ESAIM: PROCEEDINGS AND SURVEYS, Vol. ?, 2014, 1-10
Editors: Will be set by the publisher

A SEMI-LAGRANGIAN SOLVER FOR THE GUIDING CENTER MODEL ON A REGULAR HEXAGON MESH

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

Résumé. ...

Abstract. ...

Introduction

There are three kinds of regular pavings of the plane: using squares, triangles or hexagon. 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 plays an important role and introduces 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.

On the other hand tiling a regular hexagon into triangles yields a mesh of equilateral triangles having all the same size, and this 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. On the other hand such a mesh has a clear structure with three privileged directions, so that it is completely straightforward to localise points within this mesh. On the other hand derivatives along the three directions can be nicely computed using finite differencing along uniform lines. Moreover 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. Interpolation on regular hexagonal mesh

1.1. The hexagonal mesh

The unit vectors in each direction are

[1] ...
[2] Max-Planck-Institut für Plasmaphysik; e-mail : mela@ipp.mpg.de & sonnen@ipp.mpg.de
[3] ...

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

The 2D lattice sites are thus 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 fance 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).

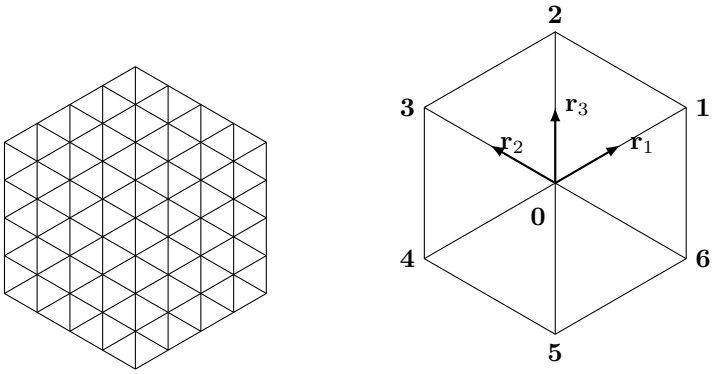

FIGURE 1. The hexagonal lattice and the vectors $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ that generate such a mesh

The mesh is based on uniform hexagons of the first type (see [6]). 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 for the points on the same ring. 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.

1.2. **Box Splines quasi-interpolation**

The accuracy of the Semi-Lagrangian method depends heavily on the interpolation method chosen. For example, for a cartesian grid is common to use cubic splines which have shown to give accurate results in an efficient manner. In our problem, with the hexagonal lattice, B-splines don't exploit the isotropy of the mesh (for more information see [5]) and therefore we need a solution better adapted. 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 [1]. Based on the latter, we have chosen to use box splines.

Let us describe such a model : 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 $f(X, V)$ where $(X, V) \notin \mathbf{Rk}$. We want a spline surface $f(\mathbf{x}) = \sum c[\mathbf{k}]\chi^n(\mathbf{x} - \mathbf{Rk})$ such that $f(\mathbf{x})$ approximates $f_0(x, v)$ and where χ^n are the box-splines of compact hexagonal support and $c[\mathbf{k}]$ are the box-splines coefficients which are obtained by [3]

$$c = s * p \tag{2}$$

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

1.2.1. *Box-Splines: General Definition*

To construct the box-splines we will use the generator vectors $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ of the hexagonal lattice and we will introduce the box-splines basis functions $\varphi_\Xi(\mathbf{x})$ where $\Xi = [\mathbf{v}_1\mathbf{v}_2]$ which are defined as follows ([2,4]):

$$\varphi_\Xi(\mathbf{x}) = \begin{cases} \dfrac{1}{|\det(\Xi)|} & \text{if } \Xi^{-1}\mathbf{x} \in [0,1)^2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and, for higher orders

$$\varphi_{\Xi\cup[v]}(\mathbf{x}) = \int_0^1 \varphi_\Xi(\mathbf{x} - t\mathbf{v})dt \tag{4}$$

1.2.2. *Box splines coefficients*

How we determine the splines coefficients is almost as important as the splines themselves. We recall we have the formula (2). Based on the literature available (notably [1]) 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 :

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

Or for the splines of order 3 :

$$p_{IIR2}[i] = \begin{cases} 244301/460800, & \text{if } i = 0 \\ 42269/576000, & \text{if } 0 < i < 7 \\ -11809/6912000, & \text{if } 6 < i < 19 \text{ and } i \text{ odd} \\ 1067/144000, & \text{if } 6 < i < 19 \text{ and } i \text{ even} \\ -23/576000, & \text{if } 18 < i < 37 \text{ and } (k_1 = 0 \text{ or } k_2 = 0 \text{ or } k_1 = k_2) \\ -109/288000, & \text{if } 18 < i < 37 \\ -1/13824000, & \text{if } 36 < i < 61 \text{ and } (k_1 = 0 \text{ or } k_2 = 0 \text{ or } k_1 = k_2) \\ 97/6912000, & \text{if } 36 < i < 61 \text{ and } (|k_1| = 2 \text{ or } |k_2| = 2) \\ 1/576000, & \text{if } 36 < i < 61 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In details, let's give the exact formula for the coefficients. We suppose we have the functions $global(k_1, k_2) = i$ and $local(i, i_0) = j$ that give respectively the global index of $\mathbf{x} = \mathbf{Rk}$ and the local index of that point regarding the point at position i_0.

$$c[\mathbf{k}] = \sum_{\mathbf{m}\in\mathbb{Z}^2} s[\mathbf{m}] \cdot p_{IIR}[local(\mathbf{m} - \mathbf{k}, \mathbf{k})] \tag{7}$$

and using (5) we obtain

$$c[\mathbf{k}] = \sum_{local(\mathbf{m}-\mathbf{k},\mathbf{k})=0}^{18} s[\mathbf{m}] \cdot p_{IIR}[local(\mathbf{m} - \mathbf{k}, \mathbf{k})] \tag{8}$$

1.2.3. *Optimizing the evaluation*

For the present state we have all the elements for the approximation of a function f with second order box splines

$$\tilde{f}(\mathbf{x}) = \sum_{\mathbf{k}\in\mathbb{Z}^2} c[\mathbf{k}]\chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}) \tag{9}$$

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 [1] : 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 :

$$\begin{aligned}\tilde{f}(\mathbf{x}) = \quad & c[\mathbf{k}_0]\ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0) \\ & + c[\mathbf{k}_0 + [1,0]]\ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{R}_1) \\ & + c[\mathbf{k}_0 + [0,1]]\ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{R}_2) \\ & + c[\mathbf{k}_0 + [1,1]]\ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{R}_1 - \mathbf{R}_2)\end{aligned} \tag{10}$$

Remarque 1.1. As the χ^2 spline has a support of radius a unity, one of the elements of (10) 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 Voronoi 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

$$\begin{cases}\partial_x f(x,y) = \partial_{H_1} f(x,y).\partial_x H_1 + \partial_{H_2} f(x,y).\partial_x H_2 \\ \partial_y f(x,y) = \partial_{H_1} f(x,y).\partial_y H_1 + \partial_{H_2} f(x,y).\partial_y H_2\end{cases} \tag{11}$$

With H_1 et H_2 the hexaedric coordinates. Since

$$\begin{cases}x = \dfrac{H_1 - H_2}{\sqrt{3}}, \\ y = H_1 + H_2,\end{cases} \tag{12}$$

We obtain :

$$\begin{cases}\partial_x H_1 = \dfrac{\sqrt{3}}{2}\ ;\ \partial_y H_1 = \dfrac{1}{2} \\ \partial_x H_2 = \dfrac{-\sqrt{3}}{2}\ ;\ \partial_y H_2 = \dfrac{1}{2}.\end{cases} \tag{13}$$

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 (?)

References

[1] L. Condat and D. Van De Ville. Quasi-interpolating spline models for hexagonally-sampled data. *IEEE, Transactions on Image Processing*, 16(5):1195–1206, May 2007.

[2] Laurent Condat and Dimitri Van De Ville. Three-directional box-splines: characterization and efficient evaluation. *IEEE Signal Process. Lett.*, 13(7):417–420, 2006.

[3] 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.

[4] Carl de Boor, Klaus Höllig, and Sherman Riemenschneider. *Box Splines*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[5] R. M. Mersereau. The processing of hexagonally sampled two-dimensional signals. 67:930–949, 1979.

[6] Robert Ulichney. *Digital Halftoning*. MIT Press, Cambridge, MA, 1987.