Solving the Vlasov equation using the Semi-Lagrangian scheme on a 2D hexagonal mesh

Laura S. Mendoza

Max-Planck-Institut für Plasmaphysik
Supervisor : Eric Sonnendrücker

June 5, 2014
Motivation

The Gyrokinetic Semi-Lagrangian (GYSELA) code:

- **Gyrokinetic model**: 5D kinetic equation on the charged particles distribution
- 5 Dimensions: 2 in velocity space, 3 in configuration space
- **Simplified geometry**: concentric toroidal magnetic flux surfaces with circular cross-sections
- Based on the **Semi-Lagrangian** scheme
Motivation

Current representation of the poloidal plane:
- Annular geometry
- Polar mesh \((r, \theta)\)

Some limitations of this choice:
- Geometric (and numeric) singular point at origin of mesh
- Unrepresented area and very costly to minimize that area
- Impossible to represent complex geometries
The Backwards Semi-Lagrangian Scheme

We consider the simplest form of the Vlasov equation, the advection equation

\[
\frac{\partial f}{\partial t} + \mathbf{a} \cdot \nabla_{\mathbf{x}} f = 0
\]

The Semi-Lagrangian scheme:

- Initial distribution known on all mesh points
- **Method of characteristics**: gives the origin of the trajectory of the particle at previous time step (density conserved along characteristics)
- Interpolate on the origin using known values of previous step at mesh points (usually **cubic B-spline interpolation**)
B(asis)-Splines basics*

B-Splines of degree \( d \) are defined by the recursion formula:

\[
B_j^{d+1}(x) = \frac{x - x_j}{x_{j+d} - x_j} B_j^d(x) + \frac{x_{j+1} - x}{x_{j+d+1} - x_{j+1}} B_{j+1}^d(x)
\]

Some important properties about B-splines

- Piecewise polynomials of degree \( d \) \( \implies \) smoothness
- Compact support \( \implies \) sparse matrix system
- Partition of unity \( \sum_j B_j(x) = 1, \ \forall x \implies \) conservation laws
Interpolating with cubic B-Splines

Initial data:
- Uniform mesh
- Initial distribution function \( f(t_0, x) \) known at all mesh points \( x_i \)

The interpolant \( f_h \) is \textbf{exact on the mesh points} and is defined by

\[
f_h(x) = \sum_{j=0}^{N} c_j B^3(x - x_j) \tag{1}
\]

where the coefficients \( c_j \) are computed using the property

\[
f_h(x_i) = \sum_{j=0}^{N} c_i B^3(x_i - x_j) = f(x_i) \tag{2}
\]

Which can be written as a \textbf{sparse matrix system}, where boundary conditions intervene.
Multi-patch: the general idea

Our original mesh:
Multi-patch: the general idea

New representation of the poloidal plane:
Multi-patch: the general idea

Specificities of the new geometry definition:

- **Additional patch(es)** with no singular point at origin
- Each patch defined as a **transformation** (or mapping) from uniform cartesian grid to new mesh
- Mappings defined with **NURBS** (Non-Uniform Rational B-Splines) $\Rightarrow$ complex geometries
- **Coupling** between patches defined by boundary condition
The 5 patches configuration

External crown divided into 4 patches and the connectivity is defined as a patch-edge to patch-edge association (creation tool: CAID)

Advantages

- Flexibility defining complex geometries
- Each patch can be treated separately
- No geometrical singularity

New challenges

- What is the best BC?
- How to treat particles which characteristics’ origin are on another patch?

4 new numerical singularities
Multi-patch: Some results

Results always showed instabilities near singular points. What we’ve tried to avoid them:

- Boundary conditions tested: strictly interdependent gradients and mean gradients between connecting patches
- Over-lapping: Impossible with interior patch and useless for others
- Squared internal mapping

**Problem:** Impossible to avoid singular points from mapping from a square to a circle
Second approach: The hexagonal mesh

**Idea:** Use a new mapping: square $\rightarrow$ circle.
We define a tiling of triangles of a hexagon as our mesh for a 2D poloidal plane.

Some advantages:
- No singular points
- (Hopefully) no need of multiple patches for the core of the tokamak
- Twelve-fold symmetry $\Rightarrow$ more efficient programming
- Easy transformation from cartesian to hexagonal coordinates
- Easy mapping to a disk
Box-splines and quasi-interpolation

Box-Splines:
- Generalization of B-Splines
- Depend on the vectors that define the mesh
- Easy to exploit symmetry of the domain

⇒ More efficient interpolation

Quasi-interpolation:
- Distribution function known at mesh points
- Of order $L$ if perfect reconstruction of a polynomial of order $L - 1$
- No exact interpolation at mesh points $f_h(x_i) = f(x_i) + O(\|x_i\|^L)$

⇒ Additional freedom to choose the coefficients $c_j$

$$f_h(x) = \sum c_j \chi^L(x - x_j)$$
Computing the spline coefficients using pre-filters

**Idea:** Coefficients obtained by discrete filtering of sample values $f(x_i)$

$$c = p * f = \sum_i f(x_i) p_i$$

(4)

**prefilters:** Obtained by solving a linear system of $L$ equations (quasi-interpolation conditions)

Example with $L = 2$:
- We use information on two hexagons from point
- Points at same radius have same weight
- Error: $O(\| x \|^2)$
Using box splines of degree 2, we obtained the following results:

<table>
<thead>
<tr>
<th>model</th>
<th>Points</th>
<th>a</th>
<th>dt</th>
<th>loops</th>
<th>$L_2$ error</th>
</tr>
</thead>
<tbody>
<tr>
<td>On mesh points</td>
<td>17101</td>
<td>0.</td>
<td>0.025</td>
<td>1</td>
<td>$4.99 \times 10^{-6}$</td>
</tr>
<tr>
<td>Constant advec.</td>
<td>17101</td>
<td>0.05</td>
<td>0.025</td>
<td>81</td>
<td>$4.70 \times 10^{-3}$</td>
</tr>
<tr>
<td>Circular advec.</td>
<td>17101</td>
<td>1.</td>
<td>0.025</td>
<td>81</td>
<td>$4.33 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

Using the pre-filter $P_{fir}$ and mirror boundary conditions. The error increments linearly on time for advections and is of order 4.
Conclusion and perspectives

Hexagonal mesh:
- Results more encouraging than multi-patch results
- No numeric problems due to the mesh
- Efficiency to be compared
- More complex models to be tested
- Results have to be tested on a disk (and not a hexagon)
- Boundary conditions to be defined properly
- Box-MOMS (Maximal order minimal support box splines)

Multi-patch:
- Schwartz iterative method: stabilize singular points
- May still be useful for more complex geometries
- Implementation in the SELALIB library
Thank you for your attention.