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

# SOLVING THE VLASOV-POISSON EQUATION 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 was developed. For the interpolation step of the Semi-Lagrangian scheme, a comparaison was made between using box-splines and Hermite finite elements. The scheme was successfully adapted to 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. Moreover, 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 construct on this mesh, called box spline. These splines have a hexagonal support and are invariant by translations along the three directions of the mesh.

[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] ...

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 feet 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 $\mathbf{R} = [\mathbf{r_1 r_2}]$ 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).

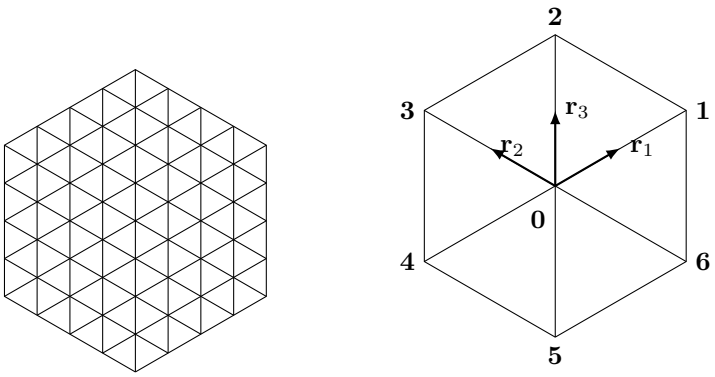


Figure 1. The hexagonal lattice and the vectors $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ that generate such a mesh

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 note $H_i$ 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 feet 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. Moreover, 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 $\Xi$. 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 $\chi^N$ 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 $\chi^2$ 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 $\chi^2$ 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 $\chi^2$ 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**

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**

### 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

### 3.1. **The Guiding Center Model**

### 3.2. **General Algorithm**

## 4. Results

## References