Six-dimensional adaptive mesh refinement for Vlasov simulation

Erwan Deriaz*

December 9, 2015

Abstract

This article presents a dynamically adaptive grid numerical scheme for the six-dimensional Vlasov-Poisson equations [29, 37]. The distribution function is represented in a non-uniform dyadic Cartesian grid which allows considerable savings in terms of computational time and memory storage. The proposed scheme combines the Adaptive Mesh Refinement framework [28, 4, 33] and a very high (higher than second) order accuracy of the finite difference method, in six dimensions.

keywords: adaptive mesh refinement, dyadic refinement, hierarchical basis, multiresolution, finite difference method, phase-space simulations, Vlasov-Poisson equations

1 Introduction

The essential idea of Adaptivity in Numerical Analysis lies in the representation of a complex system with a reduced number of elements. It aims at decreasing the memory resources and the computational time necessary to simulate this system. Often it boils down to trading off numerical volume (computer memory, number of elementary operations) against an increase in implementation complexity (data structures, algorithms).

The most common adaptive technic used in Mechanics [13, 24, 28, 31, 32], Physics [7, 19] and Astrophysics [1, 35, 36] is called the Adaptive Mesh Refinement where a Cartesian grid refines locally into finer Cartesian grids. It is often associated with finite volume methods [24, 31, 36]. Among AMR methods we can distinguish between patch-based AMR where the data structure is a collection of grids of varying sizes and accuracies embedded the ones in the others [33, 26] and the fully-threaded tree AMR where the points or cells (or groups of points or cells) are stored in the nodes or in the leaves of a large tree structure [6, 22, 31, 32, 36]. We use this latest method.

*Institut Jean Lamour, CNRS – Université de Lorraine, erwan.deriaz@univ-lorraine.fr

Other adaptive methods rely on the discretisation in functional bases: finite elements [25], wavelets [8] or hierarchical bases [4]. Sparse grids [30] enters this latest class of adaptive technics dealing with high dimensionality thanks to hyperbolic –anisotropic– bases. This scheme allows to represent structures having the same size as the grid when the number of dimensions is large [5]. But it has to be associated to AMR technics to simulate multiscale phenomena.

Passing from the classical two and three dimensional AMR to a more exotic six-dimensional one forces to have an efficient implementation and to adopt simplified numerical methods. Hence we propose to apply a straightforward recipe with the simplest numerical tools: Eulerian discretisation, polynomial interpolation and finite differences for hyperbolic partial differential equations. In order to avoid dissipation, we also enforced the use of high order –at least third order– schemes for the convection.

In the first part we describe the point-value/interpolet-expansion duality and the AMR finite differences discretisation. In the second part we introduce and detail the adaptivity process. And finally, in the numerical experiments, we apply this AMR method to Vlasov-Poisson equations simulations in two, four and six dimensions: collisionless plasma in physics and gravitational systems in astrophysics.

2 Numerical scheme

2.1 Point value discretisation, interpolet expansion

We consider a function $f : \Omega \mapsto \mathbb{R}$, $\mathbf{x} \to f(\mathbf{x})$ with $\Omega = [0,1]^d$ and periodic boundary conditions. In the frame of finite differences, we assume we know f at points of the type $(\mathbf{x}_\lambda)_{\lambda\in\Lambda}$ indexed by $\lambda = (j, k_1, \ldots, k_d)$, $j \geq 0$ and $0 \leq k_i < 2^j \ \forall i \in [1, d]$, and defined by:

$$\mathbf{x}_\lambda = (2^{-j}k_1, \ldots, 2^{-j}k_d).$$

We set $\Omega_\Lambda = \{\mathbf{x}_\lambda, \lambda \in \Lambda\}$. We notice that $\mathbf{x}_{(j,k_1,\ldots,k_d)} = \mathbf{x}_{(j+1,2k_1,\ldots,2k_d)}$, so the set Λ may contain different copies of one point.

We part the set Λ into sets Λ_j containing the indices of the level j i.e. such that $\lambda \in \Lambda_j \implies \lambda = (j, k_1, \ldots, k_d)$. We put $\Omega_j = \{\mathbf{x}_\lambda, \lambda \in \Lambda_j\}$. Hence

$$\Omega_\lambda = \bigcup_{j\geq 0} \Omega_j.$$

The intersections $\Omega_j \cap \Omega_{j+1}$ may not be empty.

For a given j, the set Ω_j may not gather all the points of Ω_λ which can be written $(2^{-j}k_1, \ldots, 2^{-j}k_d)$. But we require that if one of the k_i of $\lambda = (j, k_1, \ldots, k_d)$ is odd and $\lambda \in \Lambda$ then λ remains to Λ_j. We also require that if $\lambda = (j, k_1, \ldots, k_d) \in \Lambda_j$ and $(j + \ell, 2^\ell k_1, \ldots, 2^\ell k_d) \in \Lambda_{j+\ell}$ for $\ell \geq 0$

then $(j+m, 2^m k_1, \ldots, 2^m k_d) \in \Lambda_{j+m}$ for all $m \in [0, \ell]$. It corresponds to having a fully-threaded tree.

Remark 2.1 *For sake of simplicity of the algorithms, we impose that all the sets Ω_j be self-sufficient regarding the computations: the interpolation and the differentiation at points in Ω_j call point values from Ω_j exclusively. Hence the transfer of information between the levels $j-1$ and j is carried out only through common elements of the sets Ω_{j-1} and Ω_j i.e. $\Omega_{j-1} \cap \Omega_j$.*

The function f whose point values are known at $(\mathbf{x}_\lambda)_{\lambda\in\Lambda}$ is uniquely decomposed into the interpolet hierarchical basis $(\Phi_\lambda)_{\lambda\in\Lambda_b}$ where the set Λ_b is such that $\Omega_{\Lambda_b} = \Omega_\Lambda$ and for all $\lambda = (j, k_1, \ldots, k_d) \in \Lambda_b$ at least one k_i is odd:

$$f(\mathbf{x}) = \sum_{\lambda\in\Lambda_b} d_\lambda \Phi_\lambda. \tag{1}$$

The multi-dimensional interpolet is defined as a tensorial product of one dimensional interpolets:

$$\Phi_\lambda(\mathbf{x}) = \varphi_{j\,k_1}(x_1) \times \cdots \times \varphi_{j\,k_d}(x_d).$$

We use the familiar wavelet notation $\varphi_{j\,k}(x) = \varphi(2^j x - k)$. The interpolets $(\varphi_{j\,k})$ are interpolant scaling functions: $\varphi(0) = 1$ and $\varphi(k) = 0\ \forall k \neq 0$. The algorithm transforming $(f(\mathbf{x}_\lambda))_\lambda$ into $(d_\lambda)_\lambda$ and vice versa has a linear complexity. We refer to [16] and the Appendix D for more details.

The expansion (1) allows a straight-forward projection along *e.g.* x_1. As

$$f(\mathbf{x}) = \sum_{j=0}^{j_{\max}} \sum_{k_1,\ldots,k_d} d_{j,\mathbf{k}} \varphi_{j\,k_1}(\mathbf{x}_1) \times \cdots \times \varphi_{j\,k_d}(\mathbf{x}_d)$$

and $\int_x \varphi_{j\,k}(x)\,dx = 2^{-j}\int_x \varphi(x)\,dx = 2^{-j}$, then

$$\int_{x1} f(\mathbf{x})\,dx_1 = \sum_{j=0}^{j_{\max}} \sum_{k_2,\ldots,k_d} \left(\sum_{k_1} d_{j,\mathbf{k}} \int_{x1} \varphi_{j\,k_1}(x_1)\,dx_1 \right) \varphi_{j\,k_2}(\mathbf{x}_2) \times \cdots \times \varphi_{j\,k_d}(\mathbf{x}_d)$$

$$= \sum_{j=0}^{j_{\max}} \sum_{k_2,\ldots,k_d} 2^{-j} \left(\sum_{k_1} d_{j,\mathbf{k}} \right) \varphi_{j\,k_2}(\mathbf{x}_2) \times \cdots \times \varphi_{j\,k_d}(\mathbf{x}_d).$$

2.2 High order interpolation

During the refinement process, the sets (Ω_j) are built respecting the following rule: for a given j, any point of Ω_j is part of a cube formed by 5^d adjacent points of Ω_j whose corners remain to $\Omega_{j-1} \cap \Omega_j$. An instance of such sets (Ω_j) is presented in two dimensions in Fig. 1.

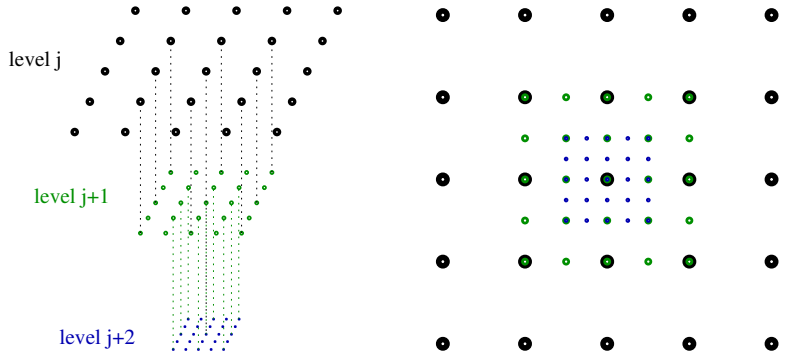


Figure 1: Refinement of a two dimensional grid: on the left the refinement process, on the right the resulting adaptive grid.

The results of the computations at level Ω_{j-1} are transmitted to the level Ω_j through $\Omega_{j-1} \cap \Omega_j$. And the 5^d cube structures allow order three interpolation from level Ω_{j-1} to level Ω_j, using the following formulas:

$$I_{3,r}\, f(x) = \frac{3\, f(x-h) + 6\, f(x+h) - f(x+3\,h)}{8}, \tag{2}$$

or its symmetric form, the letters r and l in $I_{3,r}$ and $I_{3,l}$ standing for right and left:

$$I_{3,l}\, f(x) = \frac{-f(x-3\,h) + 6\, f(x-h) + 3\, f(x+h)}{8}. \tag{3}$$

When it is possible we apply the fourth order interpolation inside Ω_j:

$$I_4\, f(x) = \frac{-f(x-3\,h) + 9\, f(x-h) + 9\, f(x+h) - f(x+3\,h)}{16}. \tag{4}$$

Fig. 2 represents the different cases when the interpolations $I_{3,r}$, $I_{3,l}$ and I_4 are applied.

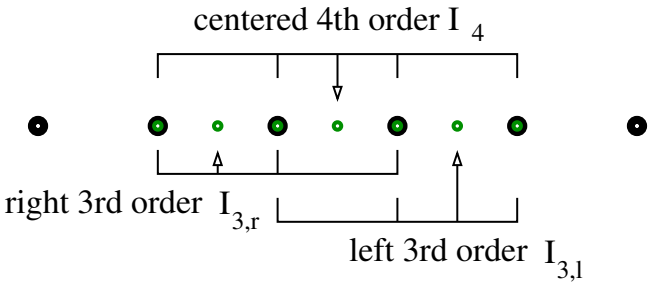


Figure 2: Third and fourth order interpolation in the one dimensional case.

In multi-dimensional simulations, the interpolation operates in a tensorial way, see Fig. 3. Hence, each point of the set $\Omega_j \setminus \Omega_{j-1}$ calls one

interpolation involving maximally four points. As a result the complexity of the interpolation algorithm remains linear with respect to the number of points independently of the dimensionality.

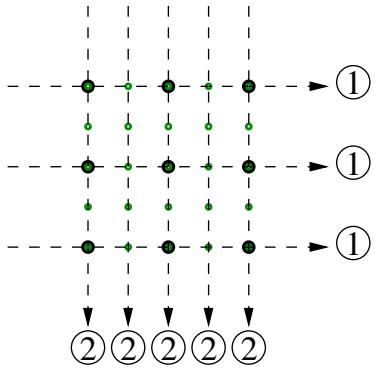


Figure 3: Interpolation of the finer-level green points by the coarser-level black points with direction by direction operations: we first interpolate horizontally and then vertically.

2.3 High order differentiation

To compute the differentiation of the function f in the i direction, many finite differences can be applied but few does not create any downwind instability in the context of Adaptive Mesh Refinement. To construct our scheme we also take into account the data structure and the computational complexity which shall remain independent of the dimensionality.

We navigate through the sets Ω_j for j going from 1 to $j_{\max}$. The set Ω_1 is a uniform grid with 2^d points. We apply the fifth order differentiation with periodic boundary conditions. Then we apply recursively the following operations for j from 2 to $j_{\max}$:

- we transfer the values obtained at the points remaining to the intersection $\Omega_{j-1} \cap \Omega_j$ from Ω_{j-1} to Ω_j,
- we interpolate all the points in Ω_j from the values in $\Omega_{j-1} \cap \Omega_j$,
- we compute the differentiation using a fifth –when possible– or third –when fifth not possible– order scheme.

We apply the following differentiations formula:

- the fifth-order upwind formula with a stencil of six points:

$$Df(x) = \frac{2\,f(x+3\,h) - 15\,f(x+2\,h) + 60\,f(x+h) - 20\,f(x) - 30\,f(x-h) + 3\,f(x-2\,h)}{60\,h} \tag{5}$$

- the third-order upwind formula with a stencil of four points:

$$Df(x) = \frac{-f(x+2\,h) + 6\,f(x+h) - 3\,f(x) - 2\,f(x-h)}{6\,h} \qquad (6)$$

- and the third-order differentiation with a stencil of three points:

$$Df(x) = \frac{5\,f(x+h) - 4\,f(x) - f(x-h)}{4\,h} - \frac{1}{2}Df(x+h) \qquad (7)$$

  where $Df(x+h)$ was computed at level $j-1$ and transmitted to level j by interpolation.

We summarize the different cases of application in Fig. 4. The numerical errors resulting from these differentiation schemes are presented in Appendix C.

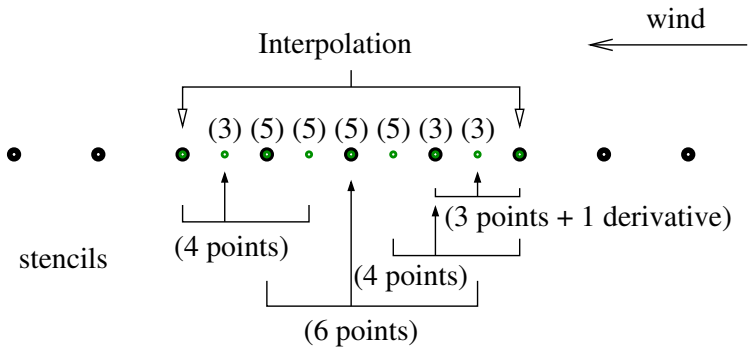


Figure 4: Overview of the different cases for the differentiation. The numbers above the points refer to the order of precision.

Choosing the fifth order when possible comes with little cost (less than 20%) and improves the precision and the conservations.

2.4 Numerical scheme for the convection in the adaptive grid

Using the grid described in Part 2.1, we propose a numerical scheme for the convection equation (8) which tackles two risks of instability:

- the upwind instability,
- the CFL[1] condition.

[1] CFL are the initials of the three authors of the founding paper [10] which first described and analyzed a stability condition linking the time step to the space step when numerically solving hyperbolic equations.

We avoid downwind instabilities by applying upwind formulas at all the points of the AMR grid.

Let us consider the following equation for the convection of a distribution function f by a velocity function $\mathbf{v} : \mathbb{T}^d \mapsto \mathbb{R}^d$:

$$\partial_t f(t, \mathbf{x}) + \mathbf{v}(\mathbf{x}) \cdot \nabla f(t, \mathbf{x}) = 0 \tag{8}$$

where $\nabla f = (\partial_1 f, \partial_2 f, \ldots, \partial_d f)$ and $\mathbf{v} \cdot \nabla f = \sum_i v_i \partial_i f$.

The transport gradient is computed by adding successively the d functions $v_i(\mathbf{x})\partial_i f(t, \mathbf{x})$: $\mathbf{v}(\mathbf{x}) \cdot \nabla f(t, \mathbf{x}) = \sum_{i=1}^{d} v_i(\mathbf{x})\partial_i f(t, \mathbf{x})$.

The CFL condition is optimized in order to benefit from the adaptivity of the grid. For the time increment, we apply a fourth-order Runge-Kutta scheme [23] to the equation (8). It is detailed in Appendix B. For this scheme associated with an upwind differentiation and a space step h, the CFL condition on the time step δt is usually given by:

$$\delta t \leq C \, \frac{h}{\max_{\mathbf{x} \in \Omega} \|\mathbf{v}(\mathbf{x})\|_{\ell^1}} \tag{9}$$

with the constant C approximately equal to 1.73 (from [14] and experiments by the author) and $\|\mathbf{v}(\mathbf{x})\|_{\ell^1} = \sum_{i=1}^{d} |v_i|$. In the case of an adaptive grid, the space step h of the grid depends on the position $\mathbf{x}$ in the computational domain Ω. As already noticed in [26] this allows to mitigate the CFL condition. Hence the CFL condition becomes:

$$\delta t \leq C \min_{\mathbf{x} \in \Omega} \frac{h(\mathbf{x})}{\|\mathbf{v}(\mathbf{x})\|_{\ell^1}}, \tag{10}$$

In phase-space coordinates, this condition on the time stepping is particularly advantageous since the high velocity parts of the grid generally present a low refinement: usually $h(\mathbf{x}) >> h(0)$ for $\mathbf{x}$ such that $\|\mathbf{v}(\mathbf{x})\| >> \|\mathbf{v}(0)\|$, so the time step δt remains large independently of $\max_{\mathbf{x} \in \Omega} \|\mathbf{v}(\mathbf{x})\|_{\ell^1}$.

3 Adaptivity

3.1 Principles and heuristics

In the method proposed in Sec. 2.1, points are disposed on a Cartesian grid which refines locally in an isotropic and dyadic way –*i.e.* one refinement corresponds in the division by two of the step in all the directions, so in d-dimensional spaces on point refines into 2^d points. In our implementation in C programming language, the points are stored in a tree structure (see Appendix A). One node of the tree corresponds to a cube of 2^d points.

The grid is refined according to a refinement criterion which relates the mesh step to the local needs of approximation evaluated through *e.g.* the

absolute value of a derivative. Ideally, as the finite difference or finite volume methods have an approximation error depending on a derivative of a certain order, we would take the magnitude of this derivative as a refinement criterion.

For instance the quality of the finite difference Eq. (6):

$$\frac{-f(x+2\,h)+6\,f(x+h)-3\,f(x)-2\,f(x-h)}{6\,h}=f'(x)-\frac{h^3}{12}f^{(4)}(x)+o(h^3),$$

applied to the C^4 regular function f depends on the fourth derivative $f^{(4)}$.

Unfortunately these derivatives of high order are not solved by the scheme and are computed incorrectly. This makes the refinement scheme unstable, versatile and unpredictable because the criterion is much to sensitive to numerical error.

Thus AMR users apply criteria on derivatives of lower orders such as: the derivative of order zero, *e.g.* in [36] the AMR maps a mass repartition function and each cell contains the same amount of mass, the gradient as in Harten's Discrete Framework [24, 32] or, more recently, the second derivatives as in [38] and here.

Such criteria allow to automatically refine or derefine the computational grid according to the local regularity of the function to map and the order of the method.

The exact form of the chosen criterion is decided based on the order of the scheme, on the characteristics of the simulation such as the regularity of the solution, the presence or not of shocks and the emergence of multiscale structures –*e.g.* filaments– that is to say, generally speaking, on the features pertinent for the computation or the physics.

We opted for the thresholding of the residue of a second order approximation: the coefficients d_λ in the expansion Eq. (1) using linear interpolets (see Appendix D). Applied to a smooth function f, the absolute value $w_j(\mathbf{x})$ of the residue can be expressed as:

$$w_j(\mathbf{x})=h^2\sum_{i\in\{1,\dots,d\}}|\partial_i^2 f(\mathbf{x})|+o(h^3)$$

with $h=2^{-j}$. In the following we replace the local sum of the derivatives by $|f^{(2)}(\mathbf{x})|$. Then the refinement level is fixed locally by the smallest $j(\mathbf{x})\in\mathbb{N}$ such that:

$$2^{-\alpha j(\mathbf{x})}w_j(\mathbf{x})\leq\varepsilon$$

with $\alpha\in\mathbb{R}$ and $\varepsilon>0$ fixed parameters.

In order to develop an heuristic leading to an adequate choice of the parameter α we make the following assumptions:

- the solution is C^∞, which is true for Vlasov-Poisson equations starting from a C^∞ initial condition,

- the error of the numerical method is dominated by the third order approximation in space controlled by the fourth derivative of the solution,
- at any given time the fourth derivative of the solution is proportional to the second derivative over all the phase-space domain. The idea behind this assumption is that one structure at one unknown scale has to be meshed at the right scale. But this is the most questionable point: although the second derivative is a better choice than the absolute value or the gradient and has the same parity, it may arbitrarily differ from the fourth derivative.

If we want to minimize the L^∞ error of the numerical approximation, we consider the following approximation –including a constant multiplier– of the error:

$$e(\mathbf{x}) = K\, h(\mathbf{x})^3\, |f^{(4)}(\mathbf{x})| \sim h(\mathbf{x})^3 |f^{(2)}(\mathbf{x})| \leq \varepsilon$$

with $h(\mathbf{x}) = 2^{-j(\mathbf{x})}$, then from $w_j(\mathbf{x}) = h(\mathbf{x})^2 |f^{(2)}(\mathbf{x})|$ we find the condition

$$h(\mathbf{x}) w_j(\mathbf{x}) = 2^{-j(\mathbf{x})} w_j(\mathbf{x}) \leq \varepsilon$$

i.e. $\alpha = 1$ which does not depend on the number of dimensions.

If we consider the minimization of the L^q-norm of the error of the numerical approximation, $\mathcal{E}_q$:

$$\mathcal{E}_q^q = \int_{[0,1]^d} e(\mathbf{x})^q \, dx \sim \int_{[0,1]^d} h(\mathbf{x})^{3\,q} |f^{(2)}(\mathbf{x})|^q \, dx,$$

while the number of points of the discretisation, given by

$$N = \int_{[0,1]^d} h(\mathbf{x})^{-d} \, dx,$$

is considered fixed. This is a problem of minimization under a constraint. It is solved thanks to the Lagrange multiplier $\lambda_0 \in \mathbb{R}$ such that $\forall g \in C^\infty([0,1]^d)$,

$$\int_{[0,1]^d} 3q\, h(\mathbf{x})^{3\,q-1} g(\mathbf{x}) |f^{(2)}(\mathbf{x})|^q \, dx + \lambda_0 \int_{[0,1]^d} (-d) h(\mathbf{x})^{-d-1}\, g(\mathbf{x})\, dx = 0.$$

that is to say

$$h^{d+3\,q}(\mathbf{x}) |f^{(2)}(\mathbf{x})|^q = \frac{\lambda_0\, d}{3q}$$

or

$$2^{-(\frac{d}{q}+1) j(\mathbf{x})} w_j(\mathbf{x}) = \varepsilon$$

which leads to $\alpha = 1 + \frac{d}{q}$. This advocates a flatter refinement in higher dimension.

Had we considered an approximation of $(p-s)$-th order of the s-th derivative of the solution leading to an error:

$$e(\mathbf{x}) = K\, h(\mathbf{x})^{p-s}\, |f^{(p)}(\mathbf{x})|$$

and an estimation based on a residue of m-th order for the refinement:

$$w_j(\mathbf{x}) = K'\, h(\mathbf{x})^m\, |f^{(m)}(\mathbf{x})|,$$

minimizing the L^q-norm of the error on the s-th derivative of the solution leads to the choice

$$\alpha = p - m - s + \frac{d}{q}.$$

In our case, we have $m = 2$, $p = 4$ and $s = 1$: we aim at minimizing the L^q-norm of the error on the gradient of the solution ($s = 1$) in order to minimize the L^q-norm of the error on the solution when simulating the advection.

Would we just have the rigorously correct estimation of the error with $m = p$, then the choice of α would be directly connected to the wavelet estimator from [8]:

$$\|f\|_{B^s_q(L^p)} \sim \| \left(2^{s\,j} 2^{d\,j(\frac{1}{2}-\frac{1}{p})} \|(\beta_\lambda)_{\lambda\in\Delta_j}\|_{\ell_p} \right)_{j\geq 0} \|_{\ell_q}$$

with $|\beta_\lambda| = 2^{-\frac{d}{2}j(\mathbf{x})} w_j(\mathbf{x})$ a L^2 normalization of the residuals, $\Delta_j \sim \Omega_j$ and $p = q$.

Minimizing the $B^s_p(L^p)$-norm of the error by thresholding the coefficients leads to discarding the coefficients β_λ such that $2^{(s+\frac{d}{2}-\frac{d}{q})j(\mathbf{x})}|\beta_\lambda| < \varepsilon$, *i.e.* $2^{(s-\frac{d}{q})j(\mathbf{x})} w_j(\mathbf{x}) < \varepsilon$, which corresponds to the choice $\alpha = -s + \frac{d}{q}$.

3.2 Details of the method for refining and coarsening

Assume we have defined an $\alpha \in \mathbb{R}$ relevant for the numerical experiment and an $\varepsilon > 0$ adapted to the available computer memory.

First, in a bottom-up algorithm –for j going from $j_{\max}$ to 2– we compute the residue of the linear interpolation of the level Ω_j by the level Ω_{j-1}. It is equivalent to a transform in the isotropic second-order hierarchical basis Eq. (1). This basis is composed of linear splines.

Then we gather the residues $d_{j,\mathbf{k}'}$ from Eq. (1) of the points $2^{-j}\mathbf{k}'$ inside cubes centered on points of the type $2^{-j}(2\ell_1+1,\ldots,2\ell_d+1)$ as indicated in Fig. 5. For $\mathbf{k} = (2\ell_1+1,\ldots,2\ell_d+1)$, this yields a refinement weight associated to this cube:

$$w^2_{j,\mathbf{k}} = \sum_{\mathbf{e}\in\{-1,0,+1\}^d} d^2_{j,\mathbf{k}+\mathbf{e}}.$$

Then we apply the following refinement instructions in this order:

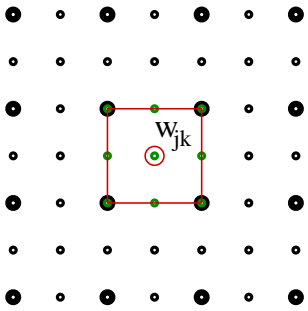


Figure 5: Elementary cube for the refinement.

- if $w_{j,\mathbf{k}} > 2^{\alpha j}\varepsilon$, the points inside the area are refined (if it is not the case already); the cube of size $(2h)^d$ and containing 3^d points –with a step equal to h– is refined into a cube of the same size but containing 5^d points –with a step equal to $\frac{h}{2}$,
- if $w_{j,\mathbf{k}} > 2^{\alpha j-1}\varepsilon$, then the points inside the area are preserved; we also activate (create or preserve) the neighbors located under the wind assuring the inclusion in a 5^d point cube necessary for the third order interpolation Eq. (2) and (3),
- if a point $2^{-j}\mathbf{k}$ is neither active at level j nor at finer levels then it is removed from the level j,

Sometimes we also fix a maximum level of refinement j_{Max} and we impose: $j \leq j_{\text{Max}}$.
The larger α, the closer the grid will be to a uniform grid. Varying ε allows to increase or decrease the number of points in the grid in order to cope with the memory storage limit of the computer (see the numerical experiments in Sec. 4).

4 Numerical results

In this Section we test the numerical scheme developed in this paper in various situations: the transport of a Gaussian in a six-dimensional box, two plasma simulations, the bump-on-the-tail instability in one dimension (two-dimensional phase space) and the two stream instability in 2D (four-dimensional phase space), and an astrophysics simulation, the merging of two halos of stars in the six-dimensional phase space.

4.1 Convection and stretching of a Gaussian in six dimensions

The numerical experiment consists in convecting a Gaussian function $f_0(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x_0}\|^2}{2\sigma^2}\right)$ with $\mathbf{x_0} = (\frac{1}{2}, \ldots, \frac{1}{2})$ in the six dimensional box $\mathbb{T}^6 = [0,1]^6$ with periodic boundary conditions, with the velocity field:

$$\mathbf{v}(x_1, x_2, \ldots, x_6) = \begin{bmatrix} 1 \\ \cos(2\pi x_1 + 1) \\ \cos(2\pi x_1 + 2) \\ \vdots \\ \cos(2\pi x_1 + 5) \end{bmatrix}. \tag{11}$$

For any particle at position $X(t) = (X_i(t))_{1 \le i \le 6}$ at time $t = 0$, the constant velocity $v_1 = 1$ gives $X_1(t) = X_1(0) + t$ and, for $2 \le i \le 6$,

$$\int_{t=0}^{1} v_i(X(t))\, dt = \int_{t=0}^{1} \cos(2\pi t + 2\pi X_1(0) + i - 1)\, dt = 0.$$

Hence the Gaussian comes back with its initial shape to its initial position at integer t. More generally, the exact solution is given by:

$$f(t, \mathbf{x}) = f_0(x_1 - t, x_2 - \frac{1}{2\pi}\left(\sin(2\pi x_1 + 1) - \sin(2\pi(x_1 - t) + 1)\right), \ldots$$

$$\ldots, x_6 - \frac{1}{2\pi}\left(\sin(2\pi x_1 + 5) - \sin(2\pi(x_1 - t) + 5)\right)) \tag{12}$$

so we can compute the error exactly.

We carried this test out thanks to a code written in C and parallelized with Open-MP. This experiment required an average of $4e{+}8$ points over 320 time steps. It took 34 hours on the 16 threads of a 2.20 Ghz Intel Xeon E5-2660 processor with 64 Go RAM memory. This approximately corresponds to a speed of $65,000$ points time-steps per second per thread.

In Fig. 6, we represent a two-dimensional projection onto (x_1, x_2) of the solution and of the adaptive grid. The maximum refinement level (in dark green) corresponds to a 128^6 uniform grid equivalent accuracy. The time step is taken constant equal to

$$\delta t = \frac{1}{320} \quad \text{close to the CFL maximum} \quad C\frac{h}{\|\mathbf{v}\|_{\ell^1}} \approx \frac{1}{344}$$

with $C = 1.73$, $h = 2^{-7}$ and $\|\mathbf{v}\|_{\ell^1} \approx 4.642$.

Since we want to minimize the L^∞ error, we take the thresholding parameter α equal to 1 (see Sec. 3.1). The initial value of ε is equal to $4e{-}4$ and the standard deviation of the Gaussian $\sigma = 0.1$.

In Fig. 6, we observe that the refinement of the grid follows the stretched Gaussian. The L^∞ error primarily appears at the boundaries between the

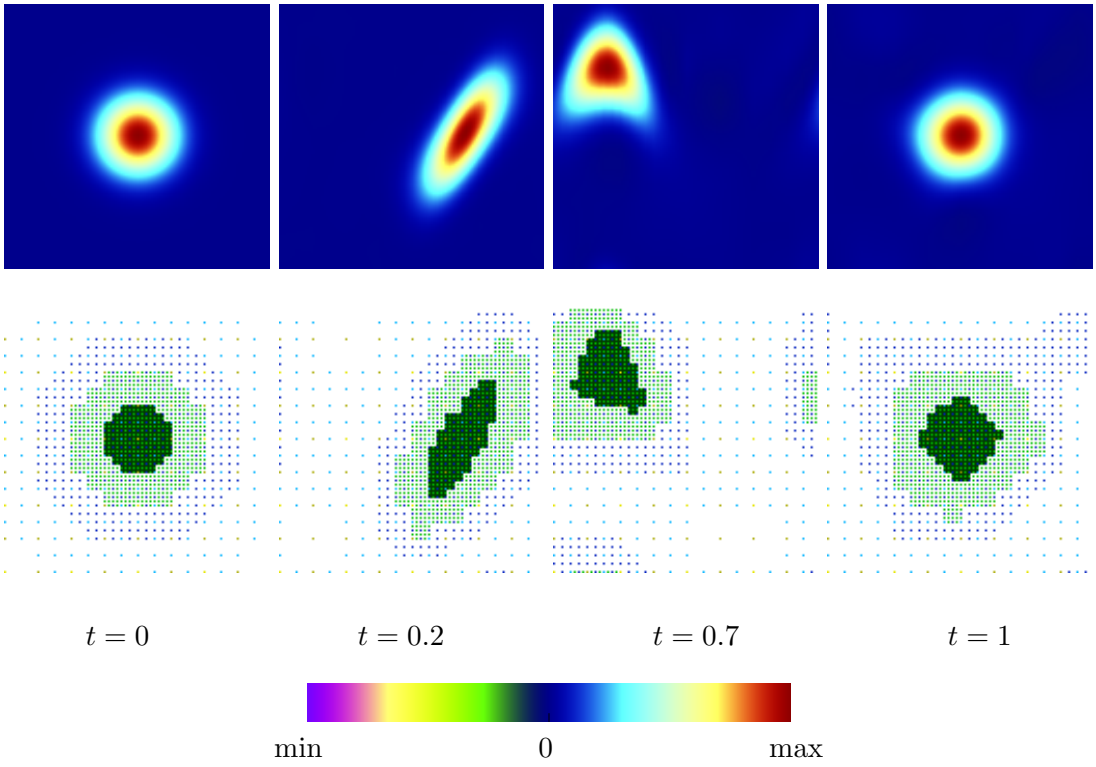


Figure 6: Evolution of the solution in projective view in the (x_1, x_2) plan (first row) and of the adaptive grid (second row) for the convection of a Gaussian in six dimensions at times $t = 0$, $t = 0.2$, $t = 0.7$ and $t = 1$.

different levels of refinement ahead of the Gaussian. As expected, the Gaussian comes back to its original position.

Regarding the color scale of Fig. 6, the dark blue color stands for zero and the dark red for the maximal value. This color scale is used in all the other figures representing a distribution function.

The L^∞-norm of the error remains inferior to 6% of the L^∞-norm of the gaussian as shown in Fig. 7. Relative error means the ratio between the norm of the error and the norm of the gaussian function. In this figure, we also represent the L^2-norm of the error, the minimum value of the solution (which shows that the scheme does not conserve the positivity), the threshold ε used to form the adaptive grid, the number of points contained in the grid and the corresponding memory needs.

The memory use coincides with the number of allocated nodes (see Appendix A) which is more or less proportional to the number of active points. In order not to exceed the memory limit and optimize the memory use, we bound the number of nodes of the tree (see Appendix A) with a maximum and a minimum. In the present test case the maximum of nodes is taken

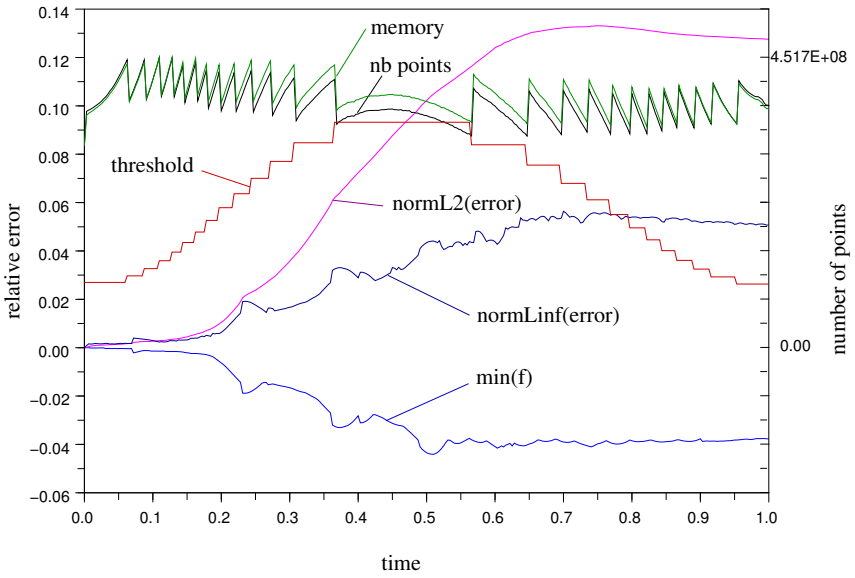


Figure 7: Error analysis, the curve for the threshold ε was rescaled fo fit the scale of the other curves.

equal to $1e{+}7$ and the minimum represents 80% of the maximum. Each time the number of tree nodes exceeds the maximum we fixed, the threshold ε is increased by 10%, leading to a drop in the number of allocated nodes and memory needs. This explains the saw-teeth aspect of the curve plotting the number of points, for $t \in [0, 0.5]$. On the contrary when the memory need decreases, we take advantage of this to increase the accuracy and we decrease the threshold by 10%. This is what happens in the second part of the graph, for $t \in [0.5, 1]$.

The error increases dramatically when the grid is at its minimum of refinement, with a high threshold ε and a major stretching of the solution, around $t = 0.5$.

In Fig. 8, we plot second-order estimates of the mass, of the L^2-norm and of the maximum of the solution. They indicate a precision of 1.5%. For a six-dimensional simulation. This is encouraging. The curves represent relative values compared to the initial one, so they equal 1 at time $t = 0$.

The fast oscillations of the maximum correspond to the shifting of the actual maximum of the solution from a point of the mesh to an other one.

The third-order convergence of the scheme is verified by comparing different runs varying the number of points. For a method of pth-order, and since $N \sim h^{-6}$, the error should satisfy:

$$e = C\,h^p = C\,(N^{1/6})^{-p} = C\,\frac{1}{N^{p/6}}.$$

Experimentally, we run the experiment until $t = 0.1$ and compare the nu-

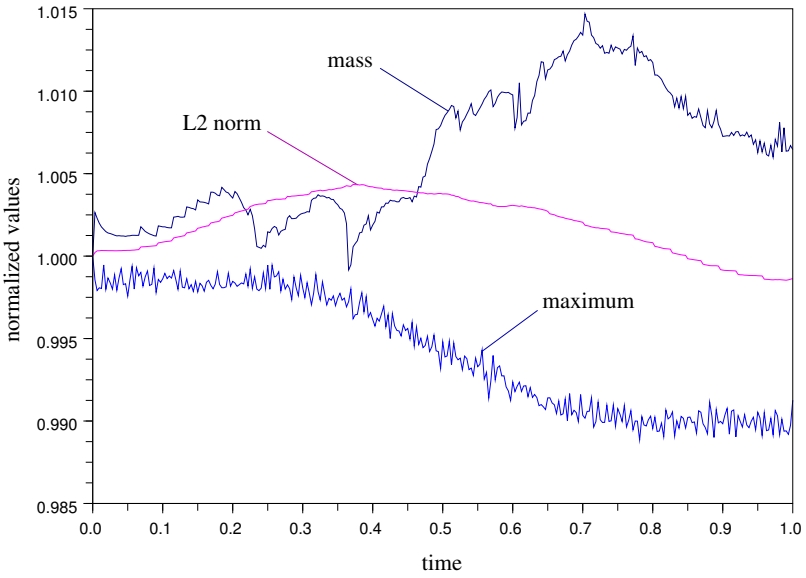


Figure 8: Conservation analysis.

merical result with the exact solution. We plot the error in L^∞-norm and L^2-norm for numbers of points from $6e{+}6$ to $4e{+}8$ in Fig. 9.

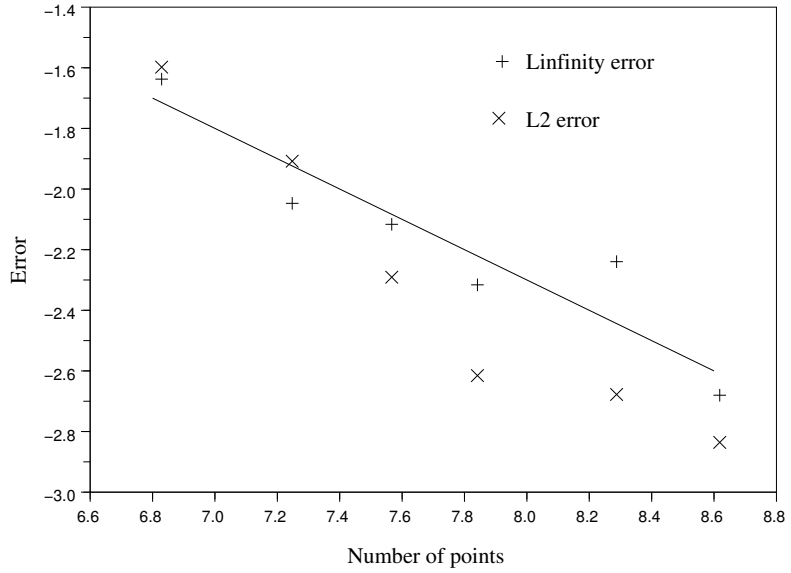


Figure 9: Drop of the error in L^∞ and L^2 norms. Both x and y axis have a Log_{10} scale. The line has a slop equal to $-\frac{1}{2}$ which corresponds to a third order convergence of the scheme.

In Fig. 9, the L^∞ norm of the error can be compared with L^∞ norm of

the Gaussian equal to one (or zero in Log_{10}). The L^2 norm of the error was multiplied by 100 in the plot and can be compared with the L^2 norm of the Gaussian approximatively equal to $5.6e{-}03$ (or -2.25 in Log_{10}).

The L^∞ norm of the error may increase drastically when interpolating new points since the interpolation has same third order as the convection scheme. This explains the jumps in the L^∞ norm of the error in Fig. 7, and the irregularity of the convergence plot Fig. 9.

4.2 Application to plasma physics

In Plasma physics and astrophysics, most of the Vlasov-Poisson simulations rely on Particle-In-Cell methods [9, 12] as they allow 6-dimensional phase-space simulations at a reasonable computational cost. In these methods, the distribution function is coarsely discretized with Dirac functions (particles) whose positions evolve through Lagrangian schemes. Recently, grid-based Eulerian schemes have been developped [20] but they rarely pass the curse of dimensionality [30] and most of the time do not exceed four dimensions for the phase-space [37].

Two-dimensional phase space case: the bump-on-tail instability In this paragraph we simulate the bump-on-tail instability with our AMR scheme. In [26], the authors tested a block-structured AMR using a high-order finite-volume scheme on this case. We obtain similar results.

We consider a density function $f : \mathbb{R}^{2d} \to \mathbb{R}_+, (\mathbf{x}, \mathbf{v}) \mapsto f(\mathbf{x}, \mathbf{v})$ subject to the Vlasov-Poisson equation:

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f + E(t, \mathbf{x}) \cdot \nabla_{\mathbf{v}} f = 0 \tag{13}$$

with

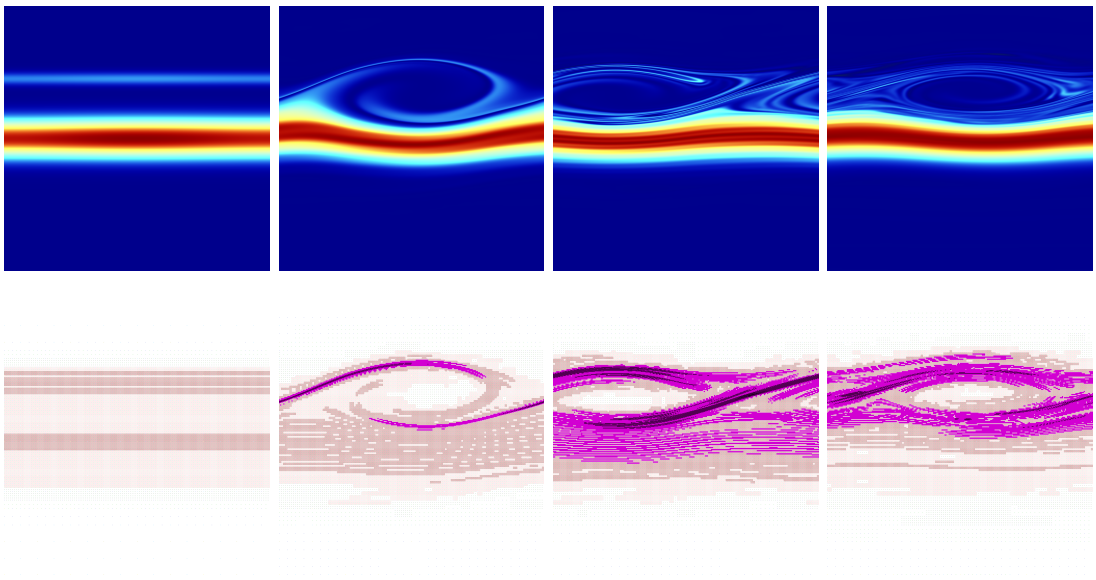
$$E(t, \mathbf{x}) = \nabla_{\mathbf{x}} \phi(t, \mathbf{x}), \tag{14}$$

$$\Delta_{\mathbf{x}} \phi(t, \mathbf{x}) = \int_{\mathbf{v} \in \mathbb{R}^d} f(t, \mathbf{x}, \mathbf{v})\, d\mathbf{v} - \int_{\mathbf{x}, \mathbf{v} \in \mathbb{R}^d} f(t, \mathbf{x}, \mathbf{v})\, d\mathbf{v}\, d\mathbf{x}. \tag{15}$$

To make the following simulation, f is taken periodic in the $\mathbf{x}$ variable, and the dimension $d = 1$. The simulation box is $(x, v) \in [-\frac{10}{3}\pi, \frac{10}{3}\pi] \times [-10, 10]$. The initial condition is given by:

$$f_0(x, v) = \left(\frac{0.9}{\sqrt{2\pi}} e^{-\frac{v^2}{2}} + \frac{0.2}{\sqrt{2\pi}} e^{-4(v-4.5)^2} \right).$$

We integrate f in eq. (15) by decomposing the discrete solution f in a linear spline hierarchical basis Eq. 1 and summing the coefficients in the velocity direction as indicated in Sec. 2.1. Then eq. (15) and (14) are solved thanks to one-dimensional Fourier transforms.



$t = 0$ $t = 22$ $t = 50$ $t = 100$

Figure 10: Evolution of the solution (first row) and of the adaptive grid (second row) for the bump-on-tail instability (1D).

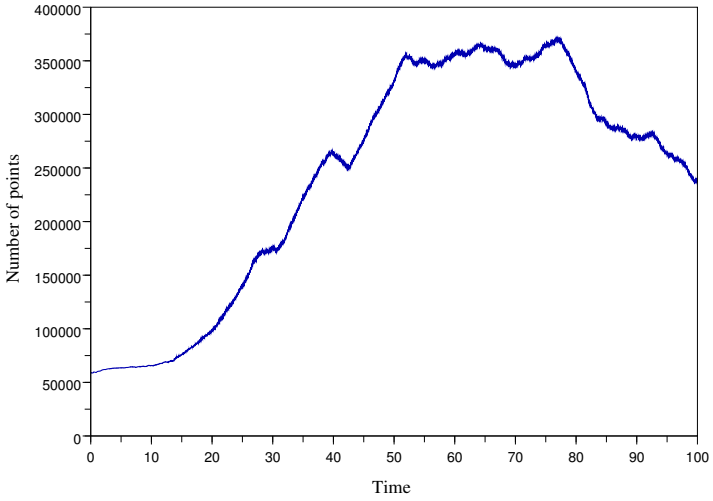


Figure 11: Evolution of the number of grid points in the AMR.

The simulation is plotted in Fig. 10 which shows a very good agreement with the results presented in [26]. We observe that the fine mesh –dark violet is equivalent to a 2048^2 uniform mesh– follows the filaments accurately.

In Fig. 11 we observe that the number of points vary from $50,000$ at the beginning to $370,000$ at the maximal complexity. The refinement parame-

ters were fixed to $\alpha = 1.7$ and $\varepsilon = 1e$–6.

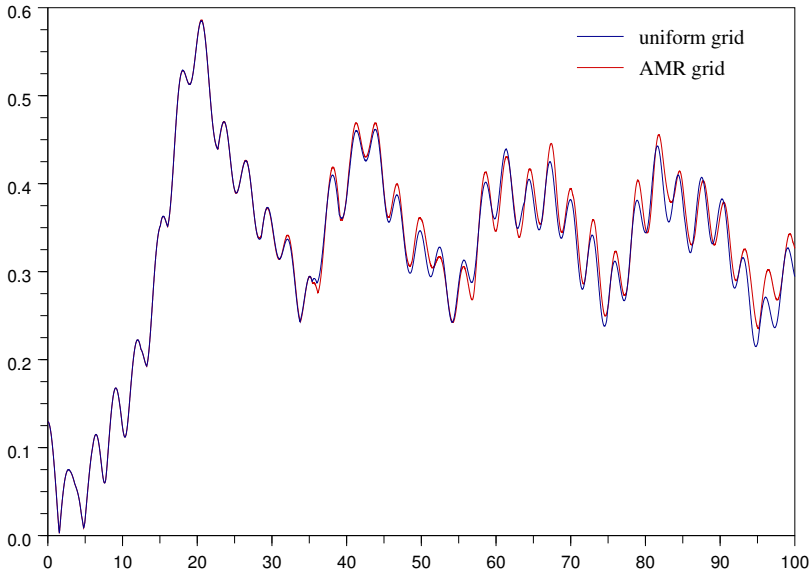


Figure 12: Plots of the maximum absolute value of the field E for two instances of the bump-on-tail instability: with an uniform grid and with an AMR grid.

The plot of the maximal electric field $\max_x |E(t,x)|$ differs with the one obtained in [26] with no obvious explanation. The uniform grid experiment was done with 256^2 points using a fourth-order in time and fifth-order in space finite difference scheme.

Fig. 13 illustrates important defects of the proposed AMR method for solving the Vlasov-Poisson equations: it does not conserve the mass and it makes negative values appear.

In Fig. 14 we can see that although the kinetic energy and the potential energy tend to compensate each other, the total energy drifts away from its initial value. In contrast, the uniform grid experiment conserves the mass and the total energy exactly: the relative variations stay below $1e$–6 *i.e.* the computer accuracy. The negative values it creates stay below a relative 0.1% to compare with the 8% appearing in the finite differences AMR experiment (see Fig. 13).

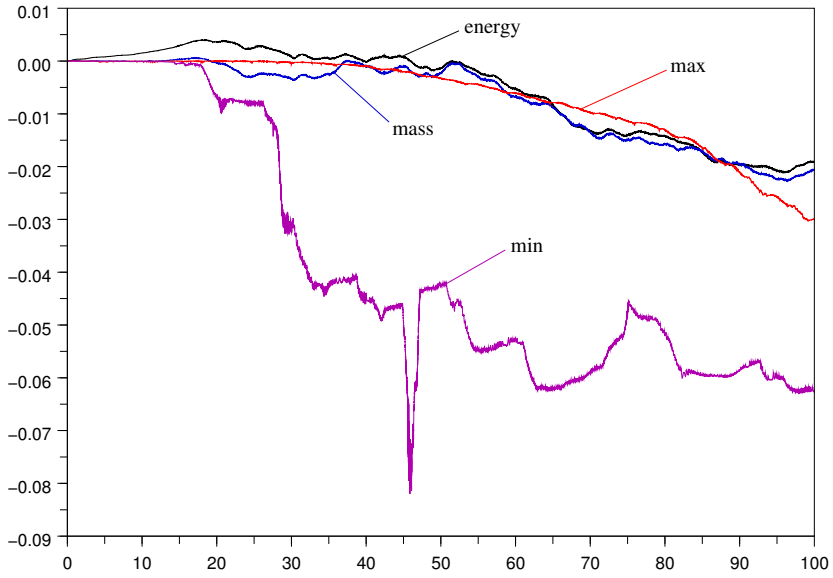


Figure 13: Relative variations ($\Delta f/f$) for the mass, the total energy and the maximum value of the distribution function. These should remain constant. For visualization purpose, the mass variation was multiplied by ten.

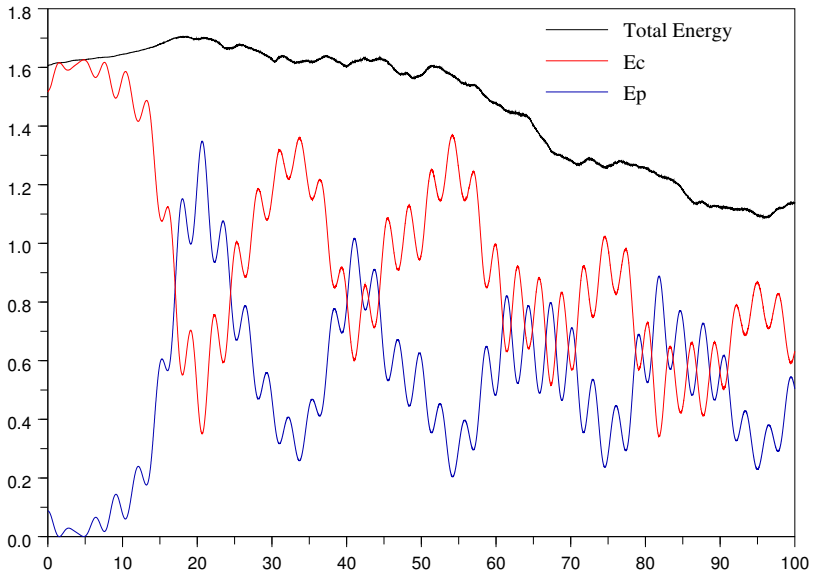


Figure 14: Variation of the kinetic (Ec) and potential (Ep) energies. The kinetic energy was vertically shifted by -23.

Four-dimensional phase space case: the two steam instability We apply the present scheme to a four-dimensional test case. Hence we begin to exploit the novelty of our AMR scheme. For instance such a high dimensionality is neither treated in [26] nor in [33].

In the simulation box $(x, y, u, v) \in [-\frac{10\,\pi}{3}, \frac{10\,\pi}{3}]^2 \times [-3\,\pi, 3\pi]^2$ with periodic boundary conditions, we consider the initial distribution function:

$$f_0(x, y, u, v) = \frac{7}{4\,\pi} \exp\left(-\frac{u^2 + 4\,v^2}{8}\right) \sin^2\left(\frac{u}{3}\right) (1 + 0.05\,\cos(0.3\,x)) \quad (16)$$

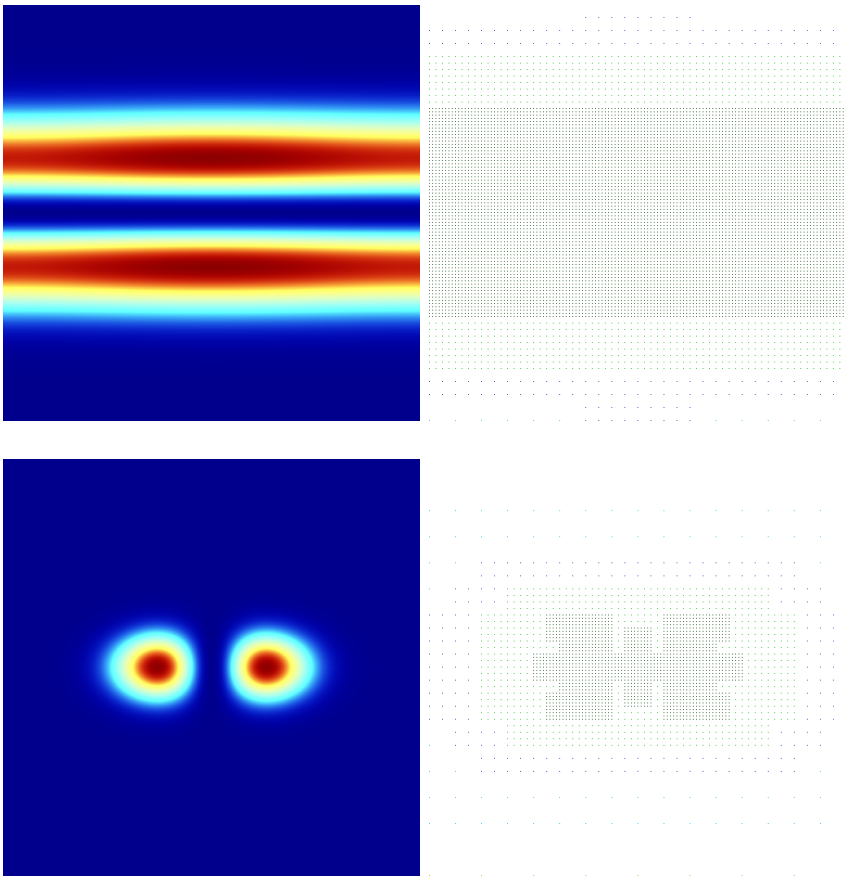


Figure 15: Phase space view of the initial condition Eq. (16). Cuts at zero along (x, u) (first row) and (u, v) (second row) of the distribution function and of the adaptive grid.

This initial distribution, represented in Fig. 15, was chosen in order to maximize the two stream instability [18]. Initially, it is discretized with $30e{+}6$ points and has a maximum level of refinement equivalent to a 128^4 uniform grid. The thresholding parameters are taken as follows: $\alpha = 1.5$ and $\varepsilon = 2.4e{-}5$ initially.

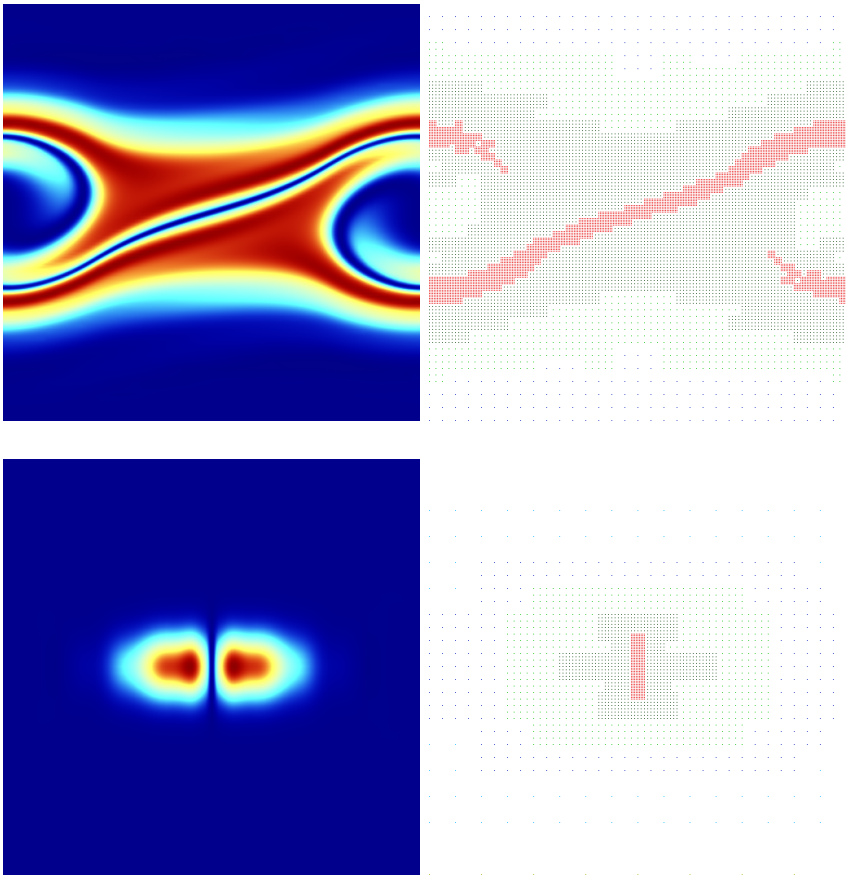


Figure 16: Phase space view of the numerical solution at time $t = 12$. Same cuts as Fig. 15.

in Fig. 16 we represented the numerical solution at $t = 12$ when the instability begins to develop. The maximum refinement goes up to 256^4 (the red points in Fig. 16 right column). The number of points increases to $62e{+}6$. And the threshold ε begins to grow, and is now $\varepsilon = 3.9e{-}5$. We have limitated the number of nodes to $5e{+}6$ with maximally $2^4 = 16$ active points each.

4.3 Six-dimensional astrophysics case: the merging of two halos of stars

In order to apply the numerical scheme to a six-dimensional phase space problem, we switch from plasma physics to astrophysics. In plasma physics the distribution function tends to occupy the whole three-dimensional physical space while in the gravitational case it collapses, allowing the adaptive scheme to refine only locally in space.

Hence the adaptive scheme refines the grid locally in the six-dimensional phase space and not only in the velocity direction as it is the case in plasma physics (see Fig. 15 and 16). And we are able to simulate the gravitational Vlasov-Poisson equations at a reasonable cost.

For a collisionless self-gravitating system such as a halo of stars or dark matter, the distribution function of matter in the phase space $(\mathbf{x}, \mathbf{v}) \in \mathbb{R}^6$, with $\mathbf{x} = (x, y, z)$ and $\mathbf{v} = (u, v, w)$

$$f : \begin{array}{ccc} \mathbb{R} \times \mathbb{R}^3 \times \mathbb{R}^3 & \rightarrow & \mathbb{R}^+ \\ t, \mathbf{x}, \mathbf{v} & \mapsto & f(t, \mathbf{x}, \mathbf{v}) \end{array} \tag{17}$$

obeys the gravitational Vlasov-Poisson equations:

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f + E(t, \mathbf{x}) \cdot \nabla_{\mathbf{v}} f = 0 \tag{18}$$

$$E(t, \mathbf{x}) = -\nabla_{\mathbf{x}} \phi(t, \mathbf{x}) \tag{19}$$

$$\Delta_{\mathbf{x}} \phi(t, \mathbf{x}) = 4\pi G \rho(t, \mathbf{x}) \tag{20}$$

where the density function ρ is given by:

$$\rho(t, \mathbf{x}) = \int_{\mathbb{R}^3} f(t, \mathbf{x}, \mathbf{v})\, d\mathbf{v}. \tag{21}$$

Comparing with plasma equations (14) and (15), please note the sign "–" in Eq. (19) and the factor 4π (not gone when the equation are nondimensionalized) in Eq. 20.

These equations admit stationary solutions [2]. One of them, the Plummer model, was successfully tested by Takao Fujiwara in [21] using the symmetries of the problem in a two-dimensional plus one invariant simulation. It is given by the following distribution function:

$$f(r, \|\mathbf{v}\|) = \frac{3M}{7\pi^3 a^3} \left(2 \left(1 + \left(\frac{r}{a} \right)^2 \right)^{-1/2} - \|\mathbf{v}\|^2 \right)^{7/2} \tag{22}$$

$$\text{if} \quad 2 \left(1 + \left(\frac{r}{a} \right)^2 \right)^{-1/2} - \|\mathbf{v}\|^2 \geq 0$$

and $f(r, \|\mathbf{v}\|) = 0$ everywhere else.
We noted $r^2 = x^2 + y^2 + z^2$ and $\|\mathbf{v}\|^2 = u^2 + v^2 + w^2$.

Then the potential function is given by:

$$\phi(r) = -\frac{GM}{a}\left(1 + \left(\frac{r}{a}\right)^2\right)^{-1/2}, \tag{23}$$

and the density function by:

$$\rho(r) = \frac{3M}{4\pi a^3}\left(1 + \left(\frac{r}{a}\right)^2\right)^{-5/2}. \tag{24}$$

In these formulas, G denotes the constant of gravity, M the total mass and a a length parameter. In the following simulations, these constants are taken equal to 1.

This distribution function Eq. (22) is compactly supported in velocity but not in space.

In Fig. 17, we represent this stationary solution in an adaptive grid. The maximum level of refinement corresponds to a 256^6 uniform grid (in red).

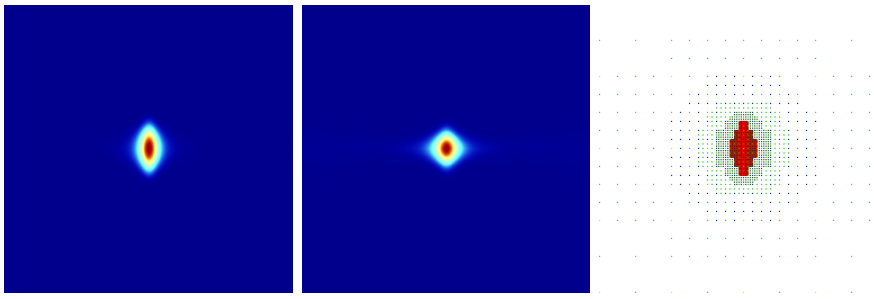


Figure 17: The Plummer Model: a two-dimensional cut (on the left), a projection (on the middle) and the corresponding adaptive grid (on the right) in (x, u)-view.

We apply the present adaptive scheme to the six-dimensional merging of two spherical Plummer models. Hence we simulate the 'collision' of two halos of stars which, when separated, form stable systems.

The convection is solved using the scheme described in Sec. 2.4, the projection using (1) at second-order, and the gravitational forces are computed in the three-dimensional Fourier space on a uniform grid of the smallest scale (i.e. with FFT in a 256^3 grid). The time step δt is computed at the beginning of each Runge-Kutta cycle by the CFL formula (10) with $C = 1$. This value is below the CFL constant for uniform domains $C = 1.73$. At the end of each Runge-Kutta cycle, we let the mesh evolve as described in Sec. 3.2 with $\alpha = 1.7$ and $\varepsilon = 2.5e{-}5$ at $t = 0$ and we prescribe the maximal number of nodes at $5e{+}6$. In order to exploite the memory optimally, we fix the minimum number of nodes at 70% of this maximum. Each node contains a maximum of $2^6 = 64$ active points.

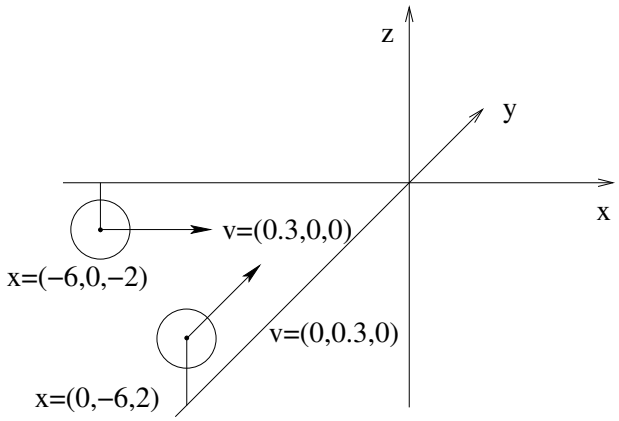


Figure 18: Initial conditions for the interaction of two Plummer models.

The initial condition is schematized in Fig. 18. Each sphere represents a stationary solution Eq. (22). The size of the box is $[-12, 12]^3 \times [-8, 8]^3$. We apply periodic boundary conditions. In order to visualize the evolution of the numerical solution, we make a cut in the (x, y)-direction at the maximum of the solution $(x_0, y_0, z_0, u_0, v_0, w_0) = \arg\max(f)$, so we plot $f(x_0 + x, y_0 + y, z_0, u_0, v_0, w_0)$ for $(x, y) \in \mathbb{T}^2$ in Fig. 19, first row. The color scale is the same as in Fig. 6.

In Fig. 19, we observe rings of the phase space localized in the physical domain at time $t = 33.6$. The mesh follows the details of the solution and adapts automatically.

In Fig. 20, we see the collision and merging of the two spheres in a three-dimensional (x, y, z) projection.

In the phase space, Fig. 21, we can observe this phenomenon. Contrarily to the physical space Fig. 20, we see in the first row Fig. 21 (and less in the second row) that the two spheres never totally merge for the time of observation $t \in [0, 60]$. Only when the small scales are under-resolved at time $t = 20$ does the numerical mixing begin: the entropy increases (Fig. 22) and later the maximum of f also increases.

In Fig. 22, we plot:

- the mass
$$M(t) = \int_{\mathbf{x},\mathbf{v}} f(\mathbf{x}, \mathbf{v}, t)\, d\mathbf{x}\, d\mathbf{v} \tag{25}$$
- the L^2-norm
$$\|f\|_2(t) = \left(\int_{\mathbf{x},\mathbf{v}} f(\mathbf{x}, \mathbf{v}, t)^2\, d\mathbf{x}\, d\mathbf{v} \right)^{\frac{1}{2}} \tag{26}$$

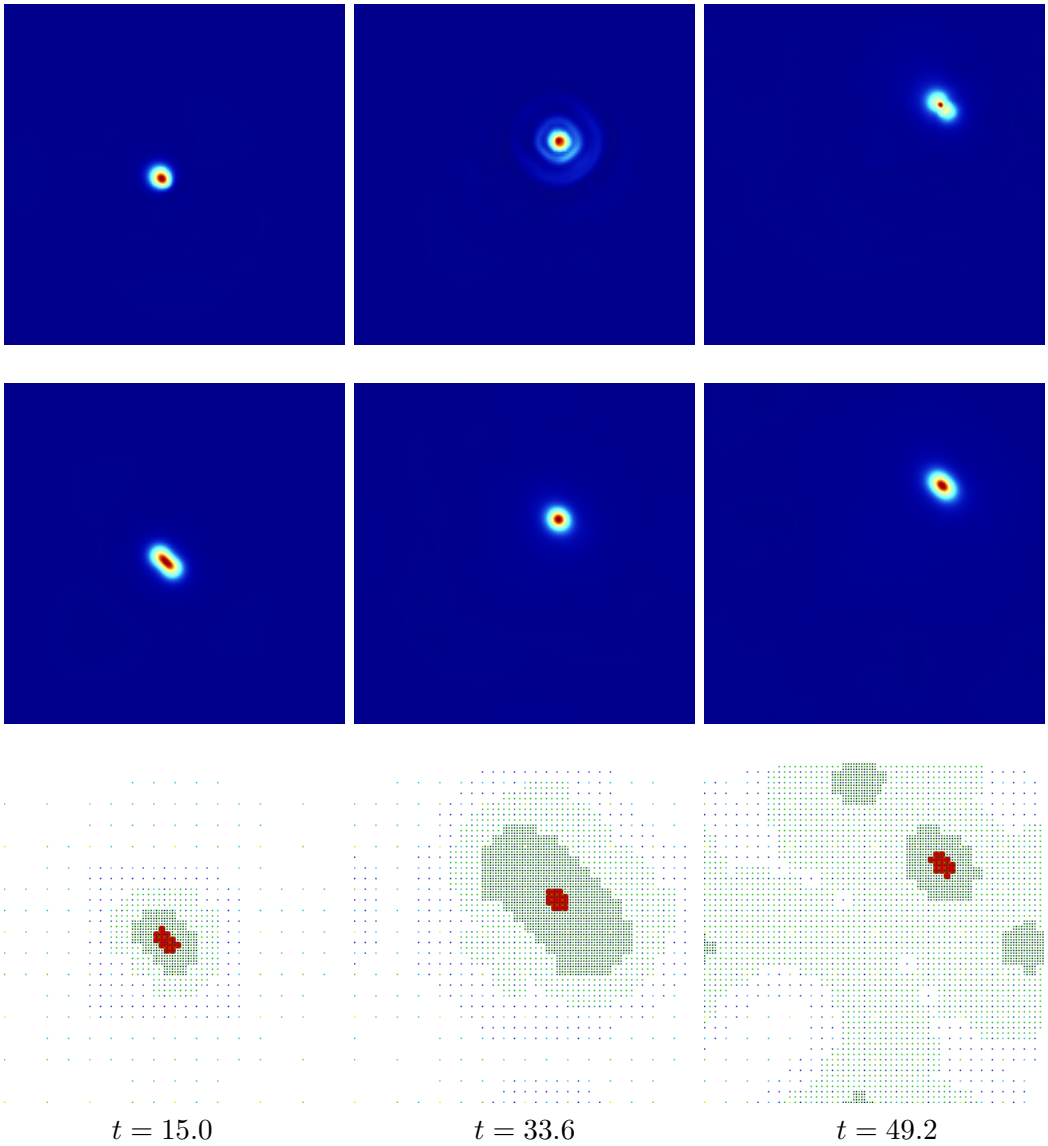


Figure 19: Collision of two Plummer spheres, a cut in the (x, y)-direction (first row), the projection onto (x, y) (second row) and the adaptive grid (third line) at times $t = 15.0$, $t = 33.6$ and $t = 49.2$.

- the entropy

$$S(t) = \int_{\mathbf{x},\mathbf{v}} -f(\mathbf{x}, \mathbf{v}, t) \ln(f(\mathbf{x}, \mathbf{v}, t))\, d\mathbf{x}\, d\mathbf{v} \tag{27}$$

which are conserved quantities for the Vlasov-Poisson equations.
In Fig. 23 we also plot

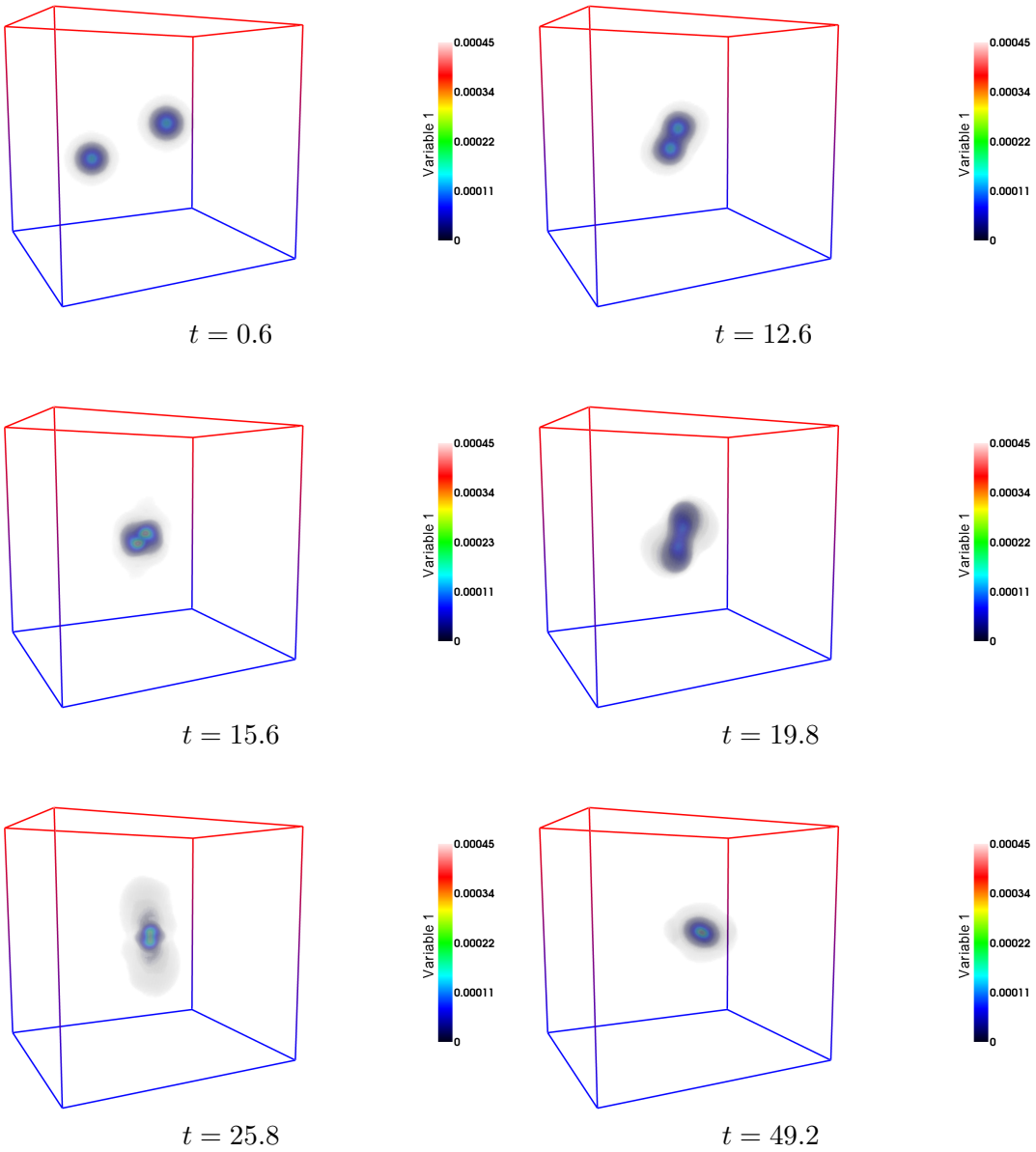


Figure 20: Three dimensional view in physical space of the collision of two Plummer spheres at times $t = 0.6, 12.6, 15.6, 19.8, 25.8, 49.2$ (this visualization was realized thanks to the IFRIT 3D Data Visualization Open Source software).

- the kinetic energy

$$E_c(t) = \int_{\mathbf{x},\mathbf{v}} \frac{\|\mathbf{v}\|^2}{2} f(\mathbf{x}, \mathbf{v}, t)\, d\mathbf{x}\, d\mathbf{v} \tag{28}$$

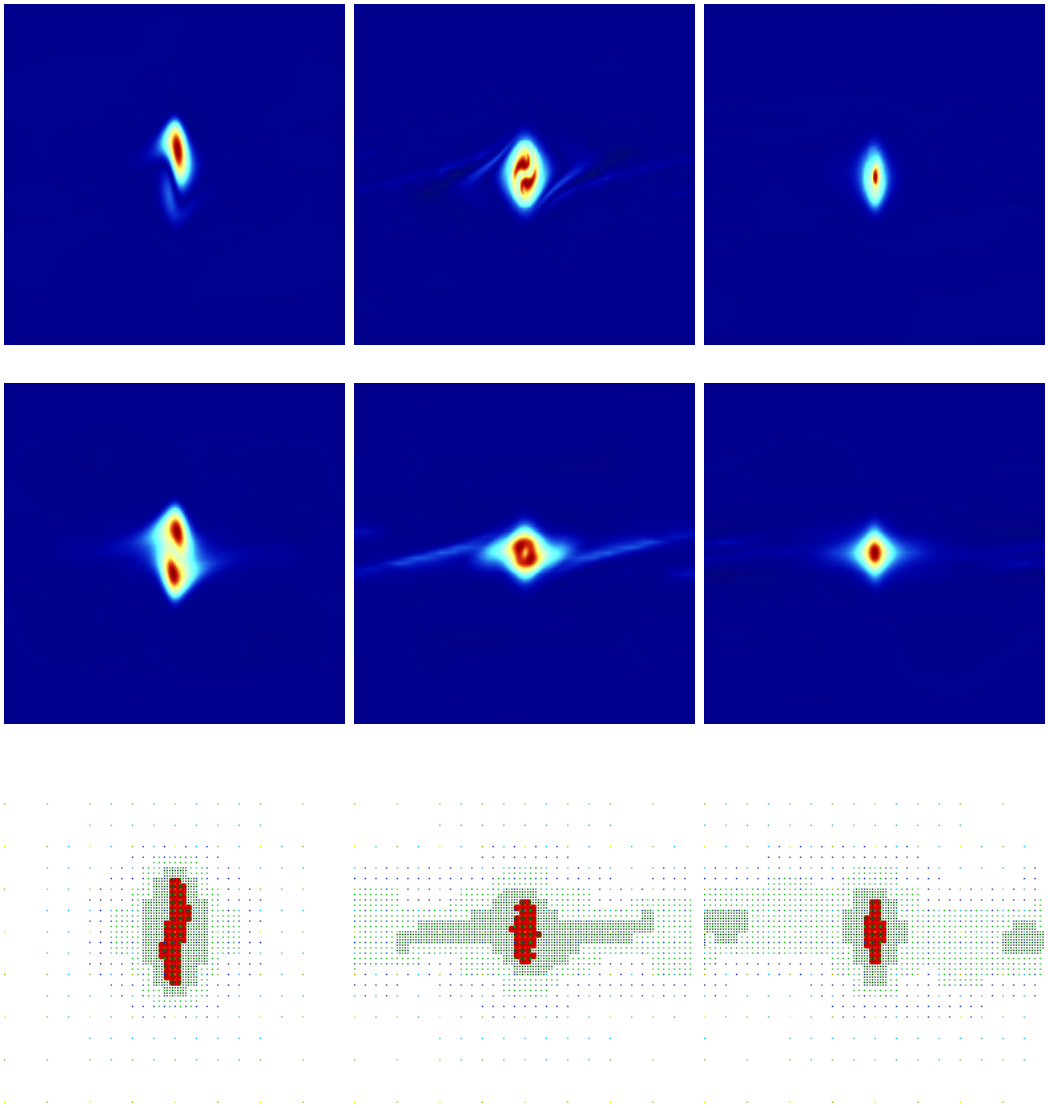


Figure 21: View in the phase space of the collision of two Plummer spheres, a cut in the (z, w)-direction (first row), the projection onto (z, w) (second row) and the corresponding adaptive grid (third row) at times $t = 15.0$, $t = 33.6$ and $t = 49.2$.

The mass is not conserved exactly but remains in a $\pm 15\%$ range. The entropy increases reasonably ($+25\%$) and this increase can be explained by the usual numerical averaging of the filamentation [35].

Watching at the threshold evolution Fig. 23, we can distinguish five phases:

1. for $t \in [0, 12]$, the two spheres get close, the complexicity stays steady,

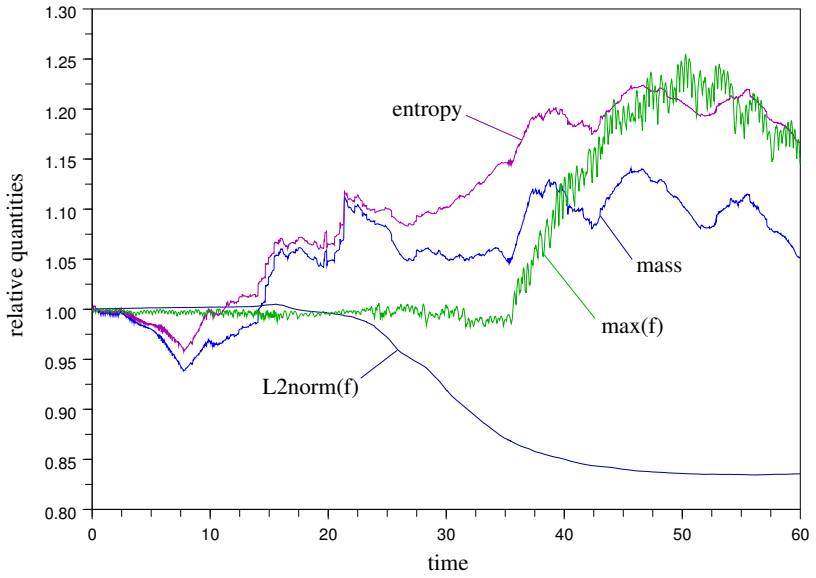


Figure 22: Conservation analysis. These are second-order estimations of the quantities.

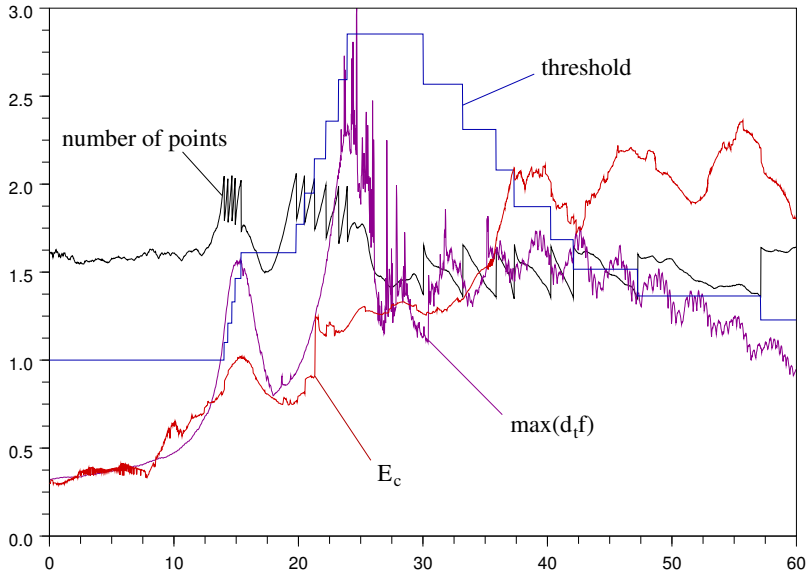


Figure 23: Evolution of the threshold ε (normalized scale), the number of points ($\times 1e{+}8$ for the scale), the kinetic energy E_c (normalized scale) and the maximum of the time derivative $|\partial_t f|$ (normalized scale).

2. for $t \in [12, 15]$, first collision, the complexity surges and the kinetic energy and the time derivative make a jump,

3. for $t \in [15, 19]$, the spheres recede one from the other, the complexity even begins to decrease slightly,

4. for $t \in [19, 24]$, the spheres interact anew, creating a complex situation,

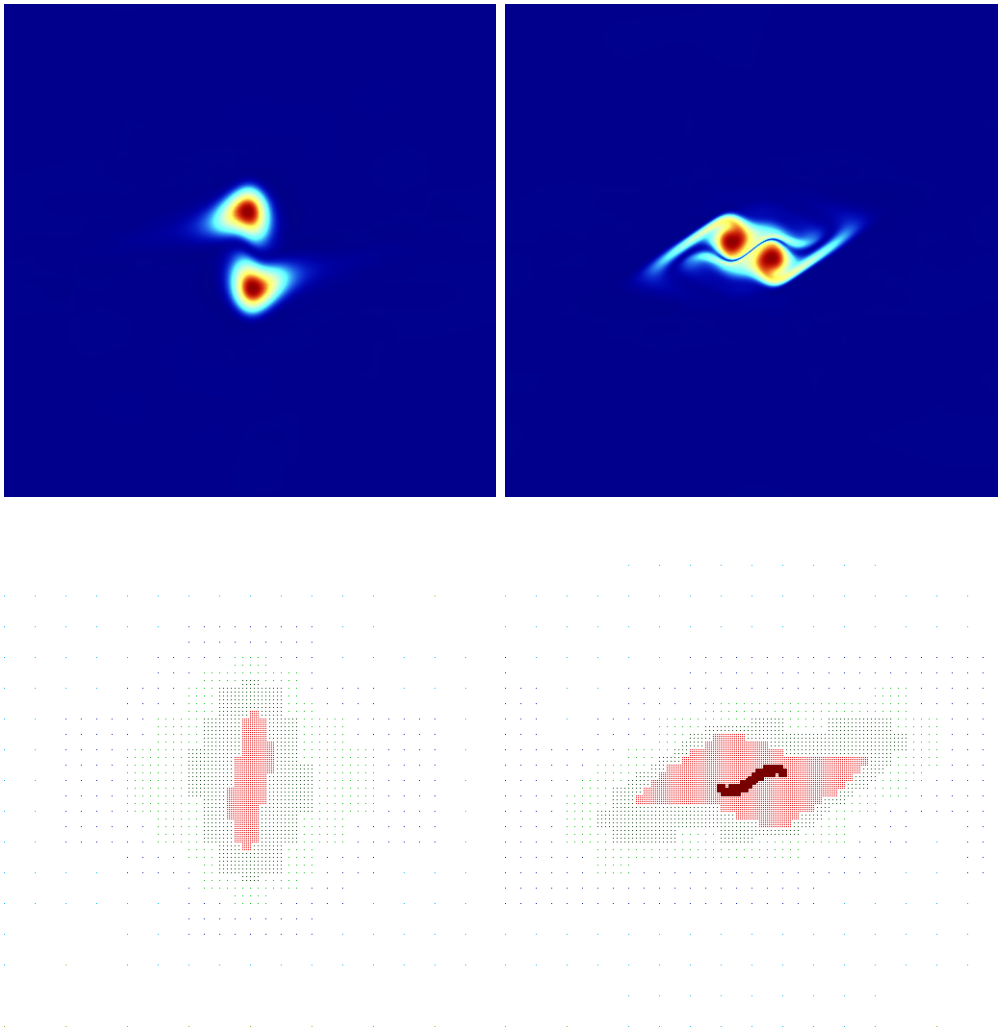
5. for $t \in [24, 60]$, damping, the solution decomplexifies, the threshold decreases to retain more and more small scales while these are dissipated by the numerical scheme or discarded by the resolution of the grid.

The number of points peaks at a maximum of $200,000,000$ triggering an increase of the threshold ε, and then, for t greater than 30, it decreases to a minimum if $140,000,000$ which makes the threshold ε increse. Running this experiment took $2,638$ time steps, and 226 hours of computation on a 2.66GHz Intel Xeon X5650 (12 threads and 48 Go RAM memory). This means an average speed of $42,000$ points time-steps per second per thread.

This experiment can be compared to the one presented in [37]. In this paper [37], two King spheres interact in a 64^6 uniform mesh ($\sim 6.87e{+}10$ points, i.e. 300 times what we are using) applying a Positive Flux Conservation (PFC) scheme [19] which is a kind of finite volume method, associated to a massive MPI/OpenMP hybrid parallelization on 1024 CPU cores on 64 nodes (i.e. 85 times the ressources we are consumming).

To further validate the AMR approach, we ran this six-dimensional numerical experiment on an Extra Large node of the supercomputer Curie: 128 cores –this large amount of cores did not allow to increase the computational speed compared with 16 cores due to the limits of the efficiency of the parallelisation– and 512 GB main memory allowing to fix to $1e{+}8$ the maximum number of tree nodes. Each tree node can contain a maximum of 64 active points. The Curie Extra Large nodes have been formed by grouping four S6010 bullx modules into a single shared-memory system, using Bull's Coherent Switch (BCS) novel architecture. BCS, an ASIC chip designed by Bull provides a global, consistent view of main memory data for all processors of the system [11].

For this experiment, a minimum threshold was fixed at $\varepsilon_{\min} = 2e\text{-}6$ and the maximum number of nodes at $100e{+}6$. The second thresholding parameter α was kept at $\alpha = 1.7$. For $t \in [0, 21.7]$, the number of nodes stayed below $71.6e{+}6$ and the threshold was not modified $\varepsilon = \varepsilon_{\min}$. It took 696 time steps to reach $t = 21.7$ and the number of active points reached $3e{+}9$ (see Fig. 25). The result at $t = 21.6$ in Fig. 24 top right is visually satisfactory: the AMR automatically refines up to a 512^6 uniform grid equivalent accuracy (in dark red on Fig. 24 bottom right) in the areas where filamentation appears.



$t = 15.0$ $t = 21.6$

Figure 24: Collision of two Plummer spheres, a cut of the distribution function in the (z, w)-direction (first row) at $(x, y, u, v) = (0, 0, 0, 0)$ and a cut of the corresponding adaptive grid (second row). Simulation with fixed threshold ε and up to $3e9$ points on the supercomputer Curie from Idris (GENCI project gen7437 (2015)).

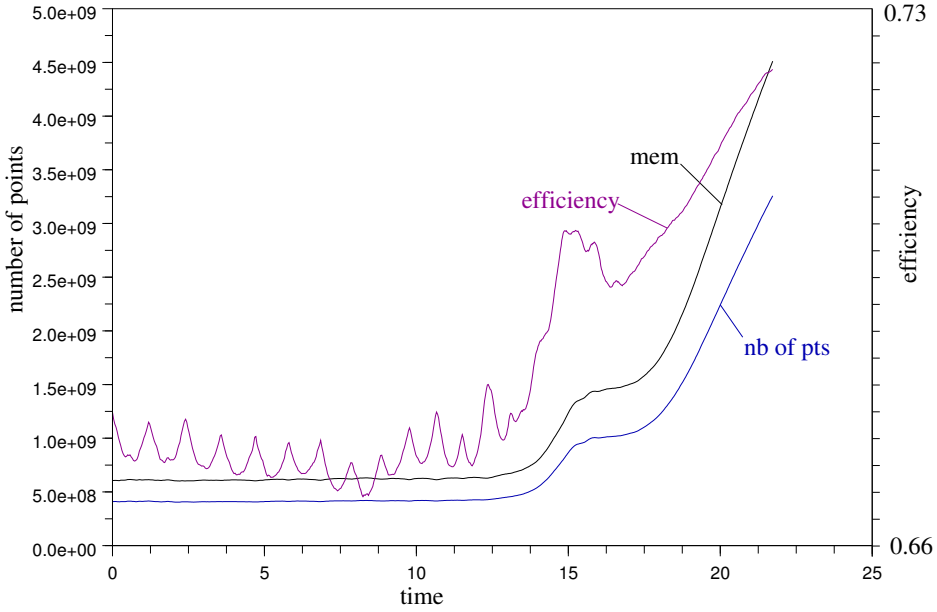


Figure 25: Point storage efficiency.

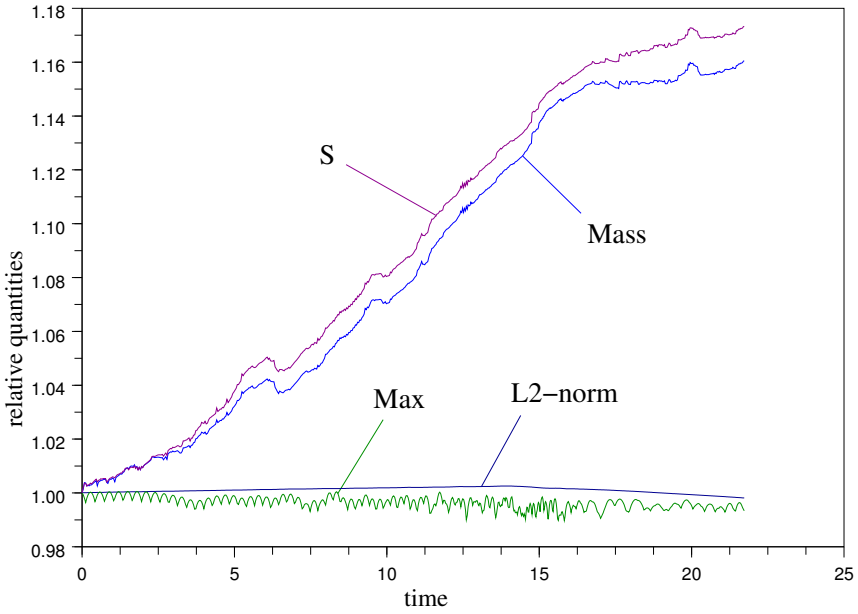


Figure 26: Conservations.

5 Conclusion

This work encourages the Adaptive Mesh Refinement approach to solve the Vlasov equations. AMR provides a good accuracy even in the six-dimensional case which was out of reach for the traditional Eulerian schemes. The basic simplicity of the numerical scheme we used –finite differences and third-order interpolation– caused some defects which must be corrected: the mass and the energy are not conserved and under-resolved shock-like structures may deteriorate the solution.

This opens the way to further work: to cope with the conservation trouble we can switch to Finite Volume schemes adapting the existing literature *e.g.* [33] to the six-dimensional constraint, or we will continue with Interpolet/Finite Difference discretisation using fourth-order centered interpolation and enforcing the mass conservation by correcting the finite difference scheme. The shock and negative value problems may be resolved using limiters and WENO schemes.

An other perspective is to further reduce the number of degrees of freedom by using six-dimensional Hyperbolic Wavelets [15] also called Sparse Grids [5]. Although this entails new algorithmic and numerical difficulties, it seems the best way to install six-dimensional AMR as a routine numerical tool.

Acknowledgements

This work has been funded in part by ANR grant ANR-13-MONU-0003 and by the Eurofusion ER14 project EURATOM-CfP-WP14-ER-01/IPP-03. The author acknowledges the help of the Maison de la Simulation Saclay and the Institute of Mathematics Polska Akademia Nauk Warsaw for two month hosting in Spring 2014. This work has also benefited a lot from interesting discussions with collaborators and particularly with Teresa Regińska, Nicolas Besse, Stéphane Colombi, Thierry Sousbie, Alain Ghizzo and Maxime Lesur, and from exchanges with Siegfried Müller and his team and with Romain Teyssier.

References

[1] T. Abel, O. Hahn, and R. Kaehler, *Tracing the dark matter sheet in phase space*, MNRAS **427** 61–76, 2012.

[2] Antonov, V. A., Vest. Leningrad Gos. Univ. **19**(96), 1962.

[3] T.D. Arber, R.G.L. Vann, *A Critical Comparison of Eulerian-Grid-Based Vlasov Solvers*, Journal of Computational Physics **180**(1) 339–357, 2002.

[4] N. Besse, G. Latu, A. Ghizzo, E. Sonnendrücker, P. Bertrand, *A wavelet-MRA-based adaptive semi-Lagrangian method for the relativistic Vlasov–Maxwell system*, Journal of Computational Physics **227**(16) 7889–7916, 2008.

[5] O. Bokanowski, J. Garcke, M. Griebel, I. Klompmaker, *An Adaptive Sparse Grid Semi-Lagrangian Scheme for First Order Hamilton-Jacobi Bellman Equations*, Journal of Scientific Computing **55**(3) 575–605, 2013.

[6] K. Brix, S. Melian, S. Müller, M. Bachmann, *Adaptive Multiresolution Methods: Practical issues on Data Structures, Implementation and Parallelization*, ESAIM: Proceedings **34** 151–183, V. Louvet, M. Massot (eds.), 2011.

[7] J. Büchner, *Vlasov-code simulation*, Advanced Methods for Space Simulations 23–46, edited by H. Usui and Y. Omura, TERRAPUB, Tokyo, 2007.

[8] A. Cohen, R. DeVore, G. Kerkyacharian and D. Picard, *Maximal spaces with given rate of convergence for thresholding algorithms*, Applied and Computational Harmonic Analysis **11** 167–191, 2001.

[9] G.-H. Cottet, P.-A. Raviart, *Particle methods for the one-dimensional Vlasov–Poisson equations*, SIAM J. Numer. Anal. **21** p. 52, 1984.

[10] R. Courant, K. Friedrichs, H. Lewy, *On the Partial Difference Equations of Mathematical Physics*, IBM Journal, march 1967, translation from a paper originally appeared in *Mathematische Annalen* **100** 32–74, 1928.

[11] J. David et al., *Best Practice Guide – Curie v1.17*, 2013.

[12] P. Degond, F. Deluzet, L. Navoret, A.-B. Sun, M.-H. Vignal, *Asymptotic-Preserving Particle-In-Cell method for the Vlasov–Poisson system near quasineutrality*, Journal of Computational Physics **229**(16) 5630–5652, 2010.

[13] S. Delage Santacreu, *Méthode de raffinement adaptatif hybride pour le suivi de fronts dans des écoulements incompressibles*, Thèse de Mécanique de l'Univérsité Bordeaux I, 2006.

[14] E. Deriaz, *Stability conditions for the numerical solution of convection-dominated problems with skew-symmetric discretizations*, SIAM J. Numer. Anal. **50**(3) 1058–1085, 2012.

[15] E. Deriaz and V. Perrier, *Direct Numerical Simulation of turbulence using divergence-free wavelets*, SIAM Multiscale Modeling and Simulation **7**(3) 1101–1129, 2008.

[16] S. Dubuc, *Interpolation through an iterative scheme*, J. Math. Anal. and Appl. **114** 185–204, 1986.

[17] M. Dumbser, V. A. Titarev, and S. V. Utyuzhnikov, *Implicit Multiblock Method for Solving a Kinetic Equation on Unstructured Meshes*, Computational Mathematics and Mathematical Physics **53**(5) 601–615, Pleiades Publishing, Ltd., 2013.

[18] E. Fijalkow, *Behaviour of phase-space holes in 2D simulations*, Journal of Plasma Physics **61**(01) 65–76, 1999.

[19] F. Filbet, E. Sonnendrucker, P. Bertrand, F. Filbet, E. Sonnendrcker, P. Bertrand, *Conservative numerical schemes for the Vlasov equation*, Journal of Computational Physics **172** 166–187, 2001.

[20] F. Filbet, E. Sonnendrcker, *Comparison of Eulerian Vlasov solvers*, Computer Physics Communications **150** 247–266, 2003.

[21] T. Fujiwara, *Integration of the Collisionless Boltzmann Equation for Spherical Stellar Systems*, Publ. Astron. Soc. Japan **35**, 547–558, 1983.

[22] M. Grandin, *Data structures and algorithms for high-dimensional structured adaptive mesh refinement*, Advances in Engineering Software **82** 75–86, 2015.

[23] Ernst Hairer, Syvert Paul Nørsett, Gerhard Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer Series in Comput. Mathematics, Vol. 8, Springer-Verlag 1987, Second revised edition 1993.

[24] A. Harten, *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Comm. Pure and Applied Math. **48** 1305–1342, 1995.

[25] F. Hecht, *New development in FreeFem++*, J. Numer. Math. **20**(3-4) 251–265, 2012.

[26] J.A.F. Hittinger, J.W. Banks, *Block-structured adaptive mesh refinement algorithms for Vlasov simulation*, Journal of Computational Physics **241** 118–140, 2013.

[27] M. Holmström, *Solving Hyperbolic PDEs Using Interpolating Wavelets*, SIAM Journal on Scientific Computing **21**(2) 405–420, 1999.

[28] Kevlahan, N.K.-R. and Vasilyev, O.V., *An adaptive Wavelet Collocation Method for Fluid-Structure Interaction*, SIAM Journal on Scientific Computing **26**(6) 1894–1915, 2005.

[29] T. Nakamura, T. Yabe, *Cubic interpolated propagation scheme for solving hyper-dimensional Vlasov-Poisson equation in phase space*, Computer Physics Communications **120** 122–154, 1999.

[30] D. Pflüger, *Spatially Adaptive Sparse Grids for High-Dimensional Problems*, Ph.D. thesis of Technischen Universität München, 2010.

[31] S. Popinet, *Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries*, Journal of Computational Physics **190**(2) 572–600, 2003.

[32] O. Roussel, K. Schneider, A. Tsigulin, H. Bockhorn, *A conservative fully adaptive multiresolution algorithm for parabolic PDEs*, Journal of Computational Physics **188**(2) 493–523, 2003.

[33] C. Shen, J.-M. Qiu, A. Christlieb, *Adaptive mesh refinement based on high order finite difference WENO scheme for multi-scale simulations*, Journal of Computational Physics **230**(10) 3780–3802, 2011.

[34] E. Sonnendrcker, J. Roche, P. Bertrand, A. Ghizzo, *The Semi-Lagrangian Method for the Numerical Resolution of the Vlasov Equation*, Journal of Computational Physics **149**(2) 201–220, 1999.

[35] V. Springel, *The cosmological simulation code* GADGET-2, MNRAS **364** 1105, 2005.

[36] R. Teyssier, *Cosmological hydrodynamics with adaptive mesh refinement. A new high resolution code called RAMSES*, Astron. Astrophys. **385** 337–364, 2002.

[37] Yoshikawa K., Yoshida, N., Umemura, M., *Direct Integration of the Collisionless Boltzmann Equation in Six-dimensional Phase Space: Self-gravitating Systems*, The Astrophysical Journal **762** 116, 2013.

[38] O. Zanotti, M. Dumbser, *A high order special relativistic hydrodynamic and magnetohydrodynamic code with spacetime adaptive mesh refinement*, Computer Physics Communications **188** 110–127, 2015.

Appendix

A Data structure and C encoding

The encodings for tree structures usually rely either on hash-tables of keys referring directly to the positions of the nodes [6] either to real tree structures built with pointers. In Fortran, these pointers are emulated thanks to tables of indices [36].

Using C programming, we chose to implement the whole fully-threaded tree structure. The basic element allocated in memory is the node. In dimension d, its building block is given by:

```
struct Node {
 double tab[nb val][2^d];   /* cube of simulation values, 2^d points    */
 struct Node *prt;          /* pointer to the parent node               */
 struct Node *chd[2^d];     /* 2^d pointers to the children             */
 struct Node *ngb[2d];      /* pointers to the neighbors                */
 int rgf;                   /* filiation rank                           */
 unsigned long long flg;    /* flag for the point current activation    */
 unsigned long long ldf;    /* flag for the point previous activation   */
 int dir;                   /* direction of the wind                    */
 int ell;                   /* parameter with different uses            */
};
```

The tree formed by multiple nodes can be read through its parent connection (see Fig. 27 left). Then a table ordered level by level (Fig. 27 right) allows to perform the algorithms presented in the present manuscript. It also permits Open-MP parallelization.

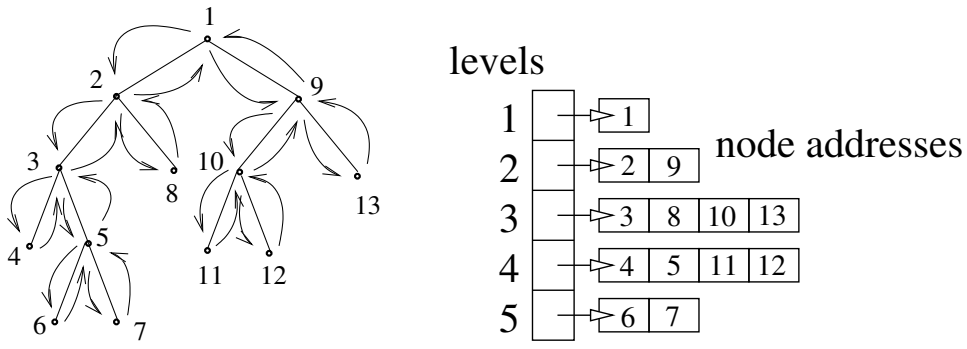


Figure 27: Accessing the nodes in the tree structure: graph transversal on the left, table of node addresses ordered level by level on the right.

B Algorithms

Three main algorithms occur in the AMR method we present, the first two rely on the level by level tables, Fig. 27 right:

1. Bottom-up computation in the tree. For instance it is used to update the results obtained at coarser level with results obtained at finer levels. These latter have a better accuracy (maximum level of refinement). It is done by transferring from level j to level $(j-1)$ the contain of the points remaining to $\Omega_{j-1} \cap \Omega_j$ for j from $j_{\max}$ to 2. Passing from the point values to the expansion Eq. (1) is also a bottom-up operation.

2. Top-down computation in the tree. As indicated in Part 2.3, we compute the finite difference differentiations for the level j going from level 1 to level $j_{\max}$. Computing the point values from the expansion Eq. (1) is a top-down computation.

3. Fourth-order Runge-Kutta scheme [23] applied to the equation (8):

$$\left\{\begin{array}{ll} k_0 = -\mathbf{v}(\mathbf{x}) \cdot \nabla f_n, & f_{n(1)} = f_n + \frac{\delta t}{2} k_0 \\ k_1 = -\mathbf{v}(\mathbf{x}) \cdot \nabla f_{n(1)}, & f_{n(2)} = f_n + \frac{\delta t}{2} k_1 \\ k_2 = -\mathbf{v}(\mathbf{x}) \cdot \nabla f_{n(2)}, & f_{n(3)} = f_n + \delta t \; k_2 \\ k_3 = -\mathbf{v}(\mathbf{x}) \cdot \nabla f_{n(3)}, & \\ k = \frac{k_0}{6} + \frac{k_1}{3} + \frac{k_2}{3} + \frac{k_3}{6}, & f_{n+1} = f_n + \delta t \; k \end{array}\right. . \tag{29}$$

   Then the iterative process for one time step unfolds as follows:

   (a) we compute the time step δt using Eq. (10),

   (b) we compute k sequentially adding the contributions from k_ℓ for $\ell \in \{0,1,2,3\}$ while $f_{n(\ell)}$ is regularly updated using Eq. (29),

   (c) we make $f_{n+1} = f_n + \delta t \, k$,

   (d) we compute the weight of the refinement details: $w(f_{n+1})$,

   (e) we refine and prune the tree using $w(f_{n+1})$ and the refinement algorithm from Sec. 3.2,

   (f) we loop to (a).

   Applying this algorithm necessitated six storages for each point:

   - room 1 $\longrightarrow f_n$,
   - room 2 $\longrightarrow k$ and $w(f_n)$,
   - room 3 $\longrightarrow v_i$,
   - room 4 $\longrightarrow v_i \partial_i f$,

- room 5 $\longrightarrow k_\ell$,
- room 6 $\longrightarrow f_{n(\ell)}$.

The resulting scheme is third-order in space and fourth-order in time. Passing to Vlasov-Poison equation just necessitates to compute $\mathbf{v}$ and does not need extra storage.

C Finite differences and interpolation

Here we summarize the interpolations and finite differences we used with the corresponding errors for sufficiently smooth functions:

- Differentiation
  - fifth-order upwind finite difference:
    $Df(x) =$
    $$\frac{2\,f(x+3\,h)-15\,f(x+2\,h)+60\,f(x+h)-20\,f(x)-30\,f(x-h)+3\,f(x-2\,h)}{60\,h} \tag{30}$$
    $$= f'(x)+\frac{h^5}{60}f^{(6)}(x)+O(h^6),$$
  - third-order upwind formula:
    $$Df(x)=\frac{-f(x+2\,h)+6\,f(x+h)-3\,f(x)-2\,f(x-h)}{6\,h} \tag{31}$$
    $$= f'(x)-\frac{h^3}{12}f^{(4)}(x)+O(h^4),$$
  - third-order differentiation using a differentiation computed at level Ω_{j-1}:
    $$Df(x)=\frac{5\,f(x+h)-4\,f(x)-f(x-h)}{4\,h}-\frac{1}{2}D_{2h}\,f(x+h) \tag{32}$$
    $$= f'(x)-\frac{h^3}{24}f^{(4)}(x)+O(h^4),$$
    or
    $$= f'(x)+\frac{7}{24}h^3f^{(4)}(x)+O(h^4),$$
    depending on how the differentiation $D_{2h}\,f(x+h)$ was computed.
- Interpolation
  - third-order interpolation
    $$I_{3,r}\,f(x)=\frac{3\,f(x-h)+6\,f(x+h)-f(x+3\,h)}{8}=f(x)-\frac{h^3}{2}f^{(3)}(x)+O(h^4), \tag{33}$$

– fourth-order interpolation

$$I_4\, f(x) = \frac{-f(x-3\,h) + 9\, f(x-h) + 9\, f(x+h) - f(x+\,h)}{16} \\ = f(x) + \frac{3}{8} h^4 f^{(4)}(x) + O(h^6). \tag{34}$$

D Interpolets

Here we detail the interpolet construction.

For the thresholding, we resort to the second-order interpolation $f(\frac{1}{2}) = \frac{f(0)+f(1)}{2}$, the corresponding scale function filter is given by:

$$\varphi_2(\frac{x}{2}) = \frac{1}{2}\varphi_2(x+1) + \varphi_2(x) + \frac{1}{2}\varphi_2(x-1) \tag{35}$$

which is also the linear B-spline function, and for the fourth-order interpolation:

$$f(\frac{1}{2}) = \frac{-f(-1) + 9\, f(0) + 9\, f(1) - f(2)}{16}, \tag{36}$$

the scaling function is given by

$$\varphi_4(\frac{x}{2}) = -\frac{1}{16}\varphi_4(x+3) + \frac{9}{16}\varphi_4(x+1) + \varphi_4(x) + \frac{9}{16}\varphi_4(x-1) - \frac{1}{16}\varphi_4(x-3). \tag{37}$$

More generally

$$\varphi_{2n}(\frac{x}{2}) = \sum_{\ell} c_\ell \varphi_{2n}(x-l), \tag{38}$$

with the symmetry $c_{-\ell} = c_\ell\ \forall \ell$, the conditions $c_0 = 1$, $c_{2\ell} = 0$ for $\ell > 0$ and

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 3^2 & \dots & (2n-1)^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 3^{2n-2} & \dots & (2n-1)^{2n-2} \end{bmatrix} \begin{pmatrix} c_1 \\ c_3 \\ \vdots \\ c_{2n-1} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{39}$$

the solution to this Vandermonde matrix system of equations is given by:

$$c_{2\ell+1} = \frac{1}{2} \prod_{j=0, j\neq\ell}^{n-1} \frac{(2j+1)^2}{(2j+1)^2 - (2\ell+1)^2} = \frac{n}{2^{4n-3}} C_{2n-1}^{n-1} \frac{(-1)^\ell}{2\ell+1} C_{2n-1}^{n+\ell}. \tag{40}$$

Then the wavelet is given by any function ψ such that

$$\psi_{2n}(\frac{x}{2}) = \varphi_{2n}(x-1) + \sum_k \alpha_k \varphi_{2n}(\frac{x}{2} - k), \tag{41}$$

with the reals $(\alpha_k)_k$ well chosen with respect to the desired properties of the wavelet: symmetry, zero-moments, small compact support. The choice $\alpha_k = 0$ for all k provides a hierarchical basis.