Sparse grids for the Vlasov–Poisson equation

Katharina Kormann* Eric Sonnendrücker‡

Abstract

The Vlasov–Poisson equation models the evolution of a plasma in an external or self-consistent electric field. The model consists of an advection equation in six dimensional phase space coupled to Poisson's equation. Due to the high dimensionality and the development of small structures the numerical solution is quite challenging. For two or four dimensional Vlasov problems, semi-Lagrangian solvers have been successfully applied. Introducing a sparse grid, the number of grid points can be reduced in higher dimensions. In this paper, we present a semi-Lagrangian Vlasov–Poisson solver on a tensor product of two sparse grids. In order to defeat the problem of poor representation of Gaussians on the sparse grid, we introduce a multiplicative delta-f method and separate a Gaussian part that is then handled analytically. In the semi-Lagrangian setting, we have to evaluate the hierarchical surplus on each mesh point. This interpolation step is quite expensive on a sparse grid due to the global nature of the basis functions. In our method, we use an operator splitting so that the advection steps boil down to a number of one dimensional interpolation problems. With this structure in mind we devise an evaluation algorithm with constant instead of logarithmic complexity per grid point. Results are shown for standard test cases and in four dimensional phase space the results are compared to a full-grid solution and a solution on the four dimensional sparse grid.

1 Introduction

In order to build fusion energy devices, it is necessary to understand the behavior of plasmas in external and self-consistent electromagnetic fields. Within the kinetic theory, the plasma is described by its distribution function in phase-space and its motion is described by the Vlasov–Maxwell or, for low-frequency phenomena, by the Vlasov–Poisson equation. Since analytic solutions are not known, numerical simulations are inevitable to understand the behavior of plasmas in a fusion device. Because the distribution is a function in phase-space, the problem is six-dimensional. Several approximative models exist with the aim of reducing the dimensionality of the problem. However, only a simulation of the full 6D problem can reveal the full structure of the problem and can help to understand the validity of lower dimensional models. Three types of methods are widely used in the simulation of the Vlasov equation: Particle-In-Cell (PIC) methods, semi-Lagrangian and Eulerian methods. Since the latter two are grid-based, they suffer from the curse of dimensionality. For this reason PIC methods are most common for high-dimensional Vlasov simulations. On the other hand, the semi-Lagrangian method [21] has proven to be an accurate alternative in simulations of the Vlasov equation in two and four dimensional phase space and there are deterministic numerical methods that are specially designed to solve high-dimensional problems efficiently.

The sparse grid method [6] is targeted at moderately high-dimensional problems with solutions of bounded mixed derivatives. The subject of the present paper is to introduce an efficient semi-Lagrangian method to solve the Vlasov–Poisson equation on a sparse grid. In [3] sparse grids are combined with the semi-Lagrangian method to solve the Hamilton–Jacobi–Bellman equation.

The evolution of small filaments in the plasma distribution function yields large mixed derivatives which causes a major problem for the performance of sparse grids. However, if the filaments evolve mainly along the coordinate axes, a tensor product of separate spatial and velocity sparse grids can yield good compression as we will demonstrate.

Sparse grids have previously been introduced to plasma physics in the context of solving an eigenvalue problem in linear gyrokinetics with the sparse grid combination technique [13]. For this application the

*Technische Universität München, Zentrum Mathematik, Boltzmannstr. 3, 85747 Garching, Germany and Max-Planck-Institut für Plasmaphysik, Boltzmannstr. 2, 85748 Garching, Germany, katharina.kormann@tum.de

‡Max-Planck-Institut für Plasmaphysik, Boltzmannstr. 2, 85748 Garching, Germany and Technische Universität München, Zentrum Mathematik, Boltzmannstr. 3, 85747 Garching, Germany, eric.sonnendruecker@ipp.mpg.de

sparse grid method is particularly suited since field-aligned coordinates are used and nonlinear effects are not considered.

The outline of the paper is as follows. First we will briefly define a semi-Lagrangian Vlasov solver in Section 2 and discuss typical features of the solution to the Vlasov equation. The sparse grid method is introduced in Section 3 before we describe the main components of our combined semi-Lagrangian solver on a sparse grid in Section 4. Section 5 explains how to use the structure of the problem in order to improve on the implementation and the accuracy of the problem. In Section 6, we explain how to improve the interpolation of Gaussians on a sparse grid and introduce our multiplicative δf method. A brief discussion on conservation and stability properties of our method is given in Section 7 before we show the numerical performance for benchmark problems in Section 8. Finally, conclusions are drawn and further research directions indicated in Section 9.

2 The Vlasov–Poisson equation

The Vlasov–Poisson equation describes the motion of a plasma in its self-consistent electric field for low-frequency phenomena. In this paper, we will consider the dimensionless Vlasov–Poisson equation for electrons with a neutralizing ion background,

$$\begin{aligned}&\partial_t f(\mathbf{x},\mathbf{v},t) + \mathbf{v}\cdot\nabla_{\mathbf{x}} f(\mathbf{x},\mathbf{v},t) - \mathbf{E}(\mathbf{x},t)\cdot\nabla_{\mathbf{v}} f(\mathbf{x},\mathbf{v},t) = 0,\\ &-\Delta\phi(\mathbf{x},t) = 1-\rho(\mathbf{x},t), \quad \mathbf{E}(\mathbf{x},t) = -\nabla\phi(\mathbf{x},t), \quad \rho(\mathbf{x},t) = \int f(\mathbf{x},\mathbf{v},t)\,\mathrm{d}\mathbf{v}.\end{aligned}$$

Here, f denotes the probability density of finding a particle at position $\mathbf{x}\in D\subset\mathbb{R}^3$ and velocity $\mathbf{v}\in\mathbb{R}^3$, $\mathbf{E}$ denotes the electric field, ϕ the electric potential, and ρ the charge density. As for any scalar hyperbolic conservation law with divergence-free advection field, the value of f is constant along the characteristics defined by

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathbf{V}, \quad \frac{\mathrm{d}\mathbf{V}}{\mathrm{d}t} = -\mathbf{E}(\mathbf{X},t). \tag{1}$$

Let us denote by $\mathbf{X}(t;\mathbf{x},\mathbf{v},s),\mathbf{V}(t;\mathbf{x},\mathbf{v},s)$ the solution of the characteristic equations (1) at time t with initial conditions $\mathbf{X}(s)=\mathbf{x}$ and $\mathbf{V}(s)=\mathbf{v}$. Given an initial distribution f_0 at time 0, the solution at time $s>0$ is given by

$$f(\mathbf{x},\mathbf{v},s) = f_0(\mathbf{X}(0;\mathbf{x},\mathbf{v},s),\mathbf{V}(0;\mathbf{x},\mathbf{v},s)), \tag{2}$$

i.e. we can find the solution at any time s by solving the characteristic equations backwards in time. However, we cannot solve (1) analytically since the right-hand side depends on f through the Poisson equation. Existence and uniqueness of the solution are shown in [9, Ch. 4]. The regularity of the solution depends on the initial condition [14] and typical features will be discussed in the following subsection. In Section 2.2, we will then discuss how to use the characteristic equations to find a numerical solution of the Vlasov equation.

2.1 Some linear analysis of the solution to the Vlasov–Poisson equation

The typical equilibrium solution to the Vlasov–Poisson problem is a Maxwellian of the form

$$f_0(\mathbf{x},\mathbf{v}) = \frac{1}{(2\pi v_{th})^{3/2}}\exp\left(-\frac{\|\mathbf{v}\|_2^2}{2v_{th}^2}\right). \tag{3}$$

In this case, the electric field is zero. Also linear combinations of several Maxwellians are possible. This can be verified by inserting the Maxwellian into the Vlasov equation. The dynamics are typically initiated by small perturbations in space from equilibrium. For example the Landau test case refers to an initial condition of the type

$$f_0(\mathbf{x},\mathbf{v}) = \frac{1}{(2\pi v_{th})^{3/2}}\exp\left(-\frac{\|\mathbf{v}\|_2^2}{2v_{th}^2}\right)\left(1+\varepsilon\sum_{i=1}^{3}\cos(kx_i)\right), \tag{4}$$

where ε is a small parameter. While the solution stays close to equilibrium for small values of ε, a filamentation in phase space depending on ε develops over time. In figure 1a-1b, the projection of $f(\mathbf{x},\mathbf{v},t)-f_0(\mathbf{x},\mathbf{v})$ to the (x_1,v_1) plane is visualized at time 5 and 15 for a four dimensional simulation of the Landau problem with $\varepsilon=0.01$ and $k=0.5$. Comparing the phase-space distribution one can clearly see the filamentation at time 15. This filamentation is typical for solutions to the Vlasov–Poisson system

(cf. [7]). Other configurations give rise to instabilities where the solution does not only show filamentation but also new structures form. In this case, the solution does not stay close to equilibrium and nonlinear effects eventually become dominant. A typical unstable problem for the two dimensional Vlasov–Poisson problem is the two stream instability defined by the initial condition

$$f_0(x,v) = \frac{1}{2\sqrt{2\pi}}\left(1 + 0.001\cos(0.2x)\right)\left(\mathrm{e}^{-0.5(v-2.4)^2} + \mathrm{e}^{-0.5(v+2.4)^2}\right). \tag{5}$$

Figures 1d-1e shows the phase-space distribution for a simulation of the two stream instability at time 5—where the distribution is close to the initial distribution—and time 30 where a hole structure has formed. A good approximation of the dynamics can often be obtained by linearizing the Vlasov–Poisson equations around the equilibrium (cf. e.g. [4, Ch. 7]). This yields a description to the first order in ε. For instance for the Landau damping problem with initial value (4), linear analysis tells us that

$$\phi^{lin}(\mathbf{x},t) \propto \varepsilon \mathrm{e}^{-\gamma t}\cos(\beta t - \phi)\sum_{i=1}^{3}\cos(kx_i), \tag{6}$$

with $\gamma, \beta, \phi \in \mathbb{R}$ depending on k. The (oscillating) electric field is damped by the rate γ. For weak Landau damping—that is if ε is small—linear theory gives a good description of the actual phenomenon. Due to the fact that the initial perturbation in (4) is separable, the dispersion relation is the sum of three two-dimensional cases. Note that the form (6) of the electric potential implies that $E_i^{lin}(\mathbf{x},t) = E_i^{lin}(x_i,t)$ for $i = 1,2,3$, and the characteristic equations separate into three independent subproblems for the tuples (x_i, v_i), $i = 1,2,3$. Therefore, the filamentation is limited to the (x_i, v_i)-planes. In Figure 1c, we have plotted the projection of $f(\mathbf{x},\mathbf{v},15) - f_0(\mathbf{x},\mathbf{v})$ to the (x_1, v_2)-plane. As opposed to the situation in (x_1, v_1)-plane, there is no filamentation.

For the two-stream instability, a linear perturbation analysis is also possible. For the given parameters, the electrical field is growing in the linear description. Hence, the perturbation is increasing over time. Figure 1f shows the time evolution of the simulated electric energy, $\frac{1}{2}\|\mathbf{E}\|_{L_2}$, together with the straight line with slope equal to the predicted growth rate of the electrical energy by the linear theory. The numerical solution of the two stream instability indeed shows that the electrical energy does not follow the linear behavior over long times but the electrical energy stops growing at a certain point where nonlinear effects become dominant. For the following two reasons, this problem will be more challenging for a solution on a sparse grid as we will discuss later: Firstly, the solution is no longer close to equilibrium and secondly nonlinear effects are not limited to couplings in the (x_i, v_i)-planes. Numerically, it has been observed that there occur considerable couplings between the different coordinate directions for a similar problem in [12] where the solution of the Vlasov–Poisson system has been represented in tensor train format. In the tensor train format, a function on a tensor product grid it compressed by higher order singular value decompositions. This can roughly be considered as an adaptive procedure to sparsify the grid.

2.2 A semi-Lagrangian Vlasov solver

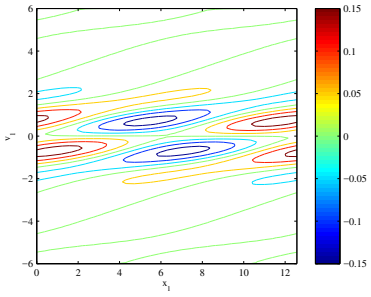
The idea of the semi-Lagrangian method is to introduce a grid $\mathcal{G}$ in phase space and to solve the characteristics successively on small time intervals Δt. Given the values f^n of the solution at the grid points for some time t_n, the values of f^{n+1} at time $t_{n+1} = t_n + \Delta t$ can be found by numerically solving the characteristics equation starting at the grid points and then interpolating the value at the origins for time t_n from the values f^n. The advantage of the semi-Lagrangian method over Eulerian solvers is the fact that usually no CFL conditions are required for stability. Stability of the semi-Lagrangian method on a bounded velocity domain has been shown in [2]. Since the Vlasov problem is posed on an unbounded domain, artificial boundary conditions have to be added to close the system. The effect of boundary conditions has not been considered in [2]. A reasonable choice of Δt is on the order of the grid spacing or slightly below.

The solution of the characteristic equations can be found using a numerical ODE solver. For the Vlasov–Poisson system, one can instead split the $\mathbf{x}$ and $\mathbf{v}$ advection steps applying a Strang splitting, solving the two problems

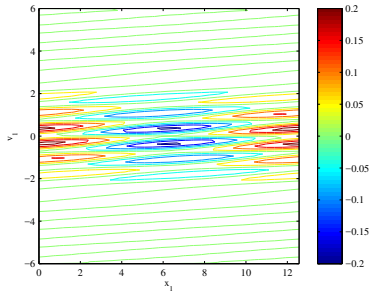
$$\partial_t f - \mathbf{E}(\mathbf{x})\cdot\nabla_{\mathbf{v}} f = 0, \quad \partial_t f + \mathbf{v}\cdot\nabla_{\mathbf{x}} f = 0 \tag{7}$$

separately. Note that the $\mathbf{v}$-advection equation is a constant-coefficient advection for each $\mathbf{x}$ and the characteristics are given as

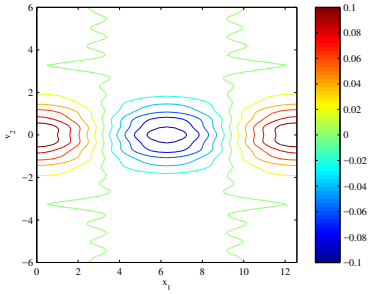
$$\mathbf{X}(t;\mathbf{x},\mathbf{v},s) = \mathbf{x}, \quad \mathbf{V}(t;\mathbf{x},\mathbf{v},s) = \mathbf{v} - (t-s)\mathbf{E}(\mathbf{x}). \tag{8}$$



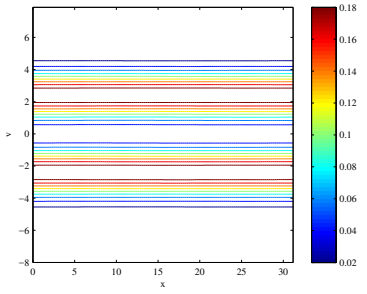
(a) Landau damping: (x_1, v_1)-projection of $f(\mathbf{x}, \mathbf{v}, 5) - f_0(\mathbf{x}, \mathbf{v})$.



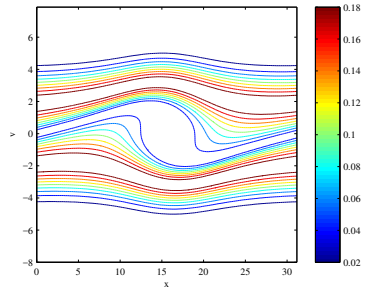
(b) Landau damping: (x_1, v_1)-projection of $f(\mathbf{x}, \mathbf{v}, 15) - f_0(\mathbf{x}, \mathbf{v})$.



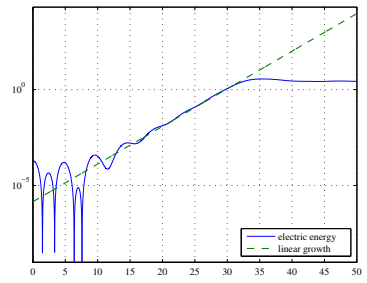
(c) Landau damping: (x_1, v_2)-projection of $f(\mathbf{x}, \mathbf{v}, 15) - f_0(\mathbf{x}, \mathbf{v})$.



(d) Two stream instability: $f(x, v, 5)$.



(e) Two stream instability: $f(x, v, 30)$.



(f) Two stream instability: electric energy.

Figure 1: Phase space distributions and electrical energy for Landau and two stream instability test cases.

Note that the density ρ and hence the field $\mathbf{E}$ do not change in the $\mathbf{v}$-advection step. Also the $\mathbf{x}$-advection equation has constant coefficients for each $\mathbf{v}$ and the solution for the characteristics are

$$\mathbf{X}(t;x,v,s) = \mathbf{x} + (t-s)\mathbf{v}, \quad \mathbf{V}(t;\mathbf{x},\mathbf{v},s) = \mathbf{v}. \tag{9}$$

This idea yields the following split semi-Lagrangian scheme originally introduced by Cheng and Knorr [7]

1. Solve $\mathbf{v}$-advection on half time step: $f^{(n,*)}(\mathbf{x_i},\mathbf{v_j}) = I_{f^{(n)}}(\mathbf{x_i},\mathbf{v_j} + \mathbf{E}^{(n)}(\mathbf{x_i})\frac{\Delta t}{2})$.
2. Solve $\mathbf{x}$-advection on full time step: $f^{(n,**)}(\mathbf{x_i},\mathbf{v_j}) = I_{f^{(n,*)}}(\mathbf{x_i} - \mathbf{v_j}\Delta t, \mathbf{v_j})$.
3. Compute $\rho(\mathbf{x_i},\mathbf{v_i})$ from $f^{(n,**)}(\mathbf{x_i},\mathbf{v_j})$ and solve the Poisson equation for $\mathbf{E}^{(n+1)}$.
4. Solve $\mathbf{v}$-advection on half time step: $f^{(n+1)}(\mathbf{x_i},\mathbf{v_j}) = I_{f^{(n,**)}}(\mathbf{x_i},\mathbf{v_j} + \mathbf{E}^{(n+1)}(\mathbf{x_i})\frac{\Delta t}{2})$.

For a given function g at all grid points, I_g evaluated at any point $(\mathbf{X},\mathbf{V})$, generally not a grid point, denotes the interpolated value at $(\mathbf{X},\mathbf{V})$ from the values of g on the grid points. In the split step method, the interpolation along the three $\mathbf{x}$ and $\mathbf{v}$ directions, respectively, is split again into three successive one-dimensional interpolations. For instance, we have the one dimensional interpolation problem

$$g^*(x_{1,i_1}, x_{2,i_2}, x_{3,i_3}, \mathbf{v_j}) = I_g(x_{1,i_1} - \Delta t v_{1,j_1}, x_{2,i_2}, x_{3,i_3}, \mathbf{v_j}) \quad \text{for all } (\mathbf{x_i},\mathbf{v_j}) \in \mathcal{G}. \tag{10}$$

Defining a one dimensional stripe of varying x_1 component from the grid by $S_{y_2,y_3,\mathbf{w}} = \{(\mathbf{x},\mathbf{v}) \in \mathcal{G} | x_2 = y_2, x_3 = y_3, \mathbf{v} = \mathbf{w}\}$, the interpolation problem (10) splits into a one-dimensional interpolation problem for each such stripe: All the points x_{1,i_1} are shifted by the constant displacement $-\Delta t v_{1,j_1}$.

The building blocks of the split semi-Lagrangian scheme are hence

1. Interpolation along one-dimensional stripes to propagate.
2. Integration over velocity dimension.
3. Solution of the three dimensional Poisson problem.

Using linear interpolation for the one-dimensional interpolation problems is too diffusive [7] and high-order interpolation needs to be used. The use of a cubic spline interpolator is common in Vlasov codes since cubic splines are a good compromise between accuracy and complexity (cf. [20, Ch. 2]). As mentioned in [7], trigonometric interpolation gives very good results for periodic problems, however, this is very specific and does not easily generalize to more complex geometries or mesh adaptivity.

3 The sparse grid method

In this section, we give a brief introduction to the concept of sparse grids and introduce the notation used throughout the rest of the paper. For the definition of a sparse grid, we restrict ourself to the domain $[0,1]^d$. However, a scaling of the domain is straightforward. Moreover, we will concentrate on piecewise linear functions first.

On the interval $[0,1]$, we define for each $\ell \in \mathbb{N}$ the grid points

$$x_{\ell,k} = \frac{k}{2^\ell}, \quad k = 0,\dots,2^\ell. \tag{11}$$

Associated to each grid point, we have the nodal hat function

$$\varphi_{\ell,k}(x) = \begin{cases} 1 - 2^\ell \text{abs}(x - x_{\ell,k}) & \text{for } x \in [x_{\ell,k} - \frac{1}{2^\ell}, x_{\ell,k} + \frac{1}{2^\ell}] \\ 0 & \text{elsewhere.} \end{cases} \tag{12}$$

with $\varphi_{\ell,k}(x_{\ell,j}) = \delta_{k,j}$ and support of size $\frac{1}{2^{\ell-1}}$ centered around $x_{\ell,k}$. In d dimensions, we define the grid of level $\boldsymbol{\ell} \in \mathbb{N}^d$ by

$$\Omega_{\boldsymbol{\ell}} := \left\{ \mathbf{x}_{\boldsymbol{\ell},\mathbf{k}} = (x_{\ell_1,k_1}, \dots, x_{\ell_d,k_d}) | k_i = 0,\dots,2^{\ell_i} \right\} \tag{13}$$

and, associated to $\Omega_{\boldsymbol{\ell}}$, the space spanned by the piecewise d-linear nodal functions

$$V_{\boldsymbol{\ell}} := \text{span}\left\{ \varphi_{\boldsymbol{\ell},\mathbf{k}}(\mathbf{x}) = \prod_{i=1}^d \varphi_{\ell_i,k_i}(x_i) | k_i = 0,\dots,2^{\ell_i} \right\}. \tag{14}$$

From the definition of Ω_ℓ in one dimension, we can see that $\Omega_{\ell-1} \subset \Omega_\ell$ contains all the points from Ω_ℓ with even index, that is Ω_ℓ is the disjoint union of $\Omega_{\ell-1}$ and

$$\Omega_\ell^{\text{odd}} := \{x_{\ell,k} \in \Omega_\ell, k \text{ odd}\}. \tag{15}$$

This leads us to the definition of the hierarchical increment

$$W_\ell := \operatorname{span}\left\{\varphi_{\ell,k} | k = 1, 3, \ldots, 2^{\ell-1} - 1\right\}. \tag{16}$$

The space $V_{\boldsymbol{\mathcal{L}}}$ can be decomposed into the direct sum of hierarchical increments with level indices smaller equal $\boldsymbol{\mathcal{L}}$, i.e.

$$V_{\boldsymbol{\mathcal{L}}} = \bigoplus_{\boldsymbol{\ell} \leq \boldsymbol{\mathcal{L}}} W_{\boldsymbol{\ell}}. \tag{17}$$

Instead of expanding a function $f \in V_{\boldsymbol{\mathcal{L}}}$ in the nodal basis, we can represent it by its hierarchical surplus $v_{\boldsymbol{\ell},\boldsymbol{k}}$ as

$$f(\mathbf{x}) = \sum_{|\boldsymbol{\ell}|_{\ell_\infty} \leq \mathcal{L}} \sum_{\boldsymbol{k} \in \Omega_{\boldsymbol{\ell}}^{\text{odd}}} v_{\boldsymbol{\ell},\boldsymbol{k}} \varphi_{\boldsymbol{\ell},\boldsymbol{k}}(\mathbf{x}). \tag{18}$$

There are two important differences between the nodal and the hierarchical representation: Firstly, the number of functions different from zero at a point $\mathbf{x} \in [0,1]^d$ increases. Except for the points on the grid, one function per hierarchical-increment basis is different from zero. On the other hand, the hierarchical surpluses express the additional information on the corresponding hierarchical increments compared to the representation of the solution on the space spanned by the hierarchical increments with smaller indices.

The sparse grid method is based on the possibility to leave out hierarchical increments where the hierarchical surpluses are smaller. In a standard sparse grid this is not done adaptively but a priori according with the aim of optimizing the cost-benefit ratio: The ℓ_1 norm of the index vector is restricted yielding

$$V_{\mathcal{L}}^s = \bigoplus_{|\boldsymbol{\ell}|_{\ell_1} \leq \mathcal{L}} W_{\boldsymbol{\ell}}. \tag{19}$$

This reduces the number of hierarchical increments to

$$\sum_{i=0}^{\mathcal{L}} \binom{d-1+i}{d-1}. \tag{20}$$

Note that the number of increments is still of the order $\frac{\mathcal{L}^d}{d!}$. On the other hand, the hierarchical increments that are left out are the ones with more points. The number of points for $V_{\mathcal{L}}^s$ is $\mathcal{O}(2^{\mathcal{L}}\mathcal{L}^{d-1})$. This means the exponential growth in the dimension is only for the basis $\mathcal{L} = \ln(N)$, while on the other hand the accuracy is only decreased to

$$\mathcal{O}(2^{-\mathcal{L}(p+1)}\mathcal{L}^{d-1}) \tag{21}$$

for functions of bounded mixed derivatives, i.e. again by the dth power of the logarithm. Here, the parameter p denotes the degree of the basis functions. So far, we have only defined linear basis functions. A construction of higher order basis functions for sparse grids has been proposed by Bungartz [5]. For each $x_{\ell,k} \in \Omega_\ell^{\text{odd}}$, the basis function $\varphi_{\ell,k}^{(p)}$ of degree p is defined by the $p+1$ conditions

$$\varphi_{\ell,k}^{(p)}(x_{\ell,k}) = 1, \quad \varphi_{\ell,k}^{(p)}(x_j) = 0, \tag{22}$$

where x_j are the two neighbors $x_{\ell,k} - \frac{1}{2^\ell}$ and $x_{\ell,k} + \frac{1}{2^\ell}$ as well as the $p-2$ next hierarchical ancestors of $x_{\ell,k}$. The support of $\varphi_{\ell,k}^{(p)}$ is restricted to $[x_{\ell,k} - \frac{1}{2^\ell}, x_{\ell,k} + \frac{1}{2^\ell}]$. In the following, let us denote by $M_{\mathcal{L}}$ the number of points of $V_{\mathcal{L}}^s$.

Now that we have exploited the benefits of a hierarchical representation, we need to discuss the downside that the support of the bulk of the basis function is increased. If we want to evaluate a representation in the nodal piecewise d-linear basis, the number of basis functions different from zero are mostly 2^d, whilst we have one function per hierarchical increment, i.e. $\mathcal{L}^d$ functions on a sparse grid of maximum level $\mathcal{L}$, in a hierarchical representation. Hence, the function evaluation has a complexity that is exponential in d for the basis $\mathcal{L}$ (cf. also [17, Sec. 2]). Also forming the hierarchical surplus in a naive implementation suffers from logarithmic scaling. However, for the piecewise d-linear basis, there is a simple relation between the hierarchical surplus and the function values that can be evaluated in linear complexity. A similar relation applies between the hierarchical surplus for a basis of order p and order $p+1$ when they are constructed as described by Bungartz [5]. Hence, we can hierarchize and also dehierarchize the representation of a

function on a sparse grid in linear complexity. Note, however, that this algorithm for hierarchization is specific to the basis functions constructed in [5] which is why using another basis is generally not advisable on sparse grids. One other basis that can efficiently be used on sparse grids is a hierarchical Fourier basis since a fast Fourier transform on sparse grids based on the so-called unidirectional principle [5] can be devised [10].

4 A semi-Lagrangian solver on a sparse grid

In the previous two sections, we have introduced the central ingredients of our novel Vlasov solver. Next, we will discuss various variants of introducing a sparse grid to the phase space. Once we have introduced the sparse grid, we discuss the design of the three building blocks of the split semi-Lagrangian method on the sparse grid.

4.1 Representation on a sparse grid

The basic ansatz would be to represent the distribution function f on a 6D sparse grid. However, since there is a natural splitting of $\mathbf{x}$ and $\mathbf{v}$ coordinates, a tensor product of a sparse grid in $\mathbf{x}$ and a sparse grid in $\mathbf{v}$ is also an interesting variant. This is actually a very special case of a dimension adaptive [11] 6D sparse grid. Using dimension or also spatial adaptivity [18] intermediate variants can be designed. However, we concentrate on non-adaptive variants in this paper where we have a simple construction principle whose structure can be exploited when implementing the building blocks of our solver.

Comparing the full 6D sparse grid (SGxv) to the tensor product of two 3D sparse grids (SGxSGv), it becomes immediately clear that the number of points will grow quadratically in $N = 2^{\mathcal{L}}$ (up to a logarithmic scaling $\mathcal{L}^d$) for the SGxSGv variant as opposed to the linear growth for the SGxv variant since we have the product of two sparse grids. On the other hand, the structure of the interpolation becomes simpler and parallelization is much simpler for the SGxSGv variant as will be discussed in Section 5. The strongest argument in favour of the tensor product variant SGxSGv lays, however, in the structure of the problem: Since the sparse grid prefers hierarchical increments with anisotropic refinement, functions that can be well approximated by the sum of univariate functions are also well approximated on a sparse grid. When we consider the Vlasov–Poisson system, we know that filaments evolve in phase-space which cause large mixed derivatives. Hence, a sparse grid representation of the solution to the Vlasov equation can be problematic. However, the filamentation is limited to the (x_i, v_i)-planes, $i = 1, 2, 3$, as long as linear effects dominate the dynamics and the initial perturbation is aligned with the coordinate axes (cf. the discussion in Section 2.1). Since the (x_i, v_i)-plane is not sparsified in the SGxSGv sparse grid, a good representation of the distribution function is expected for problems with field-aligned coordinate axes and where no instabilities arise. Also note that the SGxSGv is a structured special case of a dimension-adaptive sparse grid. Since not only the pairs (x_i, v_i) are fully resolved but also pairs with different indices, we expect that a dimension adaptive algorithm will further increase compression. However, this will be at the price of less structure, making efficient implementation more difficult.

Finally, let us discuss the boundary conditions. We note that in our definition of the sparse grid, level 0 is the level of the boundary points. Due to the constraint on the ℓ_1 norm, the points of the sparse grid are most dense on the boundary. Since the domain is periodic in $\mathbf{x}$ in our model, choosing periodic boundary conditions along the $\mathbf{x}$-coordinates is natural. In this case, only the left boundary is included along each dimension. Along the velocity dimensions, we have an unbounded domain. On the other hand, our solutions have a Gaussian shape as discussed in Section 2.1 and therefore decay fast towards infinity. We therefore truncate the computational domain where the value of the distribution function is negligible and we have to set artificial boundary conditions. Since the solution is very small, not much information is kept at the boundary points which is a bit problematic in light of the fact that the sparse grid has most points at the boundary. A simple solution would be to set the solution equal to zero at the boundary and to skip level zero of the sparse grid. However, the Vlasov equation is a first order equation that does not allow for outflow boundary conditions which lead to unstable discretizations. Zero inflow boundary conditions are more suitable but require many boundary points. Another mathematically correct boundary closure are periodic boundaries. This is, of course, unphysical but since the solution is small at the boundary, the effect of the boundary condition is small and the advantage on a sparse grid is that we only need half the boundary points. In our experiments, we have chosen periodic boundary conditions.

In order to reduce the fraction of boundary points, we can modify the definition of the sparse grid by requiring the following two conditions on the level vector of the sparse grid

$$|\boldsymbol{\ell}|_{\ell_\infty} \leq \mathcal{L}, \quad |\boldsymbol{\ell}|_{\ell_1} \leq \mathcal{L} + d - 1. \tag{23}$$

This reduces the number of points on the boundary compared to a sparse grid defined by $|\boldsymbol{\ell}|_{\ell_1} \leq \mathcal{L} + d - 1$ only. On the other hand, we have experimentally seen accuracies of the same order for both grids for the velocity sparse grid. Alternatively, we could have placed boundary and mid points on the same level which would have a similar—but not exactly the same—effect. We have chosen the former modification due to its ease of implementation. Note that the modification (23) yields worse results for the spatial sparse grid.

4.2 Interpolation

The first building block of the split-semi-Lagrangian algorithm, is one-dimensional interpolation. Interpolation on a sparse grid can be realized by computing and evaluating the hierarchical surplus. The major problem is the complexity of this operation. As long as we use piecewise d-linear basis functions or the pth order functions constructed in [5], computing the hierarchical surplus is comparably cheap as explained in Section 3. As a next step, we have to evaluate the function on the origin of each characteristic, i.e. at $M_{\mathcal{L}}$ points. As mentioned in Section 3, each evaluation of the sparse grid function has a complexity of $p\mathcal{L}^d$. In Section 5, we will design an algorithm that exploits the structure of this problem which reduces the complexity of the whole interpolation step to $\mathcal{O}(pM_{\mathcal{L}})$ for the SGxSGv variant and to $\mathcal{O}(p\mathcal{L}^3 M_{\mathcal{L}})$ for the SGxv variant.

Another question is about a suitable interpolation method. On the full grid, using a cubic spline interpolator is common as mentioned in Sec. 2.2. However, computing the hierarchical surplus for splines is generally more expensive [22, Sec. 4.1]. On sparse grids, the piecewise d-linear basis is most-frequently used. Even though linear interpolation has its advantages on sparse grids when it comes to conservation properties (cf. Sec. 7), linear interpolation is too diffusive yielding much worse results compared to higher order interpolation [7]. Hence, using sparse grids with third or fourth order polynomial basis functions constructed as in [5] are a good compromise between accuracy and computational complexity. However, the main interpolation task is only one dimensional and we will see in section 5.4 how we can efficiently combine spline interpolation along the advection direction with higher order sparse grid interpolation along all other dimensions to improve accuracy.

4.3 Integration over velocity coordinates

In order to compute the particle density ρ we need to integrate over the velocity dimensions. If we have a tensor product of a sparse grid in space and velocity, this is a simple sparse grid integration [8, 16]: For each point in the $\mathbf{x}$-sparse grid, we have a $\mathbf{v}$-sparse grid over which we have to integrate.

If we have a full sparse grid (SGxv), the velocity coordinates of all the points on the sparse grid with one particular spatial coordinate form a three dimensional sparse grid and we can perform a three dimensional sparse grid integration for each point in space. Note that the representation of the distribution function needs to be in the hierarchical surplus along the spatial dimensions before performing the integration on the three dimensional sparse grids. If we want to have the value of the density at each point, we have to dehierarchize over the spatial dimensions in the end.

In our numerical experiments, we use the sparse grid trapezoidal rule to compute the integrals.

4.4 Solution of the Poisson problem

Once we have computed ρ and hence the right-hand side of the Poisson problem, we need to solve a three dimensional Poisson problem. In this paper, we focus on problems posed on a periodic domain in $\mathbf{x}$ and use a pseudospectral Poisson solver based on the sparse grid fast Fourier transform [10].

5 Improved efficiency and accuracy

As mentioned in Section 3 the advantage of the hierarchical basis is that we can leave out points without loosing too much in accuracy. On the other hand, the computational complexity of the algorithms increases. Efficient algorithms exist for some tasks but the evaluation of a sparse grid interpolant is the task in our algorithm with highest complexity, namely $\mathcal{O}(p\mathcal{L}^d M_{\mathcal{L}})$. However, our interpolation problem exhibits a certain structure: We only have a displacement along one direction and the displacement only depends on some of the dimensions. In this section, we are going to explain how to exploit this structure to improve on the efficiency. Finally, in the last subsection we will exploit the same idea also in order to improve on the accuracy of our method.

5.1 Mixed nodal-hierarchical representation

A key ingredient to our implementation is the observation that we can represent a sparse grid interpolant by its semi-hierarchical surplus, i.e. by values that are hierarchized along all but one dimension. In the derivation, we will only consider a two dimensional function and a hierarchization along dimension one for the ease of notation. Replacing the one hierarchized dimension by several ones or interchanging the indices is straightforward.

Let us consider the sparse grid interpolant I_f of a function f. It can be expressed as

$$\begin{aligned} I_f(x_1,x_2) &= \sum_{\ell_1=0}^{\mathcal{L}} \sum_{\ell_2=0}^{\mathcal{L}-\ell_1} \sum_{k_1\in\Omega_{\ell_1}^{\text{odd}}} \sum_{k_2\in\Omega_{\ell_2}^{\text{odd}}} v_{\boldsymbol{\ell},\boldsymbol{k}}\varphi_{\boldsymbol{\ell},\boldsymbol{k}}(x_1,x_2) \\ &= \sum_{\ell_1=0}^{\mathcal{L}} \sum_{k_1\in\Omega_{\ell_1}^{\text{odd}}} \varphi_{\ell_1,k_1}(x_1) \left[\sum_{\ell_2=0}^{\mathcal{L}-\ell_1} \sum_{k_2\in\Omega_{\ell_2}^{\text{odd}}} v_{\boldsymbol{\ell},\boldsymbol{k}}\varphi_{\ell_2,k_2}(x_2) \right]. \end{aligned} \tag{24}$$

In the second step, we have reordered the sum such that the expression in brackets is a one-dimensional hierarchical sum of basis functions for all index pairs (ℓ_1,k_1). Since the nodal basis is equivalent to the hierarchical, we can dehierarchize $v_{\boldsymbol{\ell},\boldsymbol{k}}$ along dimension 2 and use a nodal representation along dimension 2. This gives

$$I_f(x_1,x_2) = \sum_{\ell_1=0}^{\mathcal{L}} \sum_{k_1\in\Omega_{\ell_1}^{\text{odd}}} \varphi_{\ell_1,k_1}(x_1) \left[\sum_{k_2\in\Omega_{\mathcal{L}-\ell_1}} v_{\ell_1,k_1}(x_{\ell_2,k_2})\varphi_{\mathcal{L}-\ell_1,k_2}(x_2) \right] \tag{25}$$

In Figure 2, we have visualized the hierarchical representation (24) and the semi-hierarchical representation (25) for a sparse grid with maximum level $\mathcal{L}=2$ and periodic boundaries.

If we want to evaluate the representation (25), we only have a hierarchical representation along dimension 1 and a nodal representation along dimension 2. Hence, the cost of one evaluation goes down from order $\mathcal{L}^2$ to $\mathcal{L}$, or in the d-dimensional case from $\mathcal{L}^d$ to $\mathcal{L}^{d-1}$. Of course this requires that we have the semi-hierarchical surplus available. If we start from the functions values of the grid points, we can compute this by just hierarchizing along $d-1$ dimensions in the same way, but even slightly cheaper than the full hierarchical surplus. If we start of from the hierarchical surplus, we have to apply dehierarchization along one dimension with a complexity of order $M_{\mathcal{L}}$. So in this case, this way of evaluating a sparse grid function is only worthwhile if we want to evaluate the functions about $M_{\mathcal{L}}$ times. But this is the case in our method. The total complexity of the interpolation step will hence reduce to $\mathcal{O}(p\mathcal{L}^{d-1}M_{\mathcal{L}})$.

5.2 Constant displacement

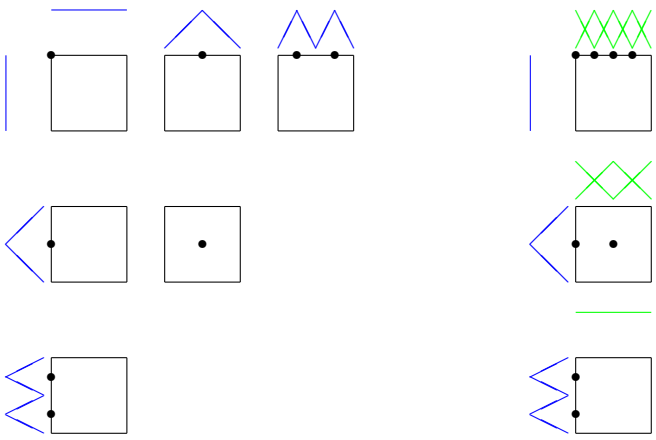
As a next step, we want to exploit the fact that we have a one-dimensional interpolation and that the displacement is constant along some dimensions. First, we consider the case where the displacement is fully constant. In the variant SGxSGv, all interpolations fall into this category. Even though the coefficients are not constant, the coefficient of the **x**-advections only depend on **v** and vice versa and the sparse grids are only including the **x** or **v** directions separately. In our analysis, we also allow for displacements that are dependent on the dimension along which we have the displacement, even though this is not the case for the interpolations in the SGxSGv method.

Let $f(x_1,x_2)$ be the original function and let I_f be its sparse grid interpolant. Now, we want to find a sparse grid representation of a function $g(x_1,x_2)$ that satisfies

$$I_g(x_1,x_2) = I_f(x_1,x_2+c(x_2)) \text{ for all } (x_1,x_2)\in\mathcal{S}, \tag{26}$$

where $\mathcal{S}$ is the set of points representing the sparse grid and $c(x_2)$ is the displacement along x_2 that is a function of x_2, i.e. constant along x_1. Note that we can again interchange the indices or add more hierarchical dimensions without displacement.

Now, we choose a semi-hierarchical representation that is nodal along dimension 2. For a point



(a) Hierarchical grid.

(b) Semi-hierarchical grid.

Figure 2: Sparse grid with maximum level $\mathcal{L} = 2$ and periodic boundary conditions. Part (a) shows the hierarchical increments with hierarchical basis functions. Part (b) shows the semi-hierarchical version where the horizontal basis functions are nodal with different level for each horizontal stripe. The hierarchical basis functions are drawn in blue and the nodal ones in green.

$(x_{L_1,K_1}, x_{L_2,K_2}) \in \mathcal{S}$, this representation reads

$$\begin{aligned} I_f(x_{L_1,K_1}, x_{L_2,K_2} + c(x_{L_2,K_2})) = \sum_{\ell_1=0}^{\mathcal{L}} \sum_{k_1 \in \Omega_{\ell_1}^{\text{odd}}} \varphi_{\ell_1,k_1}(x_{L_1,K_1}) \\ \left[\sum_{k_2 \in \Omega_{\mathcal{L}-\ell_1}} v^f_{\ell_1,k_1}(x_{L_2,K_2}) \varphi_{\mathcal{L}-\ell_1,k_2}(x_{L_2,K_2} + c(x_{L_2,K_2})) \right] \end{aligned} \tag{27}$$

$$I_g(x_{L_1,K_1}, x_{L_2,K_2}) = \sum_{\ell_1=0}^{\mathcal{L}} \sum_{k_1 \in \Omega_{\ell_1}^{\text{odd}}} \varphi_{\ell_1,k_1}(x_{L_1,K_1}) v^g_{\ell_1,k_1}(x_{L_2,K_2}).$$

If we now set

$$v^g_{\ell_1,k_1}(y_{L_2,K_2}) = \sum_{k_2 \in \Omega_{\mathcal{L}-\ell_1}} v^f_{l_1,k_1}(x_{L_2,K_2}) \varphi_{\mathcal{L}-\ell_1,k_2}(x_{L_2,K_2} + c(x_{L_2,K_2})) \tag{28}$$

for all combinations of (ℓ_1, k_1) and (L_2, K_2) on the sparse grid, Eq. (26) is satisfied. Now, we found a representation of the semi-hierarchical surplus of I_g from which we can either compute the hierarchical surplus by hierarchization along dimension 2 or the function values by dehierarchization along dimension 1. Evaluating equation (28) is independent of $\mathcal{L}$ since we have a nodal representation. In this way, we can find the representation of the interpolant of the shifted function by a combination of algorithms with linear complexity.

Following this implementation the complexity of each advection step reduces to $\mathcal{O}(pM_{\mathcal{L}})$ for a sparse grid interpolation on the SGxSGv variant.

5.3 Interpolation with coefficients constant along some dimensions

While we have found a very efficient implementation of the interpolation steps for the SGxSGv variant, the situation is more complicated for the SGxv variant since now the displacement is still dependent on some of the dimensions in the sparse grid. In order to still be able to somewhat reduce the complexity, we would like to partly apply the algorithm derived in the previous section. In order to understand the algorithm for a simple example, we consider the case of a three dimensional sparse grid with displacement along x_3 only depending on x_2, x_3. This means, we consider a function $f(x_1, x_2, x_3)$ with known interpolant I_f on the sparse grid $\mathcal{S}$ and want to compute a sparse grid representation of the function $g(x_1, x_2, x_3)$ such that

$$I_g(x_1, x_2, x_3) = I_f(x_1, x_2, x_3 + c(x_2, x_3)) \text{ for all } (x_1, x_2, x_3) \in \mathcal{S}. \tag{29}$$

Let us consider the point $(x_{L_1,K_1}, x_{L_2,K_2}, x_{L_3,K_3}) \in \mathcal{S}$ and use a representation of I_f that is nodal along x_3 and a representation of I_g that is nodal in x_2, x_3. We then have

$$\begin{aligned} I_f(x_{L_1,K_1}, x_{L_2,K_2}, x_{L_3,K_3} + c(x_{L_2,K_2}, x_{L_3,K_3})) = \\ \sum_{\ell_1=0}^{\mathcal{L}} \sum_{k_1 \in \Omega_{\ell_1}^{\text{odd}}} \varphi_{\ell_1,k_1}(x_{L_1,K_1}) \sum_{\ell_2=0}^{\mathcal{L}-\ell_1} \sum_{k_2 \in \Omega_{\ell_2}^{\text{odd}}} \varphi_{\ell_2,k_2}(x_{L_2,K_2}) \\ \left[\sum_{k_3 \in \Omega_{\mathcal{L}-\ell_1-\ell_2}} v^f_{\ell_1,k_1,\ell_2,k_2}(x_{L_3,K_3}) \varphi_{\mathcal{L}-\ell_1-\ell_2,k_3}(x_{L_3,K_3} + c(x_{L_2,K_2}, x_{L_3,K_3})) \right] \\ I_g(x_{L_1,K_1}, x_{L_2,K_2}, x_{L_3,K_3}) = \sum_{\ell_1=0}^{\mathcal{L}} \sum_{k_1 \in \Omega_{\ell_1}^{\text{odd}}} \varphi_{\ell_1,k_1}(x_{L_1,K_1}) v^g_{\ell_1,k_1}(x_{L_2,K_2}, x_{L_3,K_3}). \end{aligned} \tag{30}$$

From this representation, we can see that (29) is satisfied for all $(x_{L_1,K_1}, x_{L_2,K_2}, x_{L_3,K_3}) \in \mathcal{S}$ if we set for all combinations of (ℓ_1, k_1), (L_2, K_2) and (L_3, K_3)

$$\begin{aligned} v^g_{\ell_1,k_1}(x_{L_2,K_2}, x_{L_3,K_3}) = \sum_{\ell_2=0}^{\mathcal{L}-\ell_1} \sum_{k_2 \in \Omega_{\ell_2}^{\text{odd}}} \varphi_{\ell_2,k_2}(x_{L_2,K_2}) \\ \sum_{k_3 \in \Omega^{\text{odd}}_{\mathcal{L}-\ell_1-\ell_2}} v^f_{\ell_1,k_1,\ell_2,k_2}(x_{L_3,K_3}) \varphi_{\mathcal{L}-\ell_1-\ell_2,k_3}(x_{L_2,K_2} + c(x_{L_2,K_2}, x_{L_3,K_3})). \end{aligned} \tag{31}$$

This can now no longer be computed within linear complexity since we have one sum over hierarchical basis functions. Hence, we get a logarithmic scaling by $\mathcal{L}$. For the general case of a d dimensional sparse grid and a displacement depending on γ directions (different from the direction of the displacement), the complexity of the evaluation step is $\mathcal{O}(p\mathcal{L}^\gamma M_\mathcal{L})$. In particular, each velocity advection step on a six-dimensional sparse grid will have complexity of $\mathcal{O}(p\mathcal{L}^3 M_\mathcal{L})$ and each spatial advection step $\mathcal{O}(p\mathcal{L} M_\mathcal{L})$.

5.4 Mixed interpolation

When we solve the interpolation problem based on the efficient evaluation formulas discussed above, we always need the semi-hierarchical surplus that is hierarchical along all directions except for the one along which the displacement appears. In this section, we discuss how to exploit this fact to improve on the accuracy. As mentioned in Section 2.2, spline interpolation has proven very efficient for semi-Lagrangian methods on the full grid but computing the hierarchical surplus for a spline basis on a sparse grid has too high complexity. However, we do not need to hierarchize along the dimension along which we displace the points. This gives us the freedom to use another basis (i.e. another interpolator) along that dimension. To illustrate this, let us revisit the example of a function $f(x_1, x_2)$ of two variables where we have a displacement $c_1(x_1, x_2)$ along x_1 and a displacement $c_2(x_1, x_2)$ along x_2. For a given one-dimensional set of equidistant points Ω_ℓ as defined in (13), let us consider an arbitrary set of basis functions $(\phi_{\ell,k})_{k=1,\dots,2^\ell}$ associated with Ω_ℓ. Then, we represent f on the sparse grid by

$$I_f^{(1)}(x_1, x_2) = \sum_{\ell_1=0}^{\mathcal{L}} \sum_{k_1 \in \Omega_{\ell_1}^{\text{odd}}} \varphi_{\ell_1,k_1}(x_1) \left[\sum_{k_2 \in \Omega_{\mathcal{L}-\ell_1}} v^{(1)}_{\ell_1,k_1}(x_{\ell_2,k_2}) \phi_{\mathcal{L}-\ell_1,k_2}(x_2) \right] \tag{32}$$

for the interpolation problem along x_2. In order to solve the interpolation problem, we can thus succeed in two steps:

1. Compute the semi-hierarchical surplus $v^{(1)}_{\ell_1,k_1}(x_{\ell_2,k_2})$.
2. For each one-dimensional stripe $S_{x_{L_1,K_1}} = \{(x_1, x_2) \in \mathcal{S}_\mathcal{L} | x_1 = x_{L_1,K_1}\}$ defined by the x_1-coordinate x_{L_1,K_1} appearing on the sparse grid $\mathcal{S}_\mathcal{L}$, solve the one-dimensional interpolation problem in the basis $(\phi_{\mathcal{L}-\ell_1,k})_k$.

For the interpolation along x_1, we will then exchange the roles of the coordinates and use the following interpolant

$$I_f^{(2)}(x_1, x_2) = \sum_{\ell_2=0}^{\mathcal{L}} \sum_{k_2 \in \Omega_{\ell_2}^{\text{odd}}} \varphi_{\ell_2,k_2}(x_2) \left[\sum_{k_1 \in \Omega_{\mathcal{L}-\ell_2}} v^{(2)}_{\ell_2,k_2}(x_{\ell_1,k_1}) \phi_{\mathcal{L}-\ell_2,k_1}(x_1) \right]. \tag{33}$$

Note that unless we choose the basis $(\phi_{\mathcal{L}-\ell_1,k})_k$ equivalent to the basis used on the sparse grid, the representations (32) and (33) will not be equivalent, i.e. we use different representations for each one-dimensional interpolation problem. This also means (32) and (33) are—other than (25)—not equivalent to the representation with the fully hierarchical sparse grid interpolation. Generalization to higher dimensions is straight-forward. In our experiments, we will use cubic spline interpolations along the one-dimensional stripes combined with sparse grids with cubic basis functions in order to improve the accuracy (cf. Section 8.2).

5.5 Parallelization

Even though the number of grid points is reduced when using a sparse grid, for six-dimensional problems with reasonable resolution the number of points and the number of arithmetic operations can become so large that parallel computations become necessary. Due to the non-locality of the hierarchical basis, parallelizing sparse grid routines is not trivial.

On the other hand, the split-step semi-Lagrangian method provides trivial parallelism: Each one-dimensional interpolation step reduces to operations on one-dimensional stripes if a nodal basis is used. This is exploited in the parallelization strategy of the semi-Lagrangian library SeLaLib [1] which is the basis for our implementation: When we are computing the advection step along one dimension, the domain is decomposed along one or several of the other dimensions and distributed between the processors. Then each of the processors can work on its one-dimensional stripes independently of the other. The only thing that has to be done is a redistribution of the data once we turn to an advection step along a direction over which the data was distributed.

One of the advantages of the SGxSGv variant is the fact that we can apply this parallelization strategy. When computing the $\mathbf{x}$-advection steps, the problem is nodal along the $\mathbf{v}$ directions and vice versa. We can therefore distribute the distribution function over the points of the velocity sparse grid when performing the $\mathbf{x}$-advection steps, and along the points of the spatial sparse grid when performing the $\mathbf{v}$-advection steps.

6 Multiplicative δf method

It is well-known that sparse grids and, especially higher order polynomials on sparse grids, are badly suited to interpolate Gaussians [18, Sec. 4.2]. The major problem is that a quite large number of points is necessary before the interpolation starts to converge. Since the initial value of the distribution function is a Gaussian, the quality of the representation along the velocity dimensions needs a considerable resolution. On the other hand, we know that the solution often stays close to the equilibrium distribution (cf. Sec. 2.1). For this reason, we could only simulate the difference of the solution from the Gaussian equilibrium. Since the perturbation typically follows the same Gaussian decay, we consider a multiplicative splitting which we call multiplicative δf method.

In order to keep the presentation simple, we will explain our multiplicative splitting for the two-dimensional case. The main idea is to split the distribution function into a time-dependent part and a time-constant Gaussian part,

$$f(x,v,t) = g(x,v,t)h(v), \tag{34}$$

and to represent the function g on the sparse grid only while h is known analytically. If our solution is close to equilibrium, $g(x,v,t)$ will be close to one. For the Landau damping problem introduced in Sec. 2.1, the initial splitting would be

$$g(x,v,0) = 1 + \varepsilon\cos(kx), \quad h(v) = \frac{1}{2\pi}\exp(-0.5v^2). \tag{35}$$

Since h is independent of x, this part of the distribution function will not change for the x-advection. For a v-advection, on the other hand, we have

$$\begin{aligned} f(x, v + E(x)\Delta t, t) &= g(x, v + E(x)\Delta t)h(v + E(x)\Delta t) \\ &= g(x, v + E(x)\Delta t)\frac{h(v + E(x)\Delta t)}{h(v)}h(v). \end{aligned} \tag{36}$$

Hence, we have to perform a usual sparse grid integration for g followed by a scaling by $\frac{h(v+E(x)\Delta t)}{h(v)}$. Finally, when integrating over the velocity dimension, we analytically compute the weighted integrals

$$w_{\ell,k} := \int h(v)\varphi_{\ell,k}(v)\,\mathrm{d}v, \tag{37}$$

as quadrature weights for the hierarchical surpluses $v_{\ell,k}$. In three dimensions, on the SGxSGv grid, the density ρ at grid point $\mathbf{x}$ is thus computed as

$$\rho(\mathbf{x}) = \sum_{(\boldsymbol{\ell},\boldsymbol{k})\in G} w_{\ell_1,k_1} w_{\ell_2,k_2} w_{\ell_3,k_3} v_{\boldsymbol{\ell},\boldsymbol{k}}(\mathbf{x}), \tag{38}$$

where G denotes the index set defining the sparse grid along $\mathbf{v}$ direction. Of course, this would result in a considerable computational overhead since evaluating (37) requires evaluations of the error function. However, the weights are not depending on time and can be precomputed once.

In Section 8.3 we will show that a considerable improvement of the solution can be obtained for the Landau damping problem when the multiplicative δf method is applied. On the other hand, this procedure fails to improve accuracy when, for instance, an instability occurs, at least as long as $h(v)$ is not updated when the instability occurs.

7 A note on stability and conservation properties

A major disadvantage of a sparse grid solver is the fact that stability of sparse grid algorithms is not very well-understood or not guaranteed. A lack of L_2 stability was discussed in [6]. Also Bokanowski et al. [3] point out that they cannot provide stability estimates for their semi-Lagrangian solver.

Stability for a one-dimensional semi-Lagrangian Vlasov solver on a uniform grid was analyzed in [2]. However, effects from domain truncation are not considered. Due to the fact that we have stripes with very coarse refinement in a sparse grid, the sphere of influence of the boundary points is increased which is a potential source of numerical instability.

Indeed, if the solution is not well-resolved, unstable results have been obtained in our numerical experiments (cf. Sec. 8.4). From our experience, instabilities arise first when the solution is severly underresolved and mixed derivatives become large. As shown in Sec. 8.4, the method can be stabilized by switching to linear interpolation or alternating between linear and higher-order interpolation once the resolution becomes too bad. This introduces diffusion to the system and we observe that energy is dissipated depending on the amount of diffusion added.

Alternatively, we could add a small diffusive term on the scale of the smallest grid size as it was discussed in [19] for gyrokinetic simulations. However, a smallest grid size is not well-defined on a sparse grid. We have seen that diffusion on an increasing scale must be introduced when increasing simulation time. A better theoretical understanding of the stability of the method would be necessary to develop a robust stabilization method that does not unnecessarily deteriorate accuracy. Also the influence of the time step and the boundary conditions needs to be better understood.

Moreover, the sparse grid method does not mimic the conservation properties of the continuous model. In the continuous model, we have conservation of mass, momentum, energy and all L^p-norms. Only when using a linear interpolator and trapezoidal sparse grid integration, mass conservation is assured. We assume that improving on the conservation properties might increase the stability of the method as well.

8 Numerical results

We have implemented the sparse grid method as part of the Fortran library SeLaLib [1]. In this section, we study the performance of our method. We will often compare the sparse grid solution to a full grid solution. The full grid solution is computed with the split-step semi-Lagrangian scheme based on a cubic spline interpolator. In all simulations on the SGxSGv grid, the spatial sparse grid is a sparse grid of the form (19) with ℓ_1 bound and periodic boundary conditions and the velocity sparse grid is a modified sparse grid with ℓ_1 and ℓ_∞ bound on the level vector according to (23) and with periodic boundary conditions. The given maximal level $\mathcal{L}_x$ refers to the upper bound in the ℓ_1 norm of the level vector and $\mathcal{L}_v$ to the upper bound in the ℓ_∞ norm of the level vector. Except for the comparative study in Section 8.2 a sparse grid with cubic basis functions is used together with cubic spline interpolation along the one dimensional stripes with displacement (cf. Sec. 5.4). The interpolation steps are implemented with the efficient algorithms devised in Sections 5.1-5.3. The multiplicative δf method is only applied if this is explicitly noted.

In Section 8.1-8.3, we will consider the weak Landau problem in d dimensions ($d = 2, 3$) with initial value

$$f(\mathbf{x},\mathbf{v}) = \left(1 + 0.01\sum_{i=1}^{d}\cos(0.5x_i)\right)\frac{1}{(2\pi)^{d/2}}\exp(-0.5\|\mathbf{v}\|_2^2). \tag{39}$$

In order to assess the quality of the solution, we will compare the time evolution of the electric energy (cf. e.g. Figure 3) to the decay rate predicted by linear theory (cf. Sec. 2.1). For the given parameter of $\varepsilon = 0.01$ the electric energy should show this decay over long times. However, in numerical solutions on a grid, one typically observes an approximative recurrence of the initial state after a certain time that is proportional to the reciprocal of the velocity grid spacing, $\frac{1}{\Delta v}$ (see [15]). Of course, the simulated solution is incorrect as soon as this artificial recurrence appears.

8.1 Comparison of full grid, SGxSGv, and SGxv

Let us consider the weak Landau damping problem in four dimensions. We compare the two variants of the sparse grid with a full grid solution. The time evolution of the potential energy is shown in Figure 3. In this experiment, we use a sparse grid with cubic polynomial basis functions and cubic spline interpolation along the dimension with displacement. On the full grid, we use 32 points along each dimension. The number of levels on the SGxSGv grid is chosen such that the damping rate is recovered for at least the same time interval as on the full grid. For this we need $\mathcal{L}_x = 5$ and $\mathcal{L}_v = 7$. This means we have a grid of $M_x \times M_v = 112 \times 1024$ grid points. Compared to the full grid, we can considerably reduce the number of grid points along the $\mathbf{x}$ directions. However, there is no reduction along the $\mathbf{v}$ directions. Also using the multiplicative δf method, we cannot recover the damping rate with a maximum level less than $\mathcal{L}_v = 7$. Therefore, we can conclude that the use of a sparse grid only reduces the number of points in the $\mathbf{x}$ grid for this two-dimensional simulation. Note that it is clear from the structure of the initial value that a representation of f on a sparse grid is more difficult in velocity space where we have a function depending on v_1 multiplied by a function depending on v_2 while the perturbations along x_1 and x_2 are additive. Note that we observe an artificial recurrence on the sparse grid as well which is, however, damped and appears earlier than it would for a full grid with the same resolution of the finest level present in the sparse grid.

In Figure 3, we also show the results obtained with a full sparse grid SGxv of maximum level $\mathcal{L} = 10$ (ℓ_1 bound) and periodic boundary conditions. In this case, the number of grid points is 66304, i.e. only 6% compared to the full and 58% compared to the SGxSGv variant. On the other hand, we have to bear in mind that the complexity of the $\mathbf{v}$-advection steps is of the order $\mathcal{O}(p\mathcal{L}^2)$ per grid point, i.e. about a factor $\mathcal{L}^2 = 100$ higher than for the FG and SGxSGv variants (cf. Sec. 5). Hence, the computing time is expected to be highest for our SGxv variant. Since the solution of the SGxv method is the worst, the numerical experiments are in line with our theoretical considerations in Section 4.1 that a complete sparse grid will not be very successful in representing the distribution function governed by the Vlasov–Poisson equation.

In order to numerically verify our estimates on the complexity of our algorithms from Sec. 5, the CPU times for the three experiments are reported in Table 1. For each run, 2,000 time steps have been simulated. The simulations were performed in serial on an Intel Ivy Bridge notebook processor at 3.0 GHz and the SeLaLib library was compiled with the GNU Fortran compiler 4.8 and optimization level `-O3`. Normalizing the CPU times by the number of grid points, we see that the time per grid point is increased by a factor 5 for SGxSGv and by a factor 292 for SGxv, respectively, compared to the FG solution. Considering the fact that the complexity is increased by a factor $\mathcal{L}^2$ for the SGxv grid, this means the complexity constant is increased by a factor 5 or 2.9 for SGxSGv and SGxv, respectively, in our prototype implementation. Given the fact that the sparse grid algorithms described in Sec. 5 require (de)hierarchization steps, the constants were expected to be larger than on the full grid. Note however that the codes are not completely optimized for the complexity constant so that these values should rather be used as qualitative estimates.

Table 1: Comparison of CPU times for FG, SGxSGv, and SGxv.

method	no. of grid points	CPU time [s]	CPU time/grid point [s]
FG	1,048,576	448	$4.3 \cdot 10^{-4}$
SGxSGv	114,688	216	$2.3 \cdot 10^{-3}$
SGxv	66,304	8,270	$1.2 \cdot 10^{-1}$

8.2 Comparison of various interpolators

In this section, we again consider the four dimensional Landau problem on a $M_x \times M_v = 112 \times 1024$ SGxSGv grid and compare various interpolators. The resulting potential energy plots are shown in Figure 4.

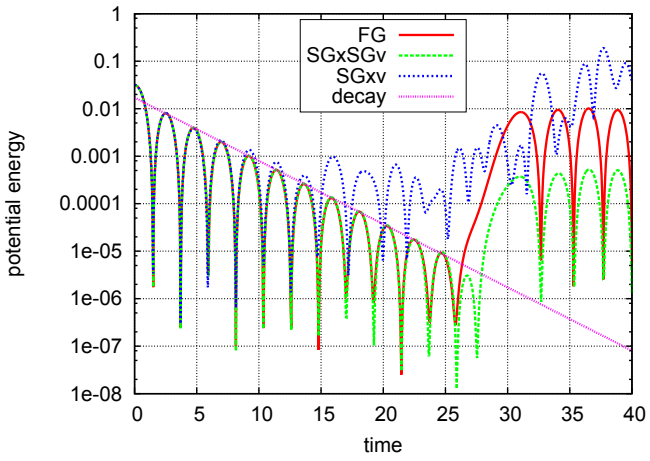


Figure 3: Weak Landau damping in 4D. Electric energy for simulations on full grid (FG), SGxSGv and SGxv

It can be clearly seen that linear interpolation is too dissipative. This is not an effect of the sparse grid but can likewise be observed on the full grid. Comparing the cubic sparse grid interpolator with the cubic sparse grid with cubic splines on the one-dimensional stripes, we can clearly see that applying mixed interpolation helps in increasing the accuracy of the interpolation.

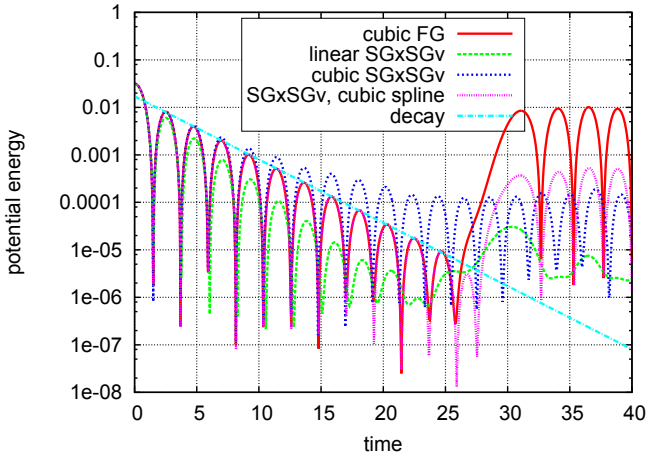


Figure 4: Weak Landau damping in 4D. Electric energy for simulations on SGxSGv grid with 112×1024 points and various interpolators.

8.3 Effects of the multiplicative δf method

So far, we have only looked at the Landau damping in four dimensions. If we use the same values for the maximum level $\mathcal{L}_x = 5$ and $\mathcal{L}_v = 7$ also in six-dimensions, the damping rate can only be recovered over a time interval of about 10 (see Figure 5). However, if we apply the multiplicative δf method as described in Section 6, we can again recover the damping rate until time 25 as in the four dimensional case. The sparse grid only contains $M_x \times M_v = 272 \times 7808$ mesh points which only amounts to 0.2% of the 32^6 points on the full grid with similar accuracy. This shows the potential of the multiplicative δf method. Note that the multiplicative δf method did not noticeably improve the results of our four dimensional experiments and is therefore not used in the four dimensional experiments reported here.

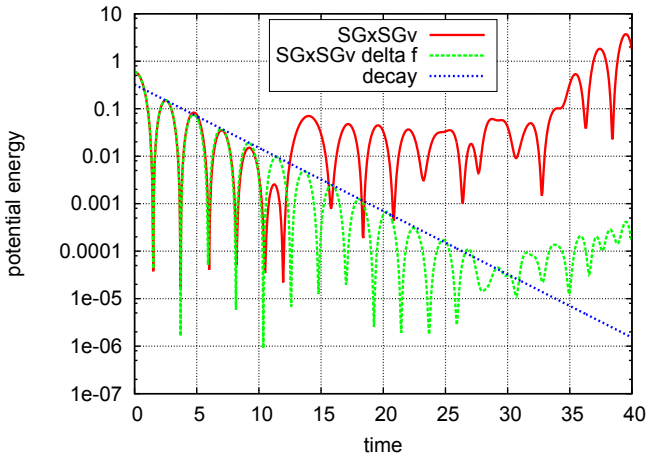


Figure 5: Weak Landau damping in 6D. Electric energy for simulations with SGxSGv grid with $M_x \times M_v = 272 \times 7808$ points with and without multiplicative δf modeling.

8.4 The two stream instability: Effects of instabilities

In this section, we want to consider a second test case: the two stream instability with initial value

$$f(\mathbf{x},\mathbf{v}) = \frac{1}{4\pi}\left(1 + 0.001\sum_{j=1}^{2}\cos(0.2x_j)\right)\left(\mathrm{e}^{-0.5(v_1-2.4)^2} + \mathrm{e}^{-0.5(v_1+2.4)^2}\right)\mathrm{e}^{-0.5v_2^2}. \tag{40}$$

The perturbation along x_1 will yield an instability as discussed in Section 2.1. During a first linear phase the energy grows and a hole structure evolves in (x_1, v_1) space. Around time 35, the energy stops to grow and nonlinear effects take over. During the nonlinear phase, particles are trapped in the hole structure and smaller and smaller filaments evolve. In the light of the discussion on the structure of the problem in Section 4.1, this problem is more difficult to represent on a sparse grid than the Landau damping problem. We can see from the results in Figure 6 that a SGxSGv sparse grid of 262,144 points in total ($\mathcal{L}_x = 6, \mathcal{L}_v = 7$) can nicely recover the linear phase. In the nonlinear phase, the electric energy only keeps an oscillating structure on the right level but the error in the solution is rather large.

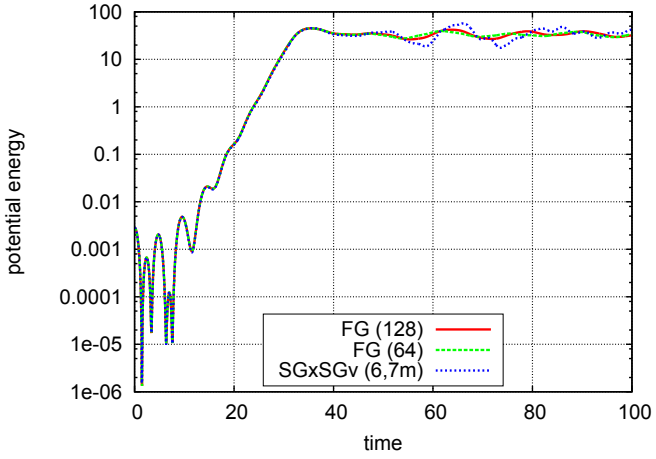


Figure 6: Two stream instability in 4D. Electric energy for simulation on full grid and SGxSGv.

When solving the extension of the problem to six dimensions

$$f(\mathbf{x},\mathbf{v}) = \frac{0.5}{(2\pi)^{1.5}} \left(1 + 0.001 \sum_{j=1}^{3} \cos(0.2 x_j)\right) \left(\mathrm{e}^{-0.5(v_1-2.4)^2} + \mathrm{e}^{-0.5(v_1+2.4)^2}\right) \mathrm{e}^{-0.5(v_2^2+v_3^2)}. \tag{41}$$

we have observed a numerical instability at the same resolution (see Fig. 7). For this simulation, we have applied the multiplicative δf method along v_2 and v_3. We can see that the simulation can be stabilized when switching to a linear interpolator at time 45. At this point in time the resolution is poor and dissipation needs to be added to keep the simulation stable. However, we can see that the propagation method becomes too diffusive and energy is dissipated. In this case, we have added too much diffusion. If we instead use a linear interpolation each 6th step from time 40, the solution still remains stable but is much less diffusive. From our experiments, we have seen that more and more diffusion needs to be added as time evolves. In order to design a stable algorithm that adds as little diffusion as possible, one would need to analyze stability of the sparse grid method.

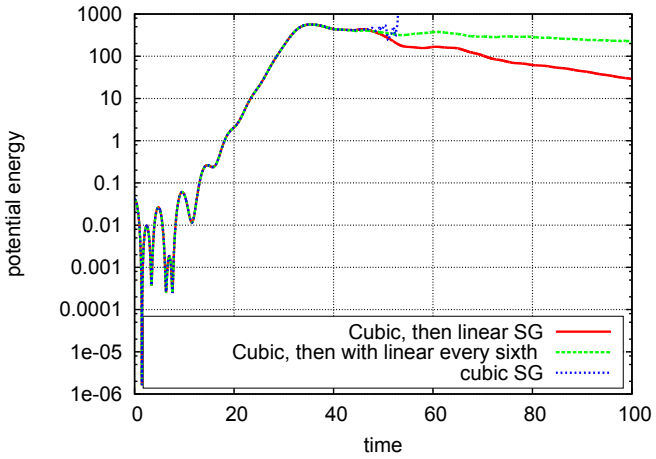


Figure 7: Two stream instability in 6D. Electric energy for simulation on SGxSGv with cubic interpolation, cubic interpolation until time 45 and linear afterwards, and linear interpolation every 6th step from time 40 onwards.

9 Conclusions

We have introduced a semi-Lagrangian Vlasov solver on a sparse grid. From both theoretical considerations and numerical performance, we have seen that a tensor product of a sparse grid in spatial and a sparse grid in velocity coordinates is better suited than a full six-dimensional sparse grid. We have introduced an efficient implementation that improves on the efficiency by exploiting the special structure of the problem. Moreover, we have devised a multiplicative δf method to defeat the problem of poor representation of Gaussians on a sparse grid.

From the results, we can conclude that good compression and reduced computational complexity can be obtained in six dimensions for problems close to equilibrium or when the filaments are aligned with the coordinate directions. For this reason, sparse grids might be interesting in a hybrid method where the bulk of the domain is resolved by a sparse grid while small structures are additionally resolved using localized full grids or particles.

References

[1] SeLaLib. http://selalib.gforge.inria.fr/.

[2] N. Besse and M. Mehrenberger. Convergence of classes of high-order semi-Lagrangian schemes for the Vlasov–Poisson system. *Math. Comput.*, 77(261):93–123, 2008.

[3] O. Bokanowski, J. Garcke, M. Griebel, and I. Klompmaker. An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton–Jacobi Bellman equations. *J. Sci. Comput.*, 55(3):575–605, 2013.

[4] T. Boyd and J. Sanderson. *The Physics of Plasmas*. Cambridge University Press, Cambridge, 2003.

[5] H.-J. Bungartz. Finite elements of higher order on sparse grids, 1998. Habilitationsschrift.

[6] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

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

[8] T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18(3-4):209–232, 1998.

[9] R. T. Glassey. *The Cauchy problem in kinetic theory*. SIAM, Philadelphia, 1996.

[10] K. Hallatschek. Fouriertransformation auf dünnen Gittern mit hierarchischen Basen. *Numer. Math.*, 63:83–97, 1992.

[11] M. Hegland. Adaptive sparse grids. *ANZIAM J.*, 44, 2003.

[12] K. Kormann. A semi-Lagrangian Vlasov solver in tensor train format. *SIAM J. Sci. Comput.*, 37(4):B613–B632, 2015.

[13] C. Kowitz, D. Pflüger, F. Jenko, and M. Hegland. The combination technique for the initial value problem in linear gyrokinetics. In J. Garcke and M. Griebel, editors, *Sparse Grids and Applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 205–222. Springer Berlin Heidelberg, 2013.

[14] P. Lions and B. Perthame. Propagation of moments and regularity for the 3-dimensional Vlasov–Poisson system. *Invent. Math.*, 105, 1991.

[15] G. Manfredi. Long-time behavior of nonlinear Landau damping. *Phys. Rev. Lett.*, 79:2815–2818, 1997.

[16] E. Novak and K. Ritter. High dimensional integration of smooth functions over cubes. *Numer. Math.*, 75(1):79–97, 1996.

[17] B. Peherstorfer, S. Zimmer, and H.-J. Bungartz. Model reduction with the reduced basis method and sparse grids. In J. Garcke and M. Griebel, editors, *Sparse Grids and Applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 223–242. Springer Berlin Heidelberg, 2013.

[18] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, 2010.

[19] M. Pueschel, T. Dannert, and F. Jenko. On the role of numerical dissipation in gyrokinetic Vlasov simulations of plasma microturbulence. *Comput. Phys. Commun.*, 181(8):1428 – 1437, 2010.

[20] E. Sonnendrücker. Numerical methods for the Vlasov equations. 2013.

[21] E. Sonnendrücker, J. Roche, P. Bertrand, and A. Ghizzo. The semi-Lagrangian method for the numerical resolution of the Vlasov equation. *J. Comput. Phys.*, 149(2):201–220, 1999.

[22] J. Valentin and D. Pflüger. Hierarchical gradient-based optimization with B-splines on sparse grids. 2015.