Parallelization of a Vlasov-Maxwell solver in four-dimensional phase space

L. K. S. Daldorff [a]

[a] *Department of Physics and Astronomy ,Uppsala university, Box 516, SE-751 20 Uppsala, Sweden*

B. Eliasson [b,1]

[b] *Department of Physics, Umeå University, SE-901 87 Umeå, Sweden*

Abstract

We present a parallelized algorithm for solving the time-dependent Vlasov-Maxwell system of equations in the four-dimensional phase space (two spatial and velocity dimensions). One Vlasov equation is solved for every particle species, from which charge and current densities are calculated for the Maxwell equations. The parallelization is divided into two different layers. For the first layer, each plasma species is given its own processor group. On the second layer, the distribution function is domain decomposed on its dedicated resources. By separating the communication and calculation steps, we have met the design criteria of good speedup and simplicity in the implementation.

Key words: Vlasov-Maxwell system, spectral method, transpose method
PACS: 52.25.Dg, 52.65.Ff

1 Introduction

The Vlasov equation describes the dynamics of the distribution function of charged particles in a collisionless plasma, in which the particles are accelerated by the long-range electric and magnetic fields. The particles are distributed statistically both in the three-dimensional configuration space $\mathbf{x}$, and

Email addresses: lars.daldorff@irfu.se (L. K. S. Daldorff), bengt@tp.umu.se (B. Eliasson).

[1] Theoretische Physik IV, Ruhr-Universität Bochum, D-44780 Bochum, Germany

in the three-dimensional velocity space $\mathbf{v}$ (or, equivalently, in momentum space $\mathbf{p}$). Numerical algorithms used to solve the Vlasov equation are divided into two main groups, particle-in-cell (PIC) methods (3; 19) and direct Vlasov solvers. In PIC methods, the particle distribution function is resolved with a large number super-particles, where each super-particle represents a large number of real particles (e.g. electrons or ions). The electromagnetic field is represented on a numerical grid in ordinary space. Averaging schemes are used to calculate current and charge densities from the particles to the grid cells and interpolation schemes are used to calculate the electric and magnetic field acting as a force on each particle. The evolution of each particle is then calculated by Newton's laws of motion and the evolution of the electromagnetic field by the Maxwell equations. PIC codes are adaptive and simple to implement but suffers from statistical noise and poor resolution of the low-density tail of the distribution function. In Vlasov solvers, the continuous particle distribution function are represented as phase fluid on a numerical grid in both the ordinary space and in velocity (or momentum) space. In this manner, the statistical noise is eliminated and the low-density tail is represented up to the accuracy of the computer's number representation. On the other hand, Vlasov codes tend to become more memory consuming and computationally demanding due to the high dimensionality of phase space. Vlasov solvers in higher dimensions therefore require efficient algorithms and parallel computing in order to solve realistic problems. Numerical schemes for solving the Vlasov equation include time-splitting schemes (5), Fourier and Hermite methods (2; 4; 17; 6; 7; 15; 9) and conservative schemes (11; 10). Comparisons of different Eulerian solvers are presented in Refs. (1; 12). Parallel algorithms for solving the non-relativistic Vlasov equation in magnetized plasmas have been developed using a Van Leer discretization (18), and for solving the two-dimensional electromagnetic and relativistic Vlasov equation using a semi-Lagrangian Vlasov model (14).

The 1×1-dimensional (one spatial and one velocity dimension) Fourier transformed Vlasov-Poisson system has previously been parallelized by means of domain decomposition in phase space (8), where the algorithms for the numerical approximations were converted into parallelized counterparts and the communication is performed locally in each algorithm. In this manner, special methods could be used to optimize the performance of the algorithms, but to the expense of a considerably more complicated parallelized code compared to the serial one. Here we present a parallelized algorithm for solving the 2×2-dimensional (2 spatial and 2 velocity dimensions) Vlasov-Maxwell system, based on a previously developed serial algorithm (6; 7). We will use a relatively simple parallelization method, where the communication step is separated from the algorithms used to approximate the differential equations. It is similar to the so-called transpose method, which is common in global atmospheric modeling (16) and other applications where spectral methods are used. In this method, one dimension is kept fixed while the other two dimensions are

decomposed, and the spectral method can be applied in parallel in the fixed dimension. After this, the system is transposed before applying the algorithm in another direction (13). The Vlasov solver uses periodic boundary conditions in configuration space, where a pseudo-spectral method is employed to calculate derivatives accurately. In velocity space, the system has been Fourier transformed analytically, and a compact fourth-order compact difference scheme is used to calculate the derivatives in the Fourier transformed velocity space. We will use two-dimensional decomposition in which one spatial and one velocity domain is decomposed on a rectangular processor grid, while the other pair of spatial and velocity dimensions are kept fixed in each processor. After having performed the necessary operations in the fixed pair of dimensions, the system is transposed so that the first pair of dimensions are kept fixed and the other pair is decomposed before applying operations on the first pair of dimensions. We will discuss the parallelization on different levels and the efficiency of different communication strategies, and the advantages of the modular design for future modifications and extensions to higher dimensions.

2 Mathematical model

Let us first present the mathematical model of a collisionless plasma consisting of electrically charged particles (electrons and ions), which interact only through long-range electric and magnetic fields. We will restrict ourselves to a two-dimensional model in which the magnetic field is directed in the x_3 direction, $\mathbf{B} = \hat{\mathbf{x}}_3 B_3$, where $\hat{\mathbf{x}}_3$ is the unit vector in the x_3 direction, and the electric field is directed in the x_1x_2-plane, $\mathbf{E} = \hat{\mathbf{x}}_1 E_1 + \hat{\mathbf{x}}_2 E_2$ where $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ are the unit vectors along the x_1 and x_2 axis. (Vectors are written with boldface characters.) We also assume that the electric and magnetic fields depend only on the spatial x_1 and x_2 coordinates, plus time t. The distribution function f_α, where α denotes the particle species (e.g. e for electrons and i for ions), has been integrated over v_3 space so that it depends only on the velocity coordinates v_1 and v_2 together with the spatial coordinates x_1 and x_2 and time t. The dynamics of the plasma is governed by the Vlasov equation which describes the evolution of the particle distribution function in the presence of electric and magnetic fields,

$$\frac{\partial f_\alpha}{\partial t} + \nabla_{\mathbf{x}} f_\alpha + \frac{q_\alpha}{m_\alpha}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}} f_\alpha = 0, \tag{1}$$

where q_α is the electric charge of the particle and m_α is the particle mass. In our two-dimensional geometry, the spatial nabla operator is given by $\nabla_{\mathbf{x}} = \hat{\mathbf{x}}_1 \partial/\partial x_1 + \hat{\mathbf{x}}_2 \partial/\partial x_2$, and the nabla operator in velocity space is given by $\nabla_{\mathbf{v}} = \hat{\mathbf{x}}_1 \partial/\partial v_1 + \hat{\mathbf{x}}_2 \partial/\partial v_2$. There is one Vlasov equation for each particle species. The electric and magnetic fields $\mathbf{E}$ and $\mathbf{B}$ are governed by the Maxwell equations

$$\nabla_{\mathbf{x}} \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0} \tag{2}$$

$$\nabla_{\mathbf{x}} \cdot \mathbf{B} = 0 \tag{3}$$

$$\nabla_{\mathbf{x}} \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{4}$$

$$\nabla_{\mathbf{x}} \times \mathbf{B} = \mu_0 \mathbf{J} - \frac{1}{c^2} \frac{\partial \mathbf{E}}{\partial t}, \tag{5}$$

where ε_0 is the electric permittivity, μ_0 is the magnetic permeability, and c is the speed of light in vacuum. By introducing the scalar and vector potentials ϕ and $\mathbf{A} = \widehat{\mathbf{x}}_1 A_1 + \widehat{\mathbf{x}}_2 A_2$ as

$$\mathbf{E} = -\nabla_{\mathbf{x}} \Phi - \frac{\partial \mathbf{A}}{\partial t} \tag{6}$$

$$\mathbf{B} = \mathbf{B}_0 + \nabla_{\mathbf{x}} \times \mathbf{A}, \tag{7}$$

into the Maxwell equations, where $\mathbf{B}_0$ is the constant external magnetic field, together with the Lorenz condition $\nabla_{\mathbf{x}} \cdot \mathbf{A} + (1/c^2)\partial\phi/\partial t = 0$, we obtain the system of evolution equations for the potentials (7),

$$\frac{\partial \mathbf{A}}{\partial t} = \mathbf{\Gamma} \tag{8}$$

$$\frac{\partial \mathbf{\Gamma}}{\partial t} = c^2 \nabla_{\mathbf{x}}^2 \mathbf{A} + \mu_0 \mathbf{J} \tag{9}$$

$$-\nabla_{\mathbf{x}}^2 \Phi = \frac{\rho}{\varepsilon_0} + \nabla_{\mathbf{x}} \cdot \mathbf{\Gamma}, \tag{10}$$

which is equivalent to the Maxwell equations. The charge and current densities ρ and $\mathbf{J}$ are obtained from the particle number densities n_α and mean velocities $\mathbf{v}_\alpha$ as

$$\rho = \sum_\alpha q_\alpha n_\alpha \tag{11}$$

$$\mathbf{J} = \sum_\alpha q_\alpha n_\alpha \mathbf{v}_\alpha, \tag{12}$$

where the summation is taken over particle species. Finally, the particle number densities and mean velocities are obtained as moments of the distribution functions as

$$n_\alpha = \iint f_\alpha \, dv_1 \, dv_2 \tag{13}$$

$$\mathbf{v}_\alpha = \frac{1}{n_\alpha} \iint \mathbf{v} f_\alpha \, dv_1 \, dv_2 \tag{14}$$

where $\mathbf{v} = \widehat{\mathbf{x}}_1 v_1 + \widehat{\mathbf{x}}_2 v_2$, and the integrals are taken over all (v_1, v_2) space. The numerical method is based on a Fourier method in velocity space in which the Vlasov equation is Fourier transformed analytically in velocity space and the resulting equation is solved numerically. Introducing the Fourier transform pair

$$f_\alpha = \iint \widetilde{f}_\alpha e^{-iv_1\eta_1 - iv_2\eta_2}\, d\eta_1\, d\eta_2 \tag{15}$$

$$\widetilde{f}_\alpha = \frac{1}{(2\pi)^2} \iint f_\alpha(\mathbf{r}, \mathbf{v}, t) e^{iv_1\eta_1 + iv_2\eta_2}\, dv_1\, dv_2 \tag{16}$$

the Vlasov equation is transformed into

$$\frac{\partial \widetilde{f}_\alpha}{\partial t} - i\frac{\partial^2 \widetilde{f}_\alpha}{\partial x_1 \partial \eta_1} - i\frac{\partial^2 \widetilde{f}_\alpha}{\partial x_2 \partial \eta_2} + \frac{q_\alpha}{m_\alpha}\left[i(E_1\eta_1 + E_2\eta_2)\widetilde{f}_\alpha + B_3\eta_1 \frac{\widetilde{f}_\alpha}{\partial \eta_2} - B_3\eta_2 \frac{\widetilde{f}_\alpha}{\partial \eta_1} \right] = 0 \tag{17}$$

and the particle number densities and mean velocities are obtained as

$$n_\alpha = (2\pi)^2 \widetilde{f}_\alpha|_{\eta_1=\eta_2=0} \tag{18}$$

$$\mathbf{v}_\alpha = -i\frac{(2\pi)^2}{n_\alpha}\left(\widehat{x}_1 \frac{\partial \widetilde{f}_\alpha}{\partial \eta_1} + \widehat{x}_2 \frac{\partial \widetilde{f}_\alpha}{\partial \eta_2} \right)\Bigg|_{\eta_1=\eta_2=0}. \tag{19}$$

Equations (17)–(19) together with (6)–(12) form a closed system, which is solved numerically.

3 Numerical approach

We here discuss the representation of our physical system on a numerical grid, the numerical algorithms used. We put special emphasis on the parallelization method which is divided into different abstraction levels for obtaining a modular design and efficient speedup.

3.1 Discretization and numerical approximations

The computational domain is discretized on a rectangular, equidistant grid. The spatial variables are discretized as $x_{1,i_1} = i_1 \Delta x_1$, $i_1 = 0, 1, \ldots, N_{x_1} - 1$, and $x_{2,i_2} = i_2 \Delta x_2$, $i_2 = 0, 1, \ldots, N_{x_2} - 1$, the Fourier transformed velocity variables are discretized as $\eta_{1,\alpha,j_1} = j_1 \Delta \eta_{1,\alpha}$, $j_1 = 0, 1, \ldots, N_{\eta_1}$, and $\eta_{2,\alpha,j_2} =$

$j_2\Delta\eta_{2,\alpha}$, $j_2 = -N_{\eta_2} \ldots -1, 0, 1, \ldots, N_{\eta_2}$. The grid sizes are $\Delta x_1 = L_1/N_{x_1}$, $\Delta x_2 = L_2/N_{x_2}$, $\Delta\eta_{1,\alpha} = \eta_{\alpha 1,max}/N_{\eta_1}$, and $\Delta\eta_{2,\alpha} = \eta_{\alpha 2,max}/N_{\eta_2}$. The time is discretized as $t_k = t_{k-1} + \Delta t_k$, $t_0 = 0$, $k = 1, 2, \ldots, N_t$. The particle distribution functions of species α are discretized and enumerated such that $\widetilde{f}_\alpha(x_{1,i_1}, x_{2,i_2}, \eta_{1,\alpha,j_1}, \eta_{2,\alpha,j_2}, t_k) \approx \widetilde{f}^k_{\alpha,i_1,i_2,j_1,j_2}$. The scalar potential is discretized as $\Phi(x_{1,i_1}, x_{2,i_2}, t_k) \approx \Phi^k_{i_1,i_2}$, and the vector components E_1, E_2, B_3, A_1, A_2, Γ_1, and Γ_2 are discretized in the same manner as Φ. Periodic boundary conditions are used in x_1, x_2 space while absorbing boundary conditions are used in η_1, η_2 space (6; 7). The time step Δt_k can either be fixed or calculated adaptively.

A pseudo-spectral method is used to approximate all derivatives in x_1, x_2 space, while a compact fourth-order difference approximation is used for the η_1 and η_2 derivatives. The time stepping is performed with a Fourth-order Runge-Kutta method. A detailed description of the numerical algorithms can be found in Refs. (6; 7).

3.2 Parallelization levels

The parallelization is performed on two levels. On the first level, the code is parallelized over particle species, so that the distribution function for different particle species are represented on different processors. In this manner, numerical integration of the Vlasov equation for each particle species can be performed in parallel, and communication is only needed when calculating the charge densities and currents needed for the Maxwell equations. The Maxwell solver takes only a minor part of the work load. We have parallelized it such that one processor calculates the scalar and vector potentials and the electric field, and one processor calculates the magnetic field and the time derivative of the vector potential, after which this data is distributed.

On the second level, as illustrated in Fig. 1, the particle distribution function for each particle species is shared between processors by means of domain decomposition of phase space. The domain decomposition is done in one spatial and one velocity dimension, say (x_2, η_2), so that numerical operations (e.g. differentiation) in the two other dimensions (x_1, η_1) can be performed in parallel. After this, the four-dimensional solution matrix is rotated, or transposed, such that the computational domain is decomposed in the other two dimensions (x_1, η_1), as illustrated in Fig. 1, and the numerical operations can be carried out in the first two dimensions (x_2, η_2). Finally the solution matrix is transposed back to its original configuration. Special care has to be taken if the domain decomposition in the Fourier transformed velocity space is dense so that each processor has less than 4 grid points, since a centered 6-point scheme (7; 8) is used to calculate the particle velocity (19) at the origin in the Fourier transformed velocity space of the particle distribution function.

In this case, an extra communication step is needed between the processor containing the origin and its neighbor.

By this parallelization strategy, the numerical operations can be carried out completely in parallel in each dimension, and there is no need to implement parallelized versions of the fast Fourier transform for calculating the x_1 and x_2 derivatives or of the compact difference schemes used to calculate the η_1 and η_2 derivatives in Eq. (17). Thus, most of the numerical algorithms can be left un-parallelized, which simplifies the implementation significantly and minimizes the risk of programming errors in the implementation. It also simplifies the work if the mathematical model is modified and new features have to be implemented into the code. This is an advantage compared to the parallelization strategy used in Ref. (8), where a considerably more complicated parallel algorithm was used to obtain a somewhat higher degree of speedup.

3.3 Speedup

We will here derive a simple speedup model for our parallelized numerical algorithm. The numerical algorithm for integrating the Vlasov-Maxwell system is described in detail in Refs. (6; 7), so we here only state shortly which numerical algorithms are used. Equations (8), (9) and (17), supplemented by Eqs. (10)–(12), (18) and (19), are advanced in time with a fourth-order Runge-Kutta scheme in which the numerical approximations in $(x_1, x_2, \eta_1, \eta_2)$ space are calculated four times per timestep. The algorithms are identical in each Runge-Kutta sub-step, and therefore we here only discuss the numerical cost for one of the sub-steps. The spatial derivatives in the Maxwell equations are calculated with a pseudo-spectral method, in which the dependent variables are Fourier transformed numerically with the fast Fourier transform (FFT), multiplied by constants and inverse transformed with the inverse fast Fourier transform (IFFT). The same holds for the spatial derivatives of the particle distribution functions in the Vlasov equation. The number of floating point operations for taking the FFT and IFFT of an $N_{x1} \times N_{x2}$ array is approximately $10 \times N_{x1} N_{x2} \log_2(N_{x1} N_{x2})$. The η_1 and η_2 derivatives in the Vlasov equation is calculated with a compact differential scheme in which tri-diagonal equation system has to be solved in η_1 and η_2 space, which for an array of size $N_{\eta 1} \times N_{\eta 2}$ requires approximately $20 \times N_{\eta 1} \times N_{\eta 2}$ floating point operations. Hence, in a simple performance model for the Vlasov-Maxwell solver, the total number of floating point operations is given approximately by $M_{\text{float}} = N_\alpha \chi N_{x1} N_{x2} [2 + \log_2(N_{x1} N_{x2})] N_{\eta 1} N_{\eta 2} + \psi N_{x1} N_{x2} \log_2(N_{x1} N_{x2})$, where the first term comes from approximations of the derivatives in the Vlasov solver and the second term from the Maxwell solver. Here N_α is the number of species, and χ and ψ are

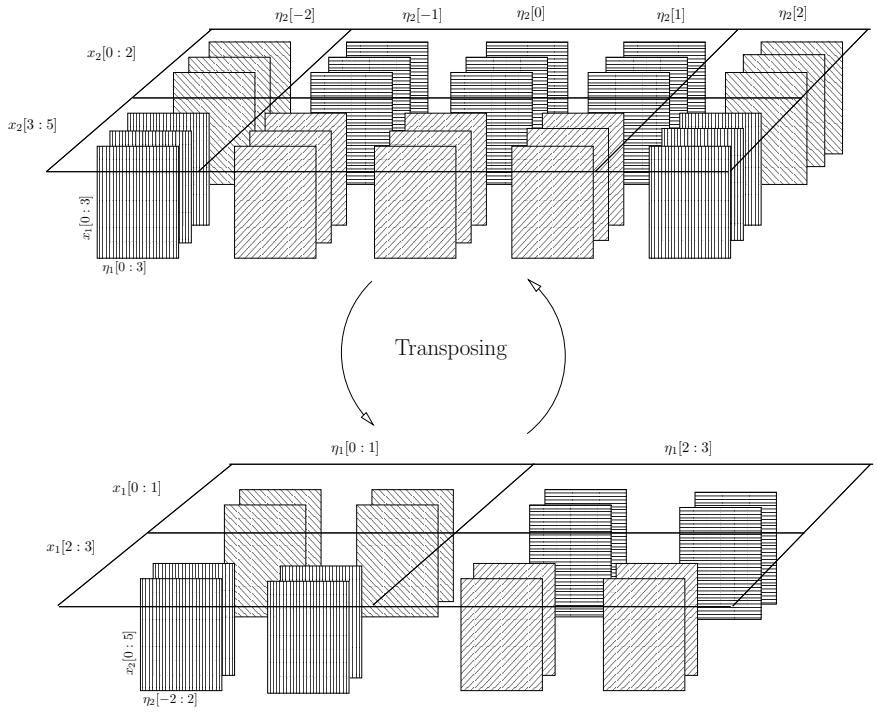

Fig. 1. In this figure we illustrate how a distribution function in the four-dimensional phase space, $\widehat{f_\alpha}\,(\eta_1[0:3], x_1[0:3], \eta_2[-2:2], x_2[0:5])$ is distributed on four processors. In this illustrated example we show the representation of $\widehat{f_\alpha}$ on the numerical grid, with 4 grid points along x_1, 4 points along η_1, 6 grid points along x_2, and 5 points along η_2. Arrays with the same pattern belong to the same processor. A redistribution of the matrix is needed when the change of dimensional dependents in the numerical operations are done. The figure shows the scheme (system) for the redistribution of the particle distribution function $\widehat{f_\alpha}$ for one particle species α. As default distribution of $\widehat{f_\alpha}$ (top panel), the domain is decomposed in the (x_2, η_2) space so each process has a full set of data of $\widehat{f_\alpha}$ in the (x_1, η_1) directions in the local processor memory, and numerical operations in (x_1, η_1) space can be performed in parallel. After transposing, the domain is decomposed in (x_1, η_1) space so that each processor has a full set of data of $\widehat{f_\alpha}$ in the (x_2, η_2) directions in the local processor memory (bottom panel). After this, operations in (x_2, η_2) can be performed in parallel. As a final step, the computational domain is transposed back to the original representation (top panel).

constants of order 10. Hence the total cost for floating point operations on one processor, $N_p = 1$, isxs

$$\begin{aligned}\Upsilon(1) = \tau_{\text{float}}\{&N_\alpha \chi N_{x1} N_{x2}[2 + \log_2(N_{x1}N_{x2})]N_{\eta 1}N_{\eta 2} \\ &+ \psi N_{x1}N_{x2}\log_2(N_{x1}N_{x2})\},\end{aligned} \tag{20}$$

where τ_{float} is the time needed to perform one floating point operation. When utilizing multiple processors, $N_p > 1$, the cost for floating point operations is divided by the number of processors, but we need to add a communication cost for transposing the distribution function, $\tau_{\text{comm}} N_{x1} N_{x2} N_{\eta 1} N_{\eta 2}$, and gather and scatter the quantities involved in the Maxwell solver $\tau_{\text{comm}} N_{x1} N_{x2}$, where τ_{comm} is the time to communicate one number in double representation. We have assumed parallelization over particle species and that $N_p > N_\alpha$. The net cost per processor becomes

$$\begin{aligned} \Upsilon(N_p) = \frac{\tau_{\text{float}}}{N_p} \{ & N_\alpha \chi N_{x1} N_{x2} [2 + \log_2(N_{x1} N_{x2})] N_{\eta 1} N_{\eta 2} \\ & + \psi N_{x1} N_{x2} \log_2(N_{x1} N_{x2}) \} \\ & + \tau_{\text{comm}} (N_{x1} N_{x2} N_{\eta 1} N_{\eta 2} + N_{x1} N_{x2}), \end{aligned} \tag{21}$$

and the speedup is calculated as

$$S_p = \frac{\Upsilon(1)}{\Upsilon(N_p)}. \tag{22}$$

For a small number of processors, so that $(\tau_{\text{float}}/N_p)(N_\alpha \chi N_{\eta 1} N_{\eta 2} + \psi) \gg \tau_{\text{comm}}(N_{\eta 1} N_{\eta 2} + 1)$ and if $N_\alpha \chi N_{\eta 1} N_{\eta 2} \gg \psi$, we have linear speedup $S_p = N_p$. In the opposite case when $(\tau_{\text{float}}/N_p)(N_\alpha \chi N_{\eta 1} N_{\eta 2} + \psi) \ll \tau_{\text{comm}}(N_{\eta 1} N_{\eta 2} + 1)$, and if $N_\alpha \chi N_{\eta 1} N_{\eta 2} \gg \psi$ and $N_{\eta 1} N_{\eta 2} \gg 1$, then the limiting speedup is

$$S_\infty = \frac{\tau_{\text{float}} N_\alpha \chi}{\tau_{\text{comm}}}. \tag{23}$$

We see that the cost of the transposing of the distribution matrix leads to a limited speedup for any number of processors.

4 Numerical performance tests

Performance tests were performed on the Sarek computer cluster at HPC2N, Umeå University. The cluster consists of 384 64-bit AMD Opteron CPUs in 192 dual nodes connected with a Myrinet-2000 High-performance interconnect (http://www.hpc2n.umu.se/resources/sarek.html). We have only used one process per cluster node to have a homogeneous environment.

We test the solver under different ways to split the distribution function and as illustrated in figure 1, and with two different implementations in MPI (`alltoallv` or `isend/irecv`) for transposing the solution matrix. The domain decomposition utilizes splitting in the spatial (x_1, x_2) and the Fourier transformed velocity (η_1 or η_2) dimensions. In the implementation of the algorithm for transposing of the distribution function we have compared the

$N_\alpha \times N_{px} \times N_{p\eta}$	`alltoallv`	`isend/irecv`
$1 \times 1 \times 1$	80:18	80:18
$2 \times 1 \times 1$	55:46	49.21
$2 \times 2 \times 1$	27:25	26:56
$2 \times 4 \times 1$	16:35	14:41
$2 \times 8 \times 1$	10:33	8:53
$2 \times 16 \times 1$	6.04	5:42
$2 \times 1 \times 1$	55:46	49.21
$2 \times 1 \times 2$	30:06	29:05
$2 \times 1 \times 4$	18:36	16:09
$2 \times 1 \times 8$	11:41	10:09
$2 \times 1 \times 16$	7:02	6:13
$2 \times 2 \times 2$	18:12	16:14
$2 \times 4 \times 4$	6:47	6:00

Table 1
Measurements of the computation time Υ (minutes:seconds) given in wall clock time for a processor grid with $N_p = N_\alpha \times N_{px} \times N_{p\eta}$ processors. We compare transposing the particle distribution matrix by means of the built-in MPI function `alltoallv` with explicit nonblocking communication by means of the `isend/irecv` functions. The speedup is illustrated in figure 2.

built-in MPI function `alltoallv` with an explicitly implemented transposing algorithm, with none-blocking send and receive (MPI `isend` and `irecv`). The results are presented in Table 1 and in Fig. 2.

In the numerical tests, we have assumed two particle species, negatively charged electrons and positively charged ions. We have used a numerical grid of size $N_{x1} = N_{x2} = 32$, $N_{\eta 1} = N_{\eta 2} = 32$ and have used $N_t = 300$ timesteps, as defined in section 3.1. In most of the simulations, we have parallelized over the particle species, so that $N_\alpha = 2$; see Table 1. This parallelization step gives a speedup of about 50%. When comparing parallelization in different dimensions, we see that there is about 10–20% higher efficiency when parallelizing in (x_1, x_2) space than in (η_1, η_2) space. We see that the parallelization in both (x_1, x_2) and (η_1, η_2) space gives an efficiency in between that of the parallelization in (x_1, x_2) space and in (η_1, η_2) space. The general method `alltoallv` is easier to implement than the explicit `isend/irecv`, while the latter method gives somewhat about 10–20% better speedup. The reason is that while `alltoallv` is blocking communication, the non-blocking `isend` and `irecv` makes it possible to desynchronize to some degree the processors and limit the wait process to a minimum. The variation in calculation time on a

multiuser computer system can be up to the same order of magnitude (10–20%) as the differences in run time. The profiling also confirmed that only a small part of the total work load is taken by the Maxwell solver. File writing and similar bottlenecks showed to be of no importance when they are done every 10th step or less frequently.

In Fig. 2, we have plotted the numerically measured speedup for the cases of parallelization in either (x_1, x_2) space or in (η_1, η_2) space, and for the two transpose methods `alltoallv` and `isend/irecv`. We see that the speedup is smaller than the theoretical linear speedup, especially for the larger number of processors, the speedup is about half of that of the linear speedup. It corresponds well with the theoretical analysis of equation (23) which predicts that the speedup becomes smaller than the linear speedup for larger number of processors. Even so, a speedup of ≈ 15 for 32 processors is acceptable, and the possibility to efficiently utilize the distributed memory on the computer cluster makes it possible to fit the particle distribution function in the computer memory for realistic problems.

5 Conclusions

We have presented a parallel algorithm for parallelizing a Vlasov-Maxwell solver in the four-dimensional phase space consisting of two spatial dimensions and two velocity dimensions. The electromagnetic fields depend on the spatial dimensions and are governed by the Maxwell equations, while the charged particles (electrons, ions) are statistically distributed in both the spatial and velocity dimensions and their dynamics is governed by one Vlasov equation per particle species. The solver uses a Fourier transform method, in which the Vlasov equation is Fourier transformed analytically in velocity space and the resulting equation is solved numerically. Pseudo-spectral methods are used for calculating the spatial derivatives and a compact, fourth-order difference scheme for the derivatives in the Fourier transformed velocity space. The parallelization is performed on different levels. On the first level, the particle distribution function for different particle species are represented on different processor groups, so that the Vlasov equation for different particle species can be performed in parallel. On a second level, within each particle species, domain decomposition is used for the particle distribution function that is is represented on a four-dimensional grid in phase space. Here we use a transpose method in which the data is redistributed among processors so that numerical approximations can be performed trivially in parallel in different directions. The challenge to make a good load balancing and a simple parallelization implementation has resulted in some compromises in performance. However, the

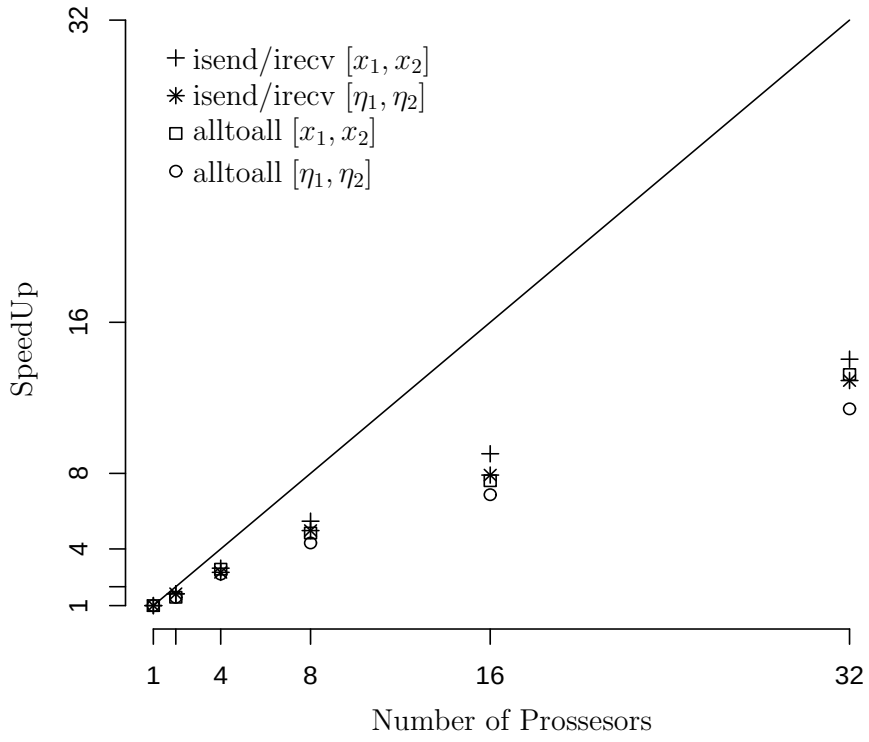

Fig. 2. The measured speedup $S = \Upsilon(1)/\Upsilon(N_p)$, derived from Table 1 for parallelization in either (x_1, x_2) space or in (η_1, η_2) space, and for the two transpose methods `alltoallv` and `isend/irecv`. The solid line shows the optimal theoretical linear speedup. Throughout the test runs we have used two particle spices resulting g in $N_\alpha = 2$ except for one single processor run, $N_\alpha = 1$.

simplicity of the algorithm gives larger possibilities to implement new features, resulting in a longer lifespan of the code. It also makes it easier to do a future extension of the parallel method to the fully 3×3-dimensional case (9).

6 Acknowledgement

This research was partially financed by the Swedish Research Council (VR). The computational resources were provided by High Performance Computing Centre North (HPC2N) and by Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX). The authors are grateful to Jarmo Rantakokko at Uppsala University for discussions and help with technical and implementation problems.

References

[1] T. D. Arber, R. G. L. Vann, A critical comparison of Eulerian-grid-based Vlasov solvers, J. Comput. Phys. 180 (2002) 339–357.

[2] T. T. Armstrong, R. C. Harding, G. Knorr, D. Montgomery, Solutions of Vlasov's equation by transform methods, vol. 9 of Methods of Computational Physics, Academic Press, New York, 1970, pp. 30-87.

[3] C. K. Birdsall, A. B. Langdon, Plasma Physics via Computer Simulation, Academic Press, 1991.

[4] P. S. Cally, Z. Sedláček, A Fourier-space description of oscillation in an inhomogeneous plasma. Part 2. Discrete approach, J. Plasma Phys. 52 (1994) 265–296.

[5] C. Z. Cheng, G. Knorr, The integration of the Vlasov equation in configuration space, J. Comput. Phys. 22 (3) (1976) 330–351.

[6] B. Eliasson, Outflow boundary conditions for the Fourier transformed two-dimensional Vlasov equation, J. Comput. Phys. 181 (1) (2002) 98–125.

[7] B. Eliasson, Numerical modelling the Fourier transformed two-dimensional Vlasov-Maxwell system, J. Comput. Phys. 190 (2) (2003) 501–522.

[8] B. Eliasson, The parallel implementation of the one-dimensional Fourier transformed Vlasov–Poisson system, Comput. Phys. Commun. 170 (2) (2005) 205–230.

[9] B. Eliasson, Outflow boundary conditions for the Fourier transformed three-dimensional Vlasov-Maxwell system, J. Comput. Phys. 225 (2007) 1508–1532.

[10] N. V. Elkina, J. Büchner, A new conservative unsplit method for the solution of the Vlasov equation, J. Comput. Phys. 213 (2006) 862–875.

[11] F. Filbet, E. Sonnedrücker, P. Bertrand, Conservative numerical schemes for the Vlasov equation, J. Comput. Phys. 172 (2001) 166.

[12] F. Filbet, E. Sonnendrücker, Comparison of Eulerian Vlasov solvers, Comput. Phys. Commun. 150 (2003) 247–266.

[13] I. T. Foster, P. H. Worley, Parallel algorithms for the spectral transform method, SIAM Journal on Scientific Computing 18 (2) (1997) 806–837.

[14] A. Ghizzo, F. Huot, P. Bertrand, A non-periodic 2D semi-Lagrangian Vlasov code for laser-plasma interaction on parallel computer, J. Comput. Phys. 186 (1) (2003) 47–69.

[15] L. Gibelli, B. D. Shizgal, Spectral convergence of the Hermite basis function solution of the Vlasov equation: The free-streaming term, J. Comput. Phys. 219 (2006) 477–488.

[16] M. Kanamitsu, H. Kanamaru, Y. Cui, H. Juang, Parallel implementation of the regional spectral atmospheric model, Tech. rep., Scripps Institution of Oceanography, University of California at San Diego, and National Oceanic and Atmospheric Administration for the California Energy Commission, PIER Energy-Related Environmental Research. CEC-500-2005-

014. (2005).
[17] A. J. Klimas, W. M. Farrel, A splitting algorithm for Vlasov simulation with filamentation filtration, J. Comput. Phys. 110 (1994) 150.
[18] A. Mangeney, F. Califano, C. Cavazzoni, P. Travnicek, A numerical scheme for the integration of the Vlasov-Maxwell system of equations, J. Comput. Phys. 179 (2) (2003) 495–538.
[19] H. Matsumoto, Y. Omura, Computer Space Plasma Physics: Simulation Techniques and Software, Terra Scientific Publishing Company, 1993.