Solving the guiding-center model on a regular hexagonal mesh

Michel Mehrenberger, Laura S. Mendoza, Charles Prouveur, Eric Sonnendrücker

ESAIM: PROCEEDINGS AND SURVEYS, Vol. ?, 2015, 1-10
Editors: Will be set by the publisher

SOLVING THE GUIDING-CENTER MODEL ON A REGULAR HEXAGONAL MESH

Michel Mehrenberger[1], Laura S. Mendoza[2,3], Charles Prouveur[4] and Eric Sonnendrücker[2,3]

Abstract. This paper introduces a Semi-Lagrangian solver for the Vlasov-Poisson equations on a regular hexagonal mesh. The latter is composed of equilateral triangles, thus it doesn't contain any singularities, unlike polar meshes. We focus on the guiding-center model, for which we need to develop a Poisson solver for the hexagonal mesh in addition to the Vlasov solver. For the interpolation step of the Semi-Lagrangian scheme, a comparison is made between the use of box-splines and of Hermite finite elements. The code will be adapted to more complex models and geometries in the future.

Résumé. Dans cet article nous présentons un solveur semi-Lagrangien pour les équations de Vlasov-Poisson sur un maillage hexagonal régulier. Ce dernier est composé de triangles équilatéraux, ainsi il ne présente aucune singularité, contrairement au maillage polaire. Nous nous concentrons ici sur le modèle centre-guide. À cette fin nous avons développé en plus du solveur pour Vlasov, un solveur de l'équation de Poisson pour maillage hexagonal. Nous comparons les résultats obtenus avec une interpolation par éléments finis d'Hermite et par des box-splines. Dans l'avenir, ce code sera adapté à des géométries et modèles plus complexes.

Introduction

There are three kinds of regular pavings of the plane: using squares, triangles or hexagons. When considering meshes, the dual mesh of a square mesh is a shifted square mesh and the regular triangle mesh is the dual of the regular hexagonal mesh.

In magnetic fusion applications the embedded magnetic flux surfaces play an important role and introduce an important anisotropy [2]. For this reason one gets favourable numerical properties when grid points align on the concentric magnetic flux surfaces. When trying to do this with a mapped cartesian grid, one ends up with a polar coordinates mesh (when the flux surfaces are circles) or something topologically equivalent. This yields smaller and smaller cells when getting closer to the center as well as a singularity at the center. This is numerically far from optimal.

Different strategies have been implemented to avoid these singularities, we can cite among others: the isoparametric analysis approach done by J. Abiteboul et al. [1] and A. Ratnani [25] or N. Besse and E. Sonnendrücker's work with unstructured meshes [5]. The methods presented in these papers are particularly interesting as not

[1] IRMA, Université de Strasbourg, 7, rue René Descartes, 67084 Strasbourg & INRIA-Nancy Grand-Est, projet TONUS, e-mail: mehrenbe@math.unistra.fr

[2] Max-Planck-Institut für Plasmaphysik, Boltzmannstr. 2, D-85748 Garching, Germany. e-mail: mela@ipp.mpg.de & sonnen@ipp.mpg.de

[3] Technische Universität München, Boltzmannstr. 3, D-85748 Garching, Germany.

[4] Université de Lyon, UMR5208, Institut Camille Jordan, 43 boulevard 11 novembre 1918, F-69622 Villeurbanne cedex, France. e-mail: prouveur@math.univ-lyon1.fr

only they avoid singularities but also they are extremely flexible and can be easily adapted to more complex geometries. However, even if these two approaches are different, they share the limitations due to the numerical complexity, the advection of the derivatives and, the localization of the feet of the characteristics. As for the Poisson equation, a recent study relevant to our problem was made by T. Nguyen et al. [24] to compare different solvers on the disk.

Tiling a regular hexagon into triangles yields a mesh of equilateral triangles having all the same area. Such a mesh was first introduced in [26] for numerical simulations. An application to particle methods is proposed in [8]. This grid can be easily mapped to a circle by slightly stretching the edges of the hexagon. This yields a nice mesh of a disk with slightly stretched triangles of almost the same size and there is no singularity in any point of the domain. Additionally, such a mesh has a structure with three privileged directions, and uniform steps in each direction, thus it is completely straightforward to localize points within this mesh. The derivatives along the three directions can also be nicely computed using regular finite differencing along the three directions. And last but not least, there is a spline construction on this mesh, called box-spline [13]. These splines have a hexagonal support and are invariant by translations along the three directions of the mesh.

The difficulties mentioned for previous approaches, are not present when dealing with this uniform hexagonal mesh. Regarding the computational efficiency, all our simulations include an analysis and comparison with more common methods. Moreover a simple and non singular mapping from this mesh can be used to handle more complex settings like the surface aligned meshes needed for tokamak simulations. This point will be left for further studies.

In this work, we focus on adapting the Semi-Lagrangian scheme to this hexagonal mesh. This scheme consists basically of two steps: computing the characteristics' origins and interpolating at these points. For the latter, we compare two different approaches: one using box-splines and the second approach using Hermite Finite Elements. Both interpolation methods, as well as the mesh, are presented in Section 1. In Section 2, we present a simple finite difference Poisson solver adapted to the hexagonal mesh. We introduce a guiding-center approximation of the 2D Vlasov Poisson system [17], and the Semi-Lagrangian scheme to solve it, in Section 3. Finally, in Section 4, we compare the results of the scheme using box-splines with the ones using Hermite finite elements. Besides the guiding-center model, the circular advection model is used to compare the numerical methods.

1. Interpolation on regular hexagonal mesh

1.1. The hexagonal mesh

The hexagonal mesh is obtained by tiling a regular hexagon into equilateral triangles. The mesh obtained can be generated by three vectors. These unit vectors are

$$\mathbf{r_1} = \begin{pmatrix} \sqrt{3}/2 \\ 1/2 \end{pmatrix} \qquad \mathbf{r_2} = \begin{pmatrix} -\sqrt{3}/2 \\ 1/2 \end{pmatrix} \qquad \mathbf{r_3} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1}$$

The 2D lattice sites are obtained by the product $\mathbf{Rk}$ where $\mathbf{R} = [\mathbf{r_1 r_2}]$ and $\mathbf{k} = [k_1, k_2]^T \in \mathbb{Z}$. To obtain exactly the mesh as in Figure 1, we need to define a few extra parameters: an origin, denoted by $P_0(x_0, y_0)$, a radius L which is the distance between the origin and any external vertex of the hexagon and the number of cells N_c on any radius.

The mesh is based on uniform hexagons of the first type (see [28]). For local and global notations we will use the following convention: the point at the center will be the point of index 0. Following the direction $\mathbf{r_1}$ the next point will be indexed 1, and the notations will follow in a counter-clockwise motion. And so on, until all the points of the domain have been indexed. See Figure 1. We will denote H_i the unit hexagon cell that is centered at the point of global index i.

Besides the fact that the hexagonal mesh contains no singularities, its regularity allows us to localize the characteristics' origins for the Semi-Lagrangian scheme by taking three integer values, similarly to what is done on cartesian grids for which only two integer values are needed. Nevertheless, the accuracy of the method

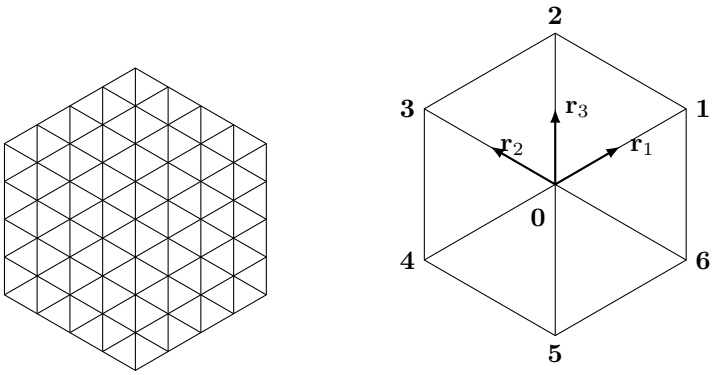

FIGURE 1. The hexagonal lattice and the vectors $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ that generate such a mesh

depends heavily on the interpolation method chosen. For example, for a Cartesian grid, it is common to use cubic splines which have shown to give accurate results in an efficient manner [27]. In our problem, with the hexagonal lattice, B-splines do not exploit the isotropy of the mesh (for more information see [23]) and are defined by a convolution in 2D, which cannot be done for our mesh. Therefore, we need to use another approach. In the following two sub-sections we present two different strategies: one using box-splines and a second one using Hermite Finite Elements.

1.2. **Box-splines quasi-interpolation**

There are mainly two families of splines that take advantage of the geometry's properties: hex-splines and the three directional box-splines. For a detailed comparison between these two types of splines we will refer to [11]. Based on the latter, we have chosen to use box-splines, as the results are more stable. And lastly, also based on the previously cited paper, we decide to use a quasi-interpolation method.

1.2.1. *Box-Splines: General Definition*

Box-splines are a generalisation of the well known B-splines. They are also piecewise polynomial and they share some properties, such as: compact support, positiveness, symmetry and partition of unity. But, unlike B-splines, box-splines are defined from a generator matrix Ξ. Therefore, to construct them on the hexagonal lattice, we will use the generator vectors $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$. The general definition is [10, 14]:

Definition 1.1 (Box-splines)**.** Let Ξ be a $d \times m$ matrix with non-null columns in $\mathbb{R}^d$. A box-spline χ_Ξ associated to the matrix Ξ, is a multivariate function $\chi_\Xi : \mathbb{R}^d \longrightarrow \mathbb{R}$. If Ξ is a square invertible matrix, *i.e.* when $m = d$ and $\det(\Xi) \neq 0$, we define a box-spline with the formula below

$$\chi_\Xi(\mathbf{x}) = \begin{cases} \dfrac{1}{|\det(\Xi)|} & \text{if } \Xi^{-1}\mathbf{x} \in [0,1)^2 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

If $\Xi \cup \mathbf{v}$ is a $d \times (m+1)$ matrix, composed by the m columns vectors from Ξ to which we append the vector $\mathbf{v}$, we define the box-spline $\chi_{\Xi \cup \mathbf{v}}$ by recursion:

$$\chi_{\Xi \cup \mathbf{v}}(\mathbf{x}) = \int_0^1 \chi_\Xi(\mathbf{x} - t\mathbf{v})\, dt \tag{3}$$

Remark 1.2. The box-splines can have different degrees in each direction. Thus, there are different definitions of the degree. We will adopt the definition below.

Definition 1.3 (Degree of a Box-spline)**.** Let Ξ be a $d \times d$ matrix with non-null columns in $\mathbb{R}^d$ such that the column vectors of Ξ form a generating basis of $\mathbb{R}^d$ and are linearly independent. Then, the box-spline of degree N of generating matrix Ξ, χ^N_Ξ, is the box-spline associated to Ξ, where all the generating vectors have multiplicity $N + 1$.

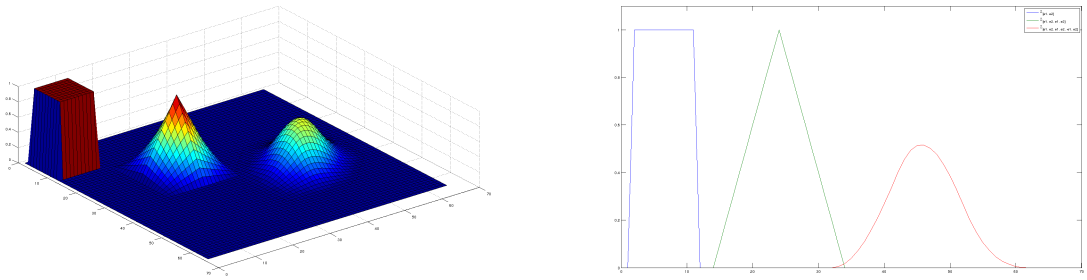

FIGURE 2. On the left: Box-splines representation of $\Xi_[e_1, e_2]$, $\Xi_[e_1, e_2, e_3]$, and $\Xi_[e_1, e_2, e_3, e_4]$. Where $e_1 = (0, 1)^T$, $e_2 = (1, 0)^T$, $e_3 = e_1 + e_2$, and $e_4 = e_1 - e_2$. On the right: the 2d projection of the box-splines onto the x plane.

1.2.2. *The quasi-interpolation scheme*

Let us describe the method: we are given an initial sample $s[\mathbf{k}] = f_0(\mathbf{Rk})$, where the points $\mathbf{Rk}$ belong to our hexagonal mesh, and we need to know the values $f(\mathbf{x})$ where $\mathbf{x} \notin \mathbf{Rk}$. We want a spline surface $f(\mathbf{x}) = \sum c[\mathbf{k}]\chi^N(\mathbf{x} - \mathbf{Rk})$, where χ^N are the box-splines of degree N of matrix $\mathbf{R}$ and $c[\mathbf{k}]$ are the coefficients associated to them. The reconstruction is defined such that $f(\mathbf{x})$ approximates $f_0(x)$ to a certain order $M = 2N$ or, in other words, the approximation is exact only if $f_0(x)$ is a polynomial of degree $M - 1$ or less. This is different from the classical interpolation method, where the reconstruction is exact on grid points for all smooth functions. The $c[\mathbf{k}]$ coefficients are the box-splines coefficients, to compute them we cannot longer solve a matrix-vector system because of the extra degree of freedom given by the quasi-interpolation method. Thus, the $c[\mathbf{k}]$ coefficients are obtained by discrete filtering [12]

$$c = s * p \tag{4}$$

where $*$ is the convolution operator, s is the initial sample data and p is a pre-filter which will be defined later on.

1.2.3. *Box-splines coefficients*

How we determine the splines coefficients is almost as important as the splines themselves. We recall we have the formula (4). Based on the literature available (notably [11]) we have chosen for second-order box-splines the quasi-interpolation pre-filters p_{IIR2} which seem to give better results within a competitive time. The pre-filter $p_{IIR2}[i]$ of the point of local index i, for splines of order 2, is defined as follows:

$$p_{IIR2}[i] = \begin{cases} 1775/2304, & \text{if } i = 0 \\ 253/6912, & \text{if } 0 < i < 7 \\ 1/13824, & \text{if } 6 < i < 19 \text{ and } i \text{ odd} \\ 11/6912, & \text{if } 6 < i < 19 \text{ and } i \text{ even} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

For higher orders, we refer to the previously mentioned papers.

1.2.4. *Optimizing the evaluation*

At the present state we have all the elements for the approximation of a function f with box-splines of degree N

$$\tilde{f}(\mathbf{x}) = \sum_{\mathbf{k}\in\mathbb{Z}^2} c[\mathbf{k}]\chi^N(\mathbf{x} - \mathbf{R}\mathbf{k}) \tag{6}$$

Even if we limit our sum to the vector $\mathbf{k}$ that defines our domain, we would like to take advantage of the fact that the splines χ^N are only non-zeros in a limited number of points. Therefore we need to know the indices $\mathbf{k}$ such that $\chi^N(\mathbf{x} - \mathbf{R}\mathbf{x}) \neq 0$. For this purpose we will use the strategy suggested in [11]: to start we need to obtain the indices on the coordinate system generated by $\mathbf{R}$: $\mathbf{k}_0 = [\lfloor u \rfloor \ \lfloor v \rfloor]$ where $[u\ v]^T = \mathbf{R}^{-1}\mathbf{x}$. Thus, for example, in the case $N = 1$, we only need 4 terms associated to the encapsulating rhomboid's vertices: $\mathbf{R}\mathbf{k}_0$, $\mathbf{R}\mathbf{k}_0 + \mathbf{r_1}$, $\mathbf{R}\mathbf{k}_0 + \mathbf{r_2}$ and $\mathbf{R}\mathbf{k}_0 + \mathbf{r_1} + \mathbf{r_2}$. Finally we obtain:

$$\begin{aligned}\tilde{f}(\mathbf{x}) = \quad & c[\mathbf{k}_0]\ \chi^1(\mathbf{x} - \mathbf{R}\mathbf{k}_0) \\ & + c[\mathbf{k}_0 + [1,0]]\ \chi^1(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{r_1}) \\ & + c[\mathbf{k}_0 + [0,1]]\ \chi^1(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{r_2}) \\ & + c[\mathbf{k}_0 + [1,1]]\ \chi^1(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{r_1} - \mathbf{r_2})\end{aligned} \tag{7}$$

Remark 1.4. As the χ^1 spline has a support of radius the unity, one of the elements of (7) is null. But this formula allows us to keep a short general formula for all points on the mesh without having to compute the indices of the cell to which $\mathbf{x}$ belongs to.

Remark 1.5. For the box-splines of degree 2, there are 12 coefficients to compute (see Figure 3). Numerically, we can take the encapsulating rhomboid of edge length 3 (instead of 1). Thus, the number of null terms for the interpolation with χ^2 is 4.

1.3. Hermite Finite Elements interpolation

Another type of interpolation method is the Hermite Finite Element interpolation. In which, to interpolate at a point $\mathbf{x}$ of barycentric coordinates $(\lambda_1, \lambda_2, \lambda_3)$ in the triangle T of vertices S_1, S_2, and S_3, we need a finite element with a local interpolation operator Π_T. This operator can be defined with the product of a set of degrees of freedom Σ_T with a set of basis functions Ξ which depends on the barycentric coordinates.
For this part we define the indices i, j and k with the following relations:

$$j = i[3] + 1\ ,\ k = j[3] + 1.$$

Here $i[3]$ (respectively $j[3]$) is the rest of the euclidean division of i (resp. j) by 3.

Several elements have been tested here: The Z9 and Z10 Zienkiewicz elements, the Hsieh-Clough-Tocher reduced (HCT-r) and complete (HCT-c), and the Ganev-Dimitrov element. These elements can be found in [19] and [4] . Here we show specifically how the hexagonal structure simplifies the interpolation with these elements.

1.3.1. *The Z9 and Z10 Zienkiewicz elements*

Z9 approach uses 9 degrees of freedom which are the values at the vertices of the triangle and the values of the derivatives in the direction of the edges at every vertex.

$$\Sigma_T = \{\forall i \in [|1;3|], f(S_i), \partial_x(S_i), \partial_v(S_i)\}$$

Z10 uses one more degree of freedom which is the value at the center of the triangle:

$$\Sigma_T = \{\forall i \in [|1;3|], f(S_i), \partial_x(S_i), \partial_v(S_i); f(C)\}$$

The advantage of using the Z10 element is the gain of one order of precision: it reproduces polynomials of total degree ≤ 3 whereas with Z9 only polynomials of degree ≤ 2 are reproduced. Let us note that although adding one degree of freedom seems harmless, adding the cells' center points represents, for an hexagonal mesh, a computational cost three times higher. Indeed there are twice as many centers as vertices in an hexagonal mesh. Therefore the number of computational points is tripled.

Let us define the basis functions needed to interpolate with the element Z9:

$$\phi = \lambda_1\lambda_2\lambda_3$$

$$\xi_i = \lambda_i^3 - \phi \text{ and } \xi_{ij} = \lambda_i^2\lambda_j + \frac{\phi}{2}$$
$$\phi_i = 3\lambda_i^2 - 2\xi_i \text{ and } \phi_{ij} = h_{ij}\xi_{ij} = h\xi_{ij}$$

ϕ_i is the basis function associated with the value of the function at S_i while ϕ_{ij} are associated with the derivatives in the direction of the edges. The fact that T is equilateral is exploited here by replacing h_{ij} with h since the length of $[S_iS_j]$ is constant. Finally for Z9 we have:

$$\Pi(X) = \sum_{i=1}^{3}[f(S_i).\phi_i + \sum_{j\neq i}\frac{\partial f(S_i)}{\partial\overrightarrow{S_iS_j}}.\phi_{ij}]$$

In the same manner, let us define the basis functions needed to interpolate with Z10:

$$\phi_i = 3\lambda_i^2 - 2\xi_i - 9\phi$$

$$\phi_{ij} = h_{ij}(\xi_{ij} - \frac{3}{2}\phi) = h(\xi_{ij} - \frac{3}{2}\phi)$$
$$\phi_{123} = 27\phi$$

therefore for $Z10$ we have:

$$\Pi(X) = \sum_{i=1}^{3}[f(S_i).\phi_i + \sum_{j\neq i}\frac{\partial f(S_i)}{\partial\overrightarrow{S_iS_j}}.\phi_{ij}] + f(C).\phi_{123}$$

1.3.2. *The HCT elements*

The HCT elements were tested as well because of their original feature which is to use a division of the triangle into three sub-triangles. This characteristic is the only difference between the interpolation with the HCT-r and the Z9 element as they both use the same 9 degrees of freedom. Unsurprisingly, they give quasi-identical results which is why we won't detail the interpolation with HCT-r and focus on HCT-c.

The HCT-c element uses the same degrees of freedom as HCT-r plus the values of the derivatives in the normal direction of the edges at the middle of the respective edge, which adds up to twelve degrees of freedom. Let us now define its interpolation operator.

Let S_i be a vertex of the triangle T, then we define respectively l_i and m_i as the length and the middle of the edge opposite to S_i. Let G be the barycenter of T, then K_l is the sub-triangle made with G, S_j and S_k.

$$\Pi_{K_l}(X) = \sum_{i=l}^{3}[f(S_i).\phi_i + \frac{\partial f(S_i)}{\partial\overrightarrow{S_iS_j}}.\phi_{ij} - ni\frac{\partial f}{\partial\nu_i}(m_i)\Phi_{\perp,l,i}].$$

The basis functions are defined by

$$\Xi_l = \Sigma_l \Lambda_l,$$

with

$$\Xi_l = (\Psi^0_{l,i}, \Psi^0_{l,j}, \Psi^0_{l,k}, \Psi^1_{l,i,k}, \Psi^1_{l,i,j}, \Psi^1_{l,j,i}, \Psi^1_{l,j,k}, \Psi^1_{l,k,j}, \Psi^1_{l,k,i})^T,$$
$$\Lambda_l = (\lambda_i^3, \lambda_j^3, \lambda_k^3, \lambda_i^2\lambda_k, \lambda_i^2\lambda_j, \lambda_j^2\lambda_i, \lambda_j^2\lambda_k, \lambda_k^2\lambda_j, \lambda_k^2\lambda_i, \lambda_i\lambda_j\lambda_k)^T.$$

The matrices Σ_l are defined with the eccentricity of each edge of the triangle T:

$$j = i[3] + 1 \ , \ k = j[3] + 1 \ , \ e_i = \frac{l_k^2 - l_j^2}{l_i^2}.$$

For an equilateral triangle, the eccentricity is null which simplifies a lot Σ_l.

$$\Sigma_l = \begin{pmatrix}
0 & 0 & 0 & \frac{9}{2} & \frac{9}{2} & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 1 & 0 & \frac{-3}{2} & 0 & 3 & 3 & 0 & 0 & 3 \\
\frac{1}{2} & 0 & 1 & 0 & \frac{-3}{2} & 0 & 0 & 3 & 3 & 3 \\
\frac{-1}{12} & 0 & 0 & \frac{5}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{12} & 0 & 0 & \frac{1}{2} & \frac{5}{4} & 0 & 0 & 0 & 0 & 0 \\
\frac{-7}{12} & 0 & 0 & \frac{-1}{2} & \frac{-1}{4} & 1 & 0 & 0 & 0 & 1 \\
\frac{2}{3} & 0 & 0 & \frac{-1}{4} & \frac{1}{4} & 0 & 1 & 0 & 0 & \frac{1}{2} \\
\frac{2}{3} & 0 & 0 & \frac{1}{4} & \frac{-1}{4} & 0 & 0 & 1 & 0 & \frac{1}{2} \\
\frac{-7}{12} & 0 & 0 & \frac{1}{4} & \frac{-1}{4} & 0 & 0 & 1 & 0 & \frac{1}{2} \\
\frac{2}{3} & 0 & 0 & \frac{1}{4} & \frac{-1}{4} & 0 & 0 & 1 & 0 & \frac{1}{2} \\
\frac{-7}{12} & 0 & 0 & \frac{1}{4} & \frac{-1}{4} & 0 & 0 & 1 & 0 & \frac{1}{2} \\
\frac{1}{4} & 0 & 0 & \frac{-1}{4} & \frac{-1}{2} & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

The advantage of HCT-c and HCT-r is that they don't require more points than the lattices of the hexagonal mesh. This is not the case for the the Z10 approach.

1.3.3. *The Ganev-Dimitrov element*

The Ganev-Dimitrov element reproduces polynomials of total degree ≤ 4 and uses 15 degrees of freedom which are the values of the function at the vertices and at the middle of the edges, plus the value of the derivatives at the vertices in the direction of the other two vertices. The computational cost for this element is four times higher than the HCT-r interpolation because of the computations needed at the middle of the edges: there are on average 3 times more edges than vertices. As a matter of fact, the vertices and the middle of the edges form another hexagonal mesh twice as fine as the original mesh. The reason why we tested such a computationally expensive element is to observe whether or not the gain in precision is interesting compared to the extra computing time allocated.
The local interpolation operator is:

$$\Pi_{K_l}(X) = \sum_{i=l}^{3} [f(S_i).\Psi_i + \frac{\partial f(S_i)}{\partial \overrightarrow{S_i S_j}}.\Psi_{ij} + f(m_i).\Psi_i^{\perp,0} - ni\frac{\partial f}{\partial \nu_i}(m_i)\Psi_i^{\perp,1}]$$

The basis function are defined by:

$$\Xi = \Sigma\Lambda,$$

with:

$$\Xi = (\Psi_1, \Psi_2, \Psi_3, \Psi_{1,3}, \Psi_{1,2}, \Psi_{2,1}, \Psi_{2,3}, \Psi_{3,2}, \Psi_{3,1}, \Psi_1^{\perp,0}, \Psi_2^{\perp,0}, \Psi_3^{\perp,0}, \Psi_1^{\perp,1}, \Psi_2^{\perp,1}, \Psi_3^{\perp,1})^T,$$
$$\Lambda = (\lambda_1^4, \lambda_2^4, \lambda_3^4, \lambda_1^3\lambda_3, \lambda_1^3\lambda_2, \lambda_2^3\lambda_1, \lambda_2^3\lambda_3, \lambda_3^3\lambda_2, \lambda_3^3\lambda_1, \lambda_2^2\lambda_3^2, \lambda_3^2\lambda_1^2, \lambda_1^2\lambda_2^2, \lambda_1^2\lambda_2\lambda_3, \lambda_1\lambda_2^2\lambda_3, \lambda_1\lambda_2\lambda_3^2).$$

$$\Sigma = \begin{pmatrix}
1 & 0 & 0 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & -5 & -5 & -4 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 4 & 4 & 0 & 0 & -5 & 0 & -5 & 0 & -4 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 4 & 4 & -5 & -5 & 0 & 0 & 0 & -4 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{-1}{2} & \frac{-1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & \frac{-1}{2} & \frac{-1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & \frac{-1}{2} & \frac{-1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & \frac{-1}{2} & \frac{-1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & \frac{-1}{2} & \frac{1}{2} & \frac{-1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & \frac{1}{2} & \frac{-1}{2} & \frac{-1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & -16 & 16 & 16 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 16 & -16 & 16 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & -16 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 4 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & -4 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & -4
\end{pmatrix}$$

Remark 1.6. Here the derivatives are not computed exactly. They are approximated with finite differences of order 6 with the values at the mesh points. Results have been found to be better with this order and no significant improvement was noticed with a higher order difference scheme.

2. The Poisson finite-difference solver

When computing the origins of the characteristics with the semi-Lagrangian method for the Vlasov-Poisson or guiding center models we need to compute the solution of the Poisson equation

$$-\Delta\phi = \rho,$$

ϕ being the potential and ρ the density. In order to solve this equation, we use a simple a finite differences scheme. Since the mesh here is hexagonal, a seven point stencil is used as shown in Figure 1. It is composed of the six vertices of an hexagon plus its center. To compute ϕ_0, the value of ϕ at the center 0, the remaining vertices of the hexagon are used. This particular stencil has the property to give a fourth order scheme at little cost [7]. Here is the previously described scheme:

$$-(\phi_1 + \phi_2 + \phi_3 + \phi_4 + \phi_5 + \phi_6 - 6\phi_0) = \frac{3h^2}{4}\rho_0 + \frac{h^2}{24}(\rho_1 + \rho_2 + \rho_3 + \rho_4 + \rho_5 + \rho_6).$$

Compared to the second order scheme on the same stencil, we notice the only difference to be the second term of the equality:

$$-(\phi_1 + \phi_2 + \phi_3 + \phi_4 + \phi_5 + \phi_6 - 6\phi_0) = h^2\rho_0.$$

Considering the gain of two order of precision at such little cost, we have used this fourth order scheme to compute ϕ.

Remark 2.1. One difficulty that arises here is to define an indexing that allows the resolution of a "computational-friendly" linear system, i.e. a sparse matrix with the non-null terms close to the diagonal to minimise filling in a Cholesky decomposition. This is done by assigning a number following one hexagonal direction, row after row, similarly to how one proceeds on a Cartesian mesh. Here however the difference is that the rows are of variable width resulting in a banded matrix. Therefore the matrix here is not constituted of 7 diagonals which makes the Poisson computation longer than on a Cartesian mesh. The width of the band is directly proportional to the number of cells in the hexagonal domain.

3. The Backward Semi-Lagrangian Scheme

When solving a Vlasov equation, one usually thinks of Lagrangian methods such as PIC [6]. However these schemes are prone to numerical noise and converge slowly in $1/\sqrt{N}$ as the number of particles increases, typical of a Monte Carlo integration. Another option to solve the Vlasov equation, are Eulerian methods like Finite Difference, Finite Element or Finite Volume methods [3, 15, 29]. The downside of this type of method is that there is a numerical limit on the time step

With the intent of overcoming the pitfalls of these methods, the Semi-Lagrangian method was introduced, first in numerical weather prediction (see [20] and articles cited within it), and then for plasma simulations [9, 27] and is used also for gyrokinetic simulations of plasma turbulence [18, 21]. This scheme consists in fixing a Eulerian grid in phase-space and following the trajectory of the equation's characteristics in time to compute the evolution of the distribution function. The advantages of this scheme are the possibility of taking large time steps and its stability. However it is still quite costly in high dimensions (5 or 6D phase space) where PIC method still largely dominate. Lastly, we can point out that there are many types of Semi-Lagrangian solvers (*e.g.* depending on the trajectories: Backward or Forward; depending on degrees of freedom on which is based: grid points, cell average, ...). We have chosen here to use the classical Backward Semi-Lagrangian (BSL) method.

3.1. Our model

We consider here a 2D linear or non linear advection equation, with a divergence free advection field $\mathbf{A}$, which can be written in general form

$$\frac{\partial \rho}{\partial t} + \mathbf{A} \cdot \nabla_{\mathbf{x}} \rho(\mathbf{x}, t) = 0 \tag{8}$$

where $\mathbf{A}$ is divergence free (*i.e.* $\nabla \cdot \mathbf{A} = 0$) and the density ρ is known at the initial time (*i.e.* $\rho(\mathbf{x}, 0) = \rho_0(\mathbf{x})$ is known). The advection field $\mathbf{A}$ will either be given and known for all times or it will depend on an electric potential computed from the solution of a Poisson equation, *i.e.*

$$\mathbf{A} = \begin{pmatrix} -\frac{\partial \phi}{\partial y} \\ \frac{\partial \phi}{\partial x} \end{pmatrix}, \quad \text{with} - \Delta \phi = \rho.$$

This model is known as the guiding center model.

We will apply the backward Semi-Lagrangian scheme to solve both the advection in a given field and the guiding center model.

3.2. Computing the origin of the characteristics

We consider the model (8) on a 2D hexagonal domain, discretized with the hexagonal mesh. The points of the lattice are denoted $\mathbf{x} = (x_1, x_2)$. The distribution function $\rho(\mathbf{x}, t)$ is known on all grid points at the initial time $t = 0$. Let A_{x_1} and A_{x_2} be respectively the first and second components of $\mathbf{A}$. We proceed to apply the BSL method to the Vlasov equation (8): First, we need to compute the origin of the characteristics ending at the grid points. These are defined for a given time $s \in \mathbb{R}$ by

$$\left\{ \begin{array}{rcl} \dfrac{d\mathbf{X}}{dt} & = & \mathbf{A} \\ \mathbf{X}(s) & = & \mathbf{x} \end{array} \right. \iff \left\{ \begin{array}{rcll} \dfrac{d\mathbf{X_1}}{dt} & = & A_{x_1} & \\ \dfrac{d\mathbf{X_2}}{dt} & = & A_{x_2} & \\ X_1(s) & = & x_1, & X_2(s) = x_2 \end{array} \right. \tag{9}$$

The solutions (X_1, X_2) of (9) are called the characteristics associated to the Vlasov equation. Now denoting by $t^n = n\Delta t$, for a given time step Δt, and $\mathbf{X}^n = \mathbf{X}(t^n)$ for any n, and setting $s = t^{n+1}$. The origin, at time t^n, $\mathbf{X}^n$ of the characteristics ending at the grid point $\mathbf{X}^{n+1} = \mathbf{x}$ can then be computed by any ODE solver, typically a

Runge-Kutta solver if $\mathbf{A}$ is known for all times. In the case of the guiding center model we use a second order scheme which is the implicit Adams-Moulton scheme of order two to compute the origin of the characteristics.

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t}=\frac{1}{2}\,(\mathbf{A}^{n+1}+\mathbf{A}^n).$$

The difficulty here is that $\mathbf{A}(t^{n+1},\mathbf{X}^{n+1})$, depends on ρ^{n+1} and is unknown, thus an approximation $\overset{*}{\mathbf{A}}$ of $\mathbf{A}$ at time t^{n+1} is made thanks to previous computations:

$$\overset{*}{\mathbf{A}}=2\;\mathbf{A}(t^n,\mathbf{X}^{n+1})-\mathbf{A}(t^{n-1},\mathbf{X}^{n+1}).$$

The unknown $\mathbf{X}^n$ is found by solving:

$$\begin{cases}\dfrac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t}=\dfrac{1}{2}\,(\overset{*}{\mathbf{A}}+\mathbf{A}^n),\\ \mathbf{X}^{n+1}=\mathbf{x}_i.\end{cases}$$

Remark 3.1. Since we need $\mathbf{A}(t^{n-1})$, the first step is done using the implicit Euler time scheme.

3.3. Updating the distribution function

We know that the density ρ is conserved along these characteristics and therefore we can write for any time t:

$$\rho(\mathbf{X}(t),t)=\rho(\mathbf{X}(s),s)=\rho(\mathbf{x},s). \tag{10}$$

So in our case, knowing the origin $\mathbf{X}^n$ of the characteristics, the new value of ρ at t^{n+1} is given by

$$\rho^{n+1}(\mathbf{x})=\rho^{n+1}(\mathbf{X}^{n+1})=\rho^n(\mathbf{X}^n) \tag{11}$$

where ρ^n is the distribution function at time step t^n.

The distribution function ρ^n is only known on the mesh points, and the origins of the characteristics $\mathbf{X}^n$ are in general not on a mesh point (see figure 3). Therefore, we need an interpolation method to compute ρ^n at the characteristic's origin, *i.e.* to approximate $\rho^n(\mathbf{X}^n)$ needed in the equation (11) to get the new value $\rho^{n+1}(\mathbf{x})$ at the grid points, using the the known data on the mesh points at its vicinity.

3.4. Localizing the characteristics' origins

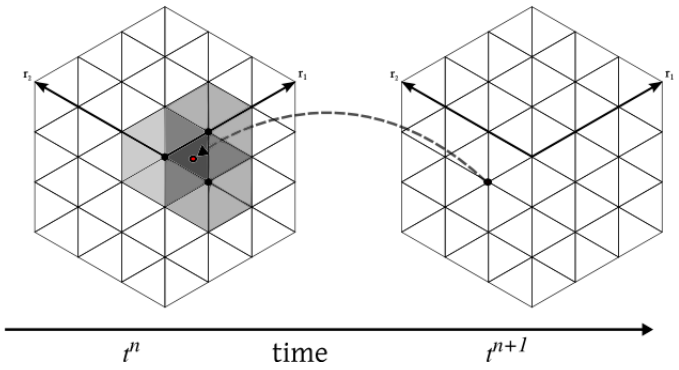

FIGURE 3. Semi-Lagrangian step: Tracing back characteristics.

One of the advantages of the hexagonal mesh is that it is a uniform mesh. Indeed, even if the mesh is not Cartesian, localizing the characteristics' origin is computationally very efficient, unlike the case of unstructured mesh where iterations are generally required. The procedure is as follows. Let (X_1, X_2) the Cartesian coordinates of the characteristics' origin, obtained by solving (9). Then to obtain the hexagonal coordinates (k_1, k_2) of the lowest point of the rhomboid encapsulating the point, we simply need to solve the system $\mathbf{x} = \mathbf{R}\mathbf{k}$, where $\mathbf{R}$ is the matrix whose columns are the unit vectors given in (1), and take the integer value. Denoting by (r_{ij}) the coefficients of the matrix $\mathbf{R}$, we get

$$\left\{ \begin{array}{rcccl} k_1 & = & \left\lfloor \dfrac{r_{22}X_1 - r_{12}X_2}{r_{11}r_{22} - r_{12}r_{21}} \right\rfloor & = & \left\lfloor \dfrac{1}{\sqrt{3}}X_1 + X_2 \right\rfloor \\ k_2 & = & \left\lfloor \dfrac{-r_{21}X_1 + r_{11}X_2}{r_{11}r_{22} - r_{12}r_{21}} \right\rfloor & = & \left\lfloor -\dfrac{1}{\sqrt{3}}X_1 + X_2 \right\rfloor \end{array} \right. \tag{12}$$

After obtaining (k_1, k_2), we know the rhomboid (composed by two cells) containing the characteristics' origin. To determine the exact cell on which the origin is located, we only need to verify if the abscissa of the point is greater than the abscissa of the mesh point at (k_1, k_2) or not. In the first case the point belongs to the cell at the right, else to the cell at the left.

3.5. General algorithm

Below, we summarize the full algorithm to compute the distribution function ρ^{n+1} solution of the non-constant advection equation (8).

Initialization: At time $t = 0$, we suppose that $\rho(\mathbf{x}, 0)$ is given and evaluate it at the grid points.

Time Loop: Incrementation of a given time step Δt, such that: $t^{n+1} = t^n + \Delta t$

- Compute the characteristics' origins using an ODE solver for (9), Runge-Kutta or Adams-Moulton as described above;
- Interpolate the distribution function ρ^n on that point using the mesh points in the vicinity;
- Update the known values: $\rho^n = \rho^{n+1}$.

Remark: Boundary conditions will need to be used between the first and the second steps of the time loop (*i.e.* before the interpolation step) for characteristics that leave the computational domain. In this paper we focus only on null Dirichlet boundary conditions.

4. Numerical results

In this section we present the numerical simulations we performed to test our methods. With the aim of studying the convergence, the dissipation, and the efficiency of the schemes, we first study the circular advection test case. To study the accuracy of the results, we compare them to a known solution. Then we proceed to the guiding-center simulation. As there is no model solution for this test case, we study quantities of the system that we know should be conserved.

4.1. Circular advection

We focus here on the circular advection test case. The model is defined by:

$$\partial_t f(x, y, t) + y\, \partial_x f(x, y, t) - x\, \partial_y f(x, y, t) = 0 \tag{13}$$

Since this equation is not coupled to a Poisson model, we can study in detail the differences between the interpolation methods previously presented. Additionally, finding the analytical solution is trivial and therefore we can study the convergence of our schemes. Here, we take a Gaussian pulse as initial distribution function:

$$f_0(x, y) = \exp\left(-\frac{1}{2} \left(\frac{(x - x_c)^2}{\sigma_x^2} + \frac{(y - y_c)^2}{\sigma_y^2} \right) \right), \tag{14}$$

On a hexagonal mesh centered at the origin of radius 8, we take $\sigma_x = \sigma_y = \frac{1}{2\sqrt{2}}$. Let us set here $x_c = 2$ and $y_c = 2$. The distance from the pulse to the limit of the domain makes the boundary effects insignificant, thus we can take a null Dirichlet boundary condition. To study the convergence in space we took $N_c = 20, 40, 60, ..., 160$. We recall that N_c is the number of cells on the radius L. With the maximum time of evaluation, t_{max}, at 6π, we chose to keep a constant CFL at 2.

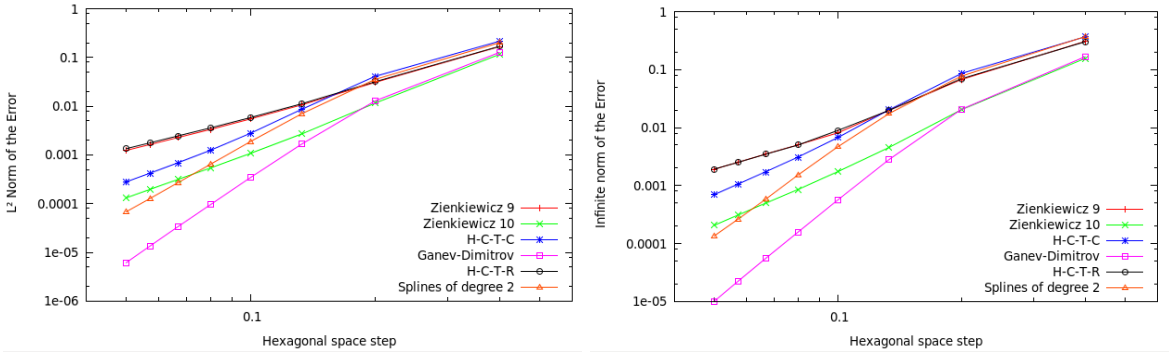

FIGURE 4. Order of convergence

In Figure 4, we plotted the L_2 and L_∞ norms for different space discretizations. We can see that for coarse meshes, all the methods are globally the same, with a slightly better accuracy for elements Z10 and Ganev-Dimitrov. But as the mesh gets finer, we can quickly see that the splines converge quicker to better results. Only the Ganev-Dimitrov elements are more accurate.

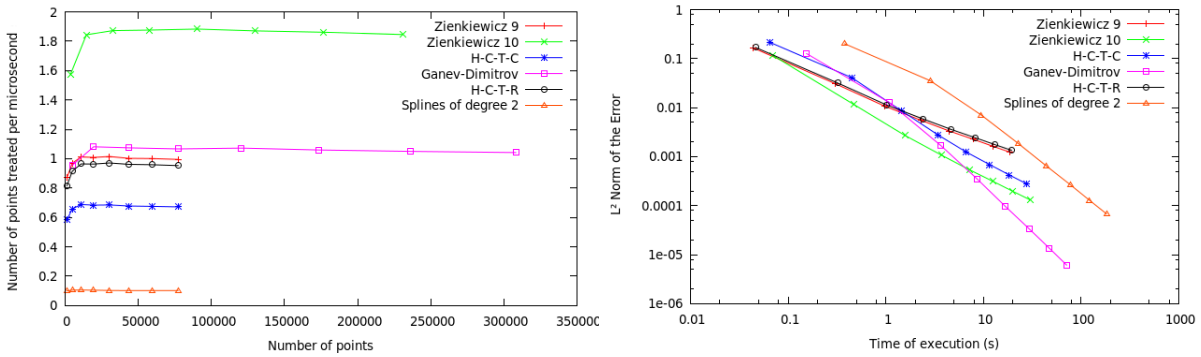

FIGURE 5. Comparison of performances for the circular advection test case.

In Figure 5 we can see that the performance convergences quite quickly, for all the methods. It is also pretty obvious that, even if the splines are more accurate, the cost is higher than most of the Hermite Finite Element methods.

4.2. **Guiding-center model - Diocotron instability test case**

We consider here a guiding-center approximation of the 2D Vlasov-Poisson system. This also corresponds to the reduced gyrokinetic model obtained [16] when all quantities are homogeneous in the direction parallel to

the magnetic field. Here the magnetic field is set to $B = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$. Then the model reads

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + E_\perp \cdot \nabla_{\mathbf{x}} \rho(\mathbf{x}, t) = 0 & \text{(15a)} \\ -\Delta \phi = \nabla \cdot E = \rho(\mathbf{x}, t) & \text{(15b)} \end{cases}$$

with $E = (E_x, E_y) = -\nabla \phi$ and $E_\perp = (-E_y, E_x)$.

By neglecting the effect of boundary conditions, the guiding center model verifies the following properties:

(1) Positivity of density ρ

$$0 \leq \rho(x, y, t).$$

(2) Mass conservation

$$\frac{d}{dt}\left(\int_D \rho \, dxdy\right) = 0.$$

(3) L^p norm conservation, for $1 \leq p \leq \infty$

$$\frac{d}{dt}||\rho||_{L^p(D)} = 0.$$

(4) Energy conservation

$$\frac{d}{dt}\left(\int_D |\nabla \phi|^2 dxdy\right) = 0.$$

This model, is commonly used in 2D simulations to study the particle density, as it describes highly magnetized plasmas in the poloidal plane of a tokamak.

We chose here to study the diocotron instability [22]. The initial density is given by:

$$\rho_0(\mathbf{x}_\perp) = \begin{cases} (1 + \varepsilon \cos(\ell\theta)) \exp\left(-4(r - 6.5)^2\right), & \text{if } r^- \leq \sqrt{x^2 + y^2} \leq r^+, \text{ with } \theta = \operatorname{atan2}(y, x). \\ 0, & \text{otherwise.} \end{cases}$$

As for the parameters, we take $\varepsilon = 0.001$, $r^- = 5$, $r^+ = 8$, $\ell = 6$, $dt = 0.1$ and the hexagonal step is $\frac{14}{160}$ with a radius of 14 and and an hexagonal parameter $N_c = 160$. In this part, we won't test the Z10 approach as it requires a special resolution of the Poisson equation that has not been implemented. Indeed, computing the values of the field at the center of the triangles can't be combined with the resolution at the vertices. Moreover to even the computational time of each method we chose to take $N_c = 80$ for the Ganev Dimitrov element as it results in the computations on a mesh with $N_c = 160$ (see Section 1.3.3).

Let us note that 6 vortices is the main mode. If we take $\ell \neq 6$, with ε small enough, we still see the mode 6. With ε big enough, i.e. at least 0.1, the modes different from 6 can be visible for a time but they are not stable and we see the fusion or the apparition of vortices until there is the sixth mode. For instance, as illustrated by Figure 6, we can see the ninth mode turning into the sixth mode by fusion of vortices. This instability can be explained with Figure 7. The influence of the geometry is clear as the potential is not round, but already deformed as an hexagon. Compared to the results obtained with a polar geometry, our results are different.

Visualisation of the results: The six vortices are developing with time without losing any symmetry. No obvious differences are visible which makes the diagnostics all the more important to compare the results computed.

After comparison of the diagnostics we see that the various interpolation methods give close results overall. They are similar in terms of positivity conservation. We notice that if box-splines conserve better the mass, the Z9 approach conserves better the L^1 norm. Also we note that box-splines and the HCTC element give very near results whichever the diagnostic considered.

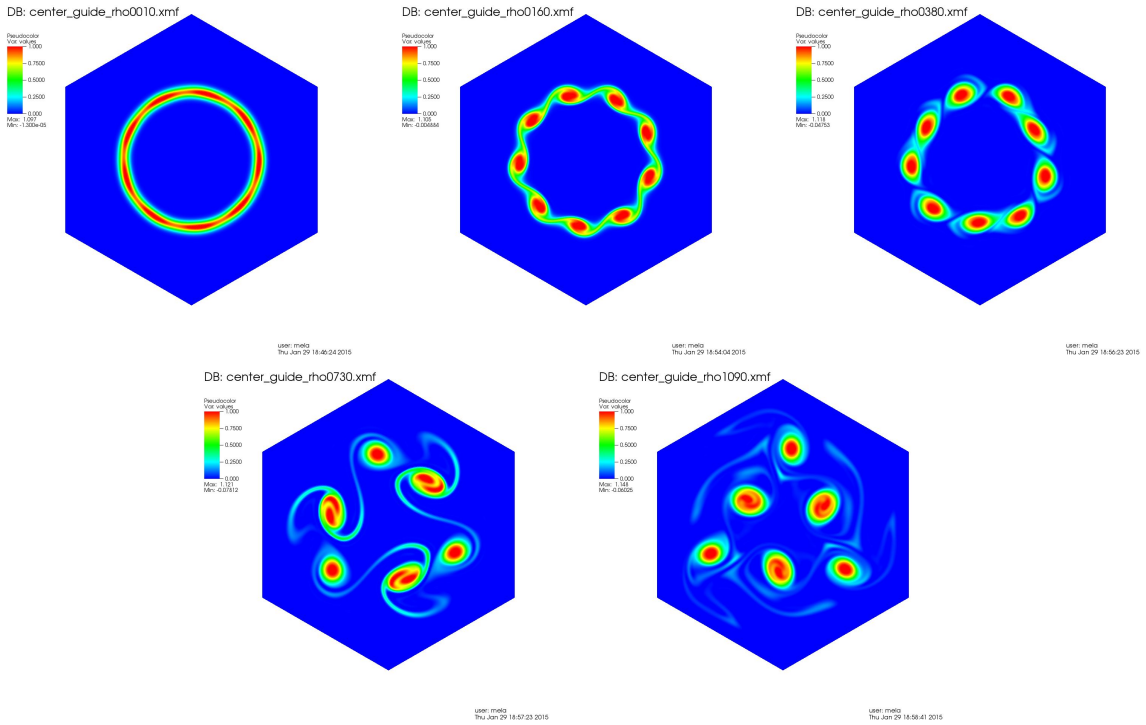

FIGURE 6. Time evolution of the guiding-center model with $\varepsilon = 0.1$, at times = 1, 16, 38, 73 and 109

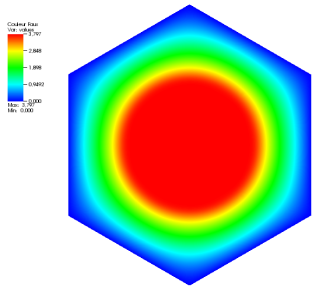

FIGURE 7. Potential at time zero for the guiding-center simulation

5. CONCLUSION

In this paper we tested Semi-Lagrangian schemes adapted to an hexagonal mesh. The strategies differentiated in the type of interpolation used: on the one hand, we developed an interpolation method based on box-splines –spline basis specific to the hexagonal mesh– and on the other hand, we introduced an interpolation method based on Hermite Finite Elements. Furthermore, we presented a Poisson solver based on finite differences on this mesh. The first simulations were made on the circular advection test case. This allowed to compare the order of the methods, as well as their efficiency. The splines approach seemed to be more accurate for finer grids, but at higher cost. Next, we simulated the 2D guiding-center model. The two methods yield comparable

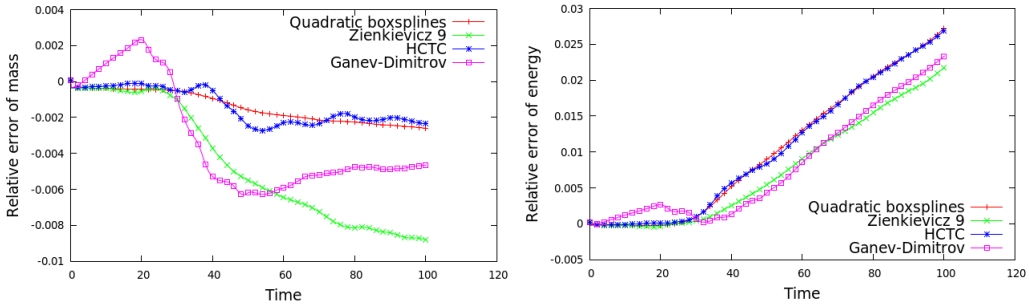

FIGURE 8. Time evolution of the relative error of mass and energy

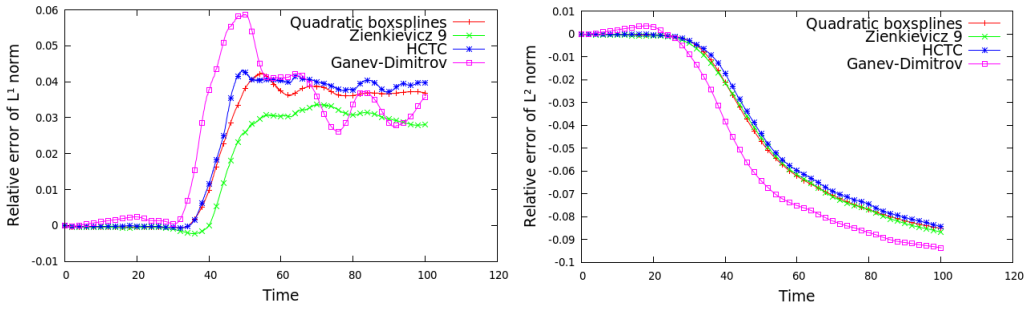

FIGURE 9. Relative error of L^1 and L^2 norms

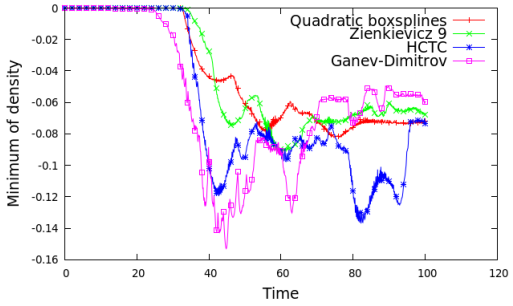

FIGURE 10. Time evolution of the density's minimum

results; more precisely, the HCTC element, in the Hermite case has almost as nice conservation properties as the splines.

REFERENCES

[1] J Abiteboul, G Latu, V Grandgirard, A Ratnani, E Sonnendrücker, and A Strugarek. Solving the vlasov equation in complex geometries. *ESAIM: Proceedings*, 32:103–117, 2011.

[2] P. Angelino, X. Garbet, L. Villard, A. Bottino, S. Jolliet, Ph. Ghendrih, V. Grandgirard, B.F. McMillan, Y. Sarazin, G. Dif-Pradalier, et al. Role of plasma elongation on turbulent transport in magnetically confined plasmas. *Physical review letters*, 102(19):195002, 2009.

[3] Jeffrey William Banks and Jeffrey Alan Furst Hittinger. A new class of nonlinear finite-volume methods for vlasov simulation. *Plasma Science, IEEE Transactions on*, 38(9):2198–2207, 2010.

[4] M. Bernadou. *Methode d'elements finis pour les problemes de coques minces*. Masson editions, 1994.

[5] N. Besse and E. Sonnendrücker. Semi-lagrangian schemes for the vlasov equation on an unstructured mesh of phase space. *Journal of Computational Physics*, 191(2):341 – 376, 2003.
[6] Charles K. Birdsall and A. Bruce. Langdon. *Plasma Physics Via Computer*. McGraw-Hill, Inc., New York, NY, USA, 1985.
[7] E. Carlson, H. Sun, D. Smith, and J. Zhang. Third order accuracy of the 4-point hexagonal net grid finite difference scheme for solving the 2d helmholtz equation, 2003. `http://www.cs.uky.edu/~jzhang/pub/PAPER/hexgrid3.html`.
[8] Philippe Chatelain and Anthony Leonard. Isotropic compact interpolation schemes for particle methods. *Journal of Computational Physics*, 227(6):3244–3259, 2008.
[9] C.Z Cheng and Georg Knorr. The integration of the vlasov equation in configuration space. *Journal of Computational Physics*, 22(3):330 – 351, 1976.
[10] Laurent Condat and Dimitri Van De Ville. Three-directional box-splines: characterization and efficient evaluation. *IEEE Signal Process. Lett.*, 13(7):417–420, 2006.
[11] Laurent Condat and Dimitri Van De Ville. Quasi-interpolating spline models for hexagonally-sampled data. *IEEE, Transactions on Image Processing*, 16(5):1195–1206, May 2007.
[12] Laurent Condat, Dimitri Van De Ville, and Michael Unser. Efficient reconstruction of hexagonally sampled data using three-directional box-splines. In *ICIP*, pages 697–700. IEEE, 2006.
[13] Laurent Condat and Dimitri Van De Ville. New optimized spline functions for interpolation on the hexagonal lattice. In *ICIP*, pages 1256–1259. IEEE, 2008.
[14] Carl de Boor, Klaus Höllig, and Sherman Riemenschneider. *Box Splines*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.
[15] Francis Filbet and Eric Sonnendrücker. Comparison of eulerian vlasov solvers. *Computer Physics Communications*, 150(3):247–266, 2003.
[16] Francis Filbet and Chang Yang. Mixed semi-lagrangian/finite difference methods for plasma simulations, September 2014. https://hal.inria.fr/hal-01068223.
[17] François Golse and Laure Saint-Raymond. L'approximation centre-guide pour l'équation de vlasov-poisson 2d. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 327(10):865 – 870, 1998.
[18] Virginie Grandgirard, Maura Brunetti, Pierre Bertrand, Nicolas Besse, Xavier Garbet, Philippe Ghendrih, Giovanni Manfredi, Yanick Sarazin, Olivier Sauter, Eric Sonnendrücker, et al. A drift-kinetic semi-lagrangian 4d code for ion turbulence simulation. *Journal of Computational Physics*, 217(2):395–423, 2006.
[19] G. Guscaglia and V. Rua. Finite element methods for the stokes system based on a zienkiewicz type n-simplex. *Computer Methods in Applied Mechanics and Engineering*, 272:83–99, 2014.
[20] Eugenia Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge university press, 2003.
[21] Jae-Min Kwon, Dokkyun Yi, Xiangfan Piao, and Philsu Kim. Development of semi-lagrangian gyrokinetic code for full-f turbulence simulation in general tokamak geometry. *Journal of Computational Physics*, 283:518–540, 2015.
[22] Eric Madaule, Sever Adrian Hirstoaga, Michel Mehrenberger, and Jérôme Pétri. Semi-lagrangian simulations of the diocotron instability. Research report, Inria, July 2013. `https://hal.inria.fr/hal-00841504`.
[23] R. M. Mersereau. The processing of hexagonally sampled two-dimensional signals. *Proceedings of the IEEE*, 67(6):930–949, 1979.
[24] Thien Nguyen, Keçstutis Karčiauskas, and Jörg Peters. A comparative study of several classical, discrete differential and isogeometric methods for solving poisson's equation on the disk. *Axioms*, 3(2):280–299, 2014.
[25] Ahmed Ratnani. Isogeometric analysis in plasmas physics and electromagnetism. In *Workshop on Higher Order Finite Element and Isogeometric Methods Program and Book of Abstracts*, page 64, 2011.
[26] Robert Sadourny, Akio Arakawa, and Yale Mintz. Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere. *Monthly Weather Review*, 96(6):351–356, 2014/11/21 1968.
[27] Eric Sonnendrücker, Jean Roche, Pierre Bertrand, and Alain Ghizzo. The semi-lagrangian method for the numerical resolution of the vlasov equation. *Journal of computational physics*, 149(2):201–220, 1999.
[28] Robert Ulichney. *Digital Halftoning*. MIT Press, Cambridge, MA, 1987.
[29] S.I Zaki, L.R.T Gardner, and T.J.M Boyd. A finite element code for the simulation of one-dimensional vlasov plasmas. i. theory. *Journal of Computational Physics*, 79(1):184 – 199, 1988.