Journal of Computational Physics 283 (2015) 436–452

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

# Coupled Vlasov and two-fluid codes on GPUs

M. Rieke, T. Trost, R. Grauer [*]

*Institut für Theoretische Physik I, Ruhr-Universität Bochum, 44801 Bochum, Germany*

ARTICLE INFO

*Article history:*
Received 20 June 2014
Received in revised form 28 November 2014
Accepted 1 December 2014
Available online 15 December 2014

*Keywords:*
Coupling
Vlasov equation
Multifluid
GPU
Whistler waves
Reconnection
GEM

ABSTRACT

We present a way to combine Vlasov and two-fluid codes for the simulation of a collisionless plasma in large domains while keeping full information on the velocity distribution in localised areas of interest. This is made possible by solving the full Vlasov equation in one region while the remaining area is treated by a 5-moment two-fluid code. In such a treatment, the main challenge of coupling kinetic and fluid descriptions is the interchange of physically correct boundary conditions between the different plasma models. In contrast to other treatments, we do not rely on any specific form of the distribution function, e.g. a Maxwellian type. Instead, we combine an extrapolation of the distribution function and a correction of the moments based on the fluid data. Thus, throughout the simulation both codes provide the necessary boundary conditions for each other. A speed-up factor of around 10 is achieved by using GPUs for the computationally expensive solution of the Vlasov equation. Additional major savings are obtained due to the coupling where the amount of savings roughly corresponds to the fraction of the domain where the kinetic equations are solved. The coupled codes were then tested on the propagation of whistler waves and on the GEM reconnection challenge.

## 1. Introduction

For many relevant problems in plasma physics, the plasma can be considered nearly collisionless. In these cases, the Vlasov–Maxwell system is sufficient for a complete description of the physically important phenomena. Nevertheless, it is close to impossible to numerically solve the Vlasov equation for global problems on fluid scales due to the vast amount of memory needed to resolve a 6-dimensional phase-space. However, in many situations a complete kinetic description is not necessary and computationally much less expensive fluid models based on the moments of the distribution function are sufficient. In addition, plasmas are typical multi-scale phenomena. Often there are only localised and small regions, where a kinetic description is needed, whereas the rest of the domain can be described by means of a fluid model. This separation of scales can be found in many typical problems of plasma physics, such as reconnection in the magnetotail of the earth, solar flares or typical phenomena in tokamaks. To determine what amount of computational savings coupling techniques might provide, a rough estimation of the numerical costs can be made on the basis of spatial scales. To do this, both the computational effort for the fluid region and the overhead for the coupling itself can be assumed as negligible compared to the costs of the kinetic part of the simulation. Thus the fraction of the domain where kinetic effects occur can indicate how much computational time might be saved. This fraction can be estimated on the basis of typical plasma parameters.

[*] Corresponding author.
*E-mail address:* grauer@tp1.rub.de (R. Grauer).

http://dx.doi.org/10.1016/j.jcp.2014.12.016

The most relevant scales are determined by the ion skin depth, below which the dynamics of ions and electrons decouple, and the ion gyro radius, which is especially important in the context of collisionless plasmas.

In the magnetotail of the earth, kinetic effects are important in and around the current sheet, where reconnection takes place. In an example of a typical setup for this problem given by Birn et al. [1], the current sheet—and thus the kinetic region—covers around 10% of the computational domain. In a typical tokamak with an overall size $\sim$1 m, the ion skin depth is $\sim$5 cm (see Woods [2]). The ion gyro radius is even one order of magnitude smaller. Thus, taking into account the rotationally symmetric geometry of a tokamak and the magnetic field therein, the fraction of the domain where localised kinetic effects as reconnection might take place is likely to scale with the fraction of 5 cm/1 m and can thus be expected to be only 5% of the domain, to give a very rough estimation of the order of magnitude. In a solar flare, the ratio is even smaller: A typical size of $\sim 10^7$ m compares to an ion skin depth of about 10 m (see for example Ramesh et al. [3]). We stress that this is a rough estimate assuming e.g. that there are isolated reconnection sites and no turbulent reconnection events where the turbulence scales meet the kinetic scales. In this situation, the fraction of the simulation domain occupied by the kinetic region would be substantially larger.

If the kinetic regions are known, this additional information might be used to reduce the computational costs drastically by restricting the expensive kinetic simulation to this smaller area and using a suitable fluid model for the rest of the domain. There have been several approaches to exploit this ansatz in numerical schemes. In the works of Degond et al. [4], Dellacherie [5], Goudon et al. [6], Klar et al. [7], and Le Tallec and Mallinger [8] methods for coupling the Boltzmann equation to some kind of fluid model like the Navier–Stokes or Euler equations are addressed. Sugiyama and Kusano [9] and especially Daldorff et al. [10] describe the coupling of PIC and MHD models, Markidis et al. [11] show a way to combine PIC and fluid models, and Kolobov and Arslanbekov [12] describe the transition from neutral gas models to models of weakly ionised plasmas with respect to coupling kinetic and fluid equations. Schulze et al. [13] use coupling techniques in the context of epitaxial growth.

Most of these schemes are based on the strong assumption that the distribution function is close to Maxwellian in the vicinity of the coupling border and use this approximation to generate boundary conditions for the kinetic region. In other approaches, an artificial field is introduced that expresses the deviation from a Maxwellian and is coupled to the underlying equations [4], or the fluxes at the boundary are used for the coupling [8].

This paper presents a different mechanism for coupling the Vlasov equation with a five-moment model and the full Maxwell equations. Our method is easy to implement and does not rely on any specific assumptions with respect to the distribution function near the coupling border. In addition, no extensive transition region between the kinetic and fluid domains is required.

The rest of this paper is organised as follows: After the physical models have been described in Section 2, the numerical schemes are presented in Section 3. The coupling procedure is explained in Section 4. Technical details on CUDA graphics cards and their use on the Vlasov code as well as parallel programming issues are discussed in Section 5. Finally, in Sections 6 and 7, first numerical results are presented and discussed.

## 2. Physical models

Consider a non-relativistic plasma consisting of different particle species $s$. The Vlasov equation

$$\partial_t f_s + \mathbf{v}\cdot\nabla_{\mathbf{x}} f_s + \frac{q_s}{m_s}(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\nabla_{\mathbf{v}} f_s = 0 \tag{1}$$

describes the time evolution of the phase-space density $f_s(\mathbf{x},\mathbf{v},t)$, where $\mathbf{E}$ is the electric field, $\mathbf{B}$ is the magnetic field, $q_s$ is the charge, and $m_s$ is the mass per particle. By calculating the moments of $f_s$ with respect to $\mathbf{v}$, we can derive important physical quantities describing the plasma:

the mass density $$\rho_s = m_s \int f_s \mathrm{d}^3 v, \tag{2}$$

the momentum density $$\mathbf{u}_s = m_s \int \mathbf{v} f_s \mathrm{d}^3 v, \tag{3}$$

and the energy density $$\mathcal{E}_s = \frac{m_s}{2}\int \mathbf{v}^2 f_s \mathrm{d}^3 v. \tag{4}$$

Their time dependence can be calculated from the Vlasov equation (1) and is given by:

$$\partial_t \rho_s = -\nabla\cdot\mathbf{u}_s \tag{5}$$

$$\partial_t \mathbf{u}_s = -\nabla\cdot\left(\frac{\mathbf{u}_s\otimes\mathbf{u}_s}{\rho_s}+\mathbb{P}_s\right)+\frac{q_s}{m_s}(\rho_s\mathbf{E}+\mathbf{u}_s\times\mathbf{B}_s) \tag{6}$$

$$\partial_t \mathcal{E}_s = -\nabla\cdot\left(\frac{\mathcal{E}_s\mathbb{1}+\mathbb{P}_s}{\rho_s}\mathbf{u}_s\right)-\nabla\cdot\mathbf{Q}_s+\frac{q_s}{m_s}\mathbf{u}_s\cdot\mathbf{E}, \tag{7}$$

where $\mathbb{P}_s$ is the pressure tensor and $\mathbf{Q}_s$ is the heat flux tensor. These quantities must be provided in order to close the system.

For our first tests of the coupling procedure, we assume that the plasma is adiabatic and the pressure is isotropic, which corresponds to

$$\mathbb{P}_s = p_s \mathbb{1}, \quad p_s = \frac{2}{3}\mathcal{E}_s - \frac{1}{3}\frac{\mathbf{u}_s^2}{\rho_s}, \quad \text{and} \quad \nabla \cdot \mathbf{Q}_s = 0 \tag{8}$$

respectively. Note that this fluid closure was chosen for simplicity and other fluid models may be used depending on the situation of the plasma application.

Furthermore Faraday's and Ampère's laws

$$\partial_t \mathbf{B} = -\nabla \times \mathbf{E} \tag{9}$$

$$\partial_t \mathbf{E} = c^2(\nabla \times \mathbf{B} - \mu_0 \mathbf{j}) \tag{10}$$

are required to determine how the electromagnetic fields evolve where current density $\mathbf{j}$ is given by

$$\mathbf{j} = \sum_s \frac{q_s}{m_s}\mathbf{u}_s. \tag{11}$$

## 3. Numerical schemes

In order to solve the coupling problem, we must first design numerical schemes for solving the Vlasov equation, the five-moment two-fluid equations and the Maxwell equations separately. It is important to note that we have chosen the fluid model with care such that we do not impose a generalised Ohm's law to obtain the electric field $\mathbf{E}$ but use the same set of Maxwell equations as for the Vlasov equations. This implies that timescales of both fluid and kinetic descriptions are comparable. This allows for a clean separation between the Maxwell solver, the Vlasov solver and the fluid solver. That is, the Maxwell solver only requires the current density as a source term, which can be obtained from the fluid and the Vlasov solver.

Coupling to even larger spatial and temporal scales which could be described with an MHD model, would thus be more related to the question of coupling Maxwell's equation to a generalised Ohm's law. This kind of coupling can be done on the fluid level (2-fluid with Maxwell equation coupled to MHD with generalised Ohm's law). This is still a major task and will be the next necessary step.

Both models, the Vlasov equation and the two-fluid equations, are implemented independently and each code can be executed alone or coupled to the other—a concept known as *multiple program multiple data* (MPMD).

Because of the high dimensionality of the distribution function $f_s$, the Vlasov code is much more computationally expensive than the fluid code. Therefore, to achieve optimal performance, the Vlasov code is executed on multiple GPUs, while the relatively low cost fluid model and Maxwell's equations (9) and (10) are solved on CPUs. The network communication is implemented via MPI. Our code solves the 2.5-dimensional problem (i.e. two space dimensions and three velocity dimensions), but a generalisation to the full 3-dimensional case is straightforward. In the following subsections we will describe the numerical schemes. In addition to that, a detailed description of semi-Lagrangian solvers is given because they are essential both to the Vlasov code and to the coupling routine that will be explained in Section 4.

### 3.1. Numerical implementation of the Vlasov equation

In this section, the numerical solver for the Vlasov equation used in this work is described. By means of splitting methods, the Vlasov equation is broken down into a set of simpler one-dimensional conservation laws. These can then be solved with a semi-Lagrangian scheme. Such a scheme requires finding the characteristics for which we use the Backsubstitution method, and approximating the integrals between these characteristics, which we do by means of the *positive flux conservative scheme* (PFC scheme).

#### 3.1.1. Conservative semi-Lagrangian schemes

The underlying idea of how to solve conservation laws by means of semi-Lagrangian schemes is now described. As an example consider

$$\partial_t f(x,t) + \partial_x\big(u(x,t) f(x,t)\big) = 0. \tag{12}$$

Let the domain be divided into equal cells and the $i$th cell be bounded by the points $x_i$ and $x_{i+1}$. Different points in time, separated by the time interval $\Delta t$, are denoted as $t^n = n\Delta t$. The scheme will then evolve the cell integrals

$$\overline{f_i}(t^n) = \int_{x_i}^{x_{i+1}} f\left(x, t^n\right) \mathrm{d}x \tag{13}$$

in time. Next, a family of curves $\mathcal{C}(t; x, s)$ is introduced, called the *characteristics* of Eq. (12), which are defined by

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{C}(t) = u\big(\mathcal{C}(t), t\big) \tag{14}$$

$$\mathcal{C}(t = s) = x. \tag{15}$$

Integrating $f$ along $x$ between the two curves that pass through the cell boundaries of cell $i$ at time $t^{n+1}$, taking the time derivative, using the chain rule for parameter integrals, and inserting the conservation law (12) leads to

$$\frac{\mathrm{d}}{\mathrm{d}t} \int\limits_{\mathcal{C}(t;x_i,t^{n+1})}^{\mathcal{C}(t;x_{i+1},t^{n+1})} f(x,t)\,\mathrm{d}x = 0. \tag{16}$$

If the velocity field $u(x,t)$ is Lipschitz continuous, the Picard–Lindelöf theorem will guarantee that two different characteristics will never intersect. Then we may integrate over time to obtain

$$\overline{f_i}\big(t^{n+1}\big) = \int\limits_{\mathcal{C}(t^n;x_i,t^{n+1})}^{\mathcal{C}(t^n;x_{i+1},t^{n+1})} f\big(x, t^n\big)\,\mathrm{d}x. \tag{17}$$

Here $\overline{f_i}(t^n)$ and the points $\mathcal{C}(t^n; x_i, t^{n+1})$ are known, the cell integrals $\overline{f_i}(t^{n+1})$ can be found by reconstructing the right hand side of Eq. (17) from the $\overline{f_i}(t^n)$ by means of an arbitrary reconstruction scheme. In other words, in order to solve the conservation law (12), it is sufficient to find the source points of the characteristics that will pass through the cell boundaries.

### 3.1.2. Splitting

The phase space density $f_S$ depends on six coordinates plus time. Trying to solve the Vlasov equation in a naive way by simply discretising it based on an arbitrary scheme would lead to severe problems: If the number of dimensions is $d$, the number of direct neighbours of a given cell is $3^d - 1$. Even if one spatial dimension can be left out because of the symmetry of the problem, this still results in 242 cells. A stencil spanning a larger part of these cells, would produce a very clumsy and slow scheme. This exponential growth of the number of dependencies can be avoided by splitting the Vlasov equation into smaller parts that are more manageable and can be solved separately. If this is done carefully, the resulting error does not exceed the one of the overall scheme.

The Vlasov equation can be written in the form

$$\partial_t f_s + (\mathcal{A} + \mathcal{B}) f_s = 0 \tag{18}$$

with operators

$$\mathcal{A} = \mathbf{v} \cdot \nabla_{\mathbf{x}}$$

and

$$\mathcal{B} = \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}}.$$

Instead of solving the full equation (18), we consider the equations

$$\partial_t f_s + \mathcal{A} f_s = \partial_t f_s + \mathbf{v} \cdot \nabla_{\mathbf{x}} f_s = 0 \tag{19}$$

and

$$\partial_t f_s + \mathcal{B} f_s = \partial_t f_s + \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}} f_s = 0. \tag{20}$$

Now, given some differential equation

$$\partial_t f + \mathcal{S} f = 0, \quad f(t = 0) = f_0 \tag{21}$$

we can write its formal solution as $f(\Delta t) = \exp(-\Delta t \mathcal{S}) f_0 =: T(\mathcal{S}) f_0$.

With this notation in mind we can decompose the solution of (18) in terms of solutions of (19) and (20) by making use of the Strang splitting method:

$$f_s(t + \Delta t) = T(\mathcal{A} + \mathcal{B}) f(t) = T\left(\frac{1}{2}\mathcal{A}\right) T(\mathcal{B}) T\left(\frac{1}{2}\mathcal{A}\right) f(t) + \mathcal{O}\big(\Delta t^3\big). \tag{22}$$

This decomposition is exact if the operators $\mathcal{A}$ and $\mathcal{B}$ commute. However, this is not the case here, but the resulting error is of third order in time and thus does not destroy the overall accuracy. Now that the problem has been split into two sub problems, each one can be solved separately. In both (19) and (20) the operators still act on a three-dimensional space. It is natural to try to decompose the operators $\mathcal{A}$ and $\mathcal{B}$ further in order to simplify the problem even more and make it possible to solve it with a semi-Lagrangian scheme which only works for one dimension. As for $\mathcal{A}$, which is the sum of three commuting operators, this splitting is exact and can easily be done by a simple composition of the single operators.

The operator $\mathcal{B}$, however, does not allow for such an easy decomposition. Here, it would be possible to use Strang splitting again, but there would be some disadvantages in this. First, the resulting scheme would be non-isotropic, as the single operators would not be used the same number of times. Second, the application of these operators is particularly expensive, so it is not desirable to apply them more than necessary.

Thus, the next problem is to find a splitting so that Eq. (20) can be solved by only three applications of the time-evolution operator (one for each dimension of $\mathbf{v}$) and obtain at least second-order accuracy in space and time. We rewrite (20) as

$$\partial_t f_s + \mathcal{B} f_s = \partial_t f_s + (\mathcal{V}_x + \mathcal{V}_y + \mathcal{V}_z) f_s = 0 \tag{23}$$

with

$$\mathcal{V}_i = \frac{q_s}{m_s}\big(E_i + (\mathbf{v}\times\mathbf{B})_i\big)\partial_i$$

and make the ansatz

$$f(t+\Delta t) = T(\tilde{\mathcal{V}}_z)T(\tilde{\mathcal{V}}_y)T(\tilde{\mathcal{V}}_x)f(t) + E_{\text{split}} + E_{\text{mod}} \tag{24}$$

with modified operators $\tilde{\mathcal{V}}_i$ that still have to be determined. The term $E_{\text{split}}$ represents the error due to the splitting scheme, while $E_{\text{mod}}$ is the error caused by modifying the operators. If the modifications are chosen correctly, both error terms cancel out and the splitting is exact. It should be noted however, that the error term of the splitting depends on the order in which the solution operators are applied. Thus it is not possible to change the order of execution of the $T(\tilde{\mathcal{V}}_i)$ once the modifications have been chosen. The implementation of (24) is done by means of the Backsubstitution method, which is described in the next section.

### 3.1.3. The Backsubstitution method

The Backsubstitution method [14,15] relies on the calculation of the characteristics needed for the semi-Lagrangian schemes that occur in the single operators in (24) by means of the Boris push method. The latter is best known from the context of PIC simulations.

We start with a review of the Boris push method. Considering the discretized equation

$$\frac{\mathbf{v}^{n+1}-\mathbf{v}^n}{\Delta t} = \frac{q_\alpha}{m_\alpha}\left(\mathbf{E}^{n+1/2} + \frac{\mathbf{v}^{n+1}+\mathbf{v}^n}{2}\times\mathbf{B}^{n+1/2}\right), \tag{25}$$

the effects of the electric and magnetic fields can be separated by introducing new variables

$$\mathbf{v}^+ = \mathbf{v}^{n+1} - \frac{q_\alpha}{m_\alpha}\frac{\Delta t}{2}\mathbf{E}^{n+1/2} \quad\text{and}\quad \mathbf{v}^- = \mathbf{v}^n + \frac{q_\alpha}{m_\alpha}\frac{\Delta t}{2}\mathbf{E}^{n+1/2}. \tag{26}$$

With that, Eq. (25) becomes

$$\frac{\mathbf{v}^+-\mathbf{v}^-}{\Delta t} = \frac{q_\alpha}{m_\alpha}\frac{\mathbf{v}^++\mathbf{v}^-}{2}\times\mathbf{B}^{n+1/2}. \tag{27}$$

This equation is solved in two steps. Introducing the quantities

$$\mathbf{t} = \frac{q_\alpha}{m_\alpha}\frac{\Delta t}{2}\mathbf{B} \quad\text{and}\quad \mathbf{s} = \frac{2\mathbf{t}}{1+t^2}, \tag{28}$$

the first step calculates an intermediate vector $\mathbf{v}'$ from $\mathbf{v}^-$, the second step uses this intermediate vector to calculate the result $\mathbf{v}^+$

$$\mathbf{v}' = \mathbf{v}^- + \mathbf{v}^- \times \mathbf{t} \tag{29}$$

$$\mathbf{v}^+ = \mathbf{v}^- + \mathbf{v}' \times \mathbf{s} \tag{30}$$

There is also a backward-in-time version of the Boris push that calculates $\mathbf{v}^-$ from $\mathbf{v}^+$:

$$\tilde{\mathbf{v}}' = \mathbf{v}^+ - \mathbf{v}^+ \times \mathbf{t} \tag{31}$$

$$\mathbf{v}^- = \mathbf{v}^+ - \tilde{\mathbf{v}}' \times \mathbf{s}. \tag{32}$$

A big advantage of the Boris push is that it correctly conserves energy respectively the magnitude of $\mathbf{v}$ if only magnetic and no electric fields are present.

Following the idea of cascade interpolation [16], the first step of the update in velocity space will be performed along the $\mathbf{v}_x$ direction, the second step along $\mathbf{v}_y$, and the last step along $\mathbf{v}_z$. For the last step, the distance along the $\mathbf{v}_z$ direction is just the $z$ component of the characteristic ending exactly at the grid point. Since we are tracing the characteristics back in time, the grid-point coordinates are denoted by $\mathbf{v}^{n+1}$. By using the new variables (26), the electric fields can be removed from the equations and the grid-point coordinates become $\mathbf{v}^+$. The distance for the $\mathbf{v}_z$ update is then given by a formula $v_z^- = v_z^-(v_x^+, v_y^+, v_z^+)$. This notation means that $v_z^-$ is a function of the variables $v_x^+$, $v_y^+$, and $v_z^+$. Using the $z$ component of Eqs. (31) and (32), one obtains

$$v_z^- = v_x^+(t_z s_x - s_y) + v_y^+(t_z s_y + s_x) + v_z^+(1 - t_x s_x - t_y s_y) \tag{33}$$

for the sought-after function. The step along the $\mathbf{v}_y$ direction follows a characteristic passing through $v_z^+$ at time $t - \Delta t$ and $v_y^+$ and $v_x^+$ at time $t$. This means that an equation of the form $v_y^- = v_y^-(v_x^+, v_y^+, v_z^-)$ is needed. This can be obtained by solving Eq. (33) for $v_z^+$ and inserting the result into the $y$ component of Eq. (32), yielding

$$\begin{aligned} v_y^- &= v_x^+\left(s_x t_y + s_z - \frac{(t_z s_x - s_y)(t_y s_z - s_x)}{1 - t_x s_x - t_y s_y}\right) \\ &\quad + v_y^+\left(1 - t_x s_x - t_z s_z - \frac{(t_z s_y + s_x)(t_y s_z - s_x)}{1 - t_x s_x - t_y s_y}\right) \\ &\quad + v_z^- \frac{t_y s_z - s_x}{1 - t_x s_x - t_y s_y}. \end{aligned} \tag{34}$$

For the last step, the equation for $v_z^-$ has the form $v_z^- = v_z^-(v_x^+, v_y^-, v_z^-)$. In the original version of the Backsubstitution method [14], this equation was obtained in a similar way by solving Eqs. (33) and (34) for $v_z^+$ and $v_y^+$, respectively, and "substituting them back" into the $x$ component of Eq. (32). However, this results in a lengthy calculation, and it is difficult to simplify the resulting expression. There is an easier way to obtain the final result using the forward-in-time version of the Boris push and solving the $x$ component of Eq. (30) for $v_x^-$, which results in

$$v_x^- = \frac{v_x^+ - v_y^-(t_x s_y + s_z) - v_z^-(t_x s_z - s_y)}{1 - t_y s_y - t_z s_z}. \tag{35}$$

Eqs. (35), (34), and (33) provide the starting points for the one-dimensional updates along the $\mathbf{v}_x$, $\mathbf{v}_y$, and $\mathbf{v}_z$-axis, respectively. The order in which these steps must be applied is $\mathbf{v}_x$, then $\mathbf{v}_y$, then $\mathbf{v}_z$.

### *3.1.4. The PFC scheme*

In this section we present a brief review of the positive flux-conservative (PFC) scheme developed by Filbet et al. [17], which is used in this work and provides a method for calculating the integral in Eq. (17). For a comparison of other schemes that might be used for the same purpose, see Crouseilles et al. [18].

The PFC scheme is used to approximate the function $f$ on the basis of its cell integral $f_i$ in the $i$th cell. Here, the case of a positive velocity $u$ is discussed. The case $u < 0$ can be derived analogously. From Eq. (17) it holds that

$$f_i^n = \int_{x_{i-1/2}}^{x_{i+1/2}} f\left(x, t^n\right)\mathrm{d}x = \int_{X(t^{n-1}; x_{i-1/2}, t^n)}^{X(t^{n-1}; x_{i+1/2}, t^n)} f\left(x, t^{n-1}\right)\mathrm{d}x. \tag{36}$$

By setting

$$\Phi_{i+1/2} = \int_{X(t^{n-1}; x_{i+1/2}, t^n)}^{x_{i+1/2}} f\left(x, t^{n-1}\right)\mathrm{d}x, \tag{37}$$

Eq. (36) can be rewritten as an update equation for the $f_i$:

$$f_i^n = f_i^{n-1} + \Phi_{i-1/2} - \Phi_{i+1/2}. \tag{38}$$

The primitive function of $f$ at the cell borders is given by

$$F\left(t^n, x_{k-1/2}\right) = \sum_{i=0}^{k-1} f_i^n. \tag{39}$$

An approximation $\widetilde{F}$ of $F$ within the $i$th cell can be obtained by the Newton interpolation of $F$ on the cell borders $\{x_{i-3/2}, x_{i-1/2}, x_{i+1/2}, x_{i+3/2}\}$:

$$\begin{aligned} \widetilde{F}(x) &= F(x_{i-3/2}) + f_{i-1}(x - x_{i-3/2}) + \frac{f_i - f_{i-1}}{2\Delta x}(x - x_{i-3/2})(x - x_{i-1/2}) \\ &\quad + \frac{f_{i+1} - 2f_i + f_{i-1}}{6(\Delta x)^2}(x - x_{i-3/2})(x - x_{i-1/2})(x - x_{i+1/2}) \end{aligned} \tag{40}$$

By differentiating and rearranging the terms, one obtains an approximation $\widetilde{f}$ in the $i$th cell:

$$\widetilde{f}(x) = \frac{\mathrm{d}\widetilde{F}(x)}{\mathrm{d}x}(x) \tag{41}$$

$$= f_i + \frac{f_{i+1} - f_i}{6(\Delta x)^2}\big(2(x - x_i)(x - x_{i-3/2}) + (x - x_{i-1/2})(x - x_{i+1/2})\big) + \frac{f_i - f_{i-1}}{6(\Delta x)^2}\big(2(x - x_i)(x - x_{i+3/2}) + (x - x_{i-1/2})(x - x_{i+1/2})\big) \tag{42}$$

This reconstruction can create negative values of $\widetilde{f}(x)$, or values larger than $f_\infty = \max(f(x))$, which would be a nonphysical solution to the Vlasov equation. Thus, slope limiters are introduced:

$$\epsilon_i^+ = \begin{cases} \min(1; 2f_i/(f_{i+1} - f_i)) & \text{if } f_{i+1} > f_i \\ \min(1; -2(f_\infty - f_i)/(f_{i+1} - f_i)) & \text{if } f_{i+1} < f_i, \end{cases} \tag{43}$$

$$\epsilon_i^- = \begin{cases} \min(1; 2(f_\infty - f_i)/(f_i - f_{i-1})) & \text{if } f_i > f_{i-1} \\ \min(1; -2f_i/(f_i - f_{i-1})) & \text{if } f_i < f_{i-1}, \end{cases} \tag{44}$$

and the approximation becomes

$$\widehat{f}_i(x) = f_i + \epsilon_i^+ \frac{f_{i+1} - f_i}{6(\Delta x)^2}\big(2(x - x_i)(x - x_{i-3/2}) + (x - x_{i-1/2})(x - x_{i+1/2})\big) + \epsilon_i^- \frac{f_i - f_{i-1}}{6(\Delta x)^2}\big(2(x - x_i)(x - x_{i+3/2}) + (x - x_{i-1/2})(x - x_{i+1/2})\big). \tag{45}$$

The flux $\Phi_{i+1/2}$ can be approximated by integrating $\widehat{f}_i(x)$ from $X(t^{n-1}; x_{i+1/2}, t^n)$ to $x_{i+1/2}$ (see Eq. (37)). It was assumed that the velocity $u$ used in calculating the characteristic $X$ is positive. Because the characteristic is followed backwards in time, its footpoint lies to the left of $x_{i+1/2}$. However, due to the semi-Lagrangian nature of this scheme, the point $X(t^{n-1}; x_{i+1/2}, t^n)$ might not be located in the $i$th cell, but multiple cells further to the left. Setting $j$ so that $X(t^{n-1}; x_{i+1/2}, t^n)$ lies in the $j$th cell and defining $\alpha_i = (x_{j+1/2} - X(t^{n-1}; x_{i+1/2}, t^n))/\Delta x$ one obtains

$$\Phi_{i+1/2} = \sum_{k=j+1}^{i} f_k + \alpha_i\left(f_j + \frac{\epsilon_j^+}{6}(1 - \alpha_i)(2 - \alpha_i)(f_{j+1} - f_j) + \frac{\epsilon_j^-}{6}(1 - \alpha_i)(1 + \alpha_i)(f_j - f_{j-1})\right). \tag{46}$$

For negative velocities, $\alpha_i = (x_{j-1/2} - X(t^{n-1}; x_{i+1/2}, t^n))/\Delta x$ can be used in

$$\Phi_{i+1/2} = -\sum_{k=j+1}^{i} f_k + \alpha_i\left(f_j - \frac{\epsilon_j^+}{6}(1 - \alpha_i)(1 + \alpha_i)(f_{j+1} - f_j) - \frac{\epsilon_j^-}{6}(2 + \alpha_i)(1 + \alpha_i)(f_j - f_{j-1})\right). \tag{47}$$

If the $\alpha_i$ are known for each cell, these formulas together with Eq. (38) provide the PFC scheme.

### *3.2. Numerical implementation of the two-fluid model*

For the discretisation in space, we use the CWENO scheme [19], an easy to implement third order finite-volume scheme. The third order strong-stability-preserving Runge–Kutta scheme by Shu and Osher [20] is employed for the time integration. See also Hakim et al. [21] as a reference for numerical two-fluid schemes.

### *3.3. Maxwell solver*

The electromagnetic fields live on a Yee grid [22]. They are evolved through the FDTD method presented in [23]. Since the speed of light exceeds all other speeds found in the plasma by far, subcycling is used in order to resolve lightwaves while keeping the global timestep as large as possible.

To obtain the electromagnetic fields and the sources at the correct positions, we use a linear interpolation. This results in a second order accurate scheme which is consistent with the Vlasov code.

## 4. Coupling

The three separate codes, the Vlasov code, the multifluid code, and the Maxwell code have to be coupled in order to complete our multiphysics approach to plasma simulations. The Maxwell solver is mostly independent of the other two schemes and can be solved globally in the complete domain. As a source term, it requires only the local current density, which can easily be calculated from the phase space distribution functions or its first moments. Similarly, the electromagnetic fields appear as (space and time dependent) source terms in the fluid and Vlasov equation and are directly provided by the Maxwell solver. Thus, no restrictions with respect to the domain or the other two schemes are implied by the way in which Maxwell's equations are solved. There is a further advantage to this approach: Because of the far-reaching separation between the Maxwell solver and the other schemes, the global timestep is not limited by the speed of light. During each step of the fluid solvers, several subcycling steps of the Maxwell scheme can be carried out in order to resolve light waves. If the interplay is organised correctly, the resulting overall error is still of second order in the global time step. As each step of the Vlasov scheme is very expensive, this separation of time scales is quite desirable and efficient.

The numerical fluid and Vlasov solvers presented above do not require any non-local operations in physical space. This makes it possible to subdivide the physical domain into arbitrary pieces or blocks, in which the equations can be solved independently. Over one time step, each block has to only provide the boundary conditions for its neighbouring blocks. In principle, this makes it possible to mix fluid and kinetic blocks in an arbitrary way as long as meaningful boundary conditions can be provided. Providing boundary data for the fluid solver from kinetic data is simple and can be achieved by calculating the necessary moments via integration of the phase space density. The difficulty comes in provisioning the boundary conditions for the Vlasov solver at the boundary to the fluid region. Even if an arbitrarily high (yet finite) number of moments is known, there is no unique way of deriving a correct phase space density. The necessary information is simply missing and can only be guessed on the basis of further assumptions. Our approach here is to extrapolate the shape of the distribution function at the border of the kinetic region into the cells, where the new distribution is needed. This extrapolated phase space density is then slightly altered such that it has the correct lowest moments, which are known from the fluid region. Thus, the only assumption we enforce is that the shape of the distribution function does not change too rapidly near the coupling border. There are neither any assumptions with respect to any specific form of the distribution function, nor is there any need for an extensive transition region. In addition, this coupling method is not limited to the simple fluid model shown above and a huge class of fluid models may be used in this spirit.

In addition, because electrons and ions are treated separately and are only coupled via the currents and electromagnetic fields, no quasi-neutrality is required in the overall scheme.

### *4.1. Boundary data for the two-fluid regions*

In the Vlasov regions, the full phase-space densities $f_s$ are known and the boundary data needed by the two-fluid code can be calculated by means of the moment equations (2)–(4). The Runge–Kutta scheme that is used for updating the fluid data from time $t$ to time $t + \Delta t$ needs boundary conditions at times $t$, $t + \Delta t/2$ and $t + \Delta t$. Thus the Vlasov data has to be advanced before the fluid data. Then, by linear interpolation in time, the intermediate boundary conditions can be calculated and the fluid simulation can be advanced.

### *4.2. Boundary data for the Vlasov regions*

To update the Vlasov data from $t$ to $t + \Delta t$, boundary conditions are required only at the initial time $t$. The full phase-space density is required though and the moments provided by the fluid code are in general not sufficient to determine $f_s$. If, however, the coupling boundary is placed in a region that changes slowly with respect to typical phenomena of the plasma, then it is appropriate to assume that the shape of $f_s$ will not change significantly over the distance of a few grid cells—otherwise the simulation would be underresolved. We may then construct the phase-space density in the boundary cells by first extrapolating $f_s$ from the Vlasov data and then modifying its shape such that it matches the moments given by the fluid data. As the extrapolation of the distribution function to the boundary cells is straightforward, we now focus on a detailed description of the procedure to adjust the moments of the distribution function.

To simplify notation, we will drop the subscript $s$ for the particle species in this section and denote the extrapolated distribution functions by $f^E$ and the modified functions—the ones that will be used as boundary conditions—by $f^M$. The modification of the zeroth moment can be performed by a simple *multiplication*. The first moment can be changed by *shifting* the distribution function in velocity space. Finally, the second moment is altered by *squeezing* or *stretching* of the function. This fitting can be done in various ways. However, here we use the same semi-Lagrangian method as in the Vlasov solver itself with an appropriately chosen advecting "velocity" field for changing the distribution function. Note that the method is not limited to a special choice of a semi-Lagrangian scheme for solving the advection problem. An exaggerated example is depicted in Fig. 1. This procedure will now be explained in more detail. For simplicity, only the one-dimensional case will be discussed here. The extension to three dimensions can be obtained by successive application of the algorithm along each direction. It is convenient to use slightly different moments than the ones defined in (2)–(4). Writing

$$n := \int f(v)\,\mathrm{d}v, \qquad V := \frac{1}{n}\int v f(v)\,\mathrm{d}v, \quad \text{and} \quad T := \frac{1}{n}\int v^2 f(v)\,\mathrm{d}v - V^2, \tag{48}$$

no generality is lost and both sets of moments can be converted into one another:

$$\rho(n, V, T) = mn, \qquad u(n, V, T) = mnV, \qquad \mathcal{E}(n, V, T) = \frac{m}{2} n\big(T + V^2\big), \tag{49}$$

$$n(\rho, u, \mathcal{E}) = \frac{\rho}{m}, \qquad V(\rho, u, \mathcal{E}) = \frac{u}{\rho}, \qquad T(\rho, u, \mathcal{E}) = \frac{2\mathcal{E}}{\rho} - \frac{u^2}{\rho^2}. \tag{50}$$

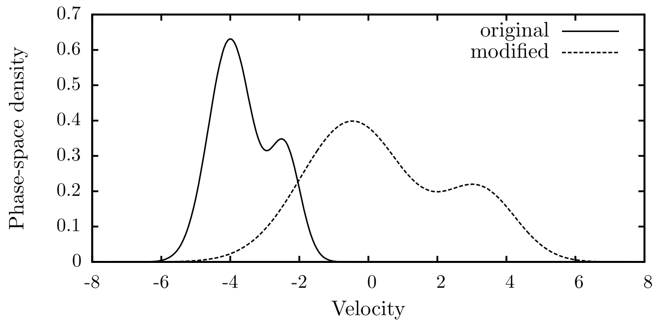


**Fig. 1.** Example of the modification of a distribution function. The solid line represents the original distribution function obtained by extrapolation. After multiplication with an appropriate factor and advection by a "velocity field" $av + b$, the area under the curve, the expectation value and the variance have changed, but the overall shape of the distribution function is preserved. Note that this plot is exaggerated for clarity.

Now, the idea is to let the distribution function be advected by a divergent velocity field $av + b$, i.e. to solve the initial value problem

$$\partial_\tau f(v,\tau) + \partial_v\big((av+b)f(v,\tau)\big) = 0, \quad f(\tau=0) = f^E, \tag{51}$$

where $\tau$ is just a dummy time variable independent of the actual time of the simulation $t$. Note that a rescaling of $\tau$ can be compensated by a rescaling of $a$ and $b$. Since this equation has the form of a conservation law, the zeroth moment $n$ of $f$ will not be changed. Furthermore it will preserve the main features of $f$, like the number of maxima etc. To solve Eq. (51) using a semi-Lagrangian scheme, we need to find the source points of the characteristics.

First we investigate the effects of this advection on the first moment:

$$\partial_\tau \frac{1}{n}\int vf\,\mathrm{d}v + \frac{1}{n}\int v\partial_v\big((av+b)f\big)\,\mathrm{d}v \tag{52}$$

$$= \partial_\tau V - b - aV = 0 \tag{53}$$

Here, partial integration and the vanishing of $f$ at infinity is used. The resulting differential equation can be solved by variation of constants to yield

$$V(\tau) = \begin{cases} \big(V(0) + \frac{b}{a}\big)\exp(a\tau) - \frac{b}{a} & \text{for } a \in \mathbb{R}\setminus\{0\} \\ V(0) + b\tau & \text{for } a = 0. \end{cases} \tag{54}$$

Solving for $b$ yields

$$b = \begin{cases} \frac{a(V(\tau) - V(0)\exp(a\tau))}{\exp(a\tau)-1} & \text{for } a \in \mathbb{R}\setminus\{0\} \\ V(\tau) - V(0) & \text{for } a = 0. \end{cases} \tag{55}$$

Next, we examine the evolution of the second moment $T$:

$$\partial_\tau \frac{1}{n}\int v^2 f\,\mathrm{d}v + \frac{1}{n}\int v^2\partial_v\big((av+b)f\big)\,\mathrm{d}v \tag{56}$$

$$= \partial_\tau T + \partial_\tau V^2 - 2bV - 2aT - 2aV^2 \tag{57}$$

$$= \partial_\tau T - 2aT = 0. \tag{58}$$

The solution of the resulting differential equation is

$$T(\tau) = T(0)\exp(2a\tau), \tag{59}$$

which can be solved for $a$, giving

$$a = \frac{1}{2}\ln\left(\frac{T(\tau)}{T(0)}\right). \tag{60}$$

We now want to find the characteristics $\mathcal{C}$ of Eq. (51), which can be calculated from

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathcal{C}(\tau) = a\mathcal{C}(\tau) + b, \tag{61}$$

with the initial condition $\mathcal{C}(\tau=0) = \mathcal{C}(0)$. The solution is

$$\mathcal{C}(\tau) = \begin{cases} \big(\mathcal{C}(0) + \frac{b}{a}\big)\exp(a(\tau-1)) - \frac{b}{a} & \text{for } a \in \mathbb{R}\setminus\{0\} \\ b(\tau-1) + \mathcal{C}(0) & \text{for } a = 0. \end{cases} \tag{62}$$

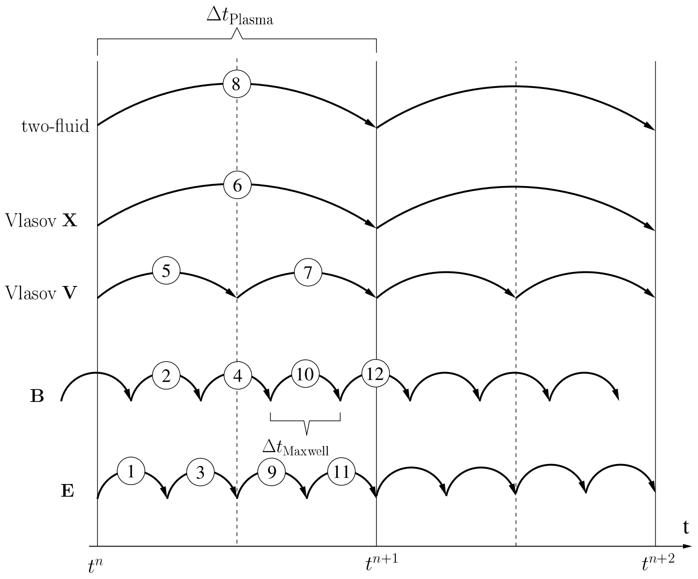


**Fig. 2.** This is the order, in which the fields are updated in each step. Each arrow represents a substep of the respective scheme. The first line (two-fluid) represents the stepping of the fluid quantities. The update of the phase space density in physical (Vlasov **X**) and velocity space (Vlasov **V**) is separated by means of Strang splitting. The electric (**E**) and magnetic (**B**) field are updated separately.

Inserting $a$ (60) and $b$ (55) yields:

$$\boxed{\mathcal{C}(\tau) = V(\tau) + \sqrt{\frac{T(\tau)}{T(0)}}\big(\mathcal{C}(0) - V(0)\big).} \tag{63}$$

As expected, all explicit dependence on $\tau$ has disappeared.

Finally, a summary of the algorithm for changing $f^E$ with moments $\rho^E$, $u^E$, and $\mathcal{E}^E$ to $f^M$ with the prescribed moments $\rho^M$, $u^M$, and $\mathcal{E}^M$:

1. Calculate $\tilde{f} = f^E \rho^M/\rho^E$ to change the zeroth moment.
   Now $\tilde{f}$ has moments $\tilde{\rho} = \rho^M$, $\tilde{u} = u^E\rho^M/\rho^E$, and $\tilde{\mathcal{E}} = \mathcal{E}^E\rho^M/\rho^E$.
2. Use the formulas (50) to obtain $\tilde{V}$, $\tilde{T}$, $V^M$, and $T^M$.
3. Set $V(0) = \tilde{V}$, $T(0) = \tilde{T}$, $V(\tau) = V^M$, and $T(\tau) = T^M$. Use a conservative semi-Lagrangian scheme with the source points of the characteristics given by Eq. (63) to obtain $f^M$.

## 4.3. *Time stepping*

As multistage and splitting methods are employed, care must be taken with respect to the points in time at which the fields and the fluxes at the cell faces live. Fig. 2 reveals the structure of time stepping for the electric **E** and magnetic **B** fields, the fluid quantities and the distribution functions. The timestep for the Maxwell solver is chosen to be smaller than the global timestep in order to resolve light waves which are much faster than all other speeds in the plasma. We choose the ratio between the smaller timestep of the Maxwell solver and the global timestep as a power of two, so the electromagnetic fields are known at all relevant times and the two different discrete times stay aligned throughout the simulation.

Initially, all the fields exist at the same point in time. Based on the currents calculated at that time, half a substep of the magnetic field is done in order to start the scheme. Afterwards, the following steps are repeated in a cyclic way:

1. Calculate currents from moments and phase space density of the different species and pass them to the Maxwell solver.
2. With these currents as source terms, advance the electromagnetic fields by half a global timestep. The magnetic field is shifted by half a time step, so linear interpolation is used for obtaining the magnetic field at the times at which the electric fields live.
3. Do a complete step of the Vlasov solver, which translates into half a step in velocity space, a full step in physical space and half a step in velocity space again. Herein, the new electromagnetic fields are used as source terms. The boundary conditions for the kinetic region are known from the last step.
4. Do a full step of the multi-fluid scheme. Once again, the newly obtained electromagnetic fields are used as source terms. The Runge–Kutta method requires boundary conditions at intermediate times, which must be obtained by temporal

interpolation of the values provided by the Vlasov solver. Because of this, it is necessary to advance the plasma in the kinetic region first.
5. Now the currents can be calculated again.
6. With these, the electromagnetic field is advanced by another half step.

A test of the order of accuracy of this rather involved time stepping scheme is given below in Section 6.2.

## 5. Implementation on a cluster of graphics cards

In this section, we will outline the principal strategy, how this code is implemented on a cluster of graphics cards. We briefly discuss memory layout, program flow, parallelisation issues and performance results. Specific details will be presented elsewhere.

### *5.1. CUDA implementation of the Vlasov solver*

The graphics cards were programmed using the CUDA programming environment [24]. For storing the distribution function $f$ in two spatial and three velocity dimensions, the following memory structure was chosen. The spatial domain is represented by an array of pointers. Each of these pointers contains the memory address of an array that stores the three-dimensional velocity distribution function at the respective coordinate. Since multiple CUDA cards are used, the spatial domain is divided into *patches*. Each card solves the Vlasov equation in its patch. The Vlasov solver then needs efficient implementations for the following tasks:

1. Operating along spatial directions.
2. Operating along velocity directions.
3. Calculating moments of the distribution function.

The problem with operating along spatial directions is that the equations for updating the value of $f$ in a cell need the values of $f$ in adjacent cells, but these values do not reside at adjacent places in memory. To solve this, the blocks are chosen to be two-dimensional with $n$ threads in the first direction and $N_x$ threads in the second direction. Here, $N_x$ is the spatial resolution of the patch in the $x$ direction and $n$ is the CUDA warp size. The grid is two-dimensional and contains $\widetilde{N}_y$ blocks along the first direction and $N_v/n$ blocks along the second direction. $\widetilde{N}_y$ is the resolution of the patch in the $y$ direction including ghost cells. $N_v$ is the total number of cells in velocity space, i.e. $N_v = N_{v_x} \cdot N_{v_y} \cdot N_{v_z}$. Then, each CUDA warp in a block reads consecutively from the GPU global memory and stores its data into an array in GPU shared memory. When the data is written, the access pattern is chosen so that the data in the shared memory is transposed, i.e. values corresponding to different $x$ coordinates are adjacent in shared memory. Now the semi-Lagrangian scheme can be used to update the distribution function and the data can be written back into global memory. Fig. 3 shows a graphical representation of this access pattern. The update along the $y$ direction uses a similar access pattern.

For the update along the $v_x$ direction, the memory is already aligned correctly. To perform the updates for $v_y$ and $v_z$, the velocity distribution function is transposed in global memory, so that the appropriate data is adjacent in memory. The transposition algorithm that was used is an extension of the algorithm by Ruetsch and Micikevicius [25].

Calculating the moments of the velocity distribution function is trivial on CPUs, but can be a difficult task on massively parallel architectures like GPUs. An optimised algorithm for calculating large sums on CUDA was developed by Harris [26], who used a binary tree that is recursively reduced until the total sum is obtained. The first step of this algorithm was modified to calculate the moments of $f$ instead of its sum.

### *5.2. Parallelisation via the message passing interface (MPI)*

In total, 64 CUDA cards were used for the solution of the Vlasov equation. Maxwell's equations and the two-fluid equations are computationally much less demanding and were solved on CPUs. To allow the exchange of information between the different solvers, the *message passing interface (MPI)* was used. This is a protocol that allows software to send packets of data over a network. There are two situations where MPI messages occur in the code:

1. Exchange of boundary conditions between different patches. These can either be boundary conditions between Vlasov patches, in which case the full distribution function is sent, or boundary conditions where at least one of the adjacent patches is a two-fluid patch. In this case, only the moments of the distribution function are exchanged.
2. Exchange of electromagnetic fields and current densities between the Maxwell solver and the Vlasov/two-fluid solvers. Since the Maxwell equations are solved over the whole domain, each patch needs to calculate its local current density and send the result to the Maxwell solver. The individual current density patches are then put together to form the global current density. Maxwell's equations are solved, the resulting electromagnetic fields are divided into patches and are sent back to their respective sources.

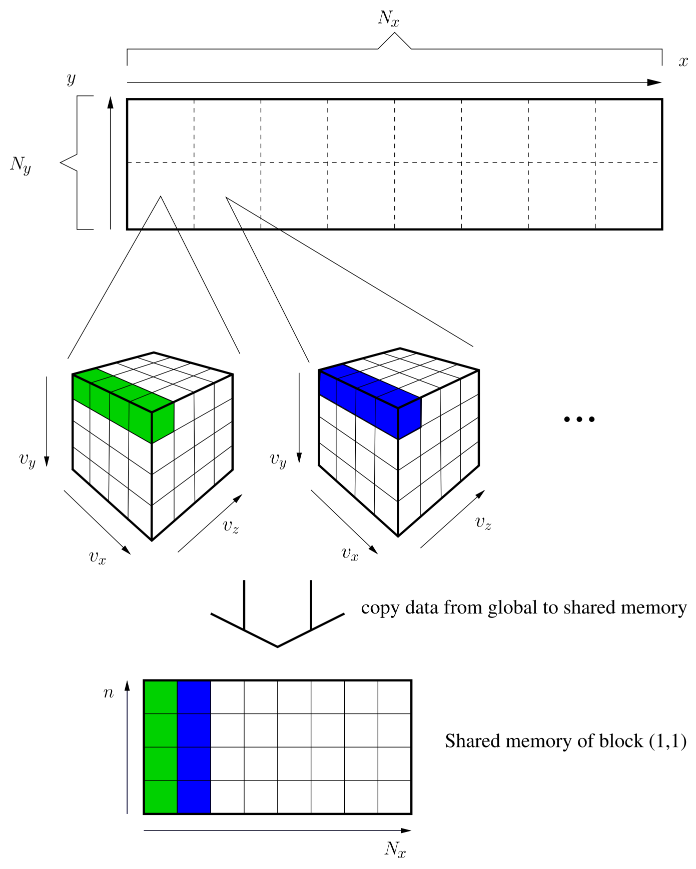

$N_x$
$x$
$y$
$N_y$
$v_y$
$v_x$
$v_z$
$v_y$
$v_x$
$v_z$
copy data from global to shared memory
$n$
Shared memory of block (1,1)
$N_x$

**Fig. 3.** Access pattern for access along the $x$ direction. The top rectangle is an array of pointers, representing a patch of the two-dimensional spatial domain. Each cell points to an array in global memory that contains the local velocity distribution function. The resolution in this example is $N_x = 8$, $N_y = 2$, $n = N_{v_x} = N_{v_y} = N_{v_z} = 4$. For simplicity, only the first two elements in the $x$ direction are shown. The block with coordinates $(1, 1)$ copies the coloured cells into its shared memory. Now, the data points needed for the step in the $x$ direction are adjacent in memory and can be accessed efficiently by the threads.

**Table 1**
Comparison of the computational time required for solving the GEM reconnection challenge problem (cf. Section 6.3).

| Code | System | Resolution | Time |
|---|---|---|---|
| Multifluid | 32 CPUs | $256 \times 128$ | 0.3 h |
| Vlasov (cf. [27]) | 64 CPUs | $256 \times 128 \times 30 \times 30 \times 30$ | 150 h |
| Vlasov (this work) | 64 CPUs / 64 GPUs | $256 \times 128 \times 32 \times 32 \times 32$ | 16 h |
| Vlasov/multifluid | 64 CPUs / 64 GPUs | $256 \times 128$ $(\times 32 \times 32 \times 32)$ | 8 h |

For the sake of an easy coupling in different situations, the two-fluid solver and the Vlasov solver are implemented as two separate programmes. Each of these programmes can run on its own and thus treat the whole domain by either the two-fluid equations or the Vlasov equation. If both programs are run simultaneously using MPI, they will split the domain into a two-fluid and a Vlasov part and automatically use the coupling routine from Section 4.

## *5.3. Performance results*

The simulations were done on the DaVinci cluster in Bochum. On this cluster, there is a total of 17 nodes. Each node is equipped with two CPUs (Xeon E5530 Quad Core, 2.4 GHz) and four GPUs (Tesla S1070). For the CPUs, there are 24 GB of RAM per node, whereas there are 16 GB RAM per node available for the GPUs. So, all in all, there are 272 CPU cores with 408 GB RAM and 16 320 GPU cores with a total of 272 GB RAM on the cluster.

Data with respect to the performance of our code on this cluster can be found in Table 1.

## 6. First results of the multiphysics approach

### 6.1. Whistler wave

In order to make it possible to compare the scheme to other recently proposed methods, a simulation of a whistler wave, similar to the one in Daldorff et al. [10] was carried out.

The whistler wave is an incompressible cold plasma wave whose wave speed $c_W = \frac{\omega}{k}$ depends on the wave number $k$. Herein, $\omega$ is the angular frequency. The dispersion relation and the ratio of amplitudes for the different fields can be gained by linearising the two-fluid/Maxwell system above in the usual way. Under the assumptions $c_W \ll c$ and $\omega \ll \Omega_e$, we obtain the four solutions

$$c_W = \pm\frac{W}{2} \pm \sqrt{\frac{W^2}{4} + c_A^2}, \quad W = \frac{c_A^2 k}{\Omega_i} \tag{64}$$

Herein, $c_A$ is the Alfven velocity. We choose the branch described by the largest $c_W$. The wave is circularly polarised. Before describing the actual initial conditions, we discuss a whistler wave that propagates along the $x$-axis. Density and pressure are constant. We can set

$$B_x = B_0$$

$$B_y = -\delta B \cos(kx - \omega t)$$

$$B_z = \delta B \sin(kx - \omega t)$$

The velocities of ions and electrons have a phase shift of $\pi$ relative to the magnetic fields and the amplitudes are $\delta u_i = \frac{\delta B}{B_0}(c_W - W)$ respectively $\delta u_e = \frac{\delta B}{B_0} c_W$. The electric field is shifted by $\frac{\pi}{2}$ and has the amplitude $\delta E = \delta B c_W$. If the assumptions above are fulfilled, this wave is a reasonable solution for the linear Vlasov regime as well (cf. [10]).

We choose the initial conditions in a way, that the wave propagates at an angle $\tan^{-1}(2)$ through the double periodic domain which has an aspect ratio of $\frac{1}{2}$. This is achieved by a simple rotation of the fields given above. The wavelength of the whistler wave is set to $\lambda = 10\delta_i$ (ion inertial lengths) and accordingly we have a dimensionality of $\sqrt{5}\lambda \times \frac{\sqrt{5}}{2}\lambda$ for the overall domain. At a spatial resolution of $64 \times 32$ this results in $\Delta x = \Delta y \approx 0.35\delta_i$. The kinetic region comprises one quarter of the domain and has a corresponding aspect ratio. Due to periodicity the actual position of the kinetic part is of minor importance. It can be seen in the figure. The resolution of the velocity space is $32^3$. All the amplitudes are chosen in a way that the ratio of the perturbed to the background magnetic field is $\frac{5}{100}$. We set $B_0 = 1$, $n_s = 1$ and $p_s = 5.12 \cdot 10^{-4}$. This way the requirements for the plasma regime that were used in the derivation are satisfied to a good degree.

An excerpt of the result can be seen in Fig. 4. The state after the propagation of one wavelength, which can be compared to the initial conditions, is shown. The wave passed through the kinetic region mostly correctly. Nevertheless, some small yet noticeable disturbances of up to about 5% of the wave amplitude have occurred, especially in the velocity field. This is mostly due to numerical heating in the kinetic region. Over the course of the simulation, the temperature in the kinetic region rises slowly and the resulting gradient induces some nonphysical flow out of this region. This can be tolerated if the resulting disturbance is negligible on the time scale of the simulation, as it can be said here, but it has to be addressed in the future.

### 6.2. Convergence test of time stepping

In order to check, whether the rather involved time stepping scheme shown above actually leads to second order accuracy in practice, a numerical test was done.

A numerical scheme is of $n$-th order in time, if

$$f_{\Delta t}(t_0) = f_{\text{exact}}(t_0) + C(\Delta t)^n \tag{65}$$

holds where $f_{\Delta t}(t_0)$ is the numerical solution with step-width $\Delta t$ at time $t_0$, $f_{\text{exact}}(t_0)$ is the exact solution at the respective time and $C$ is some constant in time which might nevertheless depend on the spatial discretisation. Both this constant and the exact solution are not known in general, so we have to find a way to eliminate them from the equations in order to find the order $n$ of a scheme. By looking at the difference

$$f_{2\Delta t} - f_{\Delta t} = C\big(2^n - 1\big)(\Delta t)^n \tag{66}$$

we get rid of $f_{\text{exact}}$. Both $C$ and $n$ can now easily be determined from a log–log plot of this quantity, as such a function appears as a straight line in such a depiction. Because we are dealing with a whole field of values rather than a single value, we consider a suitable norm of the pointwise differences of the fields. In particular, we take the discrete $L_1$-norm $\|\cdot\|_1$, which means

$$\|f_{2\Delta t} - f_{\Delta t}\| = \frac{1}{N}\sum_{i=1}^{N} |f_{2\Delta t,i} - f_{\Delta t,i}| \tag{67}$$

where $N$ is the number of grid points and $i$ represents the discrete cell values.

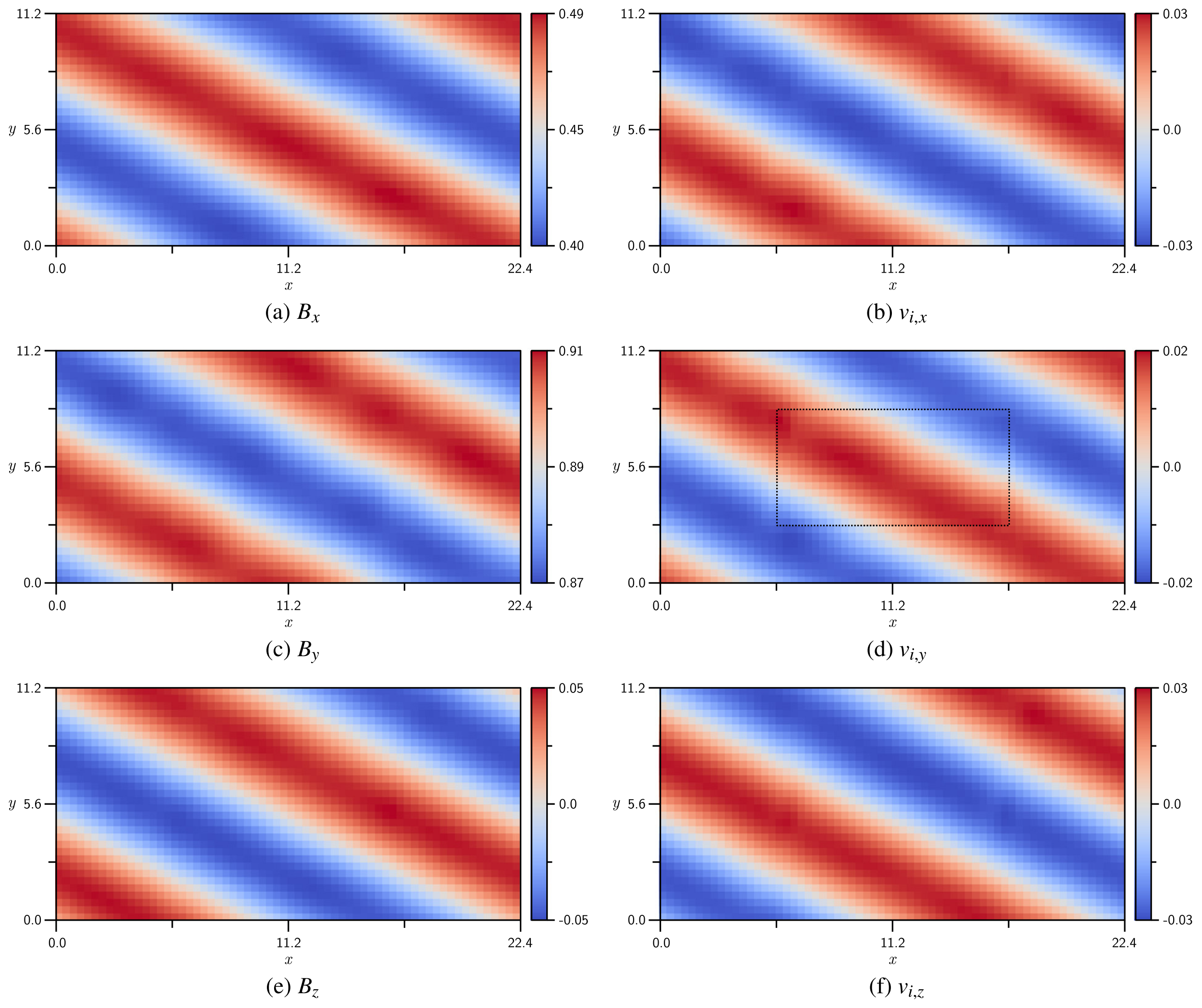


**Fig. 4.** Coupled simulation of a whistler wave at $t = 7.26\Omega_i^{-1}$, after the propagation of about a wavelength. For initial conditions and parameters, see text. The kinetic region in the middle of the domain is marked with a dashed rectangle.

The result of such a test is shown in Fig. 5. Numerical simulations with different step widths $\Delta t$ of the whistler-wave were done for the first 0.015 ion gyroperiods, using the same setup as above. In accordance with the description above, pointwise differences were taken and averaged over the whole grid. Just for comparison, the line representing $\frac{1}{2}(\Delta t)^2$ was also plotted. It must not be understood as an exact fit. For this plot, the $y$-component of the electron velocity was chosen as the data source, but the other components and fields yield similar results.

We can clearly see, that the scheme is of second order in the interval of $\Delta t$ which is considered here. As we go further to the right, other orders of $\Delta t$ become more important or the chosen time step is simple to large, so that the scheme no longer converges correctly. At the left, the accuracy of floating point numbers becomes an issue. Because graphics cards are much slower when double precision values are handled, we stick to single precision floats, which makes it useless to decrease the time step further than in the plot given above.

## 6.3. GEM reconnection challenge

As a further test, the GEM reconnection challenge setup, set forth in [1], was used. It was simulated both with the multifluid and the Vlasov code alone, as well as with our multiphysics coupling method. In this test, the plasma is advected from the fluid region into the kinetic region.

In this test case, the vicinity of the current sheet was completely simulated by means of the Vlasov model and only in the outer regions the multifluid model was applied (see Fig. 6). Nevertheless, this already leads to nearly a half of the required computation time, as only half of the domain needs to be simulated with a kinetic code. In global simulations, like space weather simulations [28], typically the regions where a fluid approach is sufficient occupies a much larger fraction

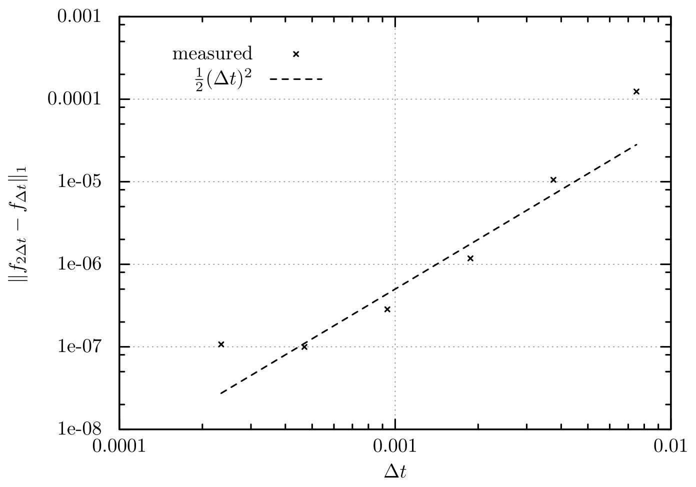


**Fig. 5.** Log–log plot of the numerical error in dependence on the time step. For a detailed description, see the text.

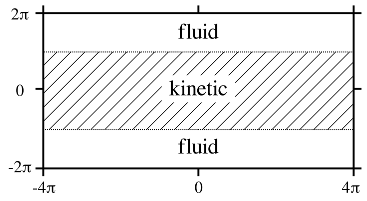


**Fig. 6.** Partition of the domain of the GEM problem. The inner region (hatched), i.e. the vicinity of the current layer, is simulated by means of the kinetic model, whereas the multifluid code is employed in the outer regions.

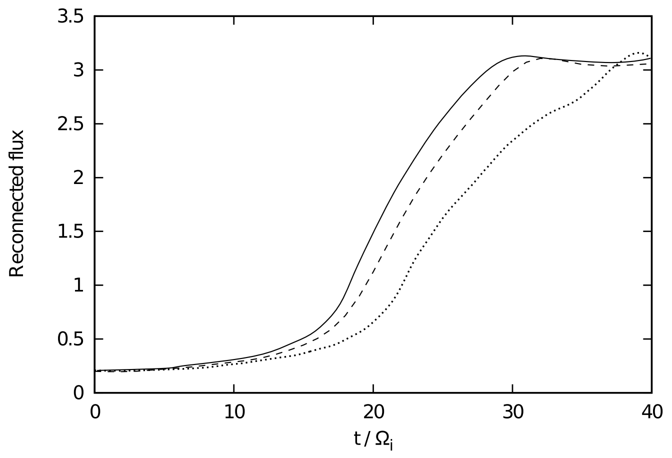


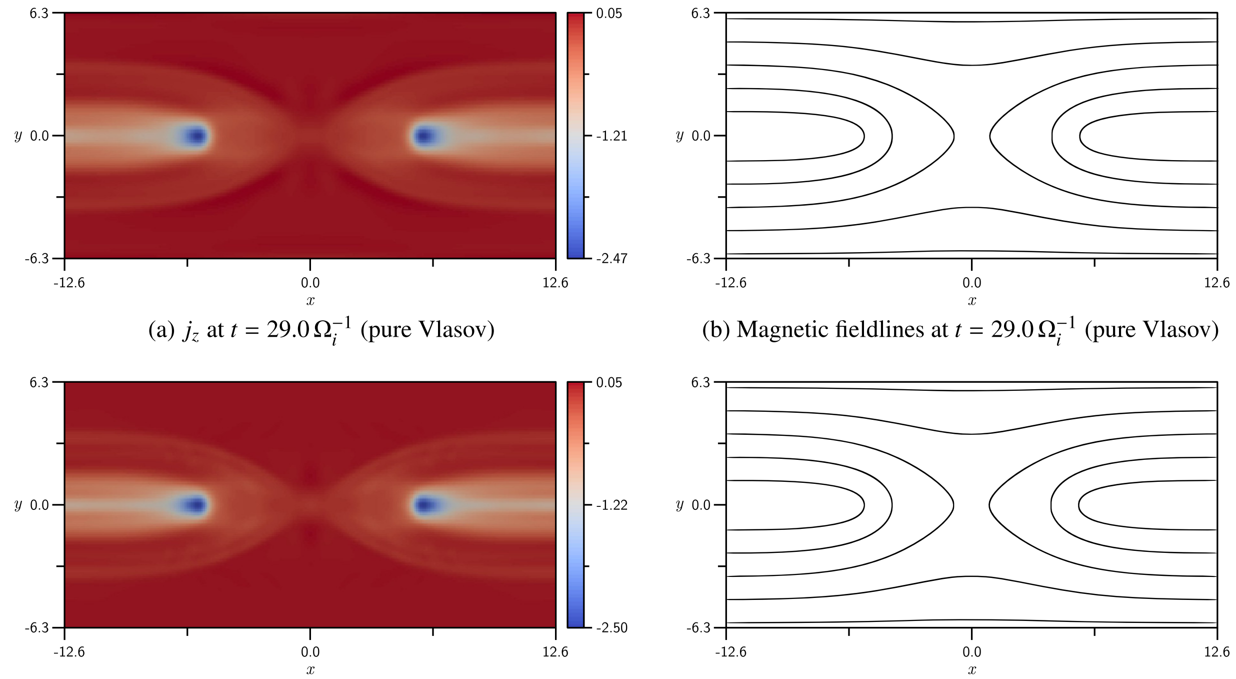
**Fig. 7.** Comparison of reconnected flux vs. time for full kinetic (solid, taken from [27]), coupled (dashed), and pure multifluid (dotted) simulations.

of the simulation domain than the regions where a kinetic treatment is necessary. Thus the aforementioned factor 2 is underestimating the potential of the coupling strategy. In this case here the time needed for the multifluid code is negligible compared to that for the Vlasov code.

A comparison of the time needed for the respective simulations can be found in Table 1.

The results of the coupled simulation look both quantitatively and qualitatively the same as for the pure Vlasov simulation, which means that the region in which the determining physical processes occur is indeed the current layer and the multifluid model is sufficient for the outer region that was chosen (see Fig. 7 and Fig. 8). Note, that the plots are chosen at slightly different times in accordance with Fig. 7 such that the reconnected flux is equal in both simulations.

As can be expected, a coupling inside the current sheet does not yield satisfactory results, because the simple 5-moment model leads to a different behaviour than the kinetic model. Thus, in future developments, new strategies will have to be developed in order to identify the regions which have to be treated by the kinetic or fluid description, respectively. This would result in an adaptive multiphysics approach to collisionless plasmas.



(a) $j_z$ at $t = 29.0\,\Omega_i^{-1}$ (pure Vlasov)
(b) Magnetic fieldlines at $t = 29.0\,\Omega_i^{-1}$ (pure Vlasov)
(c) $j_z$ at $t = 32.3\,\Omega_i^{-1}$ (coupled)
(d) Magnetic fieldlines at $t = 32.3\,\Omega_i^{-1}$ (coupled)

**Fig. 8.** Comparison of $j_z$ and magnetic fieldlines between pure Vlasov and coupled simulation.

One problem, that is observed in general is the heating in the Vlasov region, due to numerical diffusivity in the velocity space, which can hardly be avoided. Especially the fast-gyrating electrons tend to be heated noticeably. This effect does not take place when the energy conservative multifluid code is used. This gives rise to a small yet observable temperature gradient at the coupling border and corresponding changes in other fields affected by it. As a quick fix, the numerical diffusivity can be lowered by a higher resolution in $v$-space. In the long run a more robust solution for this problem needs to be found.

## 7. Summary and outlook

A Vlasov code on GPUs and a conventional multifluid code were presented that can be run both on their own as well as coupled to each other during runtime. Coupling of both codes was achieved by a combination of extrapolating and adjusting the distribution function according to the moments of the fluid description. Specific results on the propagation of whistler waves and on the GEM reconnection setup were extremely encouraging and a remarkable speed-up was observed. However, we should note that this is only a first step in the treatment of multiphysics descriptions of collisionless plasmas. Further steps needed to achieve this goal should include

i) appropriate closure relations (adiabatic, Chew–Goldberger–Low, isotropic, polytropic, Landau–fluid) depending on the physical situation;
ii) an identification of regions which have to be treated by a kinetic or fluid description, respectively;
iii) allowing the kinetic regions to change and move in an adaptive way in order to get the most efficient description of the underlying problem (analogous to our adaptive mesh framework *racoon* [29]).

These topics are currently under development in our group.

## Acknowledgements

We acknowledge all the fruitful discussions we had with Jürgen Dreher. This research was supported by the DFG Research Unit FOR 1048, project B2. Calculations were performed on the CUDA-Cluster DaVinci hosted by the Research Department *Plasmas with Complex Interactions* at the Ruhr-University Bochum.

## References

[1] J. Birn, J.F. Drake, M.A. Shay, Geospace Environmental Modeling (GEM) magnetic reconnection challenge, J. Geophys. Res. 106 (2001) 3715–3719.

[2] L. Woods, Theory of Tokamak Transport, Wiley, 2006.
[3] R. Ramesh, A. Satya Narayanan, C. Kathiravan, C.V. Sastry, N. Udaya Shankar, An estimation of the plasma parameters in the solar corona using quasi-periodic metric type III radio burst emission, Astron. Astrophys. 431 (2005) 353–357.
[4] P. Degond, G. Dimarce, L. Mieussens, A multiscale kinetic–fluid solver with dynamic localization of kinetic effects, J. Comput. Phys. 229 (2010) 4907–4933.
[5] S. Dellacherie, Kinetic–fluid coupling in the field of the atomic vapor isotopic separation: numerical results in the case of a monospecies perfect gas, AIP Conf. Proc. 663 (2003) 947–956.
[6] T. Goudon, S. Jin, J.-G. Liu, B. Yan, Asymptotic-preserving schemes for kinetic–fluid modeling of disperse two-phase flows, J. Comput. Phys. 246 (2013) 145–164.
[7] A. Klar, H. Neunzert, J. Struckmeier, Transition from kinetic theory to macroscopic fluid equations: a problem for domain decomposition and a source for new algorithms, Transp. Theory Stat. Phys. 29 (2000) 93–106.
[8] P. Le Tallec, F. Mallinger, Coupling Boltzmann and Navier–Stokes equations by half fluxes, J. Comput. Phys. 136 (1997) 51–67.
[9] T. Sugiyama, K. Kusano, Multi-scale plasma simulation by the interlocking of magnetohydrodynamic model and particle-in-cell kinetic model, J. Comput. Phys. 227 (2007) 1340–1352.
[10] L.K. Daldorff, G. Tóth, T.I. Gombosi, G. Lapenta, J. Amaya, S. Markidis, J.U. Brackbill, Two-way coupling of a global hall magnetohydrodynamics model with a local implicit particle-in-cell model, J. Comput. Phys. 268 (2014) 236–254.
[11] S. Markidis, P. Henri, G. Lapenta, K. Rönnmark, M. Hamrin, Z. Meliani, E. Laure, The fluid–kinetic particle-in-cell method for plasma simulations, J. Comput. Phys. 271 (2014) 415–429.
[12] V. Kolobov, R. Arslanbekov, Towards adaptive kinetic–fluid simulations of weakly ionized plasmas, J. Comput. Phys. 231 (2012) 839–869.
[13] T.P. Schulze, P. Smereka, W. E, Coupling kinetic Monte-Carlo and continuum models with application to epitaxial growth, J. Comput. Phys. 189 (2003) 197–211.
[14] H. Schmitz, R. Grauer, Comparison of time splitting and backsubstitution methods for integrating Vlasov's equation with magnetic fields, Comput. Phys. Commun. 175 (2006) 86–92.
[15] H. Schmitz, R. Grauer, Darwin–Vlasov simulations of magnetised plasmas, J. Comput. Phys. 214 (2006) 738–756.
[16] L.M. Leslie, R.J. Purser, Three-dimensional mass-conserving semi-Lagrangian scheme employing forward trajectories, Mon. Weather Rev. 123 (1995) 2551–2566.
[17] F. Filbet, E. Sonnendrücker, P. Bertrand, Conservative numerical schemes for the Vlasov equation, J. Comput. Phys. 172 (2001) 166–187.
[18] N. Crouseilles, M. Mehrenberger, E. Sonnendrücker, Conservative semi-Lagrangian schemes for Vlasov equations, J. Comput. Phys. 229 (2010) 1927–1953.
[19] A. Kurganov, D. Levy, A third-order semidiscrete central scheme for conservation laws and convection–diffusion equations, SIAM J. Sci. Comput. 22 (2000) 1461–1488.
[20] C.W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (1988) 439–471.
[21] A. Hakim, J. Loverich, U. Shumlak, A high resolution wave propagation scheme for ideal two-fluid plasma equations, J. Comput. Phys. 219 (2006) 418–442.
[22] K.S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Trans. Antennas Propag. 14 (1966) 302–307.
[23] A. Taflove, M.E. Brodwin, Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations, IEEE Trans. Microw. Theory Tech. 23 (1975) 623–630.
[24] nVidia Corp., nVidia CUDA C programming guide, http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html, 2014.
[25] G. Ruetsch, P. Micikevicius, Optimizing matrix transpose in CUDA, Nvidia CUDA SDK application note, 2009.
[26] M. Harris, Optimizing parallel reduction in CUDA, Nvidia CUDA SDK application note, 2009.
[27] H. Schmitz, R. Grauer, Kinetic Vlasov simulations of collisionless magnetic reconnection, Phys. Plasmas 13 (2006) 169–173.
[28] G. Tóth, D.L. De Zeeuw, T.I. Gombosi, W.B. Manchester, A.J. Ridley, I.V. Sokolov, I.I. Roussev, Sun-to-thermosphere simulation of the 28–30 October 2003 storm with the space weather modeling framework, Space Weather 5 (2007) S06003.
[29] J. Dreher, R. Grauer, Racoon: a parallel mesh-adaptive framework for hyperbolic conservation laws, Parallel Comput. 31 (2005) 913.