Block-Structured Adaptive Mesh Refinement Algorithms for Vlasov Simulation

J. A. F. Hittinger[a,*], J. W. Banks[a]

[a]*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA*

Abstract

Direct discretization of continuum kinetic equations, like the Vlasov equation, are under-utilized because the distribution function generally exists in a high-dimensional (>3D) space and computational cost increases geometrically with dimension. We propose to use high-order finite-volume techniques with block-structured adaptive mesh refinement (AMR) to reduce the computational cost. The primary complication comes from a solution state comprised of variables of different dimensions. We develop the algorithms required to extend standard single-dimension block structured AMR to the multi-dimension case. Specifically, algorithms for reduction and injection operations that transfer data between mesh hierarchies of different dimensions are explained in detail. In addition, modifications to the basic AMR algorithm that enable the use of high-order spatial and temporal discretizations are discussed. Preliminary results for a standard 1D+1V Vlasov-Poisson test problem are presented. Results indicate that there is potential for significant savings for some classes of Vlasov problems.

Keywords: kinetic simulation, Vlasov, finite-volume methods, adaptive mesh refinement

1. Introduction

The Vlasov-Maxwell system of equations is a fundamental kinetic model that describes weakly-coupled plasma dynamics. The Vlasov equation is a partial differential equation in phase space, $(\mathbf{x}, \mathbf{v}) \in \mathbb{R}^N \times \mathbb{R}^M$ for $N, M \in [1, 2, 3]$ such that $M \geq N$, that describes the evolution in time, $t \in \mathbb{R}_+$, of a particle distribution function, $f(\mathbf{x}, \mathbf{v}, t) \in \mathbb{R}_+$, in the presence of electromagnetic fields, $\mathbf{E}(\mathbf{x}, t) \in \mathbb{R}^N$ and $\mathbf{B}(\mathbf{x}, t) \in \mathbb{R}^N$. For a particle species α, the Vlasov equation is

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

where the particle charge and mass are q_α and m_α, respectively. Both imposed and self-generated electric and magnetic fields, $\mathbf{E}$ and $\mathbf{B}$, respectively, are responsible for the Lorentz force in (1) and are the solutions of Maxwell's equations (or a specialization thereof):

$$\nabla \times \mathbf{E} + \frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} = 0, \tag{2a}$$

$$c^2 \nabla \times \mathbf{B} - \frac{\partial \mathbf{E}}{\partial t} = \frac{\mathbf{j}}{\epsilon_0}, \tag{2b}$$

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}, \tag{2c}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{2d}$$

*Corresponding author. Mailing address: Center for Applied Scientific Computing, L-561, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA. Phone: 925-422-0993. Fax: 925-423-8704.

Email addresses: `hittinger1@llnl.gov` (J. A. F. Hittinger), `banks20@llnl.gov` (J. W. Banks) where ϵ_0 is the permittivity of free space and c is the *in vacuo* speed of light. The total charge density, ρ, and the total current, $\mathbf{j}$, are the sums over contributions from all species α,

$$\rho(\mathbf{x}, t) = \sum_{\alpha} \rho_{\alpha}(\mathbf{x}, t) = \sum_{\alpha} q_{\alpha} \int_{\mathbb{R}^d} f_{\alpha} \, d\mathbf{v}, \tag{3a}$$

$$\mathbf{j}(\mathbf{x}, t) = \sum_{\alpha} \mathbf{j}_{\alpha}(\mathbf{x}, t) = \sum_{\alpha} q_{\alpha} \int_{\mathbb{R}^d} \mathbf{v} f_{\alpha} \, d\mathbf{v}, \tag{3b}$$

and these moments of the distribution function nonlinearly couple Maxwell's equations to the Vlasov equation.

The Vlasov-Maxwell system and related models are fundamental non-equilibrium descriptions of plasma dynamics. Non-equilibrium kinetic effects in plasmas play a crucial role in fusion applications. Understanding and controlling wave-particle interactions is important to the success of inertial confinement fusion, where resulting resonant responses can interfere with the intended deposition of laser energy [1]. In magnetic confinement fusion, gyrokinetic models, which are a reduced form of the Vlasov equations [2, 3, 4], are used to better understand the physical mechanisms controlling the core conditions, in particular micro-turbulence, which is at the origin of the so-called anomalous transport [5].

The Vlasov model also has applicability beyond fusion plasmas. Collisionless shocks in astrophysics, which are thought to be driven by electrostatic and electromagnetic instabilities [6, 7, 8], can be accurately modeled by the Vlasov-Maxwell system. The Vlasov-Poisson system, where Gauss' Law (2c) is sufficient to describe the relationship between the electrostatic field and the charge density, is being used in particle beam accelerator design [9]. Laser isotope separation is another application area for Vlasov-Maxwell models [10].

While these kinetic models may have great relevance, their numerical approximation for problems of interest have been constrained primarily by computational cost. For $N = 3$, the distribution functions in the full Vlasov model have a phase-space domain of six dimensions. Directly discretizing phase space, an approach alternatively referred to as *grid-based*, *Eulerian*, or *continuum* methods, incurs a computational cost that scales geometrically with the number of dimensions. Thus, while for over forty years work has been done on the continuum numerical discretization of the Vlasov equation [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31], continuum Vlasov methods have been applied primarily to lower-dimensional – so-called 1D+1V and 1D+2V – problems. Application of continuum Vlasov to four dimensions (2D+2V) and above has been limited [30, 32, 31, 33]. In contrast, the *particle-based* particle-in-cell (PIC) [34] method has dominated kinetic Vlasov simulation. PIC methods use Monte-Carlo sampling techniques in velocity space to reduce the high-dimensional cost and evolve "clouds" of particles through a Lagrangian form of the Vlasov equation. Maxwell's equations, however, are solved on an overlaid computational mesh (hence, "in cell"). While this approach is generally less expensive than continuum Vlasov discretization, PIC results contain inherent statistical noise that generally vanishes only as the square root of the number of particles.

As computer speed and memory have increased, direct discretization of the Vlasov-Maxwell system has become more feasible, but hardware improvements alone are insufficient to make full-phase-space, continuum Vlasov codes practical. However, as with PIC methods, tremendous savings could be realized if continuum approaches could reduce the number of cells used to represent phase space. One means to this end is to employ adaptive mesh refinement and to resolve only those regions of phase space of greatest variation or importance. For instance, block-structured adaptive mesh refinement (AMR) in phase space could concentrate cells in the vicinity of localized structure, such as particle trapping regions. In addition, flux-based explicit Eulerian schemes have time-step restrictions that, for Vlasov, are typically dominated by the maximum particle velocity limits of the phase-space domain. AMR allows for high-aspect ratio cells in these regions, which can result in significant increases in time step size without a loss of accuracy, since little particle density or variation is present at these extreme velocity boundaries.

Adaptive mesh refinement has a limited history in Vlasov simulation. AMR has been used with PIC methods in the simulation of heavy-ion accelerators [35, 9]. Recent work in this area uses a wavelet-based approach [23, 25], where the semi-Lagrangian interpolation is based upon a multi-level wavelet basis and where the local depth of the wavelet hierarchy is used to increase or decrease the local mesh refinement. This approach generates a near-optimal grid, but progress in this direction seems to have stalled. It may be the case that the less regular grid structure may introduce other complications, for example, in the construction of moments, that make this approach less attractive.

In this paper, we present a block-structured adaptive mesh refinement approach suitable for the continuum discretization of the Vlasov-Maxwell system. As a proof-of-concept, we demonstrate the ideas and techniques in the context of a simpler system, the Vlasov-Poisson model, which is presented in Section 2 along with the basic flux-based Eulerian discretization we employ. Thus, we will not address the control of electromagnetic wave reflections at coarse-fine interfaces; methods to minimize such reflections are addressed elsewhere in the literature [36, 35]. In Section 3, we discuss the block-structured AMR strategy, its benefits, and the challenges presented by Vlasov problems. We specifically address a subset of these issues that we have resolved in order to demonstrate a successful Vlasov-AMR implementation. Sample calculations are presented in Section 4, and we conclude with a discussion of the future algorithmic advances that will allow additional gains from AMR applied to Vlasov simulation.

2. Model Problem and Discretization

Both for our purposes here as well as for many physically interesting problems, the Vlasov-Maxwell system can be significantly simplified by assuming an electrostatic limit with stationary ions. The electrostatic limit corresponds to an assumption of small magnetic field strength. The assumption of stationary ions is appropriate when the ion time scales are large compared to that of the electrons, which is typically the case. With these assumptions, the Vlasov equation (1) for the electron probability density function f becomes

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f - \mathbf{E} \cdot \nabla_{\mathbf{v}} f = 0, \tag{4}$$

under a suitable nondimensionalization. Here $\mathbf{E}$ is the electric field, $\mathbf{x}$ is the physical space and $\mathbf{v}$ is the velocity. In the electrostatic limit, only Gauss' law (2c) is relevant. Representing the electric field in terms of an electrostatic potential, $\mathbf{E} = \nabla_{\mathbf{x}}\phi$, Gauss' law becomes the Poisson equation:

$$\nabla^2_{\mathbf{x}}\phi = \rho_e - \rho_i = \int f \, d\mathbf{v} - 1. \tag{5}$$

Here the constant, unit background charge, $\rho_i = 1$, is the manifestation of the immobile ion (proton) assumption.

For the purposes of discussing the new adaptive discretization algorithms, we make one final simplifying assumption of a so-called 1D+1V phase space (*i.e.*, one spatial dimension and one velocity dimension). The final system of governing equations is thus succinctly written as

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

$$\frac{\partial^2 \phi}{\partial x^2} = \int_{-\infty}^{\infty} f \, dv - 1. \tag{6b}$$

Note that, as characteristic of all Vlasov-Maxwell-type systems, we have a higher-dimensional variable, $f(x, v)$, coupled to a lower-dimensional variable, $\phi(x)$.

In order to discretize the model Vlasov-Poisson system, we restrict our attention to a finite domain. For the physical coordinate we let $x \in [-L, L]$ and apply periodic boundary conditions. Other boundary conditions are also possible, but for the initial-value problems considered here, a periodic condition is appropriate. For the velocity coordinate, we truncate the domain and consider $v \in [v_{\min}, v_{\max}]$. This introduces an artificial boundary where we apply a characteristic boundary condition. Outgoing characteristics are extrapolated and incoming characteristics carry values from an unperturbed Maxwellian distribution.

Our discretization follows the Eulerian finite-volume formulation developed in [37, 38]. The 1D Vlasov equation (6a) is rewritten in flux-divergence form as

$$\frac{\partial}{\partial t} f + \frac{\partial}{\partial x}(vf) - \frac{\partial}{\partial v}(Ef) = 0. \tag{7}$$

Phase space is divided into cells using a Cartesian grid with mesh spacings Δx and Δv in the x- and v-dimensions respectively. Integrating over a single computational cell and dividing by the volume $\Delta x \Delta v$, we obtain the exact system of ordinary differential equations

$$\frac{d}{dt}\bar{f}_{ij} = -\frac{1}{\Delta x}\left(\langle vf\rangle_{i+\frac{1}{2},j} - \langle vf\rangle_{i-\frac{1}{2},j}\right) \quad + \frac{1}{\Delta v}\left(\langle Ef\rangle_{i,j+\frac{1}{2}} - \langle Ef\rangle_{i,j-\frac{1}{2}}\right), \tag{8}$$

where the cell average $\bar{f}_{ij}$ is defined as

$$\bar{f}_{ij} \equiv \frac{1}{\Delta x \Delta v}\int_{V_{ij}} f\,dx\,dv.$$

As in [37, 38], the angle bracket notation is used to indicate face averages, for example,

$$\langle f\rangle_{i+\frac{1}{2},j} = \frac{1}{\Delta v}\int_{v_{j-1/2}}^{v_{j+1/2}} f(x_{i+1/2}, v)\,dv.$$

The face-averaged fluxes are approximated to fourth order as

$$\begin{aligned}
\langle vf\rangle_{i+\frac{1}{2},j} &\approx \bar{v}_j \langle f\rangle_{i+\frac{1}{2},j} + \frac{\Delta v}{24}\left(\langle f\rangle_{i+\frac{1}{2},j+1} - \langle f\rangle_{i+\frac{1}{2},j-1}\right), \\
\langle Ef\rangle_{i,j+\frac{1}{2}} &\approx \bar{E}_i \langle f\rangle_{i,j+\frac{1}{2}} - \frac{1}{48}\left(\bar{E}_{i+1} - \bar{E}_{i-1}\right)\left(\langle f\rangle_{i+1,j+\frac{1}{2}} - \langle f\rangle_{i-1,j+\frac{1}{2}}\right).
\end{aligned}$$

Notice that, because v is only a function of v and E is only a function of x, the notion of a face average is redundant, and the angle bracket is replaced by an an overbar. For more details concerning this high-order finite-volume formalism, refer to [37, 38].

The quantity $\bar{v}_j$ is directly computed as an exact cell average (recall that v is an independent variable). The cell-averaged electric field is computed from a potential ϕ; to fourth-order, this is

$$\bar{E}_i \approx \frac{1}{12\Delta x}\left[8(\bar{\phi}_{i+1} - \bar{\phi}_{i-1}) - \bar{\phi}_{i+2} + \bar{\phi}_{i-2}\right].$$

The cell-averaged potential is obtained by solving a discretization of the Poisson equation (6b) :

$$30\bar{\phi}_i - 16(\bar{\phi}_{i+1} + \bar{\phi}_{i-1}) + (\bar{\phi}_{i+2} + \bar{\phi}_{i-2}) = 12\Delta x \bar{\rho}_i, \tag{9}$$

where

$$\bar{\rho}_i = 1 - \Delta v \sum_{j=-v_{\max}}^{v_{\max}} \bar{f}_{ij}.$$

This discretization leads to a linear system with a nearly pentadiagonal matrix (boundary conditions slightly alter the pentadiagonal structure).

For reasons explained in Section 3, the Poisson problem is always represented on the finest mesh in configuration space, and so the resulting linear algebra problem can be LU-decomposed once for each level of refinement and stored. Periodic boundary conditions in x lead to a singular system, which is a well-known problem that is easily addressed by projecting out the portion of $\bar{\rho}(x)$ residing in the null space of the matrix. This amounts to ensuring that $\sum_i \bar{\rho}(x_i) = 0$, and in so doing, we ensure that $\bar{\phi}(x)$ is normalized around zero. Of course, since we take a derivative of $\bar{\phi}(x)$ to get $\bar{E}(x)$, the offset has no effect on the solution.

To complete the description of the discretization, a procedure to derive face averages from cell averages must be identified. We use the scheme developed in [39, 32], which has the property that, for well-represented solutions, a fourth-order centered approximation is used. As solution features become sharp on a given mesh, upwind numerical dissipation is introduced to smooth out those features consistently. The scheme is described in detail in [39, 32], but we provide a brief overview here as well.

We focus on the determination of the face average $\langle f\rangle_{i+\frac{1}{2},j}$; other averages follow similar derivations. The scheme has many similarities to the popular WENO [40] method and uses many of the tools developed in the literature on that topic. The face average is constructed as a weighted sum of two third order approximations:

$$\langle f\rangle_{i+\frac{1}{2},j} \approx w_{i+\frac{1}{2},j,L}\langle f\rangle_{i+\frac{1}{2},j,L} + w_{i+\frac{1}{2},j,R}\langle f\rangle_{i+\frac{1}{2},j,R}, \tag{10}$$

with

$$\langle f\rangle_{i+\frac{1}{2},j,L} \approx \frac{1}{6}\left(-\bar{f}_{i-1,j} + 5\bar{f}_{i,j} + 2\bar{f}_{i+1,j}\right) \tag{11}$$

and

$$\langle f\rangle_{i+\frac{1}{2},j,R} \approx \frac{1}{6}\left(2\bar{f}_{i,j} + 5\bar{f}_{i+1,j} - \bar{f}_{i+2,j}\right). \tag{12}$$

Here the "L" and "R" indicate left- and right-biased, third-order approximations. With ideal weighting, $w_{i+\frac{1}{2},j,L} = w_{i+\frac{1}{2},j,R} = \frac{1}{2}$, equation (10) becomes the centered, fourth-order approximation. Using the standard WENO methodology, provisional weights, $\hat{w}_{i+\frac{1}{2},j,L}$ and $\hat{w}_{i+\frac{1}{2},j,R}$, are determined. To maximize the upwind diffusion in the final numerical method, we assign the larger weight to the upwind, third-order approximation and the smaller weight for the downwind, third-order stencil. Thus the final weights are determined as

$$\begin{array}{ll} \text{if } (v_j > 0), & \left\{\begin{array}{lcl} w_{i+\frac{1}{2},j,L} & = & \max(\hat{w}_{i+\frac{1}{2},j,L}, \hat{w}_{i+\frac{1}{2},j,R}), \\ w_{i+\frac{1}{2},j,R} & = & \min(\hat{w}_{i+\frac{1}{2},j,L}, \hat{w}_{i+\frac{1}{2},j,R}), \end{array}\right. \\ \text{else} & \left\{\begin{array}{lcl} w_{i+\frac{1}{2},j,L} & = & \min(\hat{w}_{i+\frac{1}{2},j,L}, \hat{w}_{i+\frac{1}{2},j,R}), \\ w_{i+\frac{1}{2},j,R} & = & \max(\hat{w}_{i+\frac{1}{2},j,L}, \hat{w}_{i+\frac{1}{2},j,R}). \end{array}\right. \end{array} \tag{13}$$

Note that, as with traditional WENO schemes, convergence rates near certain types of critical points (points with many zero derivatives) may be less than optimal. Additional modifications to the provisional weights can be made to alleviate this deficiency [41].

For the temporal discretization of the semi-discrete Vlasov equation (8), any stable method can be used. We choose the standard explicit fourth-order Runge-Kutta scheme. At each stage in the Runge-Kutta update, we solve the discrete potential equation (9) prior to evaluating the phase-space flux divergence as given by the right-hand side of (8).

Consider the ODE initial value problem

$$\frac{df}{dt} = L(f,t), \tag{14a}$$

$$f(0) = f_0. \tag{14b}$$

The RK4 discretization for the ODE between time level n and $n+1$ is

$$f^{n+1} = f^n + \Delta t \sum_{s=1}^{4} b_s k_s, \tag{15a}$$

$$k_s = L\left(f^{(s)}, t^n + c_s\Delta t\right), \tag{15b}$$

$$f^{(s)} = f^n + \alpha_s \Delta t k_{s-1}, \tag{15c}$$

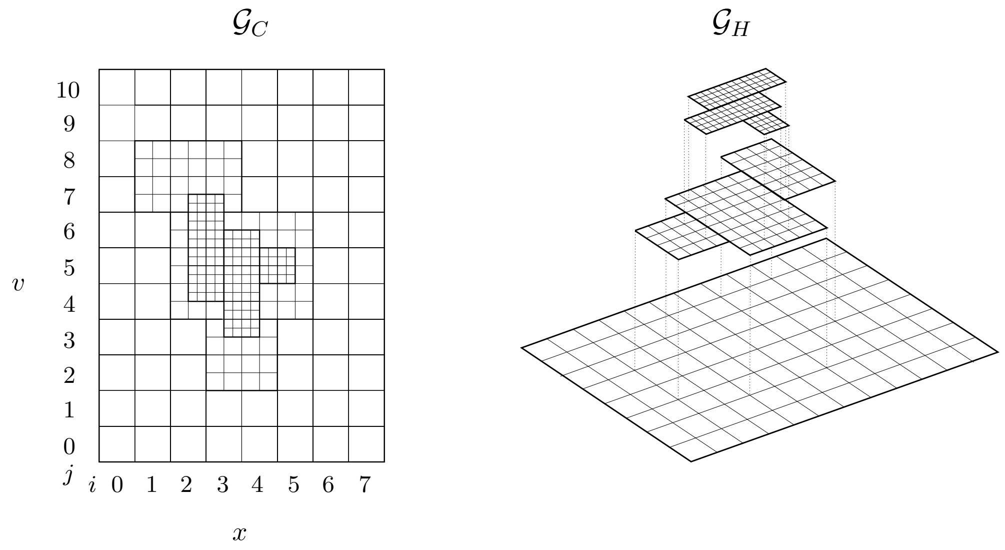

Figure 1: An example of a three-level, block-structured AMR hierarchy. On the left, the composite refined grid $\mathcal{G}_C$ is shown. On the right, the corresponding mesh hierarchy $\mathcal{G}_H$ with overlapping patches is shown. All patches on the same level have the same refinement ratio relative to the coarsest level; in this case, the refinement ratios are two and four for the intermediate and finest levels, respectively. Note that each level is comprised of a collection of patches completely contained within the patches of the next coarser level.

with $\boldsymbol{\alpha} = [0, 1/2, 1/2, 1]$, $\mathbf{b} = [1/6, 1/3, 1/3, 1/6]$, and $\mathbf{c} = [0, 1/2, 1/2, 1]$. Acknowledging that the operator L is, in our case, of flux-divergence form, we can write, for example, in one dimension,

$$
\begin{aligned}
f_i^{n+1} &= f_i^n + \Delta t \sum_{s=1}^{4} b_s k_{i,s}, && \text{(16a)}\\
&= f^n - \Delta t \sum_{s=1}^{4} b_s \left[F_{i+1/2}\left(f^{(s)}\right) - F_{i-1/2}\left(f^{(s)}\right) \right], && \text{(16b)}\\
&= f^n - \Delta t \left[\sum_{s=1}^{4} b_s F_{i+1/2}\left(f^{(s)}\right) - \sum_{s=1}^{4} b_s F_{i-1/2}\left(f^{(s)}\right) \right], && \text{(16c)}\\
&= f^n - \Delta t \left[F^*_{i+1/2} - F^*_{i-1/2} \right], && \text{(16d)}
\end{aligned}
$$

where $F^*_{i+1/2}$ are accumulated interface fluxes.

3. Block Structured AMR Algorithms

Block-structured adaptive mesh refinement [42, 43] is a natural fit for certain Vlasov-Maxwell problems. Frequently, important fine-scale features in phase space, which could substantially benefit from higher resolution, only occupy limited regions in phase space. In contrast to the semi-structured, octree-based grids that were used in the earlier Vlasov-AMR work [21], hierarchical block-structured AMR is based upon rectangular grid patches at different refinement levels in a global Cartesian index space, as shown in Figure 1. Using a local error estimate or some detection of rapid variation in the solution to identify regions of interest, cells that should be refined are tagged. Tagged cells are grouped and expanded minimally to form rectangular patches that are inserted into the next level in the hierarchy. Slightly larger refinement regions can be used to reduce the frequency of regridding. The refinement process can be repeated recursively to form a hierarchy of refinement levels, each composed of multiple patches.

Connectivity information is kept to a minimum in this scheme because the global Cartesian index space provides a simple mechanism by which to identify the relationships between patches and levels. Within a level, patches contiguous in indices are neighboring, and across levels, the same is true, after adjusting by the net refinement ratio between the two levels. In general, for explicit methods, communication between patches is accomplished through ghost cells. As an additional savings, by maintaining a consistent solution on all patches, even those covered by patches on a finer level, time refinement algorithms that allow for nested subcycling on finer levels can be devised.

Despite all of the previous work on block-structured AMR, applying the technique to Vlasov simulation introduces several new challenges. First and foremost, at least two mesh hierarchies must be maintained: one in the $\mathbb{R}^N$ configuration space and one in the $\mathbb{R}^N \times \mathbb{R}^M$ phase space. Different kinetic species will, in general, have different masses and temperatures; the bulk of the corresponding particles will therefore occupy different ranges of particle velocity, and the structures arising from resonant responses will occur in different regions of phase space. Thus, each kinetic species should have its own mesh hierarchy. Thus, new algorithms for the simultaneous advancement and coordination of multiple hierarchies are required, and more importantly, efficient algorithms to enable communication between the hierarchies are required. From a parallel implementation perspective, a hierarchy for each species also allows for increased task parallelism when each hierarchy is assigned to a subset of processors; with no collisions, kinetic species only communicate through the lower-dimensional configuration space, so, given the electromagnetic state, high-dimensional flux computations and updates can naturally be done in parallel.

In this paper, it is our goal to demonstrate solutions to the fundamental issues that must be addressed to make effective use of AMR in Vlasov simulation. Specifically, we will discuss:

- **Basic modifications due to discretization.** Using high-order finite volume and a high-order multi-stage schemes departs somewhat from the standard, nominally second-order block-structured AMR approach. We describe the modified algorithms we use, for example, the intra-hierarchy interpolation operations and the synchronous time integration algorithm.
- **Inter-hierarchy transfer operations.** The coupling of problems of different dimension and their representation on separate hierarchies necessitates the creation of inter-hierarchy reduction and injection transfer algorithms. We discuss algorithms that achieve this efficiently.
- **Regridding for multiple AMR hierarchies.** Regridding hierarchies, when multiple related hierarchies are present, requires additional constructs for coordination.

In the following subsections, we address each of these areas, describing in more details the issues involved and explaining our solution approach. We will not specifically address efficient parallel decomposition strategies in this work.

These new AMR algorithms, combined with the high-order discretizations presented in Section 2 have been implemented in the Vlasov code VALHALLA[1]. This code makes use of the block-structured AMR library SAMRAI [44], which has been used in prior plasma-fluid simulations [45]. SAMRAI is capable of handling dimensions above three as well as the simultaneous existence of multiple and lower hierarchies, possibly of different dimension. A graph-based, distributed implementation of mesh metadata is employed within SAMRAI to provide excellent scaling to tens of thousands of processors. SAMRAI also provides fairly sophisticated, high-level AMR algorithm abstractions, but the simultaneous advancement of multiple related hierarchies, as required by Vlasov-Poisson, does not fit into these integration strategies and has thus required substantial additional development.

3.1. Basic modifications due to discretization

Our base discretization uses a method-of-lines approach, where spatial operators are first discretized using a nominally fourth-order spatial discretization and then the resulting semi-discrete

[1]Vlasov Adaptive Limited High-order Algorithms for Laser Applications

system is integrated using the standard four-stage, fourth-order explicit Runge-Kutta method. Fortunately, for a high-order finite-volume implementation, the restriction algorithm to obtain a coarse cell average from fine cell averages remains the simple summation used for lower-order schemes.

3.1.1. Synchronous, multi-stage time advancement algorithm

In practice, an asynchronous process with time step subcycling on finer cells is typically used for explicit, space-time discretizations [42], but we chose to start with a synchronous update for simplicity. For a synchronized update (*i.e.*, a single Δt for all levels), the RK4 algorithm (15)-(16) for a conservation law on a single-hierarchy is summarized in Algorithm 1. Looping over stages, the predictor states and fluxes are computed, and the fluxes are accumulated. The predictor-state algorithm is laid out in Algorithm 2, and the flux divergence and accumulation algorithm is sketched in Algorithm 3. We note that, for this conservative form, we accumulate a flux variable as in (16) so that we can construct a flux divergence using a temporally fourth-order flux for the final update. Such flux accumulation eliminates an explicit re-fluxing step, since the final update can be done from finest to coarsest levels, and the accumulated flux can be averaged down so that a single update using the highest-quality flux can be done on each level. The update is computed from the accumulated fluxes as shown in Algorithm 4, and regridding is done if a user-defined number of time steps have elapsed.

Algorithm 1 Multi-Level, Single-Hierarchy Flux-Divergence RK4 Advance

```
$k \leftarrow 0$
for all Stages $s \leftarrow 1, 4$ do
    ComputePredictorState($k, s, f_{pred}$)
    ComputeRHS($f_{pred}, s, k, F_{accum}$)
end for
ComputeUpdate($F_{accum}, f_{new}$)
if time to regrid then
    Regrid all levels
end if
Compute next $\Delta t$
```

Algorithm 2 Multi-Stage Predictor State Computation

```
procedure ComputePredictorState($k, s, f_{pred}$)
    $t \leftarrow t_{old} + c_s \cdot \Delta t$
    $f_{pred} \leftarrow f_{old} + \alpha_s \cdot \Delta t \cdot k$
    Interpolate up to ghost cells on finer levels of $f_{pred}$
    Exchange ghost cells on each level of $f_{pred}$
    Apply boundary conditions to $f_{pred}$
end procedure
```

Algorithm 3 Multi-Stage Right-Hand Side Evaluation and Flux Accumulation

```
procedure ComputeRHS($f_{pred}, s, k, F_{accum}$)
    for all Levels $l \leftarrow 1, L$ do
        for all Patches $p$ do
            $F_{pred} \leftarrow$ computeFluxes($f_{pred}, t$)
        end for
        Exchange fluxes between patches on level $l$
    end for
    for all Levels $l \leftarrow L, 1$ do
        for all Patches $p$ do
            $k \leftarrow$ fluxDivergence($F_{pred}$)
            $F_{accum} \leftarrow F_{accum} + b_s \cdot F_{pred}$
        end for
    end for
end procedure
```

Algorithm 4 Multi-Stage, Multi-Level Flux-Divergence Update Computation

```
procedure ComputeUpdate($F_{accum}, f_{new}$)
    for all Levels $l \leftarrow L, 1$ do
        for all Patches $p$ do
            $\delta f \leftarrow$ fluxDivergence($F_{accum}$)
            $f_{new} \leftarrow f_{old} + \Delta t \cdot \delta f$
            Coarsen fluxes down to level $l - 1$
        end for
    end for
    Coarsen fine data down for $f_{new}$
end procedure
```

To integrate the Vlasov-Poisson system, the time advancement algorithm must be adapted to allow the simultaneous advancement of multiple phase-space hierarchies. In addition, the Poisson equation represents an instantaneous constraint, and we chose most self-consistent strategy of re-evaluating the Poisson equation at each predictor state. An alternative possibility is to extrapolate ϕ in time to avoid some of the intermediate field solves and the associated parallel synchronization; investigating this approach is left for future work.

The associated modifications to Algorithm 1 are shown in Algorithm 5. The main differences are that the major steps are each now computed for all hierarchies and that additional steps to evaluate instantaneous constraints have been inserted on predicted or updated states are obtained. Note that we do not recompute the potential until after any possible regridding since the regridding step for the configuration space hierarchy is not independent of the phase space hierarchies in our current implementation. More details about this are given in Section 3.3.

3.1.2. Conservative, limited, high-order interpolation algorithm

Fine-patch cells are filled from coarse cells either when new fine-level patches are created or when fine-patch ghost cells at coarse-fine interfaces are filled. For second-order discretizations of first-order differential operators, slope-limited linear reconstruction is generally used to obtain fine-cell averages from the coarse grid while controlling non-physical oscillations. To obtain a fourth-order reconstruction while controlling oscillations, several techniques exist, including least squares [46], unfiltered [47], and explicitly-filtered [48] high-order interpolations. We adopt a slightly different approach and make use of standard WENO5 [40] interpolants that have been analytically integrated to obtain explicit cell-average interpolation formulas.

Algorithm 5 Synchronous, Multi-Stage, Vlasov-Poisson Multi-Hierarchy Advance

```
$k \leftarrow 0$
for all Stages $s \leftarrow 1, 4$ do
    for all Hierarchies $H$ do
        ComputePredictorState($k, s, f_{pred}$)
    end for
    ComputeInstantaneousConstraints($f_{pred}, \phi$)
    for all Hierarchies $H$ do
        ComputeRHS($f_{pred}, \phi, s, k, F_{accum}$)
    end for
end for
for all Hierarchies $H$ do
    ComputeUpdate($F_{accum}, f_{new}$)
end for
if time to regrid then
    Regrid all hierarchies
end if
ComputeInstantaneousConstraints($f_{new}, \phi$)
Compute next $\Delta t$
```

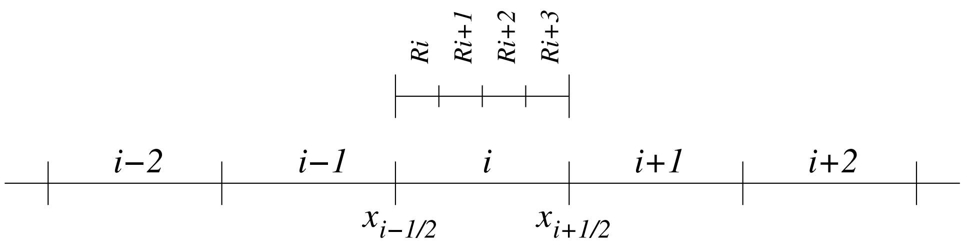

Figure 2: Relationship of fine to coarse mesh in global index space with a refinement ratio of $R = 4$. The five coarse cells shown are used to determine the cell averages in the four fine cells that subdivided cell i.

We assume cell-averaged data $\bar{u}_{\mathbf{i}}$ on a coarse mesh with mesh size $\mathbf{h}$ and an overlapping fine mesh with mesh size $\mathbf{h}^f$, such that

$$h_j^f = h_j/R_j, \qquad j = 1, 2, \ldots, D, \tag{17}$$

where each R_j is a positive integer. Our goal is to construct a high-order approximation to fine-mesh cell-averaged values $\bar{u}^f_{\mathbf{i}_f}$ such that the integral over the fine mesh exactly equals the integral over the coarse mesh. In addition, since initialization of fine mesh from coarse mesh may be done in regions of high gradients, we seek an adaptive interpolation scheme that will inhibit the creation of unphysical oscillations.

For our fourth-order discretization, the five-point WENO5 scheme is sufficient. In the general approach to obtain an interpolation in cell i, one is given five cell-averages $\bar{u}_{i+e}$, $e = -2, -1, 0, 1, 2$, and a location $x_{i-1/2} \leq x \leq x_{i+1/2}$, as shown in Figure 2. It is useful at this point to define some auxiliary quantities:

$$D_{i+n} = \bar{u}_{i+n+1} - \bar{u}_{i+n}, \qquad n = -2, -1, 0, 1, \tag{18a}$$

$$\Delta_{i+p} = D_{i+p} - D_{i+p-1}, \qquad p = -1, 0, 1. \tag{18b}$$

The algorithm proceeds for a uniform mesh as follows:

1. Compute smoothness detectors $\beta_i^{(r)}$, $r = 0, 1, 2$:

$$\beta_i^{(0)} = \frac{13}{12}\Delta_{i+1}^2 + \frac{1}{4}\left(D_{i+1} - 3D_i\right)^2, \tag{19a}$$

$$\beta_i^{(1)} = \frac{13}{12}\Delta_i^2 + \frac{1}{4}\left(D_i + D_{i-1}\right)^2, \tag{19b}$$

$$\beta_i^{(2)} = \frac{13}{12}\Delta_{i-1}^2 + \frac{1}{4}\left(3D_{i-1} - D_{i-2}\right)^2; \tag{19c}$$

2. Compute the absolute interpolation weights $\alpha_i^{(r)}$, $r = 0, 1, 2$:

$$\alpha_i^{(r)} = d_r/(\epsilon + \beta_i^{(r)})^2, \tag{20}$$

   where $d_0 = 3/10$, $d_1 = 3/5$, $d_2 = 1/10$, and ϵ is a small positive value to avoid division by zero (typically $\epsilon = 10^{-6}$);

3. Compute the relative interpolation weights $\omega_i^{(r)}$, $r = 0, 1, 2$:

$$\omega_i^{(r)} = \alpha_i^{(r)} / \left(\sum_{s=0}^{2} \alpha_i^{(s)}\right); \tag{21}$$

4. Compute the interpolants $v_i^{(r)}(x)$, $r = 0, 1, 2$:

$$v_i^{(r)}(x) = h \sum_{m=1}^{2} \left[\left(\sum_{j=0}^{m-1} \bar{u}_{i+j-r} \right) \left(\frac{\sum\limits_{\substack{l=0 \\ l\neq m}}^{2} \left[\prod\limits_{\substack{q=0 \\ q\neq m,l}}^{2} \left(x - x_{i-1/2} - h(q-r)\right) \right]}{\prod\limits_{\substack{l=0 \\ l\neq m}}^{2} h(m-l)} \right) \right]; \tag{22}$$

5. Compute the combined interpolant $v_i(x)$:

$$v_i(x) = \sum_{r=0}^{2} \omega_i^{(r)} v_i^{(r)}(x). \tag{23}$$

The result $v_i(x)$ is an interpolant constructed from cell average values such that

$$\int_{x_{i-1/2}}^{x_{i+1/2}} v_i(x)\, dx = h\bar{u}_i. \tag{24}$$

In smooth regions, $v_i(x)$ is an $O\big(h^5\big)$ approximation pointwise; in regions where under-resolution generates oscillations, the scheme drops to $O\big(h^3\big)$ pointwise (at worst) by adapting its stencil through the nonlinear weights so as to bias towards interpolations that are less oscillatory.

For adaptive mesh refinement, we can analytically integrate (22) over the fine-mesh cells to arrive at simple algebraic equations for the fine-mesh cell-averages. For a refinement ratio of R, we integrate (22) over each of the intervals $[x_{i-1/2} + sh/R, x_{i-1/2} + (s+1)h/R]$, $s = 0, 1, \ldots, R-1$, (see Figure 2). Define for $r = 0, 1, 2$, $A_i^{(r)}$, $B_i^{(r)}$, and $C_i^{(r)}$:

$$\begin{aligned} A_i^{(0)} &= \bar{u}_i + \tfrac{1}{6}(2D_{i+1} - 5D_i), \\ A_i^{(1)} &= \bar{u}_i - \tfrac{1}{6}(D_i + 2D_{i-1}), \\ A_i^{(2)} &= \bar{u}_i - \tfrac{1}{6}(4D_{i-1} - D_{i-2}), \end{aligned} \qquad \begin{aligned} B_i^{(0)} &= 2D_i - D_{i-1}, \\ B_i^{(1)} &= B_i^{(2)} = D_{i-1}, \end{aligned} \qquad \begin{aligned} C_i^{(0)} &= \Delta_{i+1}, \\ C_i^{(1)} &= \Delta_i, \\ C_i^{(2)} &= \Delta_{i-1}. \end{aligned} \tag{25}$$

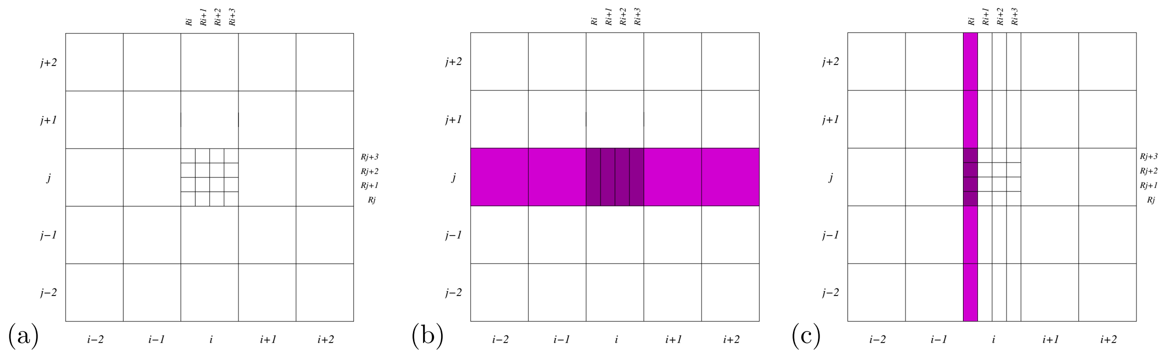

Figure 3: (a) The twenty-five coarse cells in 2D used to interpolate the sixteen fine-cell averages in cell i for a uniform refinement of $R = 4$. (b) The cells involved in the partial interpolation in the x-direction for cell i. The result are the four cell averages that are fine in the x-direction but coarse in the y-direction. (c) The cells involved in the partial interpolation in the y-direction for sub-cells at fine-grid location Ri. This is repeated for all fine-grid locations in the x-direction.

Then the three fine-mesh cell-averaged interpolated values in cell $(Ri + s)$ are

$$\left(\bar{u}^f_{Ri+s}\right)^{(r)} = A_i^{(r)} + B_i^{(r)}\left(\frac{2s+1}{2R}\right) + C_i^{(r)}\left(\frac{3s^2+3s+1}{6R^2}\right), \tag{26}$$

for $r = 0, 1, 2$. Note that, to ensure exact conservation to round-off, we renormalize the average of the fine cells to the original coarse cell average by

$$\left(\bar{u}^f_{R(i+1)-1}\right)^{(r)}_{\text{renorm}} = \left(\bar{u}^f_{R(i+1)-1}\right)^{(r)} + \bar{u}_i - \sum_{s=0}^{R-1}\left(\bar{u}^f_{Ri+s}\right)^{(r)}/R; \tag{27}$$

this ensures that the truncation errors are equidistributed amoung the sub-cells.

In implementation, advantage can be made of the many repeated factors. Notably, for any given coarse cell i, the fifteen auxiliary variables (18) and (25) and the $\omega_i^{(r)}$ only need be computed once for the R fine cells in cell i. Similarly, for a fixed refinement R, the $2(R-2)$ functions of $s = 0, 1, \cdots, R-2$ in (26) are the same for any coarse cell i.

The direct, though not most efficient, extension of the one-dimensional algorithm to multiple dimensions is to apply the method dimension-by-dimension. Thus, it is sufficient to build code to handle the 1D problem, and the multi-dimensionality is handled through data management, *i.e.*, the input provided to the 1D routines and the memory destinations to which the results are written.

Consider the 2D case where the refinement ratios are R_0 and R_1 in the x_0- and x_1-directions, respectively. An example with $R_0 = R_1 = R = 4$ is shown in Figure 3(a). In cell i, we first compute cell averages for cells refined only in x_0 using (26). This is shown in Figure 3(b). The result for each cell i is R_0 new sub-cell values.

The same operation is then applied in x_1-direction, but the input values are no longer the coarse-grid averages, but are now the partially-refined averages from the previous step. This is shown in Figure 3(c). The result for each fine cell $Ri + s$ is R_1 sub-cell values, and since there are R_0 x_1-interpolations per coarse cell i, R_0R_1 sub-cell values.

3.2. Inter-hierarchy transfer operations

Data transfer between hierarchies of different dimensionality requires the formulation of special algorithms and auxiliary data structures. While the injection of lower-dimensional data in the higher-dimensional space is a straight-forward constant continuation in the new dimensions, the reduction of higher-dimensional data into the lower-dimensional space requires the application of an operator across the dimensions that are removed, such as the integrals in the moment reductions (3).

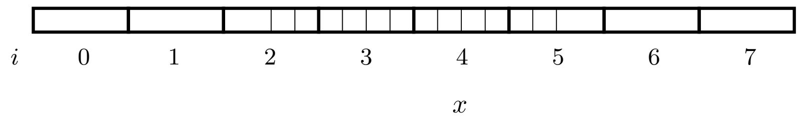

Figure 4: The configuration-space composite grid corresponding to the composite grid $\mathcal{G}_C$ depicted in Figure 1 after reduction.

The application of reductions across an AMR hierarchy is not in itself new. For example, the computation of mathematical norms is frequently executed on AMR hierarchies, and any norm is the reduction of higher-dimensional data into a scalar value. Lower-dimensional slices across a hierarchy are often used for visualization. A special case of such a slice reduction was developed for the laser-plasma interaction code ALPS [45], where the paraxial light wave sweeps required plasma densities on lower-dimensional planar slices. The challenge for the Vlasov system is that the reductions are the result of accumulation. Spatial fidelity and accuracy must be maintained while accumulating across the velocity dimensions, and such reductions must be done efficiently and without double-counting (recall the overlapping mesh hierarchy shown in Figure 1).

In addition to the need to obtain data at equivalent resolution, the act of orchestrating a reduction operation across a hierarchy in parallel requires several auxiliary structures. In fact, to preserve generality in the mesh refinement, auxiliary data structures are also helpful for injection operations. We next discuss two moment reduction algorithms that have been developed for the Valhalla code, followed by a brief description of the associated injection algorithm.

3.2.1. Moment reduction algorithm

Consider the composite grid depicted in Figure 1. Let us assume that we will accumulate on coarse-grid index j for coarse-grid index $i = 1$. Along $i = 1$ there are cells of two resolutions since there are two cells, $j = 7, 8$, that have been refined. If we do not preserve the finest resolution in the accumulation on j, we will lose some known information about the spatial structure in the remaining x-direction. To preserve the finest-resolution spatial information, we should subdivide in the x-direction to obtain cells of uniform resolution (in i). Using this principle, the corresponding composite grid after reduction is shown in Figure 4.

One might consider subdividing coarse cells without spatially reconstructing the data within the coarse cell, but this would result in an $O(h)$ error in the summation. To see this, consider that, for a refinement ratio of R, the relationship between a fine grid cell average and a coarse grid average in a single dimension is

$$\bar{u}^f_{Ri+s} = \bar{u}_i - h\left(\frac{R-2s-1}{2R}\right)\partial_x u\big|_{i+\frac{2s+1}{2R}} + O\big(h^2\big)\,, \tag{28}$$

where $s = 0, 1, \ldots, R-1$. Thus, to preserve higher-order accuracy, all coarse data must be reconstructed before averaging to the finest resolution and before applying the reduction operation.

One can make use of the limited interpolation operators defined in Section 3.1.2. However, the data to be reduced should be well-resolved (or else it would have been refined), so a less expensive option is to use a linear interpolation. One can construct such an interpolant by averaging over sub-cells the fifth-order interpolant,

$$v(x) = \sum_{j=-2}^{2} \gamma_j\left(\frac{x}{h}\right)\bar{u}_{i+j} + O\big(h^5\big)\,, \tag{29}$$

with

$$\gamma_{-2}(\eta) = \left(5\eta^4 - 20\eta^3 + 15\eta^2 + 10\eta - 6\right)/120, \tag{30a}$$
$$\gamma_{-1}(\eta) = \left(-20\eta^4 + 60\eta^3 + 30\eta^2 - 150\eta + 54\right)/120, \tag{30b}$$
$$\gamma_{0}(\eta) = \left(30\eta^4 - 60\eta^3 - 120\eta^2 + 150\eta + 94\right)/120, \tag{30c}$$
$$\gamma_{1}(\eta) = \left(-20\eta^4 + 40\eta^3 + 90\eta^2 - 10\eta - 26\right)/120, \tag{30d}$$
$$\gamma_{2}(\eta) = \left(5\eta^4 - 15\eta^2 + 4\right)/120, \tag{30e}$$

as was done to arrive at (26). The resuling formula for the fine-mesh interpolations is

$$\bar{u}^f_{Ri+s} = \sum_{j=-2}^{2} b_j(s)\bar{u}_{i+j} + O\left(h^5\right) \tag{31}$$

with

$$b_{-2}(s) = \left(p_4(s) - 5p_3(s) + 5p_2(s) + 5p_1(s) - 6\right)/120, \tag{32a}$$
$$b_{-1}(s) = \left(-4p_4(s) + 15p_3(s) + 10p_2(s) - 75p_1(s) + 54\right)/120, \tag{32b}$$
$$b_{0}(s) = \left(6p_4(s) - 15p_3(s) - 40p_2(s) + 75p_1(s) + 94\right)/120, \tag{32c}$$
$$b_{1}(s) = \left(-4p_4(s) + 5p_3(s) + 30p_2(s) - 5p_1(s) - 26\right)/120, \tag{32d}$$
$$b_{2}(s) = \left(p_4(s) - 5p_2(s) + 4\right)/120, \tag{32e}$$

where

$$p_k(s) = \sum_{j=0}^{k} \frac{k!}{j!(k-j)!} s^j. \tag{33}$$

At this point, it is helpful to introduce some notation. We denote an N-vector of integers by $\mathbf{i} = (i_0, i_1, \ldots, i_{N-1}) \in \mathbb{Z}^N$ and a patch, $\mathcal{P}$, by a pair of N-vectors that indicate the lower and upper cells of the patch: $\mathcal{P} = [\mathbf{i}_{\text{lo}}, \mathbf{i}_{\text{hi}}]$. The restriction from an N-vector to a $(N-1)$-vector by removing the j-th element is restr_j, *e.g.*,

$$\text{restr}_j\, \mathbf{i} = (i_0, i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_{N-1}) \in \mathbb{Z}^{N-1}. \tag{34}$$

and we define $\text{restr}_j\, \mathcal{P} = [\text{restr}_j\, \mathbf{i}_{\text{lo}}, \text{restr}_j\, \mathbf{i}_{\text{hi}}]$. We also define a replacement operator $\text{repl}_j(a,b)$ that operates on patches and that replaces the j-th element of the lower and upper N-vectors by a and b, respectively:

$$\text{repl}_j(a,b)\mathcal{P} = [(i_0, i_1, \ldots, i_{j-1}, a, i_{j+1}, \ldots, i_{N-1}), (i_0, i_1, \ldots, i_{j-1}, b, i_{j+1}, \ldots, i_{N-1})]. \tag{35}$$

A collection of patches at a refinement level l is denoted by $\mathcal{L}_l$, and a patch hierarchy $\mathcal{G}_H$ is defined as the set of refinement levels with an N-vector refinement ratio defined between each level and the coarsest $(l = 0)$, *i.e.*,

$$\mathcal{G}_H = \{\mathcal{L}_l, 0 \le l \le L-1 : \exists\ \mathbf{r}_l^{l+1} \in \mathbb{Z}^N,\ 0 \le l \le L-2\}. \tag{36}$$

The directional refinement and coarsening operators, $\mathcal{R}_j^{a,b}$ and $\mathcal{C}_j^{a,b}$, respectively, refine and coarsen the j-th direction of a patch by the ratio defined between refinement levels a and b, that is,

$$\mathcal{R}_j^{a,b}\mathcal{P} = \text{repl}_j(R_j^{a,b} i_j, R_j^{a,b}(i_j+1) - 1)\mathcal{P}, \tag{37a}$$
$$\mathcal{C}_j^{a,b}\mathcal{P} = \text{repl}_j(C_j^{a,b} i_j, \lfloor C_j^{a,b}(i_j+1) - 1\rfloor)\mathcal{P}, \tag{37b}$$

where $R_j^{a,b} = \prod_{l=0}^{b-1}(\mathbf{r}_l^{l+1})_j / \prod_{l=0}^{a-1}(\mathbf{r}_l^{l+1})_j$ and $C_j^{a,b} = 1/R_j^{a,b}$.

Again referring to Figure 1, we wish to compute the reduction on the composite grid $\mathcal{G}_C$, but in fact we have the hierarchy grid $\mathcal{G}_H$. The standard technique for computing a reduction across a hierarchy without double counting is to use a mask to zero out the contributions from coarse grid regions that are overlapped by fine grid. Let P_l be the number of patches in level $\mathcal{L}_l$; let $\mathcal{I}_j^{a,b}$ be the the interpolation operator in direction j that refines the local data from the resolution of level a to that of level b; and let $\mu_{\mathbf{i}}^p$ be the masking operator on patch $\mathcal{P}_p$ that sets the data in cell $\mathbf{i}$ to zero if the cell is covered by a patch at a finer level. We further assume, without loss of generality, that the phase-space integer vector $\mathbf{i}$ is ordered in such a way so that elements 0 through $N-1$ correspond to configuration-space indices, and elements N through $N+M-1$ correspond to velocity space indices.

The reduction operation to construct the charge density is

$$\begin{aligned}
\rho_{\mathbf{i}_c} &= 1 - V_v \sum_{d=N}^{N+M-1} \sum_{j_d=0}^{J_d-1} f_{\mathbf{i}} = 1 - V_v \sum_{d=N}^{N+M-1} \sum_{l=0}^{L-1} \sum_{p=0}^{P_l-1} \sum_{j_d=j_{d,\mathrm{lo}}^p}^{j_{d,\mathrm{hi}}^p} \mu_{\mathbf{i}}^p \prod_{d'=0}^{N-1} \mathcal{I}_{j_{d'}}^{l,L-1} f_{\mathbf{i}}^l, \\
&= 1 - V_v \sum_{l=0}^{L-1} \sum_{p=0}^{P_l-1} \sum_{d=N}^{N+M-1} \sum_{j_d=j_{d,\mathrm{lo}}^p}^{j_{d,\mathrm{hi}}^p} \mu_{\mathbf{i}}^p \prod_{d'=0}^{N-1} \mathcal{I}_{j_{d'}}^{l,L-1} f_{\mathbf{i}}^l,
\end{aligned} \tag{38}$$

where $\mathbf{i}_c = \prod_{d=N}^{N+M-1} \mathrm{restr}_d\, \mathbf{i} \in \mathbb{Z}^N$ is the configuration space index and where $V_v = \prod_{d=N}^{N+M-1} h_d$. In words, the distribution function on each patch p is first refined up to the finest level in the configuration space directions, then masked, and then accumulated. Note that the mask operator and the interpolation operator do not commute; after summing over the velocity -space indices, j_d, with a mask, the resulting partial sums would in general have discontinuities at mask boundaries. Thus, using a masking procedure, one must reconstruct in the higher-dimensional space, which is more expensive that reconstructing in the lower-dimensional space.

To facilitate the inter-dimensional communication, two intermediate data structures are used. The *partial reduction level* is a level of *overlapping* patches at the finest resolution in configuration space, where the patches are projections of every patch in the phase-space hierarchy:

$$\mathcal{L}_{\mathrm{partial}} = \left\{ \prod_{d'=0}^{N-1} \mathcal{R}_{d'}^{l,L-1} \prod_{d=N}^{N+m-1} \mathrm{restr}_d\, \mathcal{P}, \quad \forall\ \mathcal{P} \in \mathcal{G}_H \right\}. \tag{39}$$

The *total reduction level*, $\mathcal{L}_{\mathrm{total}}$, *disjoint* covering of patches of the configuration space domain that, aside from the resolution, is independent of all hierarchies. These structures are created given a phase-space hierarchy $\mathcal{G}_H$ and configuration space-hierarchy $\mathcal{G}_H^c$ and exist until regridding occurs.

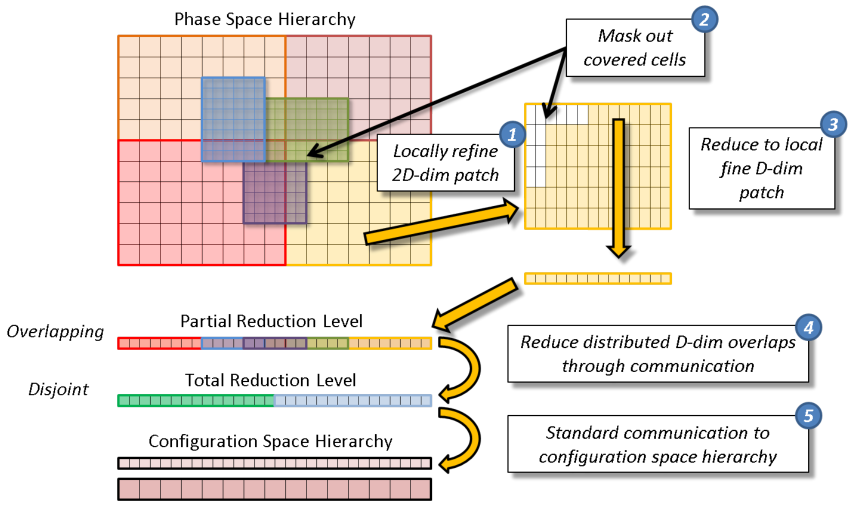
Phase Space Hierarchy
Mask out covered cells
2
Locally refine 2D-dim patch
1
Reduce to local fine D-dim patch
3
Overlapping
Partial Reduction Level
Reduce distributed D-dim overlaps through communication
4
Disjoint
Total Reduction Level
Configuration Space Hierarchy
Standard communication to configuration space hierarchy
5

Figure 5: Graphical depiction of the Mask Reduction Algorithm.

Algorithm 6 Mask Reduction

```
for all levels $\mathcal{L} \in \mathcal{G}_H$ do
    for all patches $\mathcal{P} \in \mathcal{L}$ do
        Refine data in non-reduction directions
        Mask out covered cells
        Find patch $\mathcal{P}_{\rm p}(\mathcal{P}) \in \mathcal{L}_{\rm partial}$ corresponding to patch $\mathcal{P}$
        Sum reduce to partial sum patch $\mathcal{P}_{\rm p}$
    end for
end for
for all patches $\mathcal{P}_{\rm t} \in \mathcal{L}_{\rm total}$ do
    Accumulate data from co-located patches $\mathcal{P}_{\rm p}$
end for
for all levels $\mathcal{L} \in \mathcal{G}^{\rm c}_H$ do
    for all patches $\mathcal{P} \in \mathcal{L}$ do
        Copy data from co-located patches $\mathcal{P}_{\rm t}$
    end for
end for
```

In Figure 5, a diagram of the Mask Reduction Algorithm, Algorithm 6, is presented. The algorithm proceeds as follows. For each patch in the phase space hierarchy, the data on the patch is first refined in those directions that will remain after the reduction. The covered regions are then masked out; this is accomplished by using masks that have been precomputed using a modified communication algorithm[2] and stored in the phase space hierarchy. A one-to-one mapping is used to obtain the configuration-space partial-sum patch $\mathcal{P}_p$ from the pre-computed partial summation level $\mathcal{L}_{\rm partial}$ that corresponds to the current phase space patch $\mathcal{P}$. The summation is then executed, and the results are placed in the configuration-space partial summation patch $\mathcal{P}_p$. Once all patches in the phase space hierarchy $\mathcal{G}_H$ have been reduced, a communication from the partial summation level $\mathcal{L}_{\rm partial}$ to the total reduction level $\mathcal{L}_{\rm total}$ is executed using an accumulation operation.[3] The

[2]The mask variable is set to unity everywhere. Then a standard communication algorithm from fine to coarse levels is executed on the mask variable in the phase space hierarchy, but instead of copying fine data to coarse cells where overlap occurs, zeros are copied into the coarse cells.

[3]Instead of copying values from each source patch to each destination patch, values are added from each source

total reduction level at this point contains the total reduction on a set of disjoint patches at the finest resolution. A standard communication operation from the total reduction level $\mathcal{L}_{\text{total}}$ to the configuration space hierarchy $\mathcal{G}^c_H$ completes the data transfer.

We note that the total reduction level is not necessary. One could communicate directly between the partial summation level and the configuration space hierarchy using an accumulation operation. However, for clarity and ease of implementation, we favored the use of an intermediate total reduction level.

While the Mask Reduction Algorithm is simple to implement, one might suspect that it is not as efficient as it could be because interpolation is done in phase space to the original data. Instead, consider execution of the reduction on the composite grid, $\mathcal{G}_C$. Let P_C be the number of patches in $\mathcal{G}_C$. In a single dimension:

$$\begin{aligned}
\rho_{\mathbf{i}_c} &= 1 - V_v \sum_{d=N}^{N+M-1} \sum_{j_d=0}^{J_d-1} f_{\mathbf{i}} = 1 - V_v \sum_{p=0}^{P_C-1} \sum_{d=N}^{N+M-1} \sum_{j_d=j^p_{d,\text{lo}}}^{j^p_{d,\text{hi}}} \prod_{d'=0}^{N-1} \mathcal{I}^{l(p),L-1}_{j_{d'}} f^p_{\mathbf{i}}, \\
&= 1 - V_v \sum_{p=0}^{P_C-1} \prod_{d'=0}^{N-1} \mathcal{I}^{l(p),L-1}_{j_{d'}} \sum_{d=N}^{N+M-1} \sum_{j_d=j^p_{d,\text{lo}}}^{j^p_{d,\text{hi}}} f^p_{\mathbf{i}}, \\
&= 1 - \sum_{p=0}^{P_C-1} \prod_{d'=0}^{N-1} \mathcal{I}^{l(p),L-1}_{j_{d'}} \rho^P_{\mathbf{i}_c},
\end{aligned} \tag{40}$$

where, since the composite grid has no levels, the level index l is a function of the patch index p and is merely a label indicating the refinement relative to the coarsest patches. Note the savings of the last step; instead of applying the potentially costly prolongation operator at every grid level j, it is instead applied to the lower-dimensional partial sums on each patch. To achieve this simplification, however, an efficient algorithm is needed to construct the composite grid.

Algorithm 7 Subdivision of patch into composite grid sub-patches

```
procedure ComputeSubPatches($\mathcal{P}_{\text{in}}, \mathcal{L}_{\text{in}}, \mathcal{G}_H$)
    for all levels $\mathcal{L} \in \mathcal{G}_H : \mathcal{L} > \mathcal{L}_{\text{in}}$ do
        $\mathcal{S} \leftarrow \mathcal{P}_{\text{in}}$
        for all patches $\mathcal{P} \in \mathcal{L} : \mathcal{P} \cap \mathcal{P}_{\text{in}} \neq \emptyset$ do
            $\mathcal{P}_{\text{extend}} \leftarrow \prod_{d=N}^{N+M-1} \text{repl}_d \left((\mathbf{i}^{\text{in}}_{\text{lo}})_d, (\mathbf{i}^{\text{in}}_{\text{hi}})_d\right) \mathcal{P}$
            for all patches $\mathcal{P}_s \in \mathcal{S}$ do
                $\mathcal{S} \leftarrow \mathcal{S} - \mathcal{P}_s + \mathcal{P}_s \cap \mathcal{P}_{\text{extend}} + \mathcal{P}_s \backslash (\mathcal{P}_s \cap \mathcal{P}_{\text{extend}})$
                $\mathcal{S} \leftarrow \mathcal{S} - \mathcal{P}$
            end for
        end for
    end for
    return $\mathcal{S}$
end procedure
```

Such a procedure based on basic box calculus operations is presented in Algorithm 7. Given a patch $\mathcal{P}_{\text{in}}$ and the hierarchy in which it resides, a set of sub-patches $\mathcal{S}$ is to be constructed. Initially, the set of sub-patches is just the original patch $\mathcal{P}_{\text{in}}$. All patches $\mathcal{P}$ from finer levels that overlap the patch are found. In the SAMRAI library, these relationships are already known and can be obtained directly without searching. Each overlapping patch, then, is extended in the reduction directions, for example, that is, its lower and upper indices in the reduction directions are replaced by the lower

patch to each destination patch.

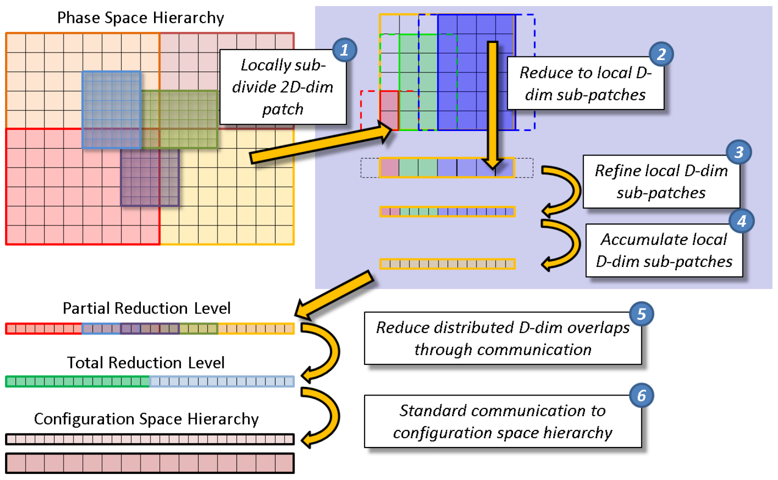
Phase Space Hierarchy
1
Locally sub-divide 2D-dim patch
2
Reduce to local D-dim sub-patches
3
Refine local D-dim sub-patches
4
Accumulate local D-dim sub-patches
Partial Reduction Level
Total Reduction Level
Configuration Space Hierarchy
5
Reduce distributed D-dim overlaps through communication
6
Standard communication to configuration space hierarchy

Figure 6: Graphical depiction of the Sub-Patch Reduction Algorithm. It is those steps in the light purple box that differ from the Mask Reduction Algorithm.

and upper limits of the input patch $\mathcal{P}_{\mathrm{in}}$:

$$\mathcal{P}_{\mathrm{extend}} = \prod_{d=N}^{N+M-1} \mathrm{repl}_d \left((\mathbf{i}_{\mathrm{lo}}^{\mathrm{in}})_d, (\mathbf{i}_{\mathrm{hi}}^{\mathrm{in}})_d \right) \mathcal{P}. \tag{41}$$

The extended overlapping patch $\mathcal{P}_{\mathrm{extend}}$ is then intersected with each patch $\mathcal{P}_s \in \mathcal{S}$, and both the intersections and the complements replace the patch $\mathcal{P}_s$ in the set. The original overlap patch $\mathcal{P}$ is then subtracted from the set $\mathcal{S}$. The extension ensures that sub-patches of the greatest extent in the reduction directions can be formed and that subsequent removal of overlap patches results in rectangular sub-domains.

Algorithm 8 Sub-Patch Reduction

```
for all levels $\mathcal{L} \in \mathcal{G}_H$ do
    for all patches $\mathcal{P} \in \mathcal{L}$ do
        Find patch $\mathcal{P}_{\mathrm{p}}(\mathcal{P}) \in \mathcal{L}_{\mathrm{partial}}$ corresponding to patch $\mathcal{P}$
        $\mathcal{S} \leftarrow \textsc{ComputeSubPatches}(\mathcal{P}, \mathcal{L}, \mathcal{G}_H)$
        for all sub-patches $\mathcal{P}_s \in \mathcal{S}$ do
            Grow ghost cells in non-reduction directions
            Sum reduce data, including ghost cells, to temporary patch $\mathcal{P}_{\mathrm{tmp}}$
            Refine data on $\mathcal{P}_{\mathrm{tmp}}$ in non-reduction directions
            Copy data on interior of $\mathcal{P}_{\mathrm{tmp}}$ to partial sum patch $\mathcal{P}_{\mathrm{p}}$
        end for
    end for
end for
for all patches $\mathcal{P}_{\mathrm{t}} \in \mathcal{L}_{\mathrm{total}}$ do
    Accumulate data from co-located patches $\mathcal{P}_{\mathrm{p}}$
end for
for all levels $\mathcal{L} \in \mathcal{G}_H^{\mathrm{c}}$ do
    for all patches $\mathcal{P} \in \mathcal{L}$ do
        Copy from co-located patches $\mathcal{P}_{\mathrm{t}}$
    end for
end for
```

Using this sub-patch construction procedure, the more efficient Sub-Patch Reduction Algorithm

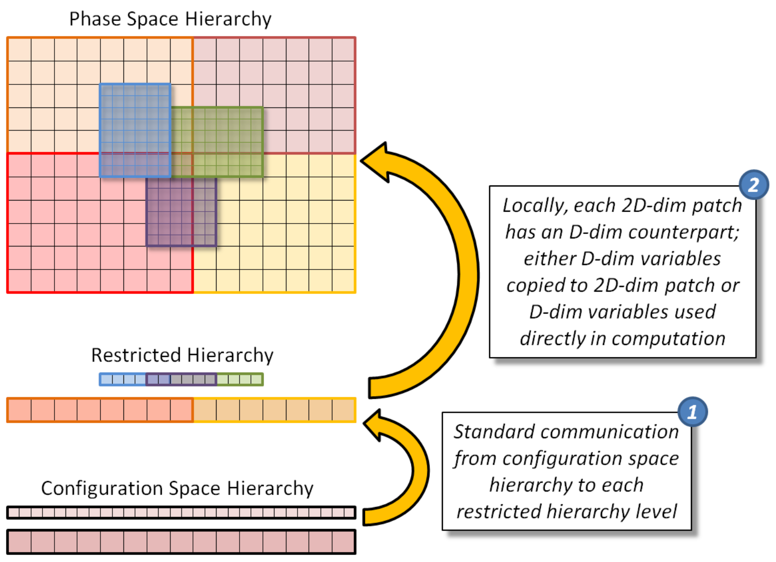
Phase Space Hierarchy
Locally, each 2D-dim patch has an D-dim counterpart; either D-dim variables copied to 2D-dim patch or D-dim variables used directly in computation
2
Restricted Hierarchy
Standard communication from configuration space hierarchy to each restricted hierarchy level
1
Configuration Space Hierarchy

Figure 7: Injection

presented in Algorithm 8 and depicted in Figure 6 can be used. As before, loops are performed over all patches, but now, for a given patch, the set of sub-patches are identified. Each of these sub-patches is first grown in the non-reduction directions by the number of ghost cells necessary for any subsequent prolongation operations. The data on each sub-patch, including the ghost cells, is sum reduced to a temporary configuration-space patch of the same resolution. This partial sum data is then refined, and the result is copied into the corresponding partial sum patch from the partial sum hierarchy. Once all contributions from all sub-patches are obtained, the reduction algorithm proceeds as before.

Finally, we note that neither reduction algorithm assumes that either the phase-space or configuration-space hierarchies have a special structure. Because intermediate data structures are used along with standard communication algorithms, arbitrary meshes could be used in either dimensions. In addition, these algorithms are applicable in parallel and for arbitrary dimension.

3.2.2. Injection algorithm

The process of injection from lower to higher dimensions is much simpler. The data, E_i, for instance, is the same for all phase space locations at index i, that is, $\hat{E}_{ij} = E_i$. Nevertheless, to facilitate the data transfer between hierarchies of different dimensions, it is convenient to first construct an intermediate configuration-space *restricted hierarchy*, $\mathcal{G}^r_H = \{\mathcal{L}^r_l, 0 \le l \le L-1\}$, where

$$\mathcal{L}^r = \left\{ \prod_{d=N}^{N+M-1} \mathrm{restr}_d\, \mathcal{P}, \quad \forall \mathcal{P} \in \mathcal{G}_H \right\}, \tag{42}$$

that is, it is composed of lower-dimensional restrictions of all of the patches in the phase space hierarchy $\mathcal{G}_H$.

Algorithm 9 Injection

```
for all Levels $\mathcal{L} \in \mathcal{G}_H^r$ do
    for all Patches $p \in \mathcal{L}$ do
        copy from co-located patches $p_c \in \mathcal{G}_H^c$
    end for
end for
for all Levels $\mathcal{L} \in \mathcal{G}_H$ do
    for all Patches $p \in \mathcal{L}$ do
        find patch $p_{\mathrm{r}}(p) \in \mathcal{G}_H^r$ corresponding to patch $p$
        either copy from $p_{\mathrm{r}}$ into $p$ or use directly
    end for
end for
```

For completeness, the injection transfer algorithm is depicted in Figure 7 and presented in Algorithm 9. Standard communication copiers are used to fill the restricted hierarchy $\mathcal{G}_H^r$ from the configuration space hierarchy $\mathcal{G}_H^c$. A one-to-one mapping exists from every restricted hierarchy patch to the corresponding phase space hierarchy patch. The restricted hierarchy data can then be injected into phase space data, *e.g.*, $\hat{E}_{ij} \leftarrow E_i$. This wastes storage with many repeated values, so in the VALHALLA code, we directly access the restricted hierarchy data when needed.

3.3. Regridding for multiple AMR hierarcies

Working with multiple hierarchies introduces regridding challenges for AMR algorithms. With a single hierarchy, the user typically defines a regrid frequency. At user-defined times, refinement criteria are used to identify cells in need of refinement (coarsening), new levels of rectangular patches are formed containing these flagged cells and are populated, and these new levels replace old levels in the hierarchy. With multiple hierarchies, one must decide to what degree to constrain the regridding of each hierarchy. Considerations include the facilitation of efficient communication between hierarchies, the cost/benefit of adaptation for each hierarchy, and the degree and nature of dependencies between hierarchies (*e.g.*, can the hierarchies refine independently, and if not, are the dependencies one-way or more complicated). In the case of Vlasov simulation, coordination is most critical between the configuration space hierarchy and the phase space hierarchies, where a variety of intermediate data structures are required to execute inter-dimensional data transfer.

For the purposes of demonstrating proof-of-principle, we made several simplifying choices. For 1D+1V Vlasov-Poisson, the electrostatic problem is solved in 1D. When restricted down to 1D, mesh refinement of features in 2D, such as particle trapping regions, will typically lead to mesh refinement almost everywhere; hence, there is little advantage to mesh refinement in configuration space. Furthermore, the cost of the higher dimensional solve by far dominates the total cost, so there should be little advantage to using mesh refinement to speed-up the lower-dimensional solve. We therefore elected to require the configuration space mesh to be uniform at the finest level of refinement of the phase space hierarchy. While the mesh was decomposed into patches, we did not distribute the configuration space hierarchy across multiple processors. For higher-dimensional problems, such as 2D+2V Vlasov-Poisson, one may benefit from distributing the configuration-space hierarchy. However, such a distribution will be over a much smaller number of processors than the phase-space hierarchy simply because there is so much less data and work to distribute in lower dimensions.[4] When the phase-space hierarchy was regridded, the configuration-space hierarchy was only regridded if a new level of refinement was added (removed) in phase space. This scheme had a secondary advantage of simplifying the Poisson solver; solves were executed on the uniform, finest mesh level in configuration space and then averaged down to coarser levels, thereby avoiding the

[4]For a uniform-grid 2D+2V Vlasov-Poisson code, we have seen in practice that the Poisson solve benefits from distributed parallelism only when the problem size has grown such that the Vlasov solve occurs across several thousand processors.

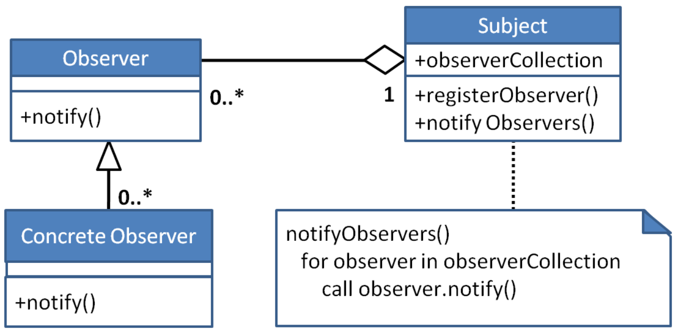

Figure 8: Unified Modeling Language depiction of the Observer Design Pattern used to create notifying hierarchies. In our case, the reduction and injection algorithms for configuration-space are `HierarchyObservers`. These observers register themselves with the phase space `NotifyingHierarchy` to receive notices about regridding.

need for a Fast Adaptive Composite (FAC) iterative algorithm [49]. We note that these are merely choices and not requirements; the algorithms for inter-hierarchy data transfers defined in Section 3.2 support more general configuration and phase-space hierarchies.

3.3.1. Managing inter-hierarchy coordination

From the descriptions of the reduction and injection transfer algorithms in Section 3.2, it is clear that the intermediate data structures, such as the partial sum level or restricted patch hierarchy, are dependent on the phase and configuration space hierarchies. When regridding of any of the primary hierarchies occurs, the intermediate data structures must be rebuilt in order to maintain consistency. To facilitate this, we made use of the Observer Design Pattern [50] as depicted in Figure 8. The SAMRAI concept of `PatchHierarchy` was generalized to allow other objects, such as the `ReductionAlgorithm` and `InjectionAlgorithm`, to subscribe to the phase space hierarchy in order to receive messages indicating that the phase space hierarchy had regridded. Reconstruction of the intermediate data structures is deferred until a subsequent reduction or injection operation is attempted and a message from an observed hierarchy is found.

3.3.2. Mesh Refinement Criteria

Finally, selection of mesh refinement criteria can be critical in obtaining optimum AMR performance. For our purposes here, we chose to apply common heuristic refinement criteria to the phase space distribution function. Specifically, we tag cells when

$$\delta_1 f_{\mathbf{i}} + \delta_2 f_{\mathbf{i}} > \text{tol} \tag{43}$$

where

$$\delta_1 f_{\mathbf{i}} = \left[\frac{1}{2} \sum_{d=1}^{D} \Delta x_d (f_{\mathbf{i}+\mathbf{e}^d} - f_{\mathbf{i}-\mathbf{e}^d})^2 \right]^{\frac{1}{2}} \quad \text{and} \quad \delta_2 f_{\mathbf{i}} = \frac{1}{2} \sum_{d=1}^{D} \Delta x_d^2 |f_{\mathbf{i}+\mathbf{e}^d} - 2 f_{\mathbf{i}} + f_{\mathbf{i}-\mathbf{e}^d}| \tag{44}$$

estimate the first two truncation error terms. We do not claim that this is the optimal choice; it is merely sufficient to demonstrate our algorithms. Other error indicators could be used, including indicators based on physical principles, such as local estimates of the location of the trapped-passing boundary. The choice of optimal refinement criteria is intimately related to problem-specific quantities of interest, so we leave this topic for future work.

4. Numerical Results

We present results from a Valhalla simulation of the bump-on-tail instability [51, §9.4] as a basic proof-of-principle of the block-structured AMR approach for Vlasov-Poisson simulation. We

Parameter	AMR1	AMR2	AMR3
`largest_patch_size`			
`level_0`	(32,32)	(16,32)	(16,32)
`level_1`	(64,64)	(32,128)	(32,128)
`level_2`	(64,64)	(128,256)	(128,256)
`smallest_patch_size`	(4,4)	(8,8)	(8,8)
`regrid_interval`	2	4	8
`tag_buffer`	(1,1)	(4,4)	(8,8)

Table 1: AMR parameters used to define the three test configurations The parameters `largest_patch_size` and `smallest_patch_size` control the largest and smallest allowable patch sizes level-by-level; if unspecified, the finest specified value is applied to all subsequent levels. The parameter `regrid_interval` is the frequency, in time steps, at which regridding occurs. Finally, `tag_buffer` is the number of buffers cells to add around a region tagged for refinement to facilitate less frequent regridding.

used the same problem specified in our previous discretization work [39]. The initial distribution function was given by

$$f = f_b(v)\left(1 + 0.04\cos(0.3x)\right), \tag{45}$$

with

$$f_b(v) = \frac{0.9}{\sqrt{2\pi}}\exp\left(-\frac{v^2}{2}\right) + \frac{0.2}{\sqrt{2\pi}}\exp\left(-4(v-4.5)^2\right). \tag{46}$$

The (x, v) domain was $[-10\pi/3, 10\pi/3] \times [-8, 10]$ and was periodic in the x-direction. We initialized the solution with a coarse grid of $N_x \times N_v = 16 \times 32$ and with an initial refinement in the box $[(0, 8), (15, 24)]$. This initial mesh configuration allowed for larger time steps, since the cells along the maximum velocity boundary have a larger aspect ratio. The initial time step was $\Delta t_0 = 0.01$, and this was allowed to adjust to 50% of the local stability condition based on the linear stability of the fourth-order scheme (See [38]). Time steps could increase no more than 10% from their previous value, but could decrease by any amount. The grid refinement criteria tolerance was tol $= 0.01$, and grid refinement ratios of $\mathbf{r}_0^1 = [2, 4]$, $\mathbf{r}_1^2 = [4, 2]$, and $\mathbf{r}_2^3 = [2, 2]$ were used. Up to four levels of AMR mesh were allowed. To isolate the AMR performance issues, we consider the serial performance on a single node of the LLNL 64-bit AMD Linux cluster `hera`.

AMR performance is very problem-dependent. When small regions of refinement are required, in particular, when there are lower-dimensional features in the solution, AMR is generally a net win. However, there is overhead associated with AMR for which sufficient problem size reduction is necessary to achieve a net gain in simulation performance. Performance is also highly dependent on the choice of parameters, such as regrid frequency and refinement tolerances, so the results presented here are meant to demonstrate that our Vlasov-AMR procedure works and can show savings. Whether or not AMR is useful in other specific cases and optimal choices for AMR parameters and regridding criteria are very important issues.

To help elucidate the AMR performance, we considered three AMR parameter configurations, as shown in Table 1. The AMR1 case represents an attempt to minimize the number of refined cells by using smaller patches and more frequent regridding. The AMR2 and AMR3 cases allow for larger patches an less frequent regridding. Thus, these three cases can give some sense of the trade-offs between reducing the amount of mesh (AMR memory reduction) and reducing the run time (AMR speed-up). All cases were run using the *Sub-Patch Reduction* algorithm with unlimited fifth-order reconstruction unless otherwise noted.

Figure 9 shows computed approximations of the phase-space distribution function at $t = 22.5$ for case AMR1. As expected, we see a concentration of the mesh only in regions of most rapid variation in the solution, and conversely, we see mesh coarsening in the the trapping region. At this point in the calculation, the total number of cells is 40784, compared to 131072 cells in the equivalent fine grid – a reduction of approximately 69%. In Figure 10, we show the time history of the number of AMR cells plotted against the instantaneous equivalent fine grid for each AMR

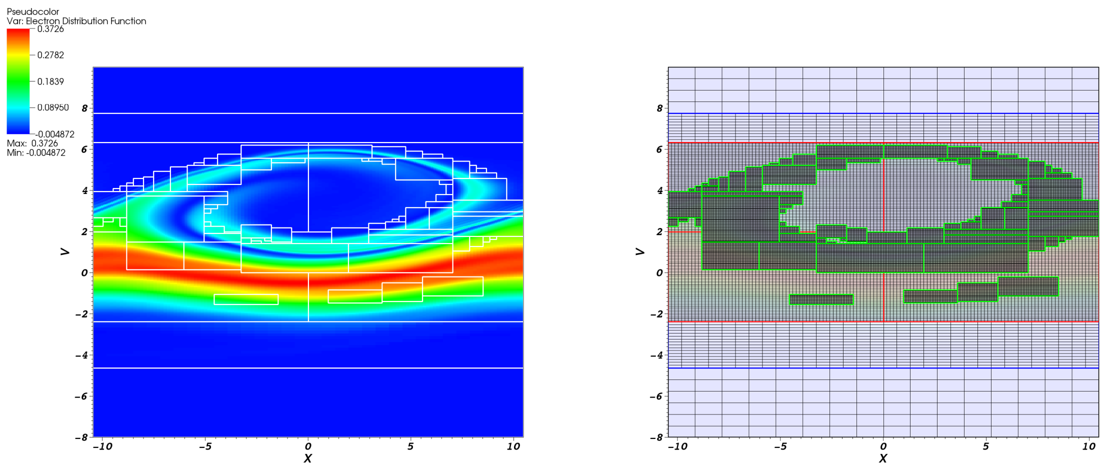

Figure 9: Example result for the bump-on-tail problem at time $t = 22.5$ for the AMR1 case. On the left, the distribution function is shown. Boxes outlined in white denote AMR patches. On the right, the corresponding four-level AMR mesh is shown. The mesh adapts to resolve the particle trapping region as it forms. Note that the minimum distribution function is small but negative; no positivity enforcement schemes were used in this calculation.

parameter configuration. We see that once adaptivity starts, we can achieve an average reduction of between forty and sixty percent, depending on the AMR parameters. As expected, the AMR1 case uses the least number of cells, and the AMR3 case, because of its increased patch size and tagging buffer, uses the most cells. Considering Figure 9, we see that a lot of the mesh reduction comes in the velocity (vertical) dimension, and this is expected for each additional velocity dimension in higher dimensions. In 1D+1V, there is little localization in configuration space. However, in 2D+2V, there is also the opportunity for spatial localization of the features, which would result in even more mesh reduction.

In Figure 11, we present the time history of the maximum of the electric field for the bump-on-tail problem for three different resolutions. This metric is a fairly sensitive measure of numerical fidelity. In addition to the three AMR parameter cases, we plot the results from three uniform-grid cases: 64×128, 128×256, and 256×512. We use the finest of these as a “reference” solution to plot the discrepancy of the electric field maximum. For the AMR calculations, the finest local resolution is equivalent to the 128×256 uniform mesh. At early times, when the solution has little structure, all of the solutions agree well. The small up/down differences in the AMR results before $t = 10$ are due to the discrete temporal resolution (the AMR cases use larger time steps) of the first two valleys of the maximum electric field. Around $t = 25$, one begins to see significant differences in the coarsest solution, since it cannot resolve as well the features being generated in the particle trapping region. We can conclude from these results that the 64×128 resolution was insufficient to accurately track the maximum electric field over this time interval; thus the increased resolution of the AMR is necessary.

By about $t = 50$, one sees a growing discrepancy between all of the AMR cases and the equivalent uniform mesh of 128×256; over the interval considered, the discrepancy is roughly twice as large at its maximum. One explanation for this could be the accumulation of error over longer integration times. Another likely explanation is that we are not capturing all of the relevant details with the refined mesh because we are using a simple heuristic gradient detection algorithm; more problem-specific refinement criteria may perform better. Nevertheless, the AMR results do track the equivalent uniform mesh results well. Compared to the finest uniform grid results, the phase of the AMR results is relatively good, but the amplitude is being under-predicted by an increasing amount over

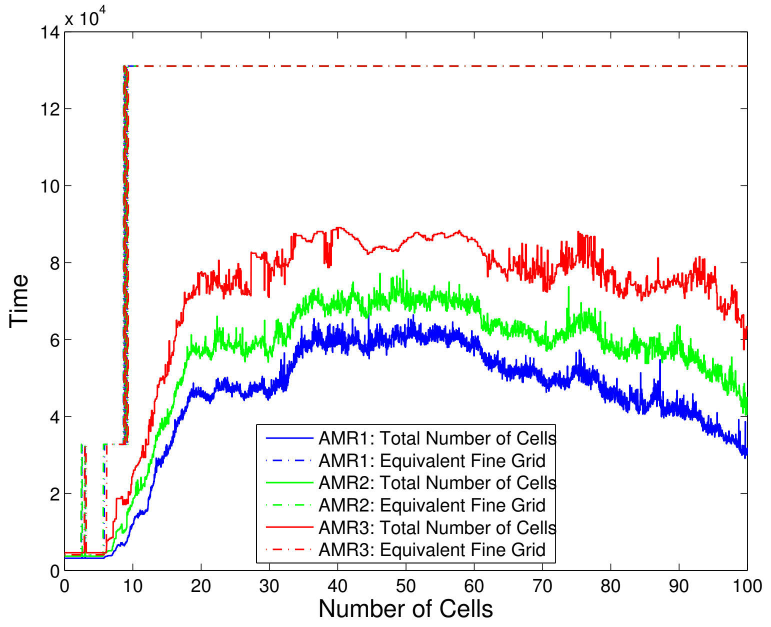

Figure 10: Time history of the number of cells for the bump-on-tail simulation for the three AMR parameter configurations. The dashed curves are the number of cells in an equivalent uniform grid based on the current maximum refinement level, while the solid curves are the actual number of cells in the AMR hierarchy. Note that around $t = 3, 6$ and 9, there is some intermittency in all cases, as the adaptivity adds and removes a small number of patches at the next finer level.

Case	**Time 1 (s)**	**Time 2 (s)**	**Time 3 (s)**	**Avg Time (s)**	**Speed-up**
Uniform	1521	1520	1519	1520	1.00
AMR1	2622	2615	2632	2623	0.58
AMR2	1355	1346	1336	1346	1.13
AMR3	960	961	960	960	1.58

Table 2: AMR speed-up on the `hera` cluster for the three AMR parameter cases. The average of three results for each case are compared to the average run time for three instances of the same problem solved using the equivalent uniform mesh.

time; there will, of course, be slightly more dissipation in the coarser results when features appear that cannot be adequately resolved. These results show that AMR can provide effectively equivalent results as a uniform mesh. Of course, one must consider the quantities of interest for the calculation, and suitable choices of AMR parameter and refinement criteria need to be selected.

In addition to mesh reduction, the potential for decreased time-to-solution by using AMR is also of interest. As indicated earlier, AMR should have the benefit that the equations are integrated on fewer cells and that larger time steps can be taken. However, traditional AMR incurs additional overhead from the regridding and the data transfers (communication, interpolation, and averaging) between patches on different levels. The Vlasov-Poisson system has additional overhead due to the reduction and injection operations between different dimensions.

In Table 2, we provide run times on the `hera` cluster for the three AMR parameter cases in comparison to the run time for the equivalent uniform mesh. The AMR1 case, with refinement every other step, causes significant slow-down of the code; however, in the other two cases, the time to solution is reduced. As expected, when the regridding frequency is reduced, the speed-up increases. We note that we have erred conservatively in favor of the uniform mesh solution; running serially and with a single patch, it incurs no inter-patch communication costs.

The reasons for the slow-down are obvious when looking at the cost of certain key operations, as

Case	Exclusive Time (s)	Total Time (s)
`MultiStageIntegrator::computeRHS()`		
`Uniform`	830.6 (55%)	830.7 (55%)
`AMR1`	169.0 (6.4%)	169.1 (6.4%)
`AMR2`	170.5 (13%)	170.6 (13%)
`AMR3`	236.0 (25%)	236.0 (25%)
`MultiStageIntegrator::evaluateInstantaneousConstraints()`		
`Uniform`	16.40 (1.1%)	115.0 (7.6%)
`AMR1`	68.89 (2.6%)	272.6 (10%)
`AMR2`	19.63 (1.5%)	132.5 (9.9%)
`AMR3`	12.35 (1.3%)	89.76 (9.4%)
`xfer::RefineSchedule::fillData()`		
`Uniform`	32.25 (2.1%)	94.65 (6.2%)
`AMR1`	21.95 (0.83%)	1564. (59%)
`AMR2`	14.54 (1.1%)	751.9 (56%)
`AMR3`	14.00 (1.5%)	384.3 (40%)
`ReductionAlgorithm::reduce()`		
`Uniform`	0.8683 (0.06%)	76.56 (5.0%)
`AMR1`	7.507 (0.29%)	153.5 (5.8%)
`AMR2`	3.179 (0.24%)	74.75 (5.6%)
`AMR3`	1.846 (0.19%)	43.31 (4.5%)
`MultiStageIntegrator::regridHierarchies()`		
`Uniform`	0.000 (0%)	0.000 (0%)
`AMR1`	72.70 (2.8%)	767.9 (29%)
`AMR2`	17.48 (1.3%)	203.6 (15%)
`AMR3`	6.150 (0.64%)	74.41 (7.8%)
`ConservativeWENORefine::WENO_2D`		
`Uniform`	0.000 (0%)	0.000 (0%)
`AMR1`	639.0 (24%)	720.2 (27%)
`AMR2`	382.2 (29%)	411.1 (31%)
`AMR3`	203.7 (21%)	215.1 (22%)

Table 3: A summary of the costs of six key routines for the three AMR cases and the equivalent uniform mesh. Exclusive time is the time strictly spent in a routine. Total time is the time spent in a routine and all subroutines called from that routine.

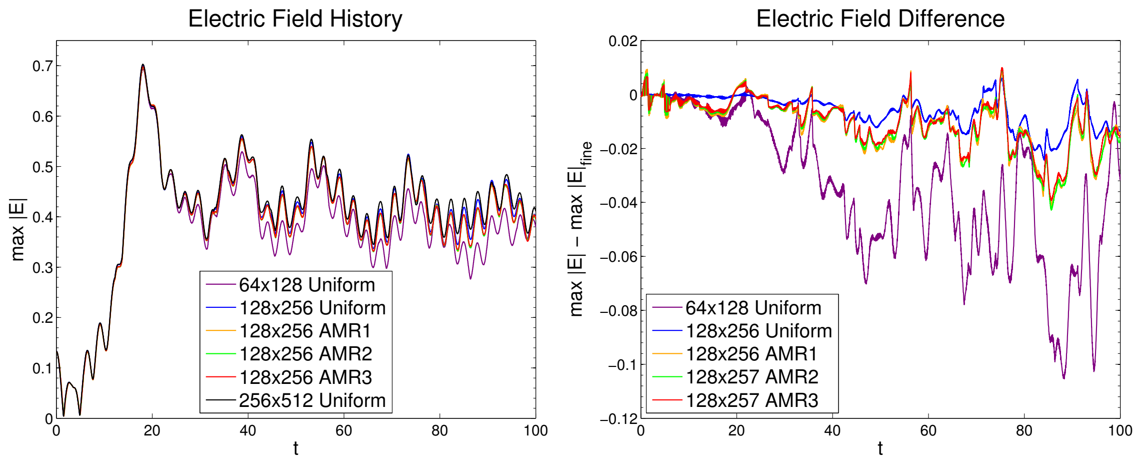

Figure 11: Time history of the maximum electric field for bump-on-tail calculations at several resolutions. Results are from the three AMR configurations in which the finest resolution is equivalent to a 128×256 uniform mesh as well as reference uniform-grid calculations on meshes of 64×128, 128×256, and 256×512. The left plot shows the maximum of the electric field. The right plot is the same data, plotted as the difference from the results from the fine 256×512 mesh.

shown in Table 3. Note that more than half the time of the uniform grid calculation is the routine `computeRHS()`. In contrast, the amount of time in this routine is significantly reduced for all AMR calculations, as one would expect, since this routine will scale with the number of cells. That the AMR time in `computeRHS()` is no more than 24% of the total time suggests that the AMR cases are spending a great deal of time in AMR overhead.

One obvious source of overhead is regridding, the cost for which is accounted in `regridHierarchies()`. As expected, we see that the AMR1 case spends the most time in regridding while the AMR3 case spends the least. The absolute total time spent in AMR1 regridding is more than ninety percent of the total time the uniform mesh case spends in `computeRHS()`. However, for the AMR3 case, which regrids on every eighth step, the regridding cost is much more reasonable.

The `fillData()` routine is the top level routine that applies boundary conditions (hence the non-zero cost even for the uniform grid case), fills patch ghost cells on the interior of the domain, and fills new patches from patches at the same (*i.e.*, copy) or coarser (*i.e.*, interpolate) mesh levels. While `fillData()` accounts for only 6% of the uniform mesh calculation total time, it represents 40-60% of the total time for the AMR calculations. One of the routines that constitutes a significant portion of `fillData()` is the finite volume WENO refinement in `WENO_2D()`; it can be seen that with less frequent refinement, less time is spent in this routine. Note that, while the absolute time in WENO interpolation decrease monotonically from AMR1 to AMR3, the relative times peak with AMR2; this is a trade-off between more mesh and less frequent regridding. Furthermore, we note that the cost of limited, high-order interpolation for the intra-hierarchy interpolations necessary in AMR is not a cost specific to the Vlasov-Poisson system; an AMR method based on a higher-order discretization for any PDE system will need to address the efficiency of such interpolations.

The other two routines shared by all four cases, `evaluateInstantaneousConstraints()` and `reduce()`, are provided to show that these operations spend roughly the same percentage of time whether for the uniform mesh or for the AMR cases. Note that `evaluateInstantaneousConstraints()` is marginally more expensive for all AMR cases because this routine includes the Poisson solve, which requires more complicated reductions across the AMR hierarchies. The AMR1 case is more expensive in absolute time because it has more patches to reduce. However, not that the reductions and constraint evaluations for AMR2 and AMR3 are about the same cost as or even cheaper than (in absolute time) the uniform case.

Finally, in Table 4, we present some timings for the routines related to the two reduction algo-

Routine	Exclusive Time (s)	Total Time (s)
`ConservativeHighOrderRefine::WENO_2D()`		
Mask Reduction	742.6 (27%)	824.3 (30%)
Sub-Patch Reduction	642.5 (24%)	727.5 (27%)
`ReductionAlgorithm::reduce()`		
Mask Reduction	27.14 (0.99%)	245.7 (9.0%)
Sub-Patch Reduction	7.505 (0.29%)	157.9 (5.9%)
`ConservativeHighOrderRefine::WENO_1D()`		
Mask Reduction	0.00 (0%)	0.00 (0%)
Sub-Patch Reduction	5.399 (0.20%)	10.88 (0.41%)

Table 4: Comparison of the processing time of the Mask Reduction and Sub-Patch Reduction algorithms. Exclusive time is the time strictly spent in a routine. Total time is the time spent in a routine and all subroutines called from that routine.

rithms described in Section 3.2. Note that these results were computed using WENO interpolation and the AMR1 parameters. For the Mask Reduction, we note that the `WENO_2D` routine is called for intra-hierarchy regridding and communication and inter-hierarchy reduction calls. With the Mask Reduction Algorithm, a great deal of time is spent in the 2D interpolation routine, and the reductions account for 9% of the total run time. For the Sub-Patch Reduction, the `WENO_2D` routine is not called, so its reported time is strictly from intra-hierarchy regridding and communication calls. With the Sub-Patch Reduction Algorithm, the time spent in the `WENO_2D` routine is reduced by roughly 10%, and it is replaced by about 1.3% of additional work in the `WENO_1D` routine. The total cost of the Sub-Patch Reduction Algorithm is 64% of the Mask Reduction Algorithm. The comparative performance is what was anticipated, although, admittedly, for 1D+1V Vlasov-Poisson, the savings are not dramatic. Nevertheless, in higher dimensions, there will be a more significant benefit; for 2D+2V Vlasov-Poisson, the Mask Reduction Algorithm will require four interpolations in each cell in the four-dimensional mesh (scaling like N^4), whereas the Sub-Patch Reduction Algorithm will require only two interpolations in cell in a two-dimensional mesh (scaling like N^2).

5. Conclusions

We have demonstrated the application of block structured adaptive mesh refinement to the 1D+1V Vlasov-Poisson system as implemented in the Valhalla code based on the SAMRAI AMR library. The primary complication comes from a solution state comprised of variables of different dimensions. The considerations and algorithms required to extend standard single-dimensional block structured AMR have been presented. In particular, algorithms for reduction and injection operations that transfer data between mesh hierarchies of different dimensions were explained in detail. In addition, modifications to the basic AMR algorithm due to our use of high-order spatial and temporal discretizations were presented. Preliminary results for a standard Vlasov-Poisson test problem were presented, and these results indicate that there is potential for savings, both in memory and in compute time, for at least some Vlasov problems. The effectiveness for any particular problem will depend intimately on the features of the solution.

There are several obvious directions for future work. Currently, we are working on generalizing the Valhalla code to 2D+2V and higher dimensions. The SAMRAI library is quite general and supports arbitrary dimension. Moving to 4D calculations and beyond opens up several new directions for investigation. When the configuration space problem is in 2D or 3D, there is potential for savings from adaptivity in configuration space. It is straightforward to enable this generalization, but it is unclear if the additional cost of the necessary FAC algorithm and of the AMR overhead will justify the complication, particularly when the solution time will be dominated by operations in the 4D or higher phase space. With larger phase-space problems, an efficient parallel decomposition will be necessary; we have already indicated potential advantages of providing each species a distinct subset of processors, but empirical results are needed. It also will prove beneficial to allow for asynchronous time stepping in the AMR advancement; an example of the necessary modifications to the time advancement algorithm has been shown [46].

Dimensions above three also require additional empirical investigation for AMR efficiency. As indicated earlier, the potential savings from AMR increases geometrically with dimension, but AMR overhead, much of which scales with the number of cells at coarse-fine boundaries, also increases with dimension. Whether the overhead costs in 4D and above negate the savings remains an open issue and must be the subject of future studies.

AMR overhead, even in lower dimensions, still requires further reduction. One clear path is to make use of hybrid parallelism as multicore architectures become more prevalent. Much of the computations contributing to AMR overhead are sequentially executed on a node but are completely independent and thus ideal for task parallelism. Implementations should also be found that further optimize the conservative, high-order intra-hierarchy interpolations.

Finally, the topic of refinement criteria requires further investigation. Heuristic or *a posteriori* error indicators that seek to minimize local error are sufficient but not optimal, depending on the desired results of the calculation. Most quantities of interest exist in configuration space, that is, the macroscopic quantities like temperature and density that can be readily measured in the laboratory. The reductions leading to configuration space quantities integrate out much of the finer details in phase space, which suggests that it may be inefficient to resolve all of the finer phase-space structure. Future investigations should consider (i) the phase space resolution requirements to obtain accurate configuration space quantities of interest and (ii) whether more efficient phase-space refinement criteria can be formulated based on these configuration-space accuracy requirements.

Acknowledgements

The authors would like to thank Dr. Bruce Cohen, Dr. Richard Berger, and Dr. Stephan Brunner for their many helpful comments and suggestions.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number DE-AC52-07NA27344. This work was funded by the Laboratory Directed Research and Development Program at LLNL under project tracking code 08-ERD-031. LLNL-JRNL-515291.

References

[1] John D. Lindl, Peter Amendt, Richard L. Berger, S. Gail Glendinning, Sigfried H. Glenzer, Steven W. Haan, Robert L. Kauffman, Otto L. Landen, and Laurence J. Suter. The physics basis for ignition using indirect drive targets on the NIF. *Phys. Plasmas*, 11(2):339–491, March 2003.

[2] E. A. Frieman and Liu Chen. Nonlinear gyrokinetic equations for low-frequency electromagnetic waves in general plasma equilibria. *Phys. Fluids*, 25(3):502–508, March 1982.

[3] T. S. Hahm. Nonlinear gyrokinetic equations for tokamak microturbulence. *Phys. Fluids*, 31 (9):2670–2673, September 1988.

[4] A. M. Dimits, L. L. LoDestro, and D. H. E. Dubin. Gyroaveraged equations for both gyrokinetic and drift-kinetic regimes. *Phys. Fluids B–Plasma*, 4(1):274–277, January 1992.

[5] W. M. Tang and V. S. Chan. Advances and challenges in computational plasma science. *Plasma Phys. Contr. F.*, 47:R1–R34, 2005.

[6] Erich S. Weibel. Spontaneously growing transverse waves in a plasma due to an anisotropic velocity distribution. *Phys. Rev. Lett.*, 2:83–84, Feb 1959. doi: 10.1103/PhysRevLett.2.83. URL `http://link.aps.org/doi/10.1103/PhysRevLett.2.83`.

[7] R. Z. Sagdeev. Cooperative Phenomena and Shock Waves in Collisionless Plasmas. *Reviews of Plasma Physics*, 4:23, 1966.

[8] D. A. Tidman and N. A. Krall. *Shock Waves in collisionless plasmas.* Wiley-Interscience, New York, 1971.

[9] Phillip Colella. An algorithmic and software framework for applied partial differential equations (APDEC): A DOE SciDAC integrated software infrastructure center, May 2003. `http://davis.lbl.gov/APDEC/old/accelerator/index.html`.

[10] Emmanuel Frénod and Frédérique Watbled. The Vlasov equation with strong magnetic field and oscillating electric field as a model for isotop resonant separation. *Elec. J. Differ. Eq.*, 2002 (6):1–20, 2002.

[11] Magdi Shoucri and Georg Knorr. Numerical integration of the Vlasov equation. *J. Comput. Phys.*, 14(1):84–92, January 1974.

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

[13] Alexander J. Klimas. A method for overcoming the velocity space filamentation problem in collisionless plasma model solutions. *J. Comput. Phys.*, 68(1):202–226, January 1987.

[14] P. Bertrand, A. Ghizzo, T. W. Johnston, M. Shoucri, E. Fijalkow, and M. R. Feix. A nonperiodic Euler-Vlasov code for the numerical simulation of laser-plasma beat wave acceleration and Raman scattering. *Phys. Fluids B–Plasma*, 2(5):1028–1037, May 1990.

[15] M. L. Bégué, A. Ghizzo, and P. Bertrand. Two-dimensional Vlasov simulation of Raman scattering and plasma beatwave acceleration on parallel computers. *J. Comput. Phys.*, 151(2): 458–478, May 1999.

[16] Eric Fijalkow. A numerical solution to the Vlasov equation. *Comput. Phys. Commun.*, 116 (2-3):319–328, February 1999.

[17] Takashi Nakamura and Takashi Yabe. Cubic interpolated propagation scheme for solving the hyper-dimensional Vlasov-Poisson equation in phase space. *Comput. Phys. Commun.*, 120: 122–154, 1999.

[18] Francis Filbet, Eric Sonnendrücker, and Pierre Bertrand. Conservative numerical schemes for the Vlasov equation. *J. Comput. Phys.*, 172:166–187, 2001.

[19] Takashi Nakamura, Ryotara Tanaka, Takashi Yabe, and Kenji Takizawa. Exactly conservative semi-Lagrangian scheme for multi-dimensional hyperbolic equations with directional splitting technique. *J. Comput. Phys.*, 174:171–207, 2001.

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

[21] N. Besse and Eric Sonnendrücker. Semi-Lagrangian schemes for the Vlasov equation on an unstructured mesh of phase space. *J. Comput. Phys.*, 191:341–376, 2003.

[22] M. Brunetti, V. Grandgirard, O. Sauter, J. Vaclavik, and L. Villard. A semi-Lagrangian code for nonlinear global simuations of electrostatic drift-kinetic ITG modes. *Comput. Phys. Commun.*, 163:1–21, 2004.

[23] Michael Gutnic, Matthieu Haefele, I. Paun, and Eric Sonnendrücker. Vlasov simulations on an adaptive phase space mesh. *Comput. Phys. Commun.*, 164:214–219, 2004.

[24] Stephan Brunner and E. Valeo. Simulations of stimulated Raman scattering in single laser hot spots. Technical report, Princton Plasma Physics Laboratory, Princeton, New Jersey, 2005.

[25] Matthieu Haefele, Guillaume Latu, and Michael Gutnic. A parallel Vlasov solver using wavelet based adaptive mesh refinement. In *Proceedings of the 2005 International Conference on Parallel Processing Workshops*. IEEE, 2005.

[26] A. J. Klimas and W. M. Farrell. A splitting algorithm for Vlasov simulation with filamentation filtering. *J. Comput. Phys.*, 110:150–163, 1994.

[27] Tomo-Hiko Watanabe, Hideo Sugamma, and Tetsuya Sato. A nondissipative simulation method for the drift kinetic equation. *J. Phys. Soc. Jpn.*, 70(12):3565–3576, December 2001.

[28] Francis Filbet and Eric Sonnendrücker. Comparison of Eulerian Vlasov solvers. *Comput. Phys. Commun.*, 150:247–266, 2003.

[29] E. Pohn, M. Shoucri, and G. Kamelander. Eulerian Vlasov codes. *Comput. Phys. Commun.*, 166:81–93, 2005.

[30] N. J. Sircombe and T. D. Arber. VALIS: A split-conservative scheme for the relativistic 2D Vlasov-Maxwell system. *J. Comput. Phys.*, 228(13):4773–4788, July 2009.

[31] N. Crouseilles, M. Mehrenberger, and E. Sonnendrücker. Conservative semi-Lagrangian schemes for Vlasov equations. *J. Comput. Phys.*, 229:1927–1953, 2010.

[32] J. W. Banks, R. L. Berger, S. Brunner, B. I. Cohen, and J. A. F. Hittinger. Two-dimensional Vlasov simulation of electron plasma wave trapping, wavefront bowing, self-focusing, and side-loss. *Phys. Plasmas*, 18(5):052102, 2011.

[33] D. J. Strozzi, E. A. Williams, H. A. Rose, D. E. Hinkel, A. B. Langdon, and J. W. Banks. Threshold for electron trapping nonlinearity in langmuir waves. *Physics of Plasmas*, 19(11): 112306, 2012. doi: 10.1063/1.4767644. URL `http://link.aip.org/link/?PHP/19/112306/1`.

[34] Charles K. Birdsall and A. Bruce Langdon. *Plasma Physics via Computer Simulation.* The Adam Hilger Series on Plasma Physics. Adam Hilger, New York, second edition, 1991.

[35] Jean-Luc Vay, P. Colella, P. McCorquodale, B. Van Straalen, A. Friedman, and D. P. Grote. Mesh refinement for particle-in-cell plasma simulations: Applications to and benefits for heavy ion fusion. *Laser Part. Bemas*, 20:569–575, 2002.

[36] Jean-Luc Vay. An extended FDTD scheme for the wave equation: Application to multiscale electromagnetic simulation. *J. Comput. Phys.*, 167:72–98, 2001.

[37] Phillip Colella, Milo R. Dorr, Jeffrey A. F. Hittinger, P. McCorquodale, and Daniel F. Martin. High-order, finite-volume methods on locally-structured grids. In N. V. Pogorelov, E. Audit, P. Colella, and G. P. Zank, editors, *Numerical Modeling of Space Plasma Flows: Astronum 2008*, volume 406 of *Astronomical Society of the Pacific Conference Series*, pages 207–216, San Francisco, 2009. Astronomical Society of the Pacific.

[38] Phillip Colella, Milo R. Dorr, Jeffrey A. F. Hittinger, and Daniel F. Martin. High-order, finite-volume methods in mapped coordinates. *J. Comput. Phys.*, 230(8):2952–2976, 2011.

[39] Jeffrey William Banks and Jeffrey Alan Furst Hittinger. A new class of nonlinear finite-volume methods for Vlasov simulation. *IEEE T. Plasma Sci.*, 38(9):2198–2207, September 2010.

[40] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Technical Report NASA-CR-97-206253, NASA Langley Research Center, November 1997.

[41] Andrew K. Henrick, Tariq D. Aslam, and Joseph M. Powers. Mapped weighted essentially non-oscillatory schemes: Achieving optimal order near critical points. *J. Comput. Phys.*, 207: 542–567, 2005.

[42] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[43] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, 1989.

[44] Richard D. Hornung and R. Kohn, Scott. Managing application complexity in the SAMRAI object-oriented framework. *Concur. Comp. Prac. Ex.*, 14(5):347–368, April 2002. `http://www.llnl.gov/CASC/SAMRAI/`.

[45] M.R. Dorr, F. X. Garaizar, and J. A. F. Hittinger. Simulation of laser plasma filamentation using adaptive mesh refinement. *J. Comput. Phys.*, 177:233–263, 2002. doi: 10.1006/jcph.2001.6985.

[46] Peter McCorquodale and Phillip Colella. A high-order finite-volume method for hyperbolic conservation laws on locally-refined grids. *Comm. App. Math. Comput. Sc.*, 6(1), 2011. doi: 10.2140/camcos.2011.6.1.

[47] Michael Barad and Phillip Colella. A fourth-order accurate local refinement method for Poisson's equation. *J. Comput. Phys.*, 209:1–18, 2007. doi: 10.1016/j.jcp.2005.02.027.

[48] Jaideep Ray, Christopher A. Kennedy, Sophia Lefantzi, and Habib N. Najm. Using high-order methods on adaptively refined block-structured meshes: Derivatives, interpolants, and filters. *SIAM J. Sci. Comput.*, 29(1):139–181, 2007. doi: 10.1137/050647256.

[49] S. F. McCormick and J. Thomas. The fast adaptive composite grid (FAC) method for elliptic equations. *Math. Comput.*, 46(174):493–456, April 1986.

[50] Allen Holub. *Hollub on Patterns: Learning Design Patterns by Looking at Code.* Apress, 2004.

[51] Nicholas A. Krall and Alvin W. Trivelpiece. *Principles of Plasma Physics.* McGraw-Hill, New York, 1973.