $\eta_{\nu,\beta}$).

From a computational cost point of view, the solution of a linear system of size N_x using the LU decomposition needs $\mathcal{O}(N_x)$ operations. This procedure has to be performed 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 account of boundary conditions for each local LU decomposition. Indeed, our strategy consists in being as close 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, we want our local determination of cubic spline coefficients to recover in the best way a usual solution on the global domain. For that purpose, some efforts have to be made 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 beginnings and ends 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 Fig. 5).

Various ways have been explored to obtain the derivatives: finite differences of different orders, cubic spline approximation, etc. In order to reconstruct a smooth approximation ($\mathcal{C}^1$ on the global domain, say), the cubic spline approximation has been chosen. Indeed, we remark that even in regions where f is smooth enough, a finite-difference approximation remains quite different from a cubic spline approximation given by (11). Hence, as we want to reconstruct the distribution function via a cubic spline approximation, the first line of the linear system whose matrix is given by (13) can introduce some numerical errors which can be propagated in the rest of the system. In the numerical experiments we have performed, the final results are incorrect, especially when one checks the mass conservation. Indeed, the finite-difference approximation leads to some variations in the mass conservation, which is inconvenient for the long-time behavior 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 as well as Eqns. (10) and (11), we manage to overcome this kind of error (see Fig. 6). Moreover, the final global reconstructed numerical solution is consistent with a numerical solution which is computed through a sequential solution. We shall explain it in the following, in the one-dimensional case (the multi-dimensional case can be easily deduced). First, (11) and (10) enable us to write

$$\begin{aligned} s'(x_i) &= \frac{1}{2h}(\eta_{i+1} - \eta_{i-1}), \\ &= \frac{1}{2h}\Big(\frac{3}{2}f_{i+1} - \frac{1}{4}\eta_i - \frac{1}{4}\eta_{i+2} \\ &\quad -\frac{3}{2}f_{i-1} + \frac{1}{4}\eta_{i-2} + \frac{1}{4}\eta_i\Big) \\ &= \frac{3}{4h}(f_{i+1}-f_{i-1}) + \frac{1}{8h}(\eta_{i-2}-\eta_{i+2}) \end{aligned} \tag{17}$$

to obtain

$$s'(x_i) = \frac{3}{4h}(f_{i+1} - f_{i-1}) - \frac{1}{4}(s'(x_{i-1}) + s'(x_{i+1})). \tag{18}$$

Substituting (17) in (18) to compute $s'(x_{i\pm1})$, we obtain

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

Then we get another expression for the derivative of s:

$$\begin{aligned} s'(x_i) &= \frac{6}{7h}(f_{i+1} - f_{i-1}) - \frac{3}{14h}(f_{i+2} - f_{i-2}) \\ &\quad + \frac{1}{14}(s'(x_{i+2}) - s'(x_{i-2})). \end{aligned} \tag{19}$$

Thanks to (19), the evaluation of $s'(x_{i+2})$ and $s'(x_{i-2})$ leads to the following new approximation of $s'(x_i)$:

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

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

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

Hence

$$\begin{aligned} &\left(1 - \frac{2}{14^2} - \frac{2}{(1 - 2/14^2)14^2}\right) s'(x_i) \\ &\quad = \sum_{j=-8}^{j=8} \omega_j f_{i+j} + \frac{1}{\alpha 14^2}(s'(x_{i+8}) + s'(x_{i-8})), \end{aligned} \tag{20}$$

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

$$\begin{aligned} &s'(x_{i+8}) \\ &\quad = \frac{-f(x_{i+10}) + 8f(x_{i+9}) - 8f(x_{i+7}) + f(x_{i+6})}{12h}, \end{aligned}$$

where h is the stepsize. Even if this choice may introduce some noise in the final evaluation of $s'(x_i)$, the resulting errors remain negligible since the use of the finite differences is now sufficiently far from the junction points. The final approximation of $s'(x_i)$ then becomes

$$\begin{aligned} s'(x_i) &= \sum_{j=-10}^{10} \tilde{\omega}_j f_{i+j}, \\ &= \sum_{j=-10}^{-1} \tilde{\omega}_j^- f_{i+j} + \sum_{j=1}^{10} \tilde{\omega}_j^+ f_{i+j}, \end{aligned} \tag{21}$$

since the coefficient $\tilde{\omega}_0$ is zero here. Note that ω_j^- and ω_j^+ are computed only once. Other iterations can also be performed, but (21) gives satisfying results.

The coefficients $\tilde{\omega}_j,\ j = -10, \ldots, 10$ are summarized in Table 1. The $\tilde{\omega}_j^+$ coefficients are defined as $\tilde{\omega}_j^+ = -\tilde{\omega}_j^-$.

Table 1. Coefficients for the approximation of the derivatives.

$\tilde{\omega}^-_{-10}$	$\tilde{\omega}^-_{-9}$	$\tilde{\omega}^-_{-8}$	$\tilde{\omega}^-_{-7}$	$\tilde{\omega}^-_{-6}$
0.22143E-5	-1.77144E-5	7.9715E-5	-3.01146E-4	1.11379E-3
$\tilde{\omega}^-_{-5}$	$\tilde{\omega}^-_{-4}$	$\tilde{\omega}^-_{-3}$	$\tilde{\omega}^-_{-2}$	$\tilde{\omega}^-_{-1}$
-4.1452E-3	0.01546474	-0.05771377	0.21539034	-0.80384758

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.

The values of the 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. Accordingly, the ghost zone should be chosen as small as possible.

Indeed, on the patch, only points (x_i, y_j) for $i = 0, \dots, N_x - 1$ and $j = 0, \dots, N_y - 1$ are known, and the interpolation step requires the knowledge of values on the patch borders. Moreover, we have to evaluate the derivative at (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 from the targeted patch. As was mentioned in the introduction, we impose a restriction on the time step to enforce the displacement to be lower than the cell size. Hence, the interpolation area becomes $[x_0 - \Delta x, x_{N_x} + \Delta x] \times [y_0 - \Delta y, y_{N_y} + \Delta y]$ (here Δx and Δy denote the stepsizes) and additional cubic spline coefficients have to be computed.

For that purpose, the solution of the linear systems described in Section 4.2 takes into account the augmented right-hand side matrix (23) (the derivatives are approximated in accordance with (21)).

The solution in the x-direction is accomplished for all j, which yields the temporary spline coefficients $\gamma_\nu(y_j)$, $\nu = -1, \dots, N_x$ and $j = -2, \dots, N_y + 1$. The coefficients corresponding to $\nu = -2$ and $\nu = N_x + 1$ are deduced from (10) for $i = -1$ and $i = N_x$.

In the same way, the solution in the y-direction is accomplished for all $\nu = -2, \dots, N_x + 1$, and gives the coefficients $\eta_{\nu,\beta}$ for $\beta = -1, \dots, N_y + 1$. The boundary values $\eta_{\nu,-2}$ and η_{ν,N_y+1} are obtained from (10).

The target processor has to gather all points needed to form the matrix (22). For that purpose, as the values of the distribution function are known at (x_i, y_j) for $i = 0, \dots, N_x - 1$ and $j = 0, \dots, N_y - 1$, the local ghost zone received from other processors is

- $f(-1, j)$ for $j = 0, \dots, N_y - 1$,
- $f(i, -1)$ for $i = 0, \dots, N_x - 1$,
- $f(N_x : N_x + 1, j)$ for $j = 0, \dots, N_y - 1$,
- $f(i, N_y : N_y + 1)$ for $i = 0, \dots, N_x - 1$,
- $f(-1, -1)$,
- $f(-1, N_y : N_y + 1)$,
- $f(N_x : N_x + 1, -1)$,
- $f(N_x : N_x + 1, N_y : N_y + 1)$.

Moreover, some weighted sums of 10 points are computed on the neighboring processors to evaluate all derivatives.

6. Numerical Simulations

In this section, some numerical results obtained with the methodology we have 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 isotropic and Maxwellian of unit density and temperature. The plasma is then perturbed and a damped periodic wave is thus 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 a solution to the Vlasov-Poisson system (1)–(2).

Linear Landau Damping. The initial condition associated with the scaled Vlasov-Poisson equation (1)–(2) has the following form:

$$f_0(x, v) = \frac{1}{\sqrt{2\pi}} \exp(-v^2/2)(1 + \alpha \cos(kx)), \qquad (23)$$

with $(x, v) \in [0, 2\pi/k] \times \mathbb{R}$, 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, which is approximated by

$$\omega^2 \simeq 1 + 3k^2. \qquad (24)$$

Then we set $v_{\max} = 6$ where the velocity domain spans from $-v_{\max}$ to $v_{\max}$. We set the number of cells as $N_v = 32$ or 64 for the velocity domain, and $N_x = 32$ or 64 in the spatial direction. The time step is given by $\mathrm{d}t = 0.25$. 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 as $k = 0.3$ or 0.5. The final time is $T = 60\,\omega_p^{-1}$, with ω_p being the plasma frequency.

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

$$\mathcal{E}_h(t) = \left(\sum_i E_i^2(t) \Delta x \right)^{1/2}, \qquad (25)$$

$$\begin{pmatrix} f(-1,-1) & \partial_y f(-1,0) & f(-1,0) & \cdots & f(-1,N_y+1) \\ \partial_x f(0,-1) & \partial^2_{xy} f(0,0) & \partial_x f(0,0) & \cdots & \partial_x f(0,N_y+1) \\ f(0,-1) & \partial_y f(0,0) & f(0,0) & \cdots & f(0,N_y+1) \\ \vdots & \vdots & \vdots & \vdots & \\ f(N_x,-1) & \partial_y f(N_x,0) & f(N_x,0) & \cdots & f(N_x,N_y+1) \\ \partial_x f(N_x,-1) & \partial^2_{xy} f(N_x,0) & \partial_x f(N_x,0) & \cdots & \partial_x f(N_x,N_y+1) \\ f(N_x+1,-1) & \partial_y f(N_x+1,0) & f(N_x+1,0) & \cdots & f(N_x+1,N_y+1) \end{pmatrix}. \tag{22}$$

where Δx is the space step. According to Landau's theory, the amplitude of $\mathcal{E}_h(t)$ is expected to be exponentially decreasing with frequency ω.

Figure 1 presents the evolution of $\log(\mathcal{E}_h(t))$ in the sequential case. Two different values of the wave number are shown, namely $k = 0.3$ 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 rates obtained are given by $\gamma = 0.0127$ for $k = 0.3$, and $\gamma = 0.154$ for $k = 0.5$, i.e., they are very similar to the predicted values available in the literature ((McKinstrie *et al.*, 1999; Besse and Sonnendrücker, 2003; Filbet *et al.*, 2001)).

We also observe the "recurrence effect" (Manfredi, 1997): for example, in Fig. 1 (b) with N_x=N_v=32 points, the amplitude of the electric energy increases at $t \simeq 31\ \omega_p^{-1}$, which appears to be in good agreement with the theoretical 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 alleviated by taking more points in velocity (see Fig. 1 with $N_x = N_v = 64$ points).

In Fig. 2, the same results are shown for 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 (21). We can observe that the results are very similar to the sequential case since the damping rate is the same and the recurrence effect occurs at the correct time.

Strong Landau Damping. The initial datum is the following:

$$f(0,x,v) = \frac{1}{\sqrt{2\pi}} \exp(-v^2/2)(1 + \alpha\cos(kx)),$$

where $(x,v) \in [0, 2\pi/k] \times \mathbb{R}$, and the amplitude of the initial perturbation in the density is $\alpha = 0.5$. Moreover, the wave number is $k = 0.5$, whereas $v_{\max}$ is equal to 6.5 in order to take into account nonlinear effects. The number of cells will be equal to $N_x = N_v = 128$, whereas the time step is $\mathrm{d}t = 0.125$.

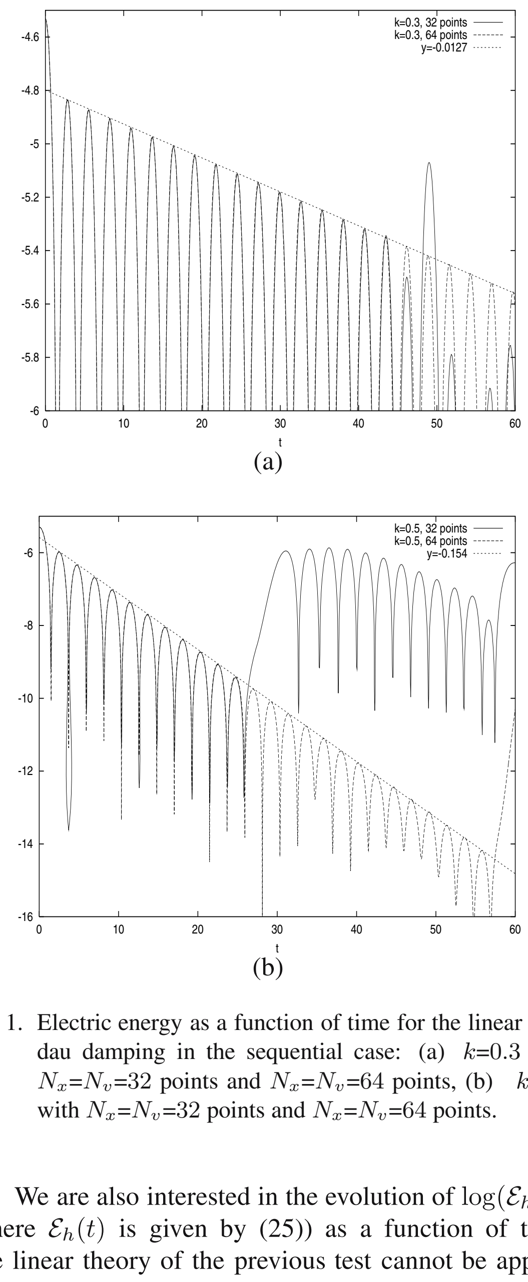
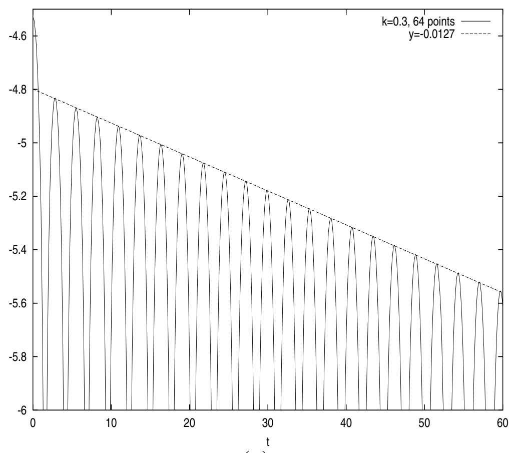
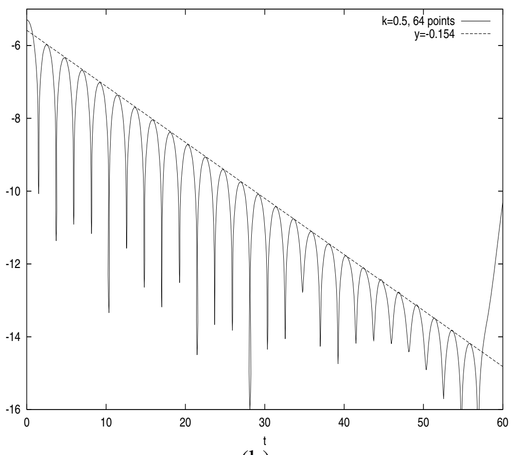


Fig. 1. Electric energy as a function of time for the linear Landau damping in the sequential case: (a) k=0.3 with N_x=N_v=32 points and N_x=N_v=64 points, (b) k=0.5 with N_x=N_v=32 points and N_x=N_v=64 points.

We are also interested in the evolution of $\log(\mathcal{E}_h(t))$ (where $\mathcal{E}_h(t)$ is given by (25)) as a function of time. The linear theory of the previous test cannot be applied in this case since 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 (Besse and Son-



(a)



(b)

Fig. 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.

nendrücker, 2003; Filbet *et al.*, 2001; Filbet and Sonnendrücker, 2003; Manfredi, 1997).

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

Moreover, in Fig. 4 we can see the projection of the distribution function as a function of the velocity. We plot the quantity

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

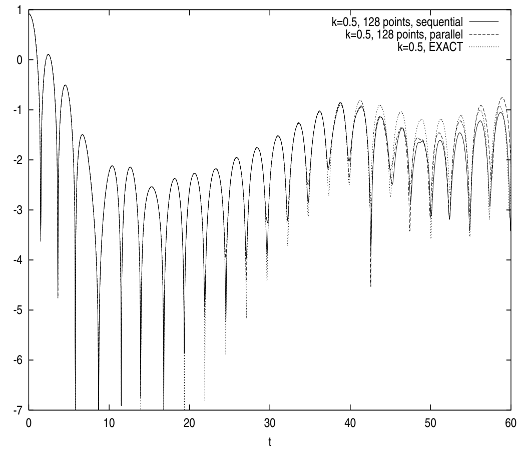


Fig. 3. Electric energy as a function of time for the strong Landau damping. Comparison between the sequential and parallel cases, k=0.5 and N_x=N_v=128. An almost "exact" solution (512×1024 points) is plotted for comparison.

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 is a consequence of the centered approximation of the derivatives (see Section 4.3). Indeed, uncentered approximation using finite-difference 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 at $v = 0$ where a decomposition point of our parallelization is located. The patches are joined to each other by the Hermite boundary conditions, which preserves an accurate global numerical approximation, even though the distribution function is very unstable (see, e.g., Fig. 4 at $t = 35\,\omega_p^{-1}$).

Moreover, to emphasize the influence of the approximation of the derivative on the results, in Figs. 5 and 6 we plot the time evolution of the total relative mass. Comparisons between the parallel and sequential cases are presented in two different contexts. In Fig. 5, all the derivatives are approximated through the following fourth-order finite-difference 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. In Fig. 6, the derivatives are replaced by the formula (21). We can observe that the finite-difference approximation is not well suited for the parallel implementation

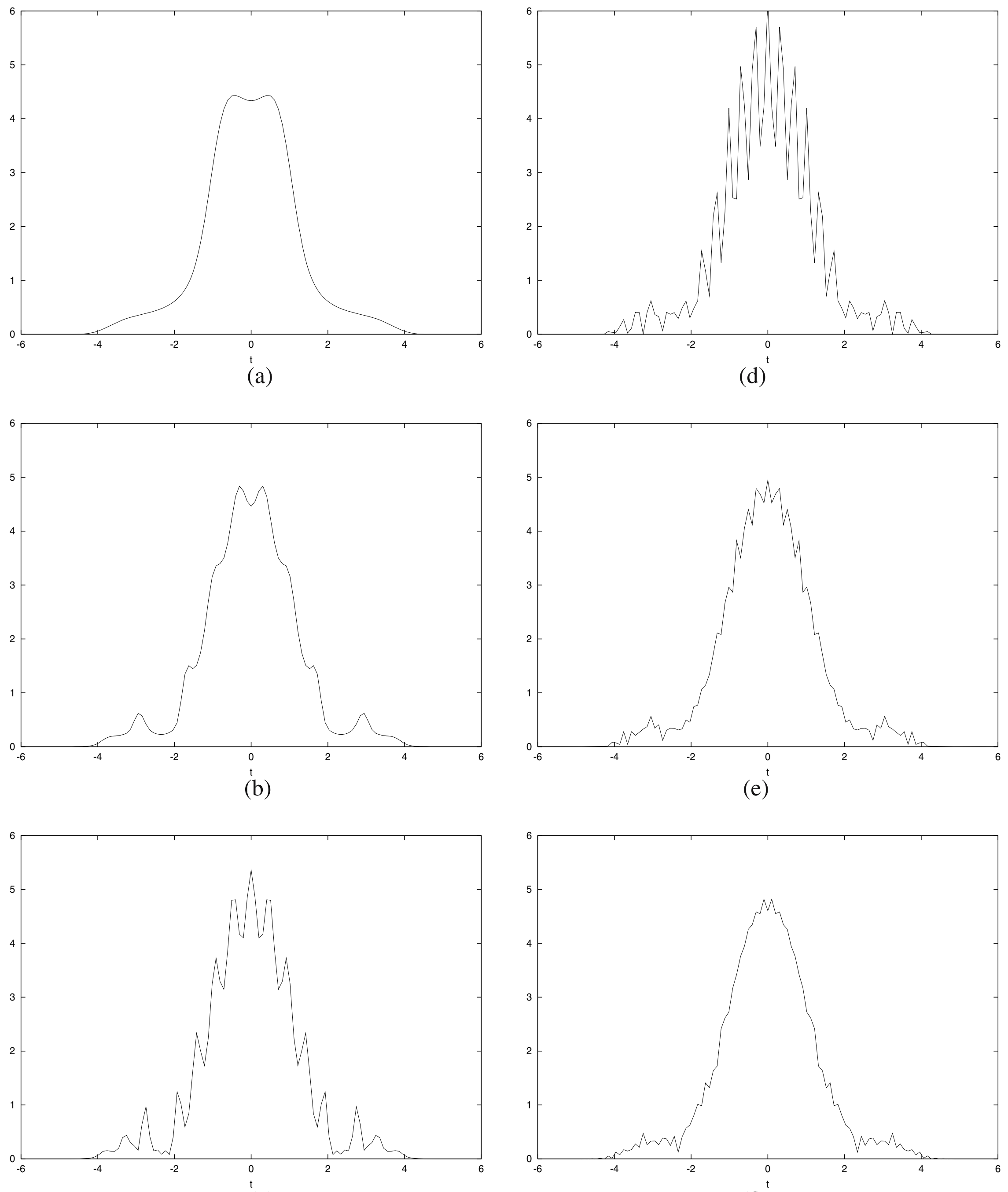


Fig. 4. Time development of the spatially integrated distribution function for the strong Landau damping: the parallel case, $N_x = N_v = 128$, (a) $t = 5\,\omega_p^{-1}$, (b) $t = 15\,\omega_p^{-1}$, (c) $t = 25\,\omega_p^{-1}$, (d) $t = 35\,\omega_p^{-1}$, (e) $t = 45\,\omega_p^{-1}$, (f) $t = 60\,\omega_p^{-1}$. .

since the total mass presents some important oscillations from $t \simeq 20\ \omega_p^{-1}$. These fluctuations become more significant 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-difference or cubic spline approximation does not affect the mass conservation in the sequential case.

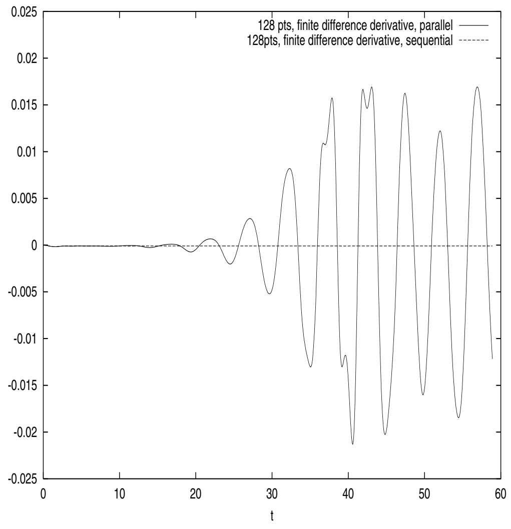


Fig. 5. Comparison between the sequential and the parallel case for the total relative mass conservation as a function of time, for the strong Landau damping. The finite-difference approximation of fourth order is used.

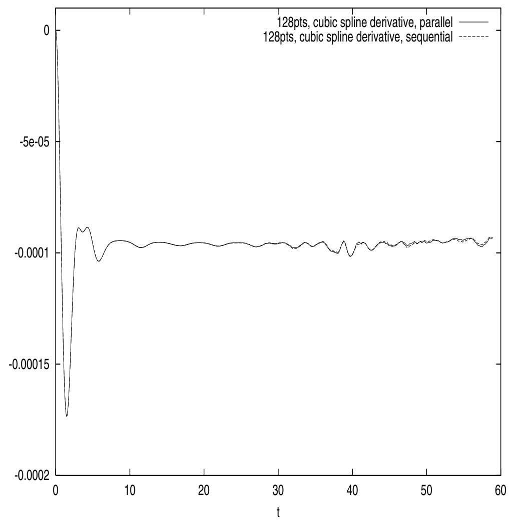


Fig. 6. Comparison between the sequential and parallel cases for the total relative mass conservation as a function of time for the strong Landau damping. The cubic spline approximation with 21 points is used.

6.2. Two-Stream Instability. In this section, we solve the Vlasov-Poisson equation considering the following initial datum:

$$f(0,x,v) = \frac{1}{\sqrt{2\pi}} v^2 \exp(-v^2/2)(1+\alpha\cos(kx)),$$

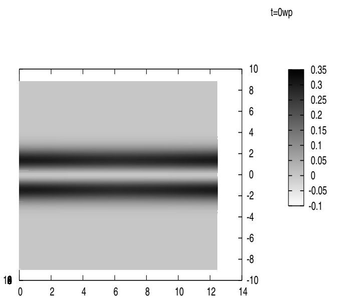
with $(x, v) \in [0, 2\pi/k] \times \mathbb{R}$, where the amplitude of the perturbation is $\alpha = 0.05$, the wave number is $k = 0.5$, and $v_{\max}$ equals 9. To get a good accuracy, the number of mesh points is $N_x = 128$ in space and $N_v = 128$ in velocity. The final time is $T = 500\ \omega_p^{-1}$.

In this model, two streams of charged particles encounter each other in the physical space with opposite velocities (see (Birdsall and Langdon, 1991) 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.

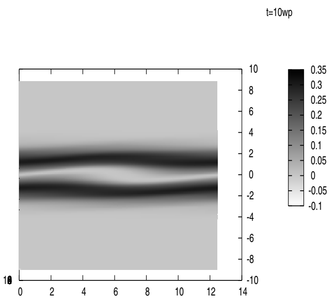
In Figs. 6 and 7, we plot the time evolution of the distribution function in the phase space in the parallel case (4 patches equally decompose the phase space domain). At time $t \simeq 10\ \omega_p^{-1}$ (where ω_p is the plasma frequency) we observe vortex creation which is associated with trapped 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 (Besse and Sonnendrücker, 2003; Filbet and Sonnendrücker, 2003).

This simulation is quite interesting since the hole has to stay in the middle of the computational domain during all the simulations. A displacement of this centered vortex can occur due to a failed numerical solution for large times. Here again, the good mass conservation depends on the derivative approximation, as explained previously. In particular, we remarked that finite-difference approximations (even centered ones) lead to uncorrect results for large times (the hole comes out early). Moreover, the inherent precision of the cubic spline interpolation allows us 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.

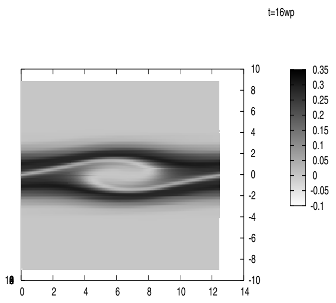
Finally, several tests have been implemented to evaluate the influence of the number of processors on the numerical results and on the simulation time. The number of points in each patch has to be substantial (for small patches, the overhead in computations to estimate the derivatives becomes too large). The performance of the parallel algorithm is summarized in Tables 2 and 3 for the two-stream instability implemented with $N_x = N_v = 512$ points. The experiments were conducted on two parallel



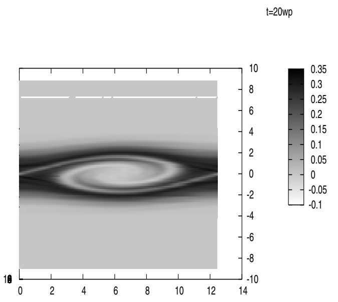
(a)



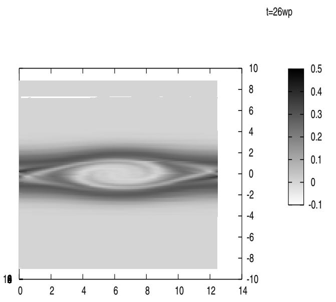
(b)



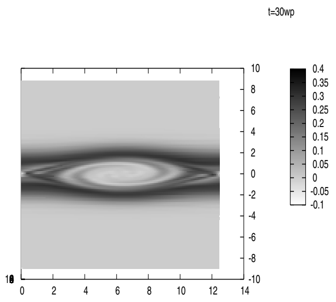
(c)



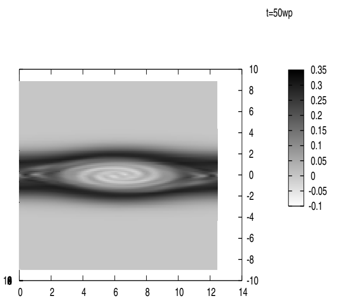
(d)



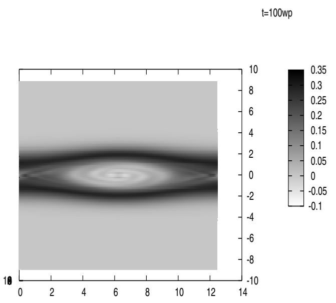
(e)



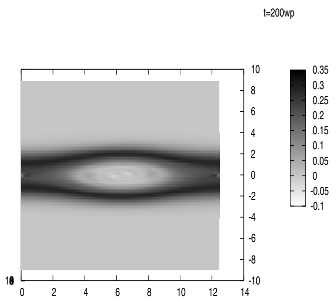
(f)



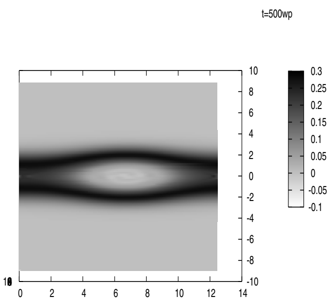
(g)



(h)



(i)



(j)

Fig. 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: $(a)\ t = 0\,\omega_p^{-1}$, $(b)\ t = 10\,\omega_p^{-1}$, $(c)\ t = 16\,\omega_p^{-1}, (d)\ t = 20\,\omega_p^{-1}, (e)\ t = 26\,\omega_p^{-1}, (f)\ t = 30\,\omega_p^{-1}$, $(g)\ t = 50\,\omega_p^{-1}, (h)\ t = 100\,\omega_p^{-1}, (i)\ t = 200\,\omega_p^{-1}, (j)\ t = 500\,\omega_p^{-1}$.

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 was stopped after 300 iterations.

Processors	1	4	8	16	32	64
Time [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 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 was stopped after 300 iterations.

Processors	1	4	8	16	32	64
Time (s.)	89.2	19.4	9.7	4.9	2.6	1.6
Speedup	1	4.6	9.2	18.2	34.3	55.75

machines: a cluster[1] of 11 IBM nodes (16-way Power5 processors at 1.9 GHz) and 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 (the corrector part of the algorithm, c.f. Section 3). This step is time consuming when the number of processor increases. Nevertheless, our methodology focuses on the interpolation step. In particular, we believe 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 affects neither the numerical results, nor 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 tested on standard plasma configurations. Indeed, the numerical results demonstrade the good efficiency of our code for the two-dimensional case and its good scalability with up to 64 processors.

Using the method 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.

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

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

Acknowledgements

Support by Commissariat à l'énergie atomique (CEA-Cadarache) is acknowledged. The authors wish to thank the CINES center and the Bordeaux 1 University for time allocation on their computers.

References

Bermejo R. (1991): *Analysis of an algorithm for the Galerkin-characteristic method*. Numerische Mathematik, Vol. 60, pp. 163–194.

Besse N. and Sonnendrücker E. (2003): *Semi-Lagrangian schemes for the Vlasov equation on an unstructured mesh of phase space*. Journal of Computational Physics, Vol. 191, pp. 341–376.

Birdsall C.K. and Langdon A.B.: *Plasma Physics via Computer Simulation*. Bristol: Institute of Physics Publishing.

Bouchut F., Golse F. and Pulvirenti M. (2000): *Kinetic Equations and Asymptotic Theory*. Paris: Gauthier-Villars.

DeBoor C. (1978): *A Practical Guide to Splines*. New-York: Springer.

Campos-Pinto M. and Merhenberger M. (2004): *Adaptive Numerical Resolution of the Vlasov Equation*.

Cheng C.Z. and Knorr G. (1976): *The integration of the Vlasov equation in configuration space*. Journal of Computational Physics, Vol. 22, p. 330.

Coulaud O., Sonnendrücker E., Dillon E., Bertrand P. and Ghizzo A. (1999): *Parallelization of semi-Lagrangian Vlasov codes*. Journal of Plasma Physics, Vol. 61, pp. 435–448.

Feix M.R., Bertrand P. and Ghizzo A. (1994): *Title?* In: Kinetic Theory and Computing, (B. Perthame, Ed.).

Filbet F., Sonnendrücker E. and Bertrand P. (2001): *Conservative numerical schemes for the Vlasov equation*. Journal of Computational Physics, Vol. 172, pp. 166–187.

Filbet F. and Sonnendrücker E. (2003): *Comparison of Eulerian Vlasov solvers*. Computer Physics Communications, Vol. 151, pp. 247–266.

Filbet F. and Violard E. (2002): *Parallelization of a Vlasov Solver by Communication Overlapping*. Proceedings PDPTA.

Glassey R.T. (1996): *The Cauchy Problem in Kinetic Theory*. Philadelphia, PA: SIAM.

Ghizzo A., Bertrand P., Begue M.L., Johnston T.W. and Shoucri M. (1996): *A Hilbert-Vlasov code for the study of high-frequency plasma beatwave accelerator*. IEEE Transactions on Plasma Science, Vol. 24.

Ghizzo A., Bertrand P., Shoucri M., Johnston T.W., Filjakow E. and Feix M.R. (1990): *A Vlasov code for the numerical simulation of stimulated Raman scattering*. Journal of Computational Physis, Vol. 90, pp. 431–457.

Grandgirard V., Brunetti M., Bertrand P., Besse N., Garbet N., Ghendrih P., Manfredi G., Sarrazin Y., Sauter O., Sonnendrücker E., Vaclavik J. and Villard L. (2006): *A drift-kinetic semi-Lagrangian 4D code for ion turbulence simulation*. Journal of Computational Physics, Vol. 217, pp. 395–423.

Gutnic M., Haefele M., Paun I. and Sonnendrücker E. (2004): *Vlasov simulation on an adaptive phase space grid*. Computer Physical Communications, Vol. 164, pp. 214–219.

Hammerlin G. and Hoffmann K.H. (1991): *Numerical Mathematics*, New-York: Springer.

Kim C.C. and Parker S.E. (2000): *Massively parallel three-dimensional toroidal gyrokinetic flux-tube turbulence simulation*. Journal of Computational Physics, Vol. 161, pp. 589–604.

McKinstrie C.J., Giacone R.E. and Startsev E.A. (1999): *Accurate formulas for the Landau damping rates of electrostatic waves*. Physics of Plasmas, Vol. 6, pp. 463–466.

Manfredi G. (1997): *Long time behaviour of strong linear Landau damping*. Physical Review Letters, Vol. 79.

Shoucri M. and Knorr G. (1974): *Numerical integration of the Vlasov equation*. Journal of Computational Physics, Vol. 14, pp. 84–92.

Sonnendrücker E., Filbet F., Friedman A., Oudet E. and Vay J.L. (2004): *Vlasov simulation of beams on a moving phase space grid*. Computer Physics Communications, Vol. 164, pp. 390–395.

Sonnendrücker E., Roche J., Bertrand P. and Ghizzo A. (1999): *The semi-Lagrangian method for the numerical resolution of the Vlasov equations*. Journal of Computational Physics, Vol. 149, pp. 201–220.

Staniforth A. and Coté J. (1991): *Semi-Lagrangian integration schemes for atmospheric models – A review*. Monthly Weather Review, Vol. 119, pp. 2206–2223.