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], Charles Prouveur[3] and Eric Sonnendrücker[2]

Abstract. This paper introduces a Semi-Lagrangian solver for the Vlasov-Poisson equation 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: a Vlasov equation, where the parallel velocity and the first two component of the magnetic field have been neglected; coupled with a Poisson equation. A Poisson solver for the hexagonal mesh is developed. For the interpolation step of the Semi-Lagrangian scheme, a comparaison is made between using box-splines and Hermite finite elements. The scheme is successfully adapted to the mesh using both interpolation methods, with different performances. We expect to adapt it to more complex models and geometries.

Résumé. ...

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 [**?**]. 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 cells that are smaller and smaller when getting closer to the center and a singularity at the center. This is numerically far from optimal.

However, tiling a regular hexagon into triangles yields a mesh of equilateral triangles having all the same area, plus the 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 on the domain. Additionally, such a mesh has a clear structure with three privileged directions, thus it is completely straightforward to localise points within this mesh. The derivatives along the three directions can also be nicely computed using finite differencing along uniform lines. And last but not least, there is a spline construction on this mesh, called box spline. These splines have a hexagonal support and are invariant by translations along the three directions of the mesh. Different strategies have been implemented to avoid this

[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, e-mail: `mela@ipp.mpg.de & sonnen@ipp.mpg.de`
[3] ...

singularities, we can cite among others: the isoparametric analysis approach done by J. Abiteboul et al. [?], N. Besse and E. Sonnendrücker's work with non-structured meshes [?].

In this work, we focus on adapting the Semi-Lagrangian scheme to this hexagonal mesh. A simplified Vlasov-Poisson model is considered. This scheme consists basically on two steps: computing the characteristics' origins and interpolating on 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 Poisson solver adapted to the hexagonal mesh. We present the guiding center model, a simplified Vlasov-Poisson system, and the Semi-Lagrangian scheme to solve it, on section 3. Finally, on section 4 we compare the results of the scheme using box splines with the ones using Hermite finite elements, we also compare them to more traditional meshes, such as the polar mesh. Besides the guiding center model, different advection models have been simulated in this section.

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 $\mathbf{R} = [\mathbf{r_1}\mathbf{r_2}]$ 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 $P_0(x_0, y_0)$, a radius L which is the distance between the origin and an external node and the number of cells N on an edge of the exterior hexagon (or in a demi-diagonal).

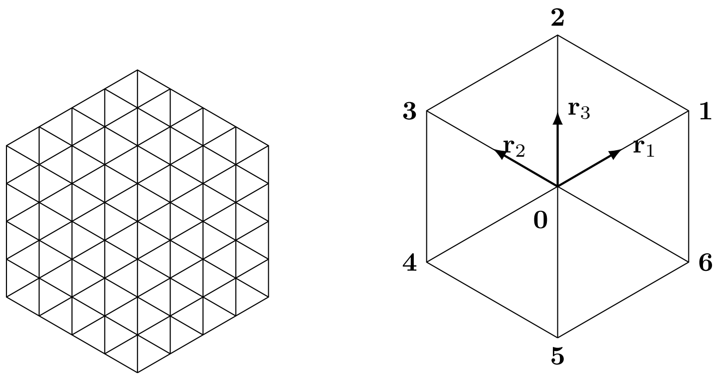

Figure 1. The hexagonal lattice and the vectors $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ that generate such a mesh

The mesh is based on uniform hexagons of the first type (see [?]). 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 centred at the point of global index i.

Besides the fact that the hexagonal mesh contains no singularities, its regularity allow us to localise effortlessly the characteristic's origin for the Semi-Lagrangian scheme. Nevertheless, the accuracy of the method 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 (see [look for a ref]). In our problem, with the hexagonal lattice, B-splines don't exploit the isotropy of the mesh (for more information see [?]) and are defined by a convolution on 2D, which cannot be done for our mesh. Therefore, we need to use another approach. On 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 splines families 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 [?]. 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 more known spline's family: B-splines. They are also piecewise polynomial and they share some properties, as: compact support, positiveness, symmetry and partition of unity. But, on the contrary to 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 [?, ?]:

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

and, for higher orders

$$\chi^N_{\Xi \cup [v]}(\mathbf{x}) = \int_0^1 \chi^{N-1}_\Xi(\mathbf{x} - t\mathbf{v})dt \tag{3}$$

Below, some representation of the first two box-splines.

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 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 N or, in other words, the approximation is exact only if $f_0(x)$ is a polynomial of degree $N-1$ or less. This is different to the classic interpolation method, where the reconstruction is exact on mesh points in any case. 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 [?]

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

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 [?]) we have chosen for second-order box-splines the quasi-interpolation pre-filters p_{IIR2} which seem to give the 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 second order box-splines

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

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^2 are only non-zeros in a limited number of points. Therefore we need to know the indices $\mathbf{k}$ such that $\chi^2(\mathbf{x}-\mathbf{R}\mathbf{x}) \neq 0$. For this purpose we will use the strategy suggested in [**?**] : 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, in our case, with splines χ^2 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 :

$$\begin{aligned} \tilde{f}(\mathbf{x}) = \quad & c[\mathbf{k}_0] \ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0) \\ & + c[\mathbf{k}_0 + [1,0]] \ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{R}_1) \\ & + c[\mathbf{k}_0 + [0,1]] \ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{R}_2) \\ & + c[\mathbf{k}_0 + [1,1]] \ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{R}_1 - \mathbf{R}_2) \end{aligned} \tag{7}$$

Remark 1.1. As the χ^2 spline has a support of radius a unity, one of the elements of (7) is null. But this formula allow us to keep a short general formula for all points on the mesh without having to compute the indices of the cell to which x belongs to.

1.3. **Hermite Finite Elements interpolation**

To interpolate at a point 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.

Several elements have been tested here: The Z2 and Z3 Zienkiewicz elements, the Hsieh-Clough-Tocher reduced (HCT-r) and complete (HCT-c), and the Ganev-Dimitrov element. These elements can be found in [**?**] and [**?**] . Here we show specifically how the hexagonal structure simplifies the interpolation with these elements.

1.3.1. *The Z2 and Z3 Zienkiewicz elements*

Z2 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 vertices:

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

Z3 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 is the interpolation is of order 3 instead of order 2 for Z2. Of course this precision comes with a price : for a hexagonal mesh there are 2 times more centers than vertices. Therefore the number of points which must be computed is tripled.
let us define the basis functions needed to interpolate with the element Z2 :

$$\phi = \lambda_1\lambda_2\lambda_3$$
$$\xi_i = \lambda_i^3 - \phi \text{ and } \xi_{ij} = \lambda_i^2\lambda_j + \frac{\phi}{2}$$
$$\phi_i = 3\lambda_i^2 - 2\xi_i \text{ and } \phi_{ij} = h_{ij}\xi_{ij} = h\xi_{ij}$$

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 Z2 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 Z3:

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

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

therefore for Z_3 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 HTC elements*

The HCT-r element only uses 9 degrees of freedom and gives quasi-identical results compared to Z2. The only difference with Z2 is that HCT-r divides the triangle into 3 sub-triangles, therefore the basis function changes depending on which sub-triangle X is.

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. The local interpolation operator is:

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

The basis function are defined by:

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

with:

$$\Xi_l = (\Psi_{l,i}^0, \Psi_{l,j}^0, \Psi_{l,k}^0, \Psi_{l,i,k}^1, \Psi_{l,i,j}^1, \Psi_{l,j,i}^1, \Psi_{l,j,k}^1, \Psi_{l,k,j}^1, \Psi_{l,k,i}^1)^T,$$

$$\Lambda_l = (\lambda_i^3, \lambda_j^3, \lambda_k^3, \lambda_i^2\lambda_k, \lambda_i^2\lambda_j, \lambda_j^2\lambda_i, \lambda_j^2\lambda_k, \lambda_k^2\lambda_j, \lambda_k^2\lambda_i, \lambda_i\lambda_j\lambda_k)^T,$$

The matrices Σ_l are defined with the excentricity 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 excentricity is null which simplifies a lot Σ_l.

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

The element HCT-c 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.

$$\Pi_{K_l}(X) = \sum_{i=l}^{3} [f(S_i).\phi_i + \frac{\partial f(S_i)}{\partial \overrightarrow{S_i S_j}}.\phi_{ij}] + f(C).\phi_{123}$$

The basis function are defined by

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

with:

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

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

The advantage of HCT-c and HCT-r is that they don't require more points than what is already included into the hexagonal mesh.

1.3.3. *The Ganev-Dimitrov element*

The Ganev-Dimitrov element is of order 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 HTC-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 local interpolation operator is:

$$\Pi_{K_l}(X) = \sum_{i=l}^{3} [f(S_i).\phi_i + \frac{\partial f(S_i)}{\partial \overrightarrow{S_i S_j}}.\phi_{ij}] + f(C).\phi_{123}$$

The basis function are defined by:

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

with:

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

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

Another possible way to interpolate is to use a 2d Hermite finite element [?]
After the root (X, V) of a characteristic is found and the triangle in which it is located has been identified...
There are ten degrees of freedom which are:

- the values at the vertex of the triangle
- the values of the derivatives

- the values at the center of the triangle

$$\begin{cases}\partial_x f(x,y) = \partial_{H_1} f(x,y).\partial_x H_1 + \partial_{H_2} f(x,y).\partial_x H_2 \\ \partial_y f(x,y) = \partial_{H_1} f(x,y).\partial_y H_1 + \partial_{H_2} f(x,y).\partial_y H_2\end{cases} \tag{8}$$

With H_1 et H_2 the hexaedric coordinates. Since

$$\begin{cases}x = \dfrac{H_1 - H_2}{\sqrt{3}}, \\ y = H_1 + H_2,\end{cases} \tag{9}$$

We obtain :

$$\begin{cases}\partial_x H_1 = \dfrac{\sqrt{3}}{2} \ ; \ \partial_y H_1 = \dfrac{1}{2} \\ \partial_x H_2 = \dfrac{-\sqrt{3}}{2} \ ; \ \partial_y H_2 = \dfrac{1}{2}.\end{cases} \tag{10}$$

At the vertexes the values are given and $f(G) = \frac{f(S_1)+f(S_2)+f(S_3)}{3}$. As for the values of the derivatives, we use the finite difference method along the hexagonal directions as it can be seen in figure (?)

2. The Poisson equation

2.1. Finite Differences Solver

Since the mesh here is hexagonal, an seven point stencil is used. It is composed of the six vertices of an hexagon plus its center. This stencil has the particularity to give a fourth order scheme at little cost [?] :

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

Indeed, the only difference with the second order scheme on the same stencil is the second terme:

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

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

2.2. Finite Elements Solver

2.2.1. *Box-spline basis functions*

2.2.2. *Finite-Element basis functions*

3. The Backward Semi-Lagrangian Scheme

When solving a Vlasov equation, one usually think in Lagrangian methods such as PIC. Unfortunately, the drawback of these schemes is that it's vulnerable to numerical noise (see [look for a ref]). Furthermore, this noise is unpredictable as it is the statistical error of a Monte Carlo integration. Looking for other possible schemes to solve the Vlasov equation, one might think of Eulerian methods, where the operators are discretized. The downside of this type of method is the dissipation (see [look for a ref]).

With the intention of neglecting the pitfalls of the method, the Semi-Lagrangian method was introduced, first in numerical weather prediction, but recently it was introduced for plasma simulations [?, ?]. This scheme consists on fixing an 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 important to note that it's very computationally costly to go to high dimensions. Thus, it is typically used with 1D and 2D models. 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 the elements on which is based: grid's point, integrated cell..). We have chosen here to use the classical Backward Semi-Lagrangian method.

3.1. The Guiding Center Model

We consider here a simplified version of the Vlasov-Poisson system, where the parallel velocity has been neglected and the magnetic field is set to $B = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$. Furthermore, we consider the model in 2D in space and 1D in time. The reduced gyrokinetic model obtained is

$$\begin{cases} \dfrac{\partial f}{\partial t} + E_\perp \cdot \nabla_{\mathbf{x}} f(\mathbf{x}, t) = 0 & \text{(11a)} \\ -\Delta\phi = \nabla \cdot E = f(\mathbf{x}, t) & \text{(11b)} \end{cases}$$

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

Explain the importance of this model. Why are we studying this model ?

3.2. General Algorithm

We consider the model (11) 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 $f(\mathbf{x}, t)$ is known on all mesh points at the initial time $t = 0$. In other words, $f_0(\mathbf{x})$ is given. To obtain the initial electric field, we solve (11b) using either poisson solver defined in 2. We proceed to apply the BSL to the Vlasov equation (11a): First, we apply the characteristic method, we obtain for a given $s \in \mathbb{R}$

$$\begin{cases} \dfrac{d\mathbf{X}}{dt} & = & E_\perp \\ \mathbf{X}(s) & = & \mathbf{x} \end{cases} \iff \begin{cases} \dfrac{d\mathbf{X_1}}{dt} & = & -E_y \\ \dfrac{d\mathbf{X_2}}{dt} & = & E_x \\ X_1(s) & = & x_1, \quad X_2(s) = x_2 \end{cases} \tag{12}$$

The solutions (X_1, X_2) of (12), obtained by any ODE solver, are called the characteristics of (12). Besides, we know that the density f is conserved along these characteristics and therefore we can write:

$$f(\mathbf{x}, dt) = f(\mathbf{X}, 0) \tag{13}$$

Or for any given time t^{n+1} when f^n, the distribution function at the previous time, is known

$$f(\mathbf{x}, t^{n+1}) = f^n(\mathbf{X}^{n+1}) \tag{14}$$

The initial distribution function is only known on the mesh points, and the characteristic's origin X^{n+1} are probably not on a mesh point (see figure 3.2). Therefore, we can use any interpolation method to compute f^n on the characteristic's origin, *i.e.* to approximate the solution of the equation (14).

Below, we summarize the full algorithm to compute the distribution function f^{n+1} solution of the guiding center model (11).

Initialization: At time $t = 0$, we suppose that $f(\mathbf{x}, 0)$ is given. We compute the electric field E by solving the poisson equation (11b).

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 (12);
- Interpolate the distribution function f^n on that point using the mesh points in the vicinity;
- Solve the poisson equation to find the electric field E^{n+1};

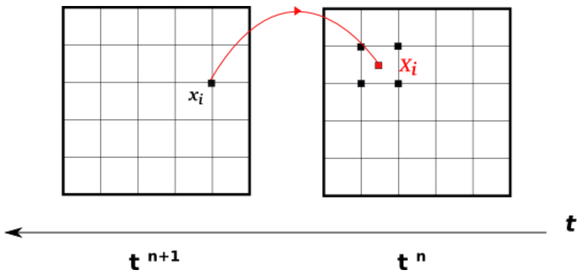

FIGURE 2. Semi-Lagrangian step: Tracing back characteristics.

- Update the known values: $f^n = f^{n+1}$, $E^{n+1} = E^n$.

Remark: the boundary conditions will intervene in between the first and the second steps of the time loop (*i.e.* before the interpolation step). On this paper we focus only on null Dirichlet boundary conditions.

4. Results