Parallelisation of Semi-Lagrangian Vlasov Codes

Olivier Coulaud, Eric Sonnendrücker, Eric Dillon, Pierre Bertrand, Alain Ghizzo

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Parallelisation of Semi-Lagrangian Vlasov Codes

Olivier Coulaud, Eric Sonnendrücker, Eric Dillon, Pierre Bertrand, Alain Ghizzo

N ° 3430

Mai 1998

THÈME 4

ISSN 0249-6399

INRIA
LORRAINE

Parallelisation of Semi-Lagrangian Vlasov Codes

Olivier Coulaud, Eric Sonnendrücker, Eric Dillon, Pierre Bertrand, Alain Ghizzo

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Numath

Rapport de recherche n ˚ 3430 — Mai 1998 — 15 pages

Abstract: In this report we describe the parallel implementation of semi-Lagrangian Vlasov solvers, which are an alternative to Particle-In-Cell simulations for the numerical investigation of the behaviour of charged particles in their self-consistent electromagnetic fields. The semi-Lagrangian method which couples the Lagrangian and Eulerian point of view is particularly interesting on parallel computers as the solution is computed on grid points the number of which remains constant in time on each processor, unlike the number of particles in PIC simulations, and thus simplifies greatly the parallelisation process.

Key-words: Vlasov equations, kinetic equations, semi-Lagrangian method, parallelism, MPI.

(Résumé : tsvp)

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifi que,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)

Parallélisation de Codes Vlasov Semi-Lagrangien

Résumé : Dans ce rapport nous pr´esentons la parall´elisation de solveurs de Vlasov semi-Lagrangiens qui repr´esentent une alternative aux codes PIC (Particle-In-Cell) pour la simulation num´erique du comportement de particules charg´ees dans leur champ ´electromagn´etique auto-consistant. La m´ethode semi-Lagrangienne qui couple les points de vue eul´erien et lagrangien est particulièrement int´eressante pour une implantation parallèle car la solution est calcul´ee aux points d'un maillage dont le nombre reste constant sur chaque processeur au cours du temps, contrairement au nombre de particules dans un code PIC, ce qui simplifie consid´erablement l'obtention d'une parall´elisation efficace.

Mots-clé : Equation de Vlasov, Equations cin´etiques, m´ethode semi-Lagrangienne, parall´elisation, MPI.

Parallelisation of Semi-Lagrangian Vlasov Codes

OLIVIER COULAUD, ERIC SONNENDRÜCKER
IECN - Projet Numath, Universit´e Henri Poincar´e Nancy 1

ERIC DILLON
INRIA Lorraine, Centre Charles Hermite

PIERRE BERTRAND, ALAIN GHIZZO
LPMI, Universit´e Henri Poincar´e Nancy 1

1 Introduction

Vlasov-Maxwell equations play a key role in plasma physics since they describe the collective motion of a collisionless plasma with a wide range of applications.

The numerical resolution of kinetic equations the solution of which depends in addition to the time on three space variables and on three velocity variables is performed most of the time using Particle-In-Cell (PIC) methods, which enable to get satisfying results with relatively few particles. However, for some applications, in particular when particles in the tail of the distribution play an important physical role or when the particle noise is important, semi-Lagrangian methods which compute the solution on a grid may better describe the physics, see [2, 4]. Such methods are all the more interesting when using parallel computers as they are, unlike PIC methods, very scalable due to their inherent parallelism as we shall try to show in this paper.

In this article, we shall first introduce the semi-Lagrangian method and discuss how it can be applied for different kind of Vlasov equations. After that we shall isolate two special cases which need different parallelisation methods and expose those methods on two examples: the electrostatic two-dimensional Vlasov-Poisson model and the two-dimensional guiding-centre Vlasov-Poisson model.

2 The semi-Lagrangian method

Let us first recall the principles of the semi-Lagrangian method for the Vlasov equations, we refer the reader to [8] for more details.

All the types of Vlasov equations we are interested in can be written in the following way

$$\frac{\partial f}{\partial t} + U(X,t) \cdot \nabla_X f = 0, \tag{1}$$

where X stands for the phase space coordinates and U is a divergence free vector field having up to six components in the full three-dimensional case. For example, in the case of the 3D non relativistic Vlasov equation, $X = (x, y, z, v_x, v_y, v_z)$ and $U(X,t) = (v_x, v_y, v_z, E_x + v_y B_z - v_z B_y, E_y + v_z B_x - v_x B_z, E_z + v_x B_y - v_y B_x)$, all components of the electric and magnetic field depending on x, y, z and t.

Note that, as U is divergence free equation (1) can also be written in conservative form

$$\frac{\partial f}{\partial t} + div_X(U(X,t)f) = 0. \tag{2}$$

Let us now introduce the characteristics of (1), which are the solutions of the dynamical system

$$\frac{dX}{dt} = U(X(t), t). \tag{3}$$

Let us denote by $X(t; x, s)$ the solution at time t whose value is x at time s. Taking $X(t)$ a solution of (3), we have

$$\frac{d}{dt}(f(X(t), t)) = \frac{\partial f}{\partial t} + \frac{dX}{dt} \cdot \nabla_X f = \frac{\partial f}{\partial t} + U(X(t), t) \cdot \nabla_X f = 0,$$

which means that f is constant along the characteristics. This can also be written

$$f(X(t; x, s), t) = f(X(s; x, s), s) = f(x, s)$$

for any times t and s and phase space coordinate x. It is this property which will be used in the semi-Lagrangian method to solve a discrete problem, which is defined by introducing a finite set of mesh points $(x_i)_{i=1,..,N}$ which may or may not be equally spaced. Then given the value of the function f at the mesh points at any given time step, we obtain the new value at mesh point x_i using that

$$f(x_i, t_n + \Delta t) = f(X(t_n - \Delta t; x_i, t_n + \Delta t), t_n - \Delta t).$$

For each mesh point x_i, f is computed in two steps:

1. Find the starting point of the characteristic ending at x_i, i.e. $X(t_n - \Delta t; x_i, t_n + \Delta t)$
2. Compute $f(X(t_n - \Delta t; x_i, t_n + \Delta t), t_n - \Delta t)$ by interpolation, f being known only at mesh points at time $t_n - \Delta t$.

In order to deal with step 1, we need to introduce a time discretisation of (3). As in general no information on the advection function U is known at any given time, we need to use a two time step scheme in order to remain second order in time. The starting point of the characteristic is obtained, to second order accuracy, by

$$\frac{x_i - X(t_n - \Delta t)}{2\Delta t} = U(X(t_n), t_n) \tag{4}$$

writing $X(t_n) = (X(t_n + \Delta t) + X(t_n - \Delta t))/2$, there exists α_i such that $X(t_n) = x_i - \alpha_i$ and $X(t_n - \Delta t) = x_i - 2\alpha_i$. Then (4) becomes

$$\alpha_i = \Delta t \; U(x_i - \alpha_i, t_n) \tag{5}$$

which can be solved iteratively for the unknown α_i by writing

$$\alpha_i^{k+1} = \Delta t \; U(x_i - \alpha_i^k, t_n),$$

or using a Newton method. Notice that U, known only on the mesh, is interpolated linearly at $x_i - \alpha_i^k$. Once α_i is known $f(x_i - 2\alpha_i)$ is interpolated by a tensor product of cubic B-splines.

The method that we have just introduced deals with the most general case. However, in many cases, the process can be simplified by using an appropriate splitting of the Vlasov equation. The

theory of splitting has been well studied for conservation laws and preserves the second order accuracy in time when applied properly. However, one has to be careful to perform the splitting such that the resulting equations can each be written in a conservative form corresponding to equation (2). This is only possible if the divergence of each separate term with respect to the corresponding advection variable does vanish and thus it does not yield simple 1D equations in all cases. We refer the reader to [8] for more details. On the other hand, even in cases where the splitting might be numerically possible, physics considerations might prescribe not to do it.

The non relativistic electrostatic Vlasov equation, that reads in one dimension

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} + E(x,t)\frac{\partial f}{\partial v} = 0,$$

is an example where the splitting procedure can be applied. Indeed it can be split into

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} = 0, \tag{6}$$

and

$$\frac{\partial f}{\partial t} + E(x,t)\frac{\partial f}{\partial v} = 0. \tag{7}$$

In this case we have two 1D equations, the advection function v and E respectively being independent of x and v respectively. Thanks to this property no iterations are needed in order to solve (5) and a one time step scheme can be used.

When the equations can be split, all the variables that do not appear in the derivatives as v in (6) or x in (7) are really just parameters when solving the corresponding equation. Hence a trivial parallelisation can be performed by distributing the computation on the processors according to the values of this parameter. It follows that there are really two distinct parallelisation methodologies which need to be followed depending on whether the equations can be split or not. We are now going to illustrate these methods on the following two example: the two-dimensional electrostatic Vlasov-Poisson model where a splitting between space and velocity is performed and the guiding-centre model where no splitting can be performed.

3 Description of the models

The first model we consider is the 2D electrostatic Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v}\cdot\nabla_x f + \mathbf{E}(\mathbf{x},t)\cdot\nabla_v f = 0,$$

coupled with $\mathbf{E} = -\nabla\phi$, and Poisson's equation

$$\Delta\phi = 1 - \int f\, dv.$$

In this case, as $\mathbf{v}$ is independent of $\mathbf{x}$ and $\mathbf{E}(\mathbf{x},t)$ is independent of $\mathbf{v}$, the above Vlasov equation can be written in conservative form

$$\frac{\partial f}{\partial t} + \nabla_x\cdot(\mathbf{v}f) + \nabla_v\cdot(\mathbf{E}(\mathbf{x},t)f) = 0,$$

and then split between space and velocity coordinates into

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_x f = 0, \tag{8}$$

and

$$\frac{\partial f}{\partial t} + \mathbf{E}(x,t) \cdot \nabla_v f = 0. \tag{9}$$

The second model we consider is the guiding-center Vlasov model which is an approximation of the full Vlasov equation valid in presence of a large constant external magnetic field $\mathbf{B_0}$. In this case the average movement of the particles is an $\mathbf{E} \times \mathbf{B}_0$ drift. The equations read

$$\frac{\partial \rho}{\partial t} + \mathbf{v}_D \cdot \nabla_x \rho = 0, \qquad \text{where } \mathbf{v}_D = \frac{\mathbf{E} \times \mathbf{B}_0}{B_0^2}.$$

the electric field being given by Poisson's equation $-\Delta\phi = \rho$ with $\mathbf{E} = -\nabla\phi$. Here both components of $\mathbf{v}_D$ depend a priori on the two space variables. Therefore splitting can not be justified theoretically, and will not be performed.

4 The specific semi-Lagrangian algorithms

4.1 The semi-Lagrangian method for the electrostatic Vlasov equation

As we saw in the previous section, the equation can be split into two 2D advections, with an advection field independent of the advection variable, namely

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_x f = 0 \tag{10}$$

and

$$\frac{\partial f}{\partial t} + \mathbf{E}(x,t) \cdot \nabla_v f = 0. \tag{11}$$

Here we could even split the equation into four 1D advections. However this would have no influence on the parallelisation strategy. Hence we shall not consider this possibility in the sequel.

In this case the feet of the characteristics can be computed explicitely. The displacement from the mesh points is the same everywhere, namely $v\Delta t$ for the advection (10) over a time step Δt and $E\Delta t$ for the advection (11) over a time step Δt. Only the second step of the algorithm involves effective computation: The distribution function at the previous time step is interpolated by cubic splines (a two-dimensional tensor product of cubic B-splines in the each case).

4.2 The semi-Lagrangian method for the guiding-centre model

Here as no splitting can be performed, a full 2D scheme is necessary. The advection function $\mathbf{v}_D(x,t)$ depends on position and time. Therefore we need to use a two time step scheme in order to remain second order in time. The algorithm we described in section 2 does not simplify. Let us explicit it for this specific case: The characteristics are the solutions of the differential system

$$\frac{dX}{dt} = v_{D_x}(x,y,t), \quad \frac{dY}{dt} = v_{D_y}(x,y,t). \tag{12}$$

Then applying the algorithm described in Section 2, for each mesh point (x_i, y_j), ρ is computed in two steps:

1. Find the starting point of the characteristic ending at (x_i, y_j), i.e. $(X(t_n - \Delta t; x_i, y_j, t_n + \Delta t), Y(t_n - \Delta t; x_i, y_j, t_n + \Delta t))$, which is done by computing for each (i, j), the displacements $(\alpha_{ij}, \beta_{ij})$ from the mesh point (x_i, y_j) by solving iteratively the non linear system

$$\begin{aligned} \alpha_{ij} &= \Delta t \; v_{D\,x}(x_i - \alpha_{ij}, y_j - \beta_{ij}, t_n) \\ \beta_{ij} &= \Delta t \; v_{D\,y}(x_i - \alpha_{ij}, y_j - \beta_{ij}, t_n). \end{aligned}$$

   Concerning the parallelisation, once $\mathbf{v}_D$ is known, the treatment of all the mesh points can be performed concurrently. Moreover, the data involved in the computation are the values of $\mathbf{v}_D$ at neighbouring mesh points. So the problem is essentially local, involving only inter-processor communication for mesh points close to a boundary of the decomposition.

2. Compute $\rho(X(t_n - \Delta t; x_i, y_j, t_n + \Delta t), Y(t_n - \Delta t; x_i, y_j, t_n + \Delta t), t_n - \Delta t)$ by a tensor product cubic B-spline interpolation, ρ being know only at mesh points at time $t_n - \Delta t$.

Let us here describe more precisely the tensor product cubic B-spline interpolation procedure as in this case it will have a direct influence on the parallelisation strategy: The first step consists in computing the coefficients $\eta_{\nu\kappa}$ of the cubic spline interpolation function $s(x, y)$ given by:

$$s(x, y) = \sum_{-2 \le \nu \le N_x - 1} \Big(\sum_{-2 \le \kappa \le N_y - 1} \eta_{\nu\kappa} B_{3\nu}(x) B_{3\kappa}(y) \Big)$$

The spline s must satisfy the interpolation conditions

$$s(x_i, y_j) = \rho(x_i, y_j, t_n - \Delta t)$$

for $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and the two boundary conditions of the function ρ in each direction, which in our case are periodic in the x-direction and natural in the y-direction.

In order to compute the $\eta_{\nu\kappa}$ coefficients, we first solve the N_y one-dimensional interpolation problems:

$$s(x, y_j) = \sum_{-2 \le \nu \le N_x - 1} \gamma_\nu^j B_{3\nu}(x) \quad \text{for } j = 1, \ldots, N_y$$

each verifying the N_x interpolation conditions $s(x_i, y_j) = \rho(x_i, y_j, t_n - \Delta t)$ and the periodic boundary conditions, where we denote by

$$\gamma_\nu(y) = \sum_{-2 \le \kappa \le N_y - 1} \eta_{\nu\kappa} B_{3\kappa}(y).$$

and by

$$\gamma_\nu^j = \gamma_\nu(y_j).$$

Using these interpolation and boundary conditions, we are brought to solve N_y linear systems, one for each value of j, involving the same $(N_x + 2)$-dimensional tridiagonal matrix corresponding to the one-dimensional B-spline interpolation, which gives us the γ_ν^j.

Then we obtain $\eta_{\nu\kappa}$ by solving the $N_x + 2$ interpolation problems:

$$\gamma_\nu(y) = \sum_{-2 \le \kappa \le N_y - 1} \eta_{\nu\kappa} B_{3\kappa}(y) \quad \text{for } \nu = -2, \ldots, N_x - 1,$$

verifying the N_y interpolation conditions $\gamma_\nu(y_j) = \gamma_\nu^j$ and natural boundary conditions. Using these interpolation and boundary conditions, we are brought to solve $N_x + 2$ linear systems, one for each value of ν, involving the same $(N_y + 2)$-dimensional tridiagonal matrix corresponding to the one-dimensional B-spline interpolation, in order to obtain the $\eta_{\nu\kappa}$, which are the needed coefficients.

Both steps described above involve a set of identical computations, namely tridiagonal solves with different right-hand-sides. Hence, taken separately these steps are perfectly parallel in nature. However, assuming a distributed architecture, there needs to be a data redistribution between the two steps.

Finally, once the B-spline coefficients $\eta_{\nu\kappa}$ for all ν and κ have been computed, the value of ρ at the origin of the characteristics is taken to be the value of the B-spline $s(x_i - 2\alpha_{ij}, y_j - 2\beta_{ij})$. This procedure is essentially local, as it involves only points at the neighborhood of the point being considered.

5 The Field Solves

The problems we deal with involve a doubly periodic domain for the 2D electrostatic model and a domain periodic in one direction and bounded in the other for the guiding center model. The algorithm used for the solves are the following:

For the first geometry (periodic in both direction) a 2D FFT enables to compute directly the electric field components.

For the second geometry: (periodic in x and bounded in y), we want to compute the electric fields to fourth order accuracy. This is done using the procedure first introduced by Knorr, Joyce and Marcus [7] which we shall recall here in order to get some insight of the intrinsic parallelism of the method. We first need a fourth order Poisson solve which consists of the following steps:

1. Perform a Fast Fourier Transform in x which yields for each discrete mode a one-dimensional problem in the y-direction which reads:

$$-\frac{\partial^2}{\partial y^2}\phi_k(y) + k^2\phi_k = \rho_k.$$

2. Use of a fourth order scheme for the resulting one-dimensional problem in the y-direction, which reads:

$$\begin{aligned} -\frac{12}{(\Delta y)^2}(\phi_{k,j+1} - 2\phi_{k,j} + \phi_{k,j-1}) &= (\rho_{k,j+1} + 10\rho_{k,j} + \rho_{k,j-1}) \\ &\quad -k^2(\phi_{k,j+1} + 10\phi_{k,j} + \phi_{k,j-1}). \end{aligned}$$

   Gathering the identical terms, this yields

$$\begin{aligned} (1 - c_k)\phi_{k,j+1} - (2 + 10c_k)\phi_{k,j} + (1 - c_k)\phi_{k,j-1} \\ = \frac{(\Delta y)^2}{12}(\rho_{k,j+1} + 10\rho_{k,j} + \rho_{k,j-1}) \end{aligned}$$

   where $c_k = \Delta y^2 k^2/12$. This holds for $j = 2, .., N_y - 1$. In addition we have $\phi_{k,1} = \phi_{k,N_y} = 0$ for vanishing Dirichlet boundary conditions. This gives us for each mode k a tridiagonal system which needs to be solved.

3. Perform an inverse Fast Fourier Transform in x to obtain the result.

¿From a computational point of view, the first step consists in N_y simultaneous independent Fast Fourier Transforms, the second steps consists in the resolution of $N_x/2+1$ tridiagonal systems which can be done concurrently. And the third step consists in N_y simultaneous independent inverse Fast Fourier Transforms. Hence each of the three steps can completely be performed in parallel, however steps 1 involves data stored in the rows of the matrix $(\rho_{i,j})$, whereas step 2 involves data stored in the columns of the matrix $(\rho_{i,j})$. So, assuming a distributed memory architecture, this means that the matrix $(\rho_{i,j})$ needs to be redistributed on the processors between the steps.

Once the potential ϕ is obtained to fourth order accuracy, the electric field can be computed still to fourth order accuracy, by simply using the following fourth order accurate one-dimensional scheme for computing a derivative:

$$f'_{i+1} + 4f'_i + f'_{i-1} = \frac{3}{\Delta x}(f_{i+1} - f_{i-1}).$$

Thanks to the periodicity of the problem in the x-direction we can use this scheme in the x-direction as $\mathbf{E}_x = -\partial_x \phi$. It yields for $j = 1, .., N_y$ and for $i = 2, .., N_x - 1$

$$E_{xi+1,j} + 4E_{xi,j} + E_{xi-1,j} = \frac{3}{\Delta x}(\phi_{i+1,j} - \phi_{i-1,j}).$$

Moreover, due to the periodicity, we have

$$E_{x\,1,j} + 4E_{xN_x-1,j} + E_{xN_x-2,j} = \frac{3}{\Delta x}(\phi_{1,j} - \phi_{N_x-2,j}),$$

and

$$E_{x2,j} + 4E_{x1,j} + E_{xN_x-1,j} = \frac{3}{\Delta x}(\phi_{2,j} - \phi_{N_x-1,j}).$$

The same scheme can also be used in the y-direction in order to compute $\mathbf{E}_y = -\partial_y \phi$ away from the boundary.

However in order to keep fourth order accuracy a special treatment is necessary at the boundary: As for the Poisson solver we can transform the problem in a set of one dimensional problems using the Fourier Transform. Then the following one dimensional formula is accurate to fourth order:

$$(f_{i+1} - f_i) = \frac{\Delta x}{2}(f'_{i+1} + f'_i) - \frac{\Delta x^2}{12}(f''_{i+1} - f''_i).$$

Applying this formula for $f = \phi_k$, $f' = -\mathbf{E}_{y\,k}$, $f'' = k^2\phi_k - \rho_k$ yields

$$\begin{aligned}
\mathbf{E}^y_{k,1} + \mathbf{E}^y_{k,2} &= -(\frac{2}{\Delta y}(1+c_k)(\phi_{k,2} - \phi_{k,1}) + \frac{\Delta y}{6}(\rho_{k,2} - \rho_{k,1})) \\
\mathbf{E}^y_{k,n} + \mathbf{E}^y_{k,n-1} &= -(\frac{2}{\Delta y}(1+c_k)(\phi_{k,n} - \phi_{k,n-1}) + \frac{\Delta y}{6}(\rho_{k,n} - \rho_{k,n-1}))
\end{aligned}$$

Then using an inverse Fourier Transform

$$\begin{aligned}
\mathbf{E}^y_{i,1} + \mathbf{E}^y_{i,2} &= -\frac{2}{\Delta y}((\phi_{i,2} - \phi_{i,1}) + \mathcal{F}^{-1}(c_k(\phi_{k,2} - \phi_{k,1}))) + \frac{\Delta y}{6}(\rho_{i,2} - \rho_{i,1}) \\
\mathbf{E}^y_{i,n} + \mathbf{E}^y_{i,n-1} &= -\frac{2}{\Delta y}((\phi_{i,n} - \phi_{i,n-1}) + \mathcal{F}^{-1}(c_k(\phi_{k,n} - \phi_{k,n-1}))) + \frac{\Delta y}{6}(\rho_{i,n} - \rho_{k,n-1})
\end{aligned}$$

where $c_k = k^2(\Delta y)^2/12$. These two equations complete the system which remains fourth order.

As for the Poisson solves, there is an inherent parallelism in the computation of the electric field which consists for both components in the resolution of a set of tridiagonal systems. But here again the distribution of the data on the processors needs to be different for the computation of E_x and E_y.

6 The parallel algorithms

6.1 The guiding-centre model

Going through the different parts of the algorithm, we can explicit the different kinds of computation which have to be done. These are for the density advance in time: fixed point iterations over the displacements from the mesh points, tridiagonal solves for computing the cubic B-spline coefficients (a whole set in each direction as we are using a tensor product interpolant), evaluation of the interpolated spline at the feet of the characteristics. On the other hand for the field solve the computations to be performed are multiple one-dimensional Fast Fourier Transforms and inverse Fast Fourier Transforms, as well as multiple tridiagonal solves.

We notice that a few stages are completely local: finding the origin of the characteristics, computing the spline values which only involves the mesh points in the neighbourhood of the one being computed. And most stages (FFT, tridiagonal solves) involve whole lines of the domain, but unfortunately not always in the same direction.

Normally, for a 2D problem like ours, a global 2D decomposition is optimal as it has the smallest inter-processor boundary and involves no data redistribution on the processors. However, tridiagonal solves and FFTs are well known not to have a very good scalability when performed on multiple processors. On the other hand, we saw that for each stage of the algorithm, there is a 1D band decomposition for which there is a natural optimal parallelism. Therefore our final choice was to take several distinct decompositions even though this would involve global communication. One primary decomposition in 1D bands in the direction (Ox) would be used most of the time, the global data being stored according to this decomposition, and several secondary decompositions, not always the same, would be used where required. That is in our specific case: the tridiagonal solves for Poisson, for E_y, and one direction of the spline interpolation.

Using these considerations, we start with a 1D band distribution along the x-direction and proceed at each time step through the following stages:

1. Compute the new electric field. For this we first perform a Poisson solve, which consists of
   (a) a multiple 1D FFT for ρ on each processor, the data being distributed evenly according to their j index,
   (b) a data redistribution, the data is now distributed across the processors according to their mode number,
   (c) multiple tridiagonal solves on each processor,
   (d) a data redistribution to go back to the initial distribution according to the j index,
   (e) a multiple inverse FFT.

   Then we compute the derivatives in each direction of the electric field, for which the steps are:

(a) multiple tridiagonal solves on each processors where the data is distributed according to the j index for computing the derivative with respect to x,

(b) a data redistribution, so that the distribution is along the i index,

(c) multiple tridiagonal solves on each processor where the data is distributed according to the i index for computing the derivative with respect to y,

(d) a data redistribution, to get back to the original distribution for the density advance.

2. Advance the charge density ρ, which is performed as follows

(a) Compute feet of characteristics: this needs information from neighbourhood, hence it involves inter-processor communication for mesh points at the edge of the decomposition.

(b) Compute the spline coefficients in the x-direction. This involves multiple tridiagonal solves on each processor.

(c) data redistribution so that the distribution is along the i index.

(d) Compute the tensor product spline coefficients. This involves multiple tridiagonal solves on each processor which can be performed locally thanks to the actual data distribution.

(e) Data redistribution to go back to the original distribution for the beginning of the next time step.

(f) Evaluation of splines at feet of characteristics: This needs information from the neighbourhood and thus a slight amount of local inter-processor communication.

6.2 The electrostatic model

Let us now go through the steps of the algorithm for the electrostatic model in the same way. Thanks to the splitting method we use, the semi-Lagrangian method is applied once for a 2D physical space advection with the velocity coordinate $\mathbf{v}$ as a parameter and once for a 2D velocity space advection with the physical space coordinate $\mathbf{x}$ as a parameter. Thus, if the data is distributed in each case according to the parameter, there will be no communication at all, not even the almost local communications that occured in the previous example. The computations in this case, which are fully local involve multiple tridiagonal solves for the cubic spline interpolation, and explicit computations for the spline evaluations. For the field solve, we need a 2D Fast Fourier Transform which is also achieved best by multiple 1D Fast Fourier Transforms involving data redistribution.

Finally the algorithm reads in this case for each time step, starting with a 1D band distribution along the v_x-direction.

1. perform a spatial shift over $\Delta t/2$, that is apply the simplified semi-Lagrangian algorithm for equation (8) over a time step of $\Delta t/2$.

2. redistribute the distribution function in order to get a 1D band distribution along x-direction. Then integrate to obtain ρ

3. Compute the electrical field

(a) perform a multiple 1D FFT along the y-direction for the electric field

(b) redistribute the field data along the y-direction

(c) perform a multiple 1D FFT along the x-direction for the electric field

(d) redistribute the distribution the electric field in order to get a 1D band distribution along the y-direction.

4. redistribute the distribution function in order to get a 1D band distribution along v_x-direction.

5. perform a velocity shift over Δt, that is apply the simplified semi-Lagrangian algorithm for equation (9) over a time step of Δt.

6. perform a second spatial shift over $\Delta t/2$.

7 The implementations

7.1 A shared memory implementation of the guiding-centre model

The easiest way to implement the parallel algorithm we described for the guiding-centre model is to use shared distributed memory concept available on Silicon Graphics' Origin 2000. This concept means that even though the memory of the computer is physically distributed, there exists a software system giving access to the whole memory. When programming on such a platform one still needs to take care of the data distribution on the processors, but the communications are handled by the system through compiler directives or system calls. Let us now give a few more details:

The parallel implementation uses just one parallel loop over the number of processors which appears after the initialisation part of the code and is ended at the end of the computation. This is implemented using the `!$DOACROSS` directive. The arrays through which the communication will be carried over are declared as shared at the beginning of this parallel loop, all the others are local to each processor. Then a matrix transpose is implemented by simple array copy:

```
do i=1,n
  a(i,j)=b(j,i)
end do
```

where b is a shared array. Moreover the almost local communications which appear in the computation of the feet of the characteristics and the spline evaluations are transparent through the use of shared arrays for the drift velocity and the spline coefficients.

Before each global data redistribution, there is a call to the system routine `mp_barrier` for synchronisation.

7.2 A message passing implementation of the electrostatic model

In this case we noticed that with our parallel algorithm, the only communications are the row-columns exchanges involved in the global data redistribution. There is no other communication necessary. The implementation we used in this case was the MPI message passing library.

The most obvious approach for implementing the data redistribution consists in having the processors send all at once the rows of the arrays to the processors which need them for further computations and then have them all receive their data in the apropriate columns. This worked fine on

Cray's T3E but overflowed the Origin 2000's message passing buffers. Indeed, this communication scheme requires each processor to bufferize the rows to be sent. As a consequence, the number of required MPI buffers increases very quickly with the number of processors, leading to deadlock on Origin 2000 : each processor blocks because all its local buffers are full, but still need to execute "send" actions before actually starting to receive columns, freeing by the way others processors' buffers.

For this reason a more refined algorithm was needed. The main idea that lead the design of the new algorithm for this row-column exchange, was to avoid buffering most of the time. To do so, we choose to refine the algorithm into a sequence of steps (not only "bulk send followed by bulk receive"), where each step would only require one MPI buffer, whatever the total number of processors. Thanks to this, it is possible to reduce the memory needs, and, most important, to avoid deadlock, even if the architecture has only few MPI buffers available.

The algorithm designed for this row-column exchange is to be applied recursively by each processor on their local band. Its main advantage is that it ensures that, at each step of the global exchange, each processor will only perform one point to point communication. As a consequence, all congestions and bufferisations can be avoided during communications.

Let us see how the recursive algorithm works on an example. For this example, we will distribute the global matrix by bands on 4 processors. Figure 1 shows the successive steps performed in parallel on the matrix.

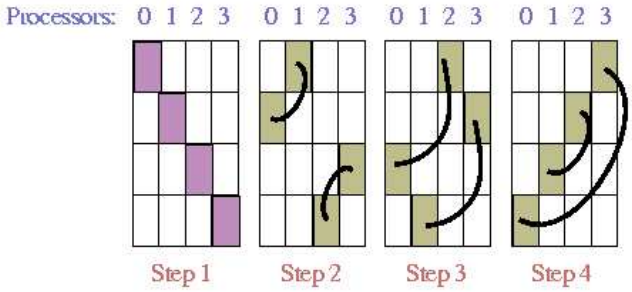

Figure 1: Successive steps of the transposition

At each step, the global matrix is virtually split into 16 blocks, so that each processor will only have one partner to send his own data to.

The first step does not include any communication, since this step only involves the data stored on the diagonal of the global matrix, which means that this very block does not need to be moved.

The three next steps then consist in point to point exchanges between processors. You might notice that at each step, each processor only has one send-receive operation to complete, so that it only needs one buffer for the exchange.

Since our 2D Vlasov-Poisson application also required the use of 2D FFT over the global matrix, we chose to extend this row-column global exchange to a true matrix transposition in order to use it for a 2D parallel FFT. Indeed, the extension to the previous algorithm was to actually transpose the data during the exchange. This transposition was performed, on the fly, using MPI derived datatype.

To perform the parallel 2D FFT over a matrix distributed in band on processors, we can use the following algorithm :

1. each processor computes a multiple real to complex 1D-FFT on its local columns,

2. the result of this first multiple complex 1D-FFT is globally transposed, to swap directions,

3. each processor computes a new complex to real 1D-FFT on these new columns,

4. the result is transposed once again, to get each direction in the right way.

Multiple 1D-FFT are available from the *scsl* library on CRAY and SGI platform. We implemented the row-column exchange algorithm with "on the fly" transposition thanks to MPI derived data types, to get a true transposition algorithm.

8 The parallel speed-ups

The performance of the parallel algorithms are summarized in the following tables. For the guiding center model the speed-up is given for the whole code, whereas for the electrostatic code the speed-up is only for the transposition, the rest of the algorithm being perfectly parallel.

- Speed-up for the guiding-centre code

Processors	1	2	4	8	16
Time (in s)	608	324	163	85	47
Speed-Up	1	1.88	3.73	7.15	12.94

Table 1: Speed-Up for a 1024×1024 grid

- Speed up for the MPI transposition algorithm

Processors	8	16	32
Speed-Up	$7,53$	$11,1$	$16,75$
Efficacity	$0,94$	$0,69$	$0,52$

Table 2: Speed-Up of a 1024x1024 complex matrix transposition

As we showed in this paper, the semi-Lagrangian algorithm can be applied such that most steps are perfectly parallel. In fact, in the case of the electrostatic Vlasov-Poisson model, where the feet of the characteristics are known, all stages of the computation are perfectly parallel provided data redistributions are performed at some points. Moreover, we have found a way to perform the data redistribution in an efficient manner which enables us to have a good parallel efficiency.

9 Conclusion

Parallelizing Vlasov codes is an essential task since the numerical solution of Vlasov equations is very time and memory consuming. The two special models discussed above, have been chosen to reflect the specific difficulties associated with the parallelisation process of more general Vlasov

codes. The performance obtained on these examples are very satisfying. Let us also mention that the examples we developed here were essentially classical problems picked to test the numerical algorithm and benchmark the codes. More complex and realistic problems related to laser-plasma interaction are adressed in a companion paper submitted in this issue of the journal [6].

Building on the methods we introduced here we are now ready to develop a fortran 90 module library implementing the different kinds of advection types that are needed. Assembling these modules will then enable us to treat many problems occuring in plasma physics using the semi-Lagrangian methodology.

References

[1] R. Bermejo, *Numer. Math. 60* (1991)

[2] A. Ghizzo, P. Bertrand, M. Shoucri, T.W. Johnston, E. Filjakow, M.R. Feix*J. Comput Phys. 90*, 431 (1990)

[3] A. Ghizzo, P. Bertrand, M.L. Begue, T.W. Johnston, M. Shoucri *IEEE Transaction on Plasma Science 24*, 370 (1996)

[4] M.R. Feix, P. Bertrand, A. Ghizzo, *in Kinetic Theory and Computers*, World Scientific Vol. 22, B. Perthame ed., (1994)

[5] A. Ghizzo, P. Bertrand, M. Shoucri, E. Filjakow, M.R. Feix *J. Comput Phys. 108*, 105 (1993)

[6] M.L. B´egu´e, A. Ghizzo, P. Bertrand, E. Sonnendr¨ucker, O. Coulaud, T.W. Johnston, M. Shoucri, *submitted to Journal of Plasma Physics, Special issue of the 16th International Conference on the Numerical Simulation of Plasmas.*

[7] G. Knorr, G. Joyce and A.J. Marcus., J. Comput. Phys. **38**, 227-236 (1980).

[8] E. Sonnendr¨ucker, J. Roche, P. Bertrand, A. Ghizzo, INRIA report 3393 (March 1998), submitted to J. Comput. Phys.

[9] A. Staniforth and J. Cˆot´e, *Monthly Weather Review 119* (1991).

Unit´e de recherche INRIA Lorraine, Technopˆole de Nancy-Brabois, Campus scientifi que,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhˆone-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399