Abstract.

Keywords. Finite difference method; Finite element method; semi-Lagrangian scheme; Vlasov-Poisson model; Guiding-center model; Plasma physics.

1. Introduction

2. Theory

2.1. Spline. 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 [**?**]) 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 [**?**]. 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 [**?**]

$$c = s * p \tag{2.1}$$

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

2.1.1. *Three directional box-splines.* 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 ([**?**, **?**]):

$$\varphi_\Xi(\mathbf{x}) = \begin{cases} \dfrac{1}{|\det(\Xi)|} & \text{if } \Xi^{-1}\mathbf{x} \in [0,1)^2 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

and, for higher orders

$$\varphi_{\Xi\cup[v]}(\mathbf{x}) = \int_0^1 \varphi_\Xi(\mathbf{x} - t\mathbf{v})dt \tag{2.3}$$

We can define the Courant element [**?**] as $\chi^1 = (\sqrt{3}/2)\varphi_{[\mathbf{r_1 r_2 - r_3}]}$ where $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ are the generator vectors. For higher orders we have the recursive expression : $\chi^n = (2/\sqrt{3})\chi^{n-1} * \chi^1, \quad n > 1$ where the operator $*$ represents the convolution. For a complete analytical expression we refer to [**?**] where we find the formula for $\chi^n(\mathbf{x})$, which we have generalized to any hexagonal grid generated by a matrix $\mathbf{R}$ such that $\mathbf{R} = [\mathbf{r_1 r_2}] = \begin{bmatrix} r_{11} & r_{21} \\ r_{12} & r_{22} \end{bmatrix}$. The generalized algorithm is as follows

$$
\begin{aligned}
\chi^n(x_1,x_2) = & \sum_{k_1,k_2=-n}^{n} \sum_{i=\max(k_1,k_2,0)}^{\min(k_1+n,k_2+n,n)} (-1)^{k_1+k_2+i} \binom{n}{i-k_1} \binom{n}{i-k_2} \binom{n}{i} \\
& \sum_{d=0}^{n-1} \binom{n-1+d}{d} \frac{1}{(2n-1+d)!(n-1-d)!} \\
& \left| \frac{2}{\sqrt{3}} (x_2 - r_{12}k_1 - r_{22}k_2) \right|^{n-1-d} \\
& \left(x_1 - r_{11}k_1 - r_{21}k_2 - \frac{1}{\sqrt{3}} |x_2 - r_{12}k_1 - r_{22}k_2| \right)_+^{2n-1+d}
\end{aligned}
\tag{2.4}
$$

where $(x)_+^n = \{x^n \text{ for } x > 0;\ 0, \text{ otherwise}\}$.

This formula derives from a convolution between a particular Green function and a prefilter. For more information we refer to [**?**]. In the latter we find as well an algorithm which exploits the twelve-fold symmetry of the mesh. Unfortunately this algorithm is specific to the second type of hexagon and doesn't take into account an eventual scaling. Nevertheless, if we denote by $\bar{\mathbf{R}} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}$ the generating matrix of a second type hexagonal-mesh of spacing 1, we can re-write the coordinates $\mathbf{x}$ in the basis $\mathbf{R}$ to the basis $\bar{\mathbf{R}}$ by using the formula :

$$
\bar{\mathbf{R}}\mathbf{R}^{-1}\mathbf{x} = \bar{\mathbf{x}} \tag{2.5}
$$

We will choose the definition in (2.4) mostly for $n = 2$ for higher orders we will opt for Box-MOMS (box-splines of maximum order and with minimal support) as presented in [**?**]. The results for Box-MOMS of order 4 (BM_4) are encouraging specially when compared with normal Box-splines of the same order.

2.1.2. *Box splines coefficients.* How we determine the splines coefficients is almost as important as the splines themselves. We recall we have the formula (2.1). 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 :

$$
p_{IIR2}[i] = \begin{cases} 1775/2304, & \text{if } i = 0 \\ 253/6912, & \text{if } 0 < i < 7 \\ 1/13824, & \text{if } 6 < i < 19 \text{ and } i \text{ odd} \\ 11/6912, & \text{if } 6 < i < 19 \text{ and } i \text{ even} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}
$$

Or for the splines of order 3 :

$$p_{IIR2}[i] = \begin{cases} 244301/460800, & \text{if } i = 0 \\ 42269/576000, & \text{if } 0 < i < 7 \\ -11809/6912000, & \text{if } 6 < i < 19 \text{ and } i \text{ odd} \\ 1067/144000, & \text{if } 6 < i < 19 \text{ and } i \text{ even} \\ -23/576000, & \text{if } 18 < i < 37 \text{ and } (k_1 = 0 \text{ or } k_2 = 0 \text{ or } k_1 = k_2) \\ -109/288000, & \text{if } 18 < i < 37 \\ -1/13824000, & \text{if } 36 < i < 61 \text{ and } (k_1 = 0 \text{ or } k_2 = 0 \text{ or } k_1 = k_2) \\ 97/6912000, & \text{if } 36 < i < 61 \text{ and } (|k_1| = 2 \text{ or } |k_2| = 2) \\ 1/576000, & \text{if } 36 < i < 61 \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

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 $\mathbf{x} = \mathbf{R}\mathbf{k}$ and the local index of that point regarding the point at position i_0.

$$c[\mathbf{k}] = \sum_{\mathbf{m}\in\mathbb{Z}^2} s[\mathbf{m}] \cdot p_{IIR}[local(\mathbf{m} - \mathbf{k}, \mathbf{k})] \tag{2.8}$$

and using (2.7) we obtain

$$c[\mathbf{k}] = \sum_{local(\mathbf{m}-\mathbf{k},\mathbf{k})=0}^{18} s[\mathbf{m}] \cdot p_{IIR}[local(\mathbf{m} - \mathbf{k}, \mathbf{k})] \tag{2.9}$$

2.1.3. *Optimizing the evaluation.* For the present state we have all the elements for the approximation of a function f with second order box splines

$$\tilde{f}(\mathbf{x}) = \sum_{\mathbf{k}\in\mathbb{Z}^2} c[\mathbf{k}]\chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}) \tag{2.10}$$

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 :

$$\begin{aligned} \tilde{f}(\mathbf{x}) = \quad & c[\mathbf{k}_0]\ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0) \\ & + c[\mathbf{k}_0 + [1,0]]\ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{r}_1) \\ & + c[\mathbf{k}_0 + [0,1]]\ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{r}_2) \\ & + c[\mathbf{k}_0 + [1,1]]\ \chi^2(\mathbf{x} - \mathbf{R}\mathbf{k}_0 - \mathbf{r}_1 - \mathbf{r}_2) \end{aligned} \tag{2.11}$$

Remark 1: As the χ^2 spline has a support of radius a unity, one of the elements of (2.11) 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.

3. General algorithm

To conclude we want to write the entire procedure.

Assumptions and initialization.

- Mesh : defined by the matrix $\mathbf{R}$, its center $\mathbf{x}_0$ (typically the origin), its radius L and the number of cells N;
- Points : The points of the mesh can be initialized as follows $\mathbf{x}_i = \sum_i \mathbf{R}\mathbf{k}_i$;
- Initial distribution : We assume we have a sample data such that $s^0[i] = f_0(\mathbf{x}_i)$ is given on the mesh points;
- Computing the characteristics : as the characteristic's feet are time-independent we can compute at the initialization step. We denote them $\tilde{\mathbf{x}}_i$.

Time loop.

- Computing of the spline's coefficient : using the algorithm in (2.9) we compute the 19 elements sum on each point and pre-compute the spline coefficients using the pre-filter's values and the sample data s^n ;
- Element by element interpolation :
  - First we compute $\mathbf{k}_0$ such that $\mathbf{k}_0 = \mathbf{R}^{-1}\tilde{\mathbf{x}}_i$. Remark 2: We will need a test case to see if $\mathbf{k}_0 \in \Omega$ and that uses the boundary conditions to compute a new $\mathbf{k}_0$ otherwise.
  - Then we need to change the points' basis, such that they are defined in the spline basis. We use (2.5) to find the coordinates of $\tilde{\mathbf{x}}_i$ in the basis of the second-type hexagonal mesh $\bar{\mathbf{R}}$, we will denote the solution $\bar{\mathbf{x}}_i$
  - We use the formula (2.11) to interpolate the value $\tilde{f}_0(\bar{\mathbf{x}}_i)$, the final formula is given below

$$\begin{aligned}\tilde{f}_0(\bar{\mathbf{x}}_i) = \quad & c[\mathbf{k}_0]\, \chi^2(\bar{\mathbf{R}}\mathbf{R}^{-1}(\tilde{\mathbf{x}}_i - \mathbf{R}\mathbf{k}_0)) \\ & + c[\mathbf{k}_0 + [1,0]]\, \chi^2(\bar{\mathbf{R}}\mathbf{R}^{-1}(\tilde{\mathbf{x}}_i - \mathbf{R}\mathbf{k}_0 - \mathbf{r}_1)) \\ & + c[\mathbf{k}_0 + [0,1]]\, \chi^2(\bar{\mathbf{R}}\mathbf{R}^{-1}(\tilde{\mathbf{x}}_i - \mathbf{R}\mathbf{k}_0 - \mathbf{r}_2)) \\ & + c[\mathbf{k}_0 + [1,1]]\, \chi^2(\bar{\mathbf{R}}\mathbf{R}^{-1}(\tilde{\mathbf{x}}_i - \mathbf{R}\mathbf{k}_0 - \mathbf{r}_1 - \mathbf{r}_2))\end{aligned} \tag{3.12}$$

3.1. **Hermite finite element.** 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

$$\begin{cases}\partial_x f(x,y) = \partial_{H_1} f(x,y).\partial_x H_1 + \partial_{H_2} f(x,y).\partial_x H_2 \\ \partial_y f(x,y) = \partial_{H_1} f(x,y).\partial_y H_1 + \partial_{H_2} f(x,y).\partial_y H_2\end{cases} \tag{3.1}$$

With H_1 et H_2 the hexaedric coordinates. Since

$$\begin{cases}x = \dfrac{H_1 - H_2}{\sqrt{3}}, \\ y = H_1 + H_2,\end{cases} \tag{3.2}$$

We obtain :

$$\begin{cases}\partial_x H_1 = \dfrac{\sqrt{3}}{2} \;;\; \partial_y H_1 = \dfrac{1}{2} \\ \partial_x H_2 = \dfrac{-\sqrt{3}}{2} \;;\; \partial_y H_2 = \dfrac{1}{2}.\end{cases} \tag{3.3}$$

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 (?)

4. Numerical tests

5. Conclusion and perspectives

In this paper

References

[1] E. Sonnendrcker, J. Roche, P. Bertrand, A. Ghizzo The Semi-Lagrangian Method for the Numerical Resolution of Vlasov Equations, *Comput. Phys* ,**149, 201220** (1999)

[2] Gustavo Buscaglia, Vitoriano Ruas Finite element methods for the Stokes system based on a Zienkiewicz type N-simplex, *Comput. Methods Appl. Mech. Engrg.*,**272, 83-99** (2014)