Guiding center simulations on curvilinear grids

Adnane Hamiaz, Michel Mehrenberger, Aurore Back, Pierre Navaro

▶ **To cite this version:**

Adnane Hamiaz, Michel Mehrenberger, Aurore Back, Pierre Navaro. Guiding center simulations on curvilinear grids. ESAIM: Proceedings, 2016, 53, pp.99-119. 10.1051/proc/201653007 . hal-00908500v3

HAL Id: hal-00908500
https://hal.science/hal-00908500v3

Submitted on 8 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Guiding center simulations on curvilinear grids

A. Hamiaz, M.Mehrenberger, A. Back, P. Navaro

October 8, 2015

Abstract

Semi-Lagrangian guiding center simulations are performed on sinusoidal perturbations of cartesian grids, and on deformed polar grid with different boundary conditions. Key ingredients are: the use of a B-spline finite element solver for the Poisson equation and the classical backward semi-Lagrangian method (BSL) for the advection. We are able to reproduce standard Kelvin-Helmholtz and diocotron instability tests on such grids. When the perturbation leads to a strong distorted mesh, we observe that the solution differs if one takes standard numerical parameters that are used in the cartesian reference case. We can recover good results together with correct mass conservation, by diminishing the time step.[1]

1 Introduction

Semi-Lagrangian schemes often deal with cartesian mesh; the extension to curvilinear grids is important in order to be able to deal with specific geometries and also for adapting the grid to save computational effort. This study is part of a general work on adding curvilinear capabilities for the simulation of drift kinetic and gyrokinetic equations in a semi-Lagrangian framework, and is in current development in the SeLaLib library [30]. One motivation is that it is standard to use specific curvilinear grids for turbulence simulations of magnetic fusion plasmas [22]. This is due to the specific geometry of magnetic fusion devices and the strong magnetic field for which it helps to have a mesh aligned on magnetic flux surfaces. In order to treat the case of a general geometry, semi-Lagrangian schemes on unstructured triangular meshes have been developed and applied to Vlasov-Poisson simulations [5]. Recently, a new approach has been developed which permits to stick on a cartesian mesh, with a suitable technique to treat boundary conditions [19]. Another option is to use curvilinear grids, with analytical or discrete transformation [1]. There, the choice has been made to keep the expressions of the advection equations in the physical space, rather than to rewrite these equations in the reference space. Here, we choose the other

[1]This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

option (as in [3], for PIC simulations), which permits to avoid the cost overhead of the localization of the feet of the characteristics (which is also present on unstructured meshes), since the localisation is performed on the cartesian grid. The semi-Lagrangian method can be adapted to the new equation, which is of similar nature, and the cost of this step remains of the same order.
The work is devoted to the case of the $2D$ guiding center model, which is an advection equation of the form

$$\partial_t f(t,\mathbf{x}) + \mathbf{a}\cdot\nabla_{\mathbf{x}} f(t,\mathbf{x}) = 0. \tag{1}$$

The vector field $\mathbf{a}$ is divergence free. It writes $\mathbf{a} = (\partial_{x_2}\Phi, -\partial_{x_1}\Phi)^\top$, where $^\top$ stands for the transposition and $\mathbf{x} = (x_1, x_2)$. The stream function Φ satisfies the Poisson equation

$$-\Delta\Phi = f.$$

This reduced model has been widely used in the literature in recent years (see e.g. [34],[14] for cartesian geometry, [29],[12],[18] for circular geometry and [19] for D-shaped geometry). A reference on such models and different variants (change of geometry, extension to 3D) is the work of M. Shoucri (see e.g. [31] and references in [14, 12], in particular [32]).
Such system can already describe some physics; as an example, it can be seen as a limit of a Vlasov-Poisson system (see e.g. [6] and [13, 9, 20] for the numerics). As emphasized in [12], it is a building block for future drift kinetic simulations. Guiding center semi-Lagrangian simulations on curvilinear grids have already been performed in [8]; but there the mesh was only made oblic and the Poisson solver was solved on cartesian grid. We refer also to [2], for recent work on curvilinear semi-Lagrangian schemes, in the context of Navier-Stokes equations, and to [21], where the semi-Lagrangian CIP method is adapted to curvilinear geometry. In [26], conservative schemes are used and applied in analytical cases and $1D \times 1D$ Vlasov-Poisson simulations on curvilinear grids.

We will use here the classical backward semi-Lagrangian method (BSL) with cubic splines [34] and one originality of the work is the use of a B-spline finite element solver for the Poisson solver [4]. Getting expertise on such solver inside a library like Selalib is a key point, as such tool can be used for different solvers (semi-Lagrangian, PIC) and different geometries. Previous Poisson solvers using FFT (for polar or cartesian mesh) are no more valid in this context. Alternative possible strategies are to interface the code with other existing softwares (like Mudpack, used in [24]). Note that these difficulties were not present in the context of semi-Lagrangian simulations on simple grids. To see the robustness of the numerical method, we test the method both on sinusoidal perturbations of cartesian grids [11] and polar grids. In another work [4], the method is successfully tested for $2D \times 2D$ Vlasov-Poisson simulations. Next steps of validation concern drift kinetic and gyrokinetic simulations.
Note that a further study about semi-Lagrangian schemes on curvilinear geometry is performed in [24], written after the first submission of this work. We focus

here on the Poisson solver and on the ability of the same code to perform simulations on different geometries, such points being not discussed in [24]. We refer to [24] for more references on semi-Lagrangian schemes, motivations, context description and other validations of the code that is used for the simulations. We mention also the work on the SELHEX project [27], which is an alternative strategy allowing to avoid geometrical singularity of the Jacobian, as it is the case for example using polar mesh. Here we suppose that the Jacobian is defined everywhere and non zero; this limits the range of applications, but multi-patch extension could be envisaged for dealing with more complex domains (see [28], for a preliminary work on the subject).
In Section 2, we briefly write the equations in curvilinear coordinates. We then detail the numerical method in Section 3, taking special attention to the Poisson solver. Numerical results are given in Section 4. Conclusion and perspectives are presented in Section 5.

2 Curvilinear framework

In this section, we define the mapping from the reference domain (called logical domain) to the physical domain and then give the expression of the different equations (advection and Poisson equations) in the logical domain. We refer to [24] for the derivation of the equations and only put here the results.
We denote by $\Omega \in \mathbb{R}^2$ the physical domain where the equations are set. To solve these equations, we consider a curvilinear coordinates system which is a mapping $\mathcal{F}$, defined on a rectangular logical domain Q:

$$\mathcal{F} : Q = [\eta_{1_{min}}, \eta_{1_{max}}] \times [\eta_{2_{min}}, \eta_{2_{max}}] \to \Omega \tag{2}$$
$$\begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \to \begin{bmatrix} x_1(\eta_1, \eta_2) \\ x_2(\eta_1, \eta_2) \end{bmatrix}.$$

Denoting $\tilde{u} = u \circ \mathcal{F}$ for a function $u : \Omega \to \mathbb{R}$, equation (1) rewrites in the logical domain

$$\frac{\partial \tilde{f}}{\partial t} + \frac{1}{\sqrt{g}} \frac{\partial \tilde{\Phi}}{\partial \eta_2} \frac{\partial \tilde{f}}{\partial \eta_1} - \frac{1}{\sqrt{g}} \frac{\partial \tilde{\Phi}}{\partial \eta_1} \frac{\partial \tilde{f}}{\partial \eta_2} = 0, \tag{3}$$

where

$$\sqrt{g} = \frac{\partial x_1}{\partial \eta_1} \frac{\partial x_2}{\partial \eta_2} - \frac{\partial x_1}{\partial \eta_2} \frac{\partial x_2}{\partial \eta_1}$$

is the jacobian which is the determinant of the Jacobian matrix

$$D\mathcal{F}(\eta_1, \eta_2) = \begin{pmatrix} \frac{\partial x_1}{\partial \eta_1} & \frac{\partial x_1}{\partial \eta_2} \\ \frac{\partial x_2}{\partial \eta_1} & \frac{\partial x_2}{\partial \eta_2} \end{pmatrix}.$$

Poisson equation has the form of a general elliptic equation

$$-\nabla_{\mathbf{x}} \cdot (\mathbf{b}(x_1, x_2) \nabla_{\mathbf{x}} \Phi(x_1, x_2)) + c(x_1, x_2) \Phi(x_1, x_2) = f(x_1, x_2), \tag{4}$$

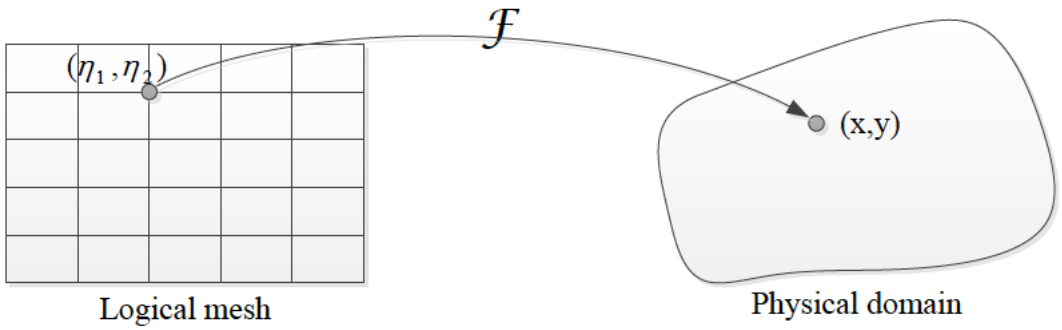

Figure 1: Representation of a mapped mesh in two dimensions.

with $\mathbf{b}$ a 2×2 matrix function and c a scalar function. Here, we have $\mathbf{b} = I_2, c = 0$ (where I_2 is the identity matrix), as we solve the Poisson equation. It rewrites in the logical domain

$$- \nabla_\eta \cdot (\sqrt{g}\, D\mathcal{F}^{-1} \tilde{\mathbf{b}}\, D\mathcal{F}^{-\top} \nabla_\eta \tilde{\Phi}) + \sqrt{g}\, \tilde{c}\, \tilde{\Phi} = \sqrt{g}\, \tilde{f}, \tag{5}$$

where we use the notation $D\mathcal{F}^{-\top} = (D\mathcal{F}^{-1})^\top$. Note that the expression of the mass and electric energy (which are conserved quantities in this model, up to specifying boundary conditions [12]) are respectively given by

$$\mathcal{M}(t) = \int_\Omega f dxdy = \int_Q \tilde{f} |\sqrt{g}\, |d\eta_1 d\eta_2,$$

and

$$\mathcal{E}(t) = \int_\Omega (\nabla_{\mathbf{x}} \Phi)^\top \nabla_{\mathbf{x}} \Phi\, dxdy = \int_Q (\nabla_\eta \tilde{\Phi})^\top D\mathcal{F}^{-1} D\mathcal{F}^{-\top} \nabla_\eta \tilde{\Phi}\, |\sqrt{g}\, |d\eta_1 d\eta_2.$$

Using these expressions, we will check numerically the conservation of the mass and electric energy in the Kelvin-Helmholtz instability case on a deformed mesh.

3 Numerical method

In this section, we first detail the numerical procedure to compute the Poisson equation and then describe the semi-Lagrangian solver.

3.1 Finite element method with B-splines for the Poisson equation on curvilinear coordinates

Equation (5) leads to the elliptic equation of same type as (4):

$$- \nabla_\eta \cdot \sqrt{g}\, \mathbf{A} \nabla_\eta \tilde{\Phi} + \tilde{c} \sqrt{g}\, \tilde{\Phi} = \tilde{f} \sqrt{g}\, , \tag{6}$$

where the matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} = D\mathcal{F}^{-1}\tilde{\mathbf{b}}\, D\mathcal{F}^{-\top}.$$

This means that developing a general elliptic solver in cartesian geometry, a Poisson solver in curvilinear geometry or a general elliptic solver in curvilinear geometry are all three of the same difficulty; we detail here the main steps for the implementation of such a solver, using B-splines. We could have used more classical finite element solvers; B-splines are a natural choice, as we will use cubic splines for the interpolation. We will consider here B-splines of maximal regularity; extension could be made to deal with lower regularity by repeating the knots which are defined here at the boundary of the uniform cells. This would however lead to increase the number of degrees of freedom.

3.1.1 B-splines computations

We first consider to $1D$ case. Let $\eta_{\min} < \eta_{\max} \in \mathbb{R}$. We use a uniform mesh with $N \in \mathbb{N}^*$ cells and define

$$\eta_j = \eta_{\min} + (j-1)\Delta\eta,\ j = 1, \dots, N+1,\ \Delta\eta = \frac{\eta_{\max} - \eta_{\min}}{N}.$$

Let $d \in \mathbb{N}$. We define knots $\tau_j,\ j = 1, \dots N + 2d + 1$. The knots inside the domain are the mesh points:

$$\tau_{d+j} = \eta_j,\ j = 1, \dots, N+1.$$

We suppose that the other $2d$ knots satisfy

$$\tau_j \leq \tau_{d+1},\ j = 1, \dots, d,\ \tau_{N+d+1} \leq \tau_j,\ j = N+d+2, \dots, N+2d+1.$$

Definition 3.1 (B-Splines series [7])**.** *The i-th B-Spline of degree $d \in \mathbb{N}$ (or order $d+1$) denoted by B_i^{d+1} is defined by: $B_i^1(x) = \begin{cases} 1,\ \tau_i \leq x < \tau_{i+1} \\ 0,\ \text{elsewhere.} \end{cases}$, for $d = 0$, and*

$$B_i^{d+1}(x) = \frac{x - \tau_i}{\tau_{i+d} - \tau_i} B_i^d(x) + \left(1 - \frac{x - \tau_{i+1}}{\tau_{i+1+d} - \tau_{i+1}}\right) B_{i+1}^d(x),\ 1 \leq i \leq N+d,\ d \geq 1,$$

if $\tau_i \neq \tau_{i+d+1}$, and $B_i^{d+1}(x) = 0$, otherwise, and using the convention $\frac{0}{0} = 0$. The B-spline B_i^{d+1} of degree d has compact support $[\tau_i, \tau_{i+d+1}]$.

In order to have something more explicit, and as we will need to evaluate both the non zero B-splines and their derivatives, we can use Deboor's Fortran routine `bsplvd`. We have used the following version of the algorithm, which may have its interest, as it is self-consistant, it does not use any extra memory, except two variables, and it is of similar cost, as the `bsplvd` implementation. For $x \in [\tau_m, \tau_{m+1}[$, we can compute $b_j = b_j(x),\ j = 1, \dots, d+1$, where $b_j(x) = B_{m-d-1+j}^{d+1}(x)$, together with $b'_j = b'_j(x),\ j = 1, \dots, d+1$, where $b'_j(x) = (B_{m-d-1+j}^{d+1})'(x)$ by the following algorithm, which works for $d \geq 2$,

$b_1 \leftarrow 1$
for $\ell = 1, \dots, d$ **do**
  $\alpha \leftarrow \frac{x - \tau_{m+1-\ell}}{\tau_{m+1} - \tau_{m+1-\ell}} b_1$
  $b_1 \leftarrow b_1 - \alpha$
  for $k = 2, \dots, \ell$ **do**
    $\beta \leftarrow \frac{x - \tau_{m+k-\ell}}{\tau_{m+k} - \tau_{m+k-\ell}} b_k$
    $b_k \leftarrow b_k + \alpha - \beta$
    $\alpha \leftarrow \beta$
  end for
  $b_{\ell+1} \leftarrow \alpha$
  if $\ell = d - 1$ **then**
    $\alpha \leftarrow \frac{d}{\tau_{m+1} - \tau_{m+1-d}} b_1$
    $b'_1 \leftarrow -\alpha$
    for $k = 2, \dots, d$ **do**
      $b'_k \leftarrow \alpha$
      $\alpha \leftarrow \frac{d}{\tau_{m+k} - \tau_{m+k-d}} b_k$
      $b'_k \leftarrow b'_k - \alpha$
    end for
    $b'_{d+1} = \alpha$
  end if
end for

As the B-splines depend only on the knots sequence $\boldsymbol{\tau} = (\tau_i)_{i=1}^{N+2d+1}$, we will use the notation $B_{i,\boldsymbol{\tau}} = B_i^{d+1},\ i = 1, \dots, N + d$.
Now, we consider the $2D$ extension. We use a uniform mesh with N_δ cells in direction η_δ, for $\delta = 1, 2$ and define

$$\eta_{\delta,j} = \eta_{\delta,\min} + (j-1)\Delta\eta_\delta,\ j = 1, \dots, N_\delta + 1,\ \Delta\eta_\delta = \frac{\eta_{\delta,\max} - \eta_{\delta,\min}}{N_\delta},\ \delta = 1, 2,$$

and similarly degrees and knots in $2D$: $d_\delta,\ \boldsymbol{\tau}_\delta,\ \delta = 1, 2$. A function $S_h : Q \to \mathbb{R}$ in the vector space generated by the B-spline will be of the form

$$S_h(\eta_1, \eta_2) = \sum_{i=1}^{N_1+d_1} \sum_{j=1}^{N_2+d_2} c_{i,j}^{S_h} B_{i,\boldsymbol{\tau}_1}(\eta_1) B_{j,\boldsymbol{\tau}_2}(\eta_2),\ (\eta_1, \eta_2) \in Q, \tag{7}$$

and $(c_{i,j}^{S_h})_{i,j=1}^{N_1+d_1, N_2+d_2}$ are called the spline coefficients of S_h.
Thanks to the compact support of the B-splines, we have for $\eta_{1,i} \le \eta_1 < \eta_{1,i+1},\ \eta_{2,j} \le \eta_2 < \eta_{2,j+1},\ i = 1, \dots, N_1 + 1,\ j = 1, \dots, N_2 + 1$,

$$S_h(\eta_1, \eta_2) = \sum_{k=i}^{i+d_1} \sum_{\ell=j}^{j+d_2} s_{k,\ell} B_{k,\boldsymbol{\tau}_1}(\eta_1) B_{\ell,\boldsymbol{\tau}_2}(\eta_2) \tag{8}$$

3.1.2 Finite element formulation

Now, we want to solve (6) using a finite element formulation in this setting. So we look for a function $\tilde{\Phi}_h \simeq \tilde{\Phi}$ of the form (7), which is equivalent to find its

spline coefficients $(c_{i,j}^{\tilde{\Phi}_h})_{i,j=1}^{N_1+d_1,N_2+d_2}$. We replace $\tilde{\Phi}$ by $\tilde{\Phi}_h$ in (6), multiply by a test function $B_{k,\boldsymbol{\tau}_1}B_{\ell,\boldsymbol{\tau}_2}$, for $k=1,\dots,N_1+d_1$ and $\ell=1,\dots,N_2+d_2$, and integrate on Q. On the right hand side, we suppose that we have a reconstruction (not necessarily the spline decomposition) $\tilde{f}_h \simeq \tilde{f}$, which is defined on Q. This leads to

$$\begin{aligned}\sum_{i,j=1}^{N_1+d_1,N_2+d_2} c_{i,j}^{\tilde{\Phi}_h} \int_Q \left(A_{1,1}B'_{i,\boldsymbol{\tau}_1}B_{j,\boldsymbol{\tau}_2} + A_{1,2}B_{i,\boldsymbol{\tau}_1}B'_{j,\boldsymbol{\tau}_2}\right) B'_{k,\boldsymbol{\tau}_1}B_{\ell,\boldsymbol{\tau}_2}|\sqrt{g}\,|d\eta \\ +c_{i,j}^{\tilde{\Phi}_h} \int_Q \left(A_{2,1}B'_{i,\boldsymbol{\tau}_1}B_{j,\boldsymbol{\tau}_2} + A_{2,2}B_{i,\boldsymbol{\tau}_1}B'_{j,\boldsymbol{\tau}_2}\right) B_{k,\boldsymbol{\tau}_1}B'_{\ell,\boldsymbol{\tau}_2}|\sqrt{g}\,|d\eta \\ +c_{i,j}^{\tilde{\Phi}_h} \int_Q \tilde{c}B_{i,\boldsymbol{\tau}_1}B_{j,\boldsymbol{\tau}_2}B_{k,\boldsymbol{\tau}_1}B_{\ell,\boldsymbol{\tau}_2}|\sqrt{g}\,|d\eta = \int_Q \tilde{f}_h B_{k,\boldsymbol{\tau}_1}B_{\ell,\boldsymbol{\tau}_2}|\sqrt{g}\,|d\eta,\end{aligned} \tag{9}$$

up to boundary conditions which will be discussed later. We obtain a linear system on the spline coefficients $\mathbf{U}=(U_I)\in\mathbb{R}^{(N_1+d_1)(N_2+d_2)}$, where $U_{i+(j-1)(N_1+d_1)} = c_{i,j}^{\tilde{\Phi}_h}$ which reads

$$\mathbf{MU}=\boldsymbol{V}. \tag{10}$$

The right hand side vector is $\mathbf{V}=(V_K)\in\mathbb{R}^{(N_1+d_1)(N_2+d_2)}$ given by

$$V_K = \int_Q \tilde{f}_h B_{k,\boldsymbol{\tau}_1}B_{\ell,\boldsymbol{\tau}_2}|\sqrt{g}\,|d\eta,\ K=k+(\ell-1)(N_1+d_1),\ k=1,\dots,N_1+d_1,\ \ell=1,\dots,N_2+d_2.$$

Using the property of the support of the B-splines, we have

$$V_K = \sum_{r=\max(k-d_1,1)}^{\min(k,N_1)} \sum_{s=\max(\ell-d_2,1)}^{\min(\ell,N_2)} \int_{\eta_{1,r}}^{\eta_{1,r+1}} \int_{\eta_{2,s}}^{\eta_{2,s+1}} \tilde{f}_h B_{k,\boldsymbol{\tau}_1}B_{\ell,\boldsymbol{\tau}_2}|\sqrt{g}\,|d\eta,$$

So, $\mathbf{V}$ can be computed by the following algorithm:

$\mathbf{V}\leftarrow 0$
for $r=1,\dots,N_1$ and $s=1,\dots,N_2$ **do**
  for $k=r,\dots,r+d_1$ and $\ell=s,\dots,s+d_2$ **do**
    $K\leftarrow k+(\ell-1)(N_1+d_1)$
    $V_K\leftarrow V_K+\int_{\eta_{1,r}}^{\eta_{1,r+1}}\int_{\eta_{2,s}}^{\eta_{2,s+1}} \tilde{f}_h B_{k,\boldsymbol{\tau}_1}B_{\ell,\boldsymbol{\tau}_2}|\sqrt{g}\,|d\eta$
  end for
end for

The integral in this algorithm will be approximated by a Gauss quadrature formula. The matrix $\mathbf{M}=(M_{I,K})\in\mathbb{R}^{(N_1+d_1)(N_2+d_2)}\times\mathbb{R}^{(N_1+d_1)(N_2+d_2)}$ is formed of several pieces. Let $\mathbf{M}^{1,1}$ be the contribution of the matrix concerning the term $A_{1,1}$. Contribution of other terms are similar and will not be detailed. We have

$$\mathbf{M}^{1,1}_{I,K} = \int_Q A_{1,1}B'_{i,\boldsymbol{\tau}_1}B_{j,\boldsymbol{\tau}_2}B'_{k,\boldsymbol{\tau}_1}B_{\ell,\boldsymbol{\tau}_2}|\sqrt{g}\,|d\eta,\ I=i+(j-1)(N_1+d_1),\ K=k+(\ell-1)(N_1+d_1),$$

and can compute $\mathbf{M}^{1,1}$ by the following algorithm (using again the property of the support of the B-splines and using also a Gauss quadrature formula for the integral):

$\mathbf{M}^{1,1} \leftarrow 0$
for $r = 1, \dots, N_1$ and $s = 1, \dots, N_2$ **do**
  for $i, k = r, \dots, r + d_1$ and $j, \ell = s, \dots, s + d_2$ **do**
    $I \leftarrow i + (j-1)(N_1 + d_1)$ and $K \leftarrow k + (\ell - 1)(N_1 + d_1)$
    $M^{1,1}_{I,K} \leftarrow M^{1,1}_{I,K} + \int_{\eta_{1,r}}^{\eta_{1,r+1}} \int_{\eta_{2,s}}^{\eta_{2,s+1}} A_{1,1} B'_{i,\boldsymbol{\tau}_1} B_{j,\boldsymbol{\tau}_2} B'_{k,\boldsymbol{\tau}_1} B_{\ell,\boldsymbol{\tau}_2} |\sqrt{g}\, | d\eta$
  end for
end for

This a first construction of the matrix $\mathbf{M}$ and right hand side $\mathbf{V}$.

3.1.3 Treatment of boundary conditions

We have to make some adaptions in order to treat different boundary conditions. We will refer to $\mathbf{M}^{\mathrm{bc}_1,\mathrm{bc}_2}$ and $\mathbf{V}^{\mathrm{bc}_1,\mathrm{bc}_2}$ the new matrix adapted to the boundary condition bc_1 in direction η_1 and bc_2 in direction η_2. We will write $\mathrm{bc}_\delta = \mathrm{bc}_{\delta,\min}/\mathrm{bc}_{\delta,\max}$, where $\mathrm{bc}_{\delta,\min}$ (resp. $\mathrm{bc}_{\delta,\max}$) is the boundary condition at $\eta_{\delta,\min}$ (resp. $\eta_{\delta,\max}$). For boundary conditions, we will consider the periodic case ($\mathrm{bc}_{\delta,\min} = \mathrm{bc}_{\delta,\max} = \mathrm{per}$), Dirichlet and Neumann cases. For example, we can consider $\mathrm{bc}_\delta = \mathrm{neu}/\mathrm{dir}$:

- $\mathrm{bc}_{\delta,\min} = \mathrm{neu}$: Neumann at $\eta_{\delta,\min}$, which corresponds to set the derivative te be zero when $\eta_\delta = \eta_{\delta,\min}$ (homogeneous Neumann boundary condition)
- $\mathrm{bc}_{\delta,\max} = \mathrm{dir}$: Dirichlet at $\eta_{\delta,\max}$, which corresponds to set the function te be zero when $\eta_\delta = \eta_{\delta,\max}$ (homogeneous Dirichlet boundary condition).

Extensions could be made to treat Robin and non homogeneous boundary conditions but are not discussed here. First, we will adapt the knots to the boundary conditions; we describe this procedure in $1D$ as before. For Dirichlet and Neumann boundary conditions, knots are chosen by repeating the values at the boundary :

$$\tau_j = \tau_d,\ j = 1, \dots, d,\ \tau_j = \tau_{N+d+1},\ j = N + d + 2, \dots, N + 2d + 1,$$

and for periodic boundary conditions, we use

$$\tau_j = \eta_{j-d+N} - (\eta_{\max} - \eta_{\min}),\ j = 1, \dots, d,\ \tau_j = \eta_{j-d-N} + (\eta_{\max} - \eta_{\min}),\ j = N+d+2, \dots, N+2d+1.$$

Now, we describe how to construct the matrices. We have $\mathbf{M}^{\mathrm{bc}_1,\mathrm{bc}_2} \in \mathbb{R}^{N^1_{\mathrm{bc}_1} N^2_{\mathrm{bc}_2}} \times \mathbb{R}^{N^1_{\mathrm{bc}_1} N^2_{\mathrm{bc}_2}}$ and $\mathbf{V}^{\mathrm{bc}_1,\mathrm{bc}_2} \in \mathbb{R}^{N^1_{\mathrm{bc}_1} N^2_{\mathrm{bc}_2}}$. Sizes are fixed in the following way:

$$N^\delta_{\mathrm{per/per}} = N_\delta,\ N^\delta_{\mathrm{dir/dir}} = N_\delta + d_\delta - 2,\ N^\delta_{\mathrm{dir/neu}} = N^\delta_{\mathrm{neu/dir}} = N_\delta + d_\delta - 1,\ N^\delta_{\mathrm{neu/neu}} = N_\delta + d_\delta,\ \delta = 1, 2.$$

In the case of periodic boundary conditions in both directions, or combined with Neumann boundary conditions (i.e. $\mathrm{bc}_1,\ \mathrm{bc}_2 \in \{\mathrm{per/per},\ \mathrm{neu/neu}\}$) the matrix can be not invertible when $\tilde{c} = 0$ and a procedure which will be detailed thereafeter has to be added to deal with such case.
We now give an example of construction, for a specific boundary condition. The vector $\mathbf{V}^{\mathrm{per/per,neu/dir}}$ can be constructed as follows :

$$V^{\mathrm{per/per,neu/dir}}_{i+(j-1)N_1} = V_{i+(j-1)N_1} + V_{N_1+i+(j-1)N_1},\ i = 1,\dots,d_1,\ j = 1,\dots,N_2+d_2-1,$$
$$V^{\mathrm{per/per,neu/dir}}_{i+(j-1)N_1} = V_{i+(j-1)N_1},\ i = d_1+1,\dots,N_1,\ j = 1,\dots,N_2+d_2-1,$$

and other quantities can be computed in a similar fashion. In the case of matrix $\mathbf{M}^{\mathrm{per/per,per/per}}$, the matrix is not invertible when $\tilde{c} = 0$ and a further procedure to deal which such case will be adopted thereafter. Solution of the system is defined as $\mathbf{U}^{\mathrm{bc}_1,\mathrm{bc}_2} \in \mathbb{R}^{N^1_{\mathrm{bc}_1} N^2_{\mathrm{bc}_2}}$, and we can get finally $\mathbf{U} \in \mathbb{R}^{(N_1+d_1)(N_2+d_2)}$. Again, as example, from $\mathbf{U}^{\mathrm{per/per,neu/dir}}$, we can get $\mathbf{U}$ as follows:

$$U_{i+(j-1)N_1} = 0,\ i = 1,\dots,N_1+d_1,\ j = N_2+d_2,$$
$$U_{i+(j-1)N_1} = U^{\mathrm{per/per,neu/dir}}_{i-N_1+(j-1)N_1},\ i = N_1+1,\dots,N_1+d_1,\ j = 1,\dots,N_2+d_2-1.$$

Now, we detail also the case $\mathrm{bc}_1 = \mathrm{per/per}$ and $\mathrm{bc}_2 = \mathrm{dir/dir}$:

$$V^{\mathrm{per/per,dir/dir}}_{i+(j-1)N_1} = V_{i+jN_1} + V_{N_1+i+jN_1},\ i = 1,\dots,d_1,\ j = 1,\dots,N_2+d_2-2,$$
$$V^{\mathrm{per/per,dir/dir}}_{i+(j-1)N_1} = V_{i+jN_1},\ i = d_1+1,\dots,N_1,\ j = 1,\dots,N_2+d_2-2,$$
$$U_{i+(j-1)N_1} = 0,\ i = 1,\dots,N_1+d_1,\ j = 1, N_2+d_2,$$
$$U_{i+(j-1)N_1} = U^{\mathrm{per/per,dir/dir}}_{i-N_1+(j-2)N_1},\ i = N_1+1,\dots,N_1+d_1,\ j = 2,\dots,N_2+d_2-1.$$

3.1.4 Additional step, when the matrix is not invertible

In the next section, we use the Poisson equation with $\tilde{c} = 0$ in (6). In that case, dealing with periodic boundary conditions (or Neumann boundary conditions as seen before), in order to have a solution, the right hand side should satisfy the compatibility condition

$$\int_Q \tilde{f}_h |\sqrt{g}\ |d\eta = 0 \tag{11}$$

and then when this condition is satisfied the solution is known upon an additive constant. In order to have unicity of the solution, we impose

$$\int_Q \tilde{\Phi}_h |\sqrt{g}\ |d\eta = 0. \tag{12}$$

We omit here the subscripts $^{\mathrm{per/per,per/per}}$ on $\mathbf{M}, \mathbf{U}$ and $\mathbf{V}$, as we deal with this specific boundary condition case, in order to simplify notations. So, for $\mathbf{V}$ with compatibility condition $\sum_{k,\ell=1}^{N_1,N_2} V_{k+(\ell-1)N_1} = 0$, the solution $\mathbf{U}$ then satisfies

$$\mathbf{M}\mathbf{U} = \mathbf{V},\ B_h^\top \mathbf{U} = 0, \tag{13}$$

with

$$(B_h)_K = \int_Q B_{k,\boldsymbol{\tau}_1} B_{\ell,\boldsymbol{\tau}_2} |\sqrt{g}\ |d\eta,\ K = k+(\ell-1)N_1,\ k = 1,\dots,N_1,\ \ell = 1,\dots,N_2.$$

Penalization method, Lagrange multipliers or conjugate gradient method can be used to solve this problem.

Penalization method

It consists in replacing $\tilde{c} = 0$ in (6), by $\tilde{c} = \varepsilon$, where $\varepsilon > 0$ is a small parameter. Denoting by $\tilde{\Phi}_h^\varepsilon$ the corresponding solution, we get,

$$\varepsilon \int_Q \tilde{\Phi}_h^\varepsilon |\sqrt{g}\,|d\eta = \int_Q \tilde{f}_h |\sqrt{g}\,|d\eta,$$

summing (9) over $k = 1, \ldots, N_1$ and $\ell = 1, \ldots, N_2$ and using the partition of unity of the B-splines. Now, for $\varepsilon > 0$ and having (11), we get $\int_Q \tilde{\Phi}_h^\varepsilon |\sqrt{g}\,|d\eta = 0$, which is coherent with (12). As (11) may not be true, we can enforce it, replacing $\tilde{f}_h$ by $\tilde{f}_h - \lambda$, such that (11) is true for $\tilde{f}_h - \lambda$, that is taking $\lambda = \int_Q \tilde{f}_h |\sqrt{g}\,|d\eta / \int_Q |\sqrt{g}\,|d\eta$.

Lagrange multipliers method

We are lead to find $(\mathbf{U}, \lambda)$ with multiplier $\lambda \in \mathbb{R}$ solution of

$$\left(\begin{array}{c|c} \mathbf{M} & B_h \\ \hline B_h^T & 0 \end{array}\right) \left(\begin{array}{c} \mathbf{U} \\ \lambda \end{array}\right) = \left(\begin{array}{c} \mathbf{V} \\ 0 \end{array}\right). \tag{14}$$

Note that, by adding the first $N_1 N_2$ lines of the system (14), we get

$$\lambda \int_Q |\sqrt{g}\,|d\eta = \int_Q \tilde{f}_h |\sqrt{g}\,|d\eta,$$

so that when the compatibility condition (11) is satisfied, we obtain $(\mathbf{U}, 0)$ as solution and $\mathbf{U}$ is then solution of (13). If condition (11) is not satisfied, we can make the same correction than for the case of the penalization method; but it is not necessary as this formulation automatically finds this solution and the multiplier λ is the same than the λ of the correction described in the penalization method. With respect to the previous method, we have the additionnal cost to deal with a bigger matrix, but there is no parameter ε to fix. In practice, we will use this method.

Conjugate gradient method

Using the conjugate gradient method, we just have to ensure that the right hand side is of zero mean (that is (11) holds), which is again ensured by doing the correction, as before. In this case, we have the advantage not to have to construct a bigger matrix, but the method remains iterative.

3.1.5 Practical implementation

Previous descriptions have permitted to describe how to get the solution of the Poisson solver from a mathematical point of view. Even if detailed, this still does not give the full implementation. We have used here local, global and *local to global* indices. After that, we can construct the matrix in CSR format, and

add a constraint in the same format, when it is necessary. For the linear solver, we have used UMFPACK [16]. For dealing with different boundary conditions, we have adopted a strategy based on the use of several $1D$ vectors of integers containing boundary condition information for a given direction $\eta_\delta,\ \delta \in \{1,2\}$. This permits to reduce the combinatory of the multidimensional case (here 2D case).

3.2 The Semi-Lagrangian method for the guiding center model

Now, we consider the transport part, which is done by a semi-Lagrangian method and the coupling with the Poisson solver. We first give the basic elements of a semi-Lagrangian scheme and then detail the whole algorithm. Let $s,t \in \mathbb{R}$. The characteristics $(\gamma_{1,s}(t),\ \gamma_{2,s}(t))$ associated with equation (3) are solution of

$$\frac{\partial \gamma_{1,s}(t)}{\partial t} = \frac{\partial_{\eta_2}\tilde{\Phi}(t,(\gamma_{1,s}(t),\gamma_{2,s}(t)))}{\sqrt{g(\gamma_{1,s}(t),\gamma_{2,s}(t))}},\ \gamma_{1,s}(s)=\eta_1,$$
$$\frac{\partial \gamma_{2,s}(t)}{\partial t} = -\frac{\partial_{\eta_1}\tilde{\Phi}(t,(\gamma_{1,s}(t),\gamma_{2,s}(t)))}{\sqrt{g(\gamma_{1,s}(t),\gamma_{2,s}(t))}},\ \gamma_{2,s}(s)=\eta_2.$$

In the sequel, we will write classically

$$\Gamma(t;\eta,s) = \begin{bmatrix}\gamma_{1,s}(t)\\ \gamma_{2,s}(t)\end{bmatrix},\ \eta=(\eta_1,\eta_2).$$

The distribution function $\tilde{f}$ is constant along the characteristic curves which reads

$$\tilde{f}(t,\Gamma(t;\eta,s)) = \tilde{f}(s,\Gamma(s;\eta,s)) = \tilde{f}(s,\eta), \qquad \forall\, t,s,\eta. \tag{15}$$

We define the $2D$ mesh points as $\eta_{ij}=(\eta_{1,i},\eta_{2,j})$ for $i=1,\dots,N_1+1$ and $j=1,\dots,N_2+1$. Using the property (15), the classical semi-Lagrangian method, or backward (BSL: Backward Semi Lagrangian see article [34]) is divided into two steps to compute the distribution function $\tilde{f}_{ij}^{n+1}=\tilde{f}(t_{n+1},\eta_{ij})$ at time t_{n+1} from the distribution function $\tilde{f}_{ij}^{n}=\tilde{f}(t_n,\eta_{ij})$ at time t_n:
For each mesh point η_{ij}

1. Calculating $\Gamma(t_n;\eta_{ij},t_{n+1})$ the value of the characteristic at time t_n which is equal to η_{ij} at time t_{n+1} (we solve the characteristics backward in time).

2. As the distribution function solution of the guiding center equation reads:

$$\tilde{f}_{ij}^{n+1} = \tilde{f}^n(\Gamma(t_n;\eta_{ij},t_{n+1})),$$

and since usually the point $\Gamma(t_n;\eta_{ij},t_{n+1})$ is not a point of the logical grid, the value of $\tilde{f}_{ij}^{n+1}$ is obtained by interpolation of the function $\tilde{f}(t_n,\cdot)$ at mesh points $\Gamma(t_n;\eta_{ij},t_{n+1})$ for $i=1,\dots,N_1+1$ and $j=1,\dots,N_2+1$ at time t_n.

The interpolation that is used here is cubic splines, as in [34]. For the computation of the origin of the characteristics, we use Verlet algorithm, and cubic splines for the interpolation of the advection field $(\partial_{\eta_2}\tilde{\Phi}, -\partial_{\eta_1}\tilde{\Phi})/\sqrt{g}$. The derivatives of $\tilde{\Phi}$ are computed using cubic splines (we could have used here the B-spline decomposition of arbitrary degree d, but we restrict ourselves here to the cubic splines case, that corresponds to $d = 3$). In this way, we ensure conservation of mass at first order, see [24]. A predictor-corrector method is also used for having a second order in time scheme: first predict the advection field at time $t_n + \Delta t/2$, and then use it to evolve the solution for time t_n to time $t_{n+1} = t_n + \Delta t$.

Computational algorithm

We now detail the whole algorithm. Unknowns are $\tilde{f}^n_{i,j} \simeq \tilde{f}(t_n, \eta_{1,i}, \eta_{2,j})$. We write $\tilde{f}^n$ the cubic splines reconstruction from the interpolating conditions

$$\tilde{f}^n(\eta_{1,i}, \eta_{2,j}) = \tilde{f}^n_{i,j},\ i = 1, \dots, N_1 + 1,\ j = 1, \dots, N_2 + 1, \tag{16}$$

together with boundary conditions. Here, we will specify them for each direction and use periodic or Hermite. For Hermite, in direction δ the derivative is set to 0 at the boundary $\eta_{\delta,\min}$ and $\eta_{\delta,\max}$, $\delta = 1, 2$. Other *interpolators* could be used (see [24] for other reconstructions of third degree). Spline interpolation of arbitrary degree would be a natural choice, in the coupling with the B-splines finite element solver. The way to compute the spline coefficients has then to be provided. For the cubic splines case, we refer to [33], using an algorithm defined in [35]. Note that there can be conditions so that the system is invertible (more precisely, for even degree and even N_δ, the system not invertible in the periodic case). Position of the knots and number of knots may also be changed. We have not considered here such options and have preferred to stick to the classical cubic splines case, that is widely used in the context of semi-Lagrangian schemes.

1. **Initialization:**
   - $\tilde{f}^0_{i,j} = \tilde{f}^0(\eta_{1,i}, \eta_{2,j}),\ i = 1, \dots, N_1 + 1,\ j = 1, \dots, N_2 + 1.$
2. **Time loop:**
   - **Prediction:** $t_n \to t_n + \Delta t/2$
     - compute $\tilde{\Phi}_h$, solving the Poisson equation, using $\tilde{f}_h = \tilde{f}^n$.
     - evaluate $\tilde{\Phi}_h$ at grid points $\tilde{\Phi}^n_{i,j} = \tilde{\Phi}_h(\eta_{1,i}, \eta_{2,j}),\ i = 1, \dots, N_1 + 1,\ j = 1, \dots, N_2 + 1$
     - compute $\frac{\partial \tilde{\Phi}^n}{\partial \eta_1}, \frac{\partial \tilde{\Phi}^n}{\partial \eta_2}$ at grid points, using derivatives of the cubic splines of $\tilde{\Phi}^n$
     - For each $i = 1, \dots, N_1 + 1,\ j = 1, \dots, N_2 + 1$, find the feet $\eta^*_{ij} = (\gamma_1(t^n),\ \gamma_2(t^n))$ of the characteristics

$$\frac{\partial \gamma_1(t)}{\partial t} = \frac{1}{\sqrt{g}} \frac{\partial \tilde{\Phi}^n}{\partial \eta_2}, \quad \frac{\partial \gamma_2(t)}{\partial t} = -\frac{1}{\sqrt{g}} \frac{\partial \tilde{\Phi}^n}{\partial \eta_1},$$

ending at grid point $(\gamma_1(t^{n+1/2}),\ \gamma_2(t^{n+1/2})) = (\eta_{1,i}, \eta_{2,j})$.

- Interpolate at feet of characteristics: $\tilde{f}_{i,j}^{n+1/2} = \tilde{f}^n(\eta_{ij}^*),\ i = 1, \ldots, N_1 + 1,\ j = 1, \ldots, N_2 + 1$.
- compute $\tilde{\Phi}_h$, solving the Poisson equation, using $\tilde{f}_h = \tilde{f}^{n+1/2}$.
- evaluate $\tilde{\Phi}_h$ at grid points $\tilde{\Phi}_{i,j}^{n+1/2} = \tilde{\Phi}_h(\eta_{1,i}, \eta_{2,j}),\ i = 1, \ldots, N_1 + 1,\ j = 1, \ldots, N_2 + 1$
- compute $\frac{\partial \tilde{\Phi}^{n+1/2}}{\partial \eta_1}, \frac{\partial \tilde{\Phi}^{n+1/2}}{\partial \eta_2}$ at grid points, using derivatives of the cubic splines of $\tilde{\Phi}^{n+1/2}$

- **Correction:** $t_n \to t_{n+1}$
  - For each $i = 1, \ldots, N_1 + 1,\ j = 1, \ldots, N_2 + 1$, find the feet $\eta_{ij}^* = (\gamma_1(t^n),\ \gamma_2(t^n))$ of the characteristics

$$\frac{\partial \gamma_1(t)}{\partial t} = \frac{1}{\sqrt{g}} \frac{\partial \tilde{\Phi}^{n+1/2}}{\partial \eta_2}, \quad \frac{\partial \gamma_2(t)}{\partial t} = -\frac{1}{\sqrt{g}} \frac{\partial \tilde{\Phi}^{n+1/2}}{\partial \eta_1},$$

    ending at grid point $(\gamma_1(t^{n+1}),\ \gamma_2(t^{n+1})) = (\eta_{1,i}, \eta_{2,j})$.
  - Interpolate at feet of characteristics: $\tilde{f}_{i,j}^{n+1} = \tilde{f}^n(\eta_{ij}^*), i = 1, \ldots, N_1 + 1,\ j = 1, \ldots, N_2 + 1$.

Note that, for the specific case where, the elliptic solver is used with $d_1 = d_2 = 3$, we could use directly the vector of splines coefficients $c_{i,j}^{\tilde{\Phi}_h}$, which is the solution of the elliptic solver, to get the derivatives $\frac{\partial \tilde{\Phi}^n}{\partial \eta_1}, \frac{\partial \tilde{\Phi}^n}{\partial \eta_2}$ at grid points.
Using a spline decomposition of $\tilde{f}_h$ permits to save computational time for the computation of the right hand side, as the contribution of the B-splines basis to the right hand side can be precomputed. Then, from $\tilde{f}_h$, we compute its spline coefficients (we consider here the case of cubic splines as described previously, see around (16)) and then we only have a matrix vector product to perform.

4 Numerical results

Kelvin-Helmholtz instability in a periodic box with Colella mesh

We refer for example to [14] for this test case. The initial distribution f_0 is given by the formula :

$$f_0(x, y) = \sin(y) + \beta \cos(\sigma x)$$

where $\beta = 0.015$ and $\sigma = 0.5$. Periodic conditions are considered both in x and y direction. The domain is $[0, L_x] \times [0, L_y]$, with $L_x = \frac{2\pi}{\sigma},\ L_y = 2\pi$. B-splines of degree $d = 3$ ($d_1 = d_2 = 3$) are taken for the Poisson solver. We test here the

robustness of the numerical method on a Colella mesh [11] in order to see the influence of the mesh. The mapping is given by

$$x(\eta_1,\eta_2) = \eta_1 + \alpha \sin(\frac{2\pi}{L_x}\eta_1)\sin(\frac{2\pi}{L_y}\eta_2),\ y(\eta_1,\eta_2) = \eta_2 + \alpha \sin(\frac{2\pi}{L_x}\eta_1)\sin(\frac{2\pi}{L_y}\eta_2),$$

for $(\eta_1,\eta_2) \in [0,L_x]\times[0,L_y]$ and $0 \leq \alpha < 1$. We take here $\alpha \in \{10^{-6},\ 0.25,\ 0.5,\ 0.75,\ 0.9\}$, $2^i \times 2^i$ grids, for $i = 7,8,9$ and time step $\Delta t \in \{2^{-i},\ i = 3,\ldots,7\}$. Final time T is set to $T = 100$.

We illustrate the distorted meshes on Figure 2: we show examples of meshes for $\alpha \in \{0.25,\ 0.5,\ 0.75,\ 0.9\}$. A 32×32 grid is used on the left of the Figure and a finer grid 128×128 is used on the right . We observe that the mesh becomes more and more distorted as α increases and approaches 1. The case $\alpha = 10^{-6}$ is not plotted, as we cannot distinguish it with a uniform mesh.

On Figure 3, we plot distribution function $f(t,x,y)$ at time $t = 50$ for $\alpha \in \{0.25,\ 0.5,\ 0.75\}$ (from top to bottom). Time step is fixed to $\Delta t = 2^{-3}$. In order to see the influence of the number of points in the grid, we consider a quite coarse grid 128×128 on the left, and a finer grid 512×512 on the right. We observe that the solution on the coarse grid is more diffusive with less details than the solution on the finer grid. We can see that the solution is not much sensible with the different values of α and the results are also coherent with those of [24]. Note that in [24], the Poisson solver is different (it uses a MUDPACK solver) and the semi-Lagrangian method is validated on a rotation test case.

On Figure 4, we plot distribution function $f(t,x,y)$ at time $t = 50$ in the more distorted case $\alpha = 0.9$. We consider again the coarse grid 128×128 on the left, and the finer grid 512×512 on the right. In order to see the influence of the time step, we take $\Delta t = 2^{-3}$, on the top, $\Delta t = 2^{-5}$, on the middle and $\Delta t = 2^{-7}$ on the bottom. Here the results are different.

-There are always more details in the finer mesh, but we observe that $\Delta t = 2^{-3}$ is not small enough (which was fine for less distorted meshes) as it leads to a different solution (Figures on the top), both on the fine and coarse mesh. We also observe appearance of oscillations on the fine mesh (top right).

-When the time step becomes smaller, we find the solution of the previous figure, for the fine mesh, which confirms that the solution of the previous figure was good.

-For the coarse mesh, we observe oscillations that increase when the time step get smaller; this is due to the fact that we use cubic spline interpolation which behave bad when the time step becomes small (see [24, 10]).

-For the fine grid, we also observe a begin of oscillations (bottom right).

In conclusion, we have to take a sufficiently small time step, in order to get enough accurate solution; on the other hand, the time step should not be too small to prevent from numerical oscillations. By refining the mesh, the range of

valid time steps (not too big and not too small) becomes bigger.

On Figure 5, we plot the time evolution of mass, electric energy, L^1, L^2 and L^∞ norms (from top to bottom) which should be theoretically conserved. We consider here on each plot $\alpha \in \{10^{-6}, 0.25, 0.5, 0.75\}$. Grids of size $N \times N$, with $N \in \{128, 256, 512\}$ are used (from left to right). Time step is fixed $\Delta t = 2^{-3}$. We observe that mass is better preserved when α becomes smaller; the bad mass conservation for big α is not much sensible to the space discretization. It is linked to the time discretization. Electric energy is surprisingly better conserved when α is large in the case $N = 128$. Refining the grid leads to a better electric energy conservation, with errors of same amplitude with respect to α. We remark more oscillations of electric energy in time, when α is big. L^1 and L^2 norms have a classical and expected behaviour which here do not much depend on α. L^∞ norm is not well preserved, for all the cases; in the simulation we see appearance of peaks which have luckily not much influence on the other quantities.

So, to conclude, the main concern is the loss of mass conservation for the not-too-much deformed meshes; one remedy is to take a smaller time step as we will see on next Figure (see also [24]).

On Figure 6, we plot the same quantities, for the case $\alpha = 0.9$. This time on each plot, different values of time steps are used: $\Delta t \in \{2^{-i},\ i = 3, \dots 7\}$. The plots share similar behaviour but there are some differences. We observe, as just previously said, better mass conservation as Δt diminishes, this almost independently of the number of points of the grid. We see that the case $\Delta t = 2^{-3}$ strongly differ form the other values, confirming previous remarks on Figure 4, where we have seen that the solution differ: the time step is clearly too big to get the right solution. Electric energy is also better preserved as Δt decreases up to a certain point. If Δt is too small, we observe numerical instabilities that appear, especially on the 256×256 grid. Looking at the solution, we see that the values of the peaks become bigger and bigger and as they are so big, they interfere with the solution and all the diagnostics (not only the L^∞ norm). On the coarse grid, we do not observe this phenomenon; maybe the grid is too coarse which permits to get a more diffusive solution. On the finest grid, we have less problems: except at the end of the simulation; strangely, the problem appears for $\Delta t = 2^{-6}$ and not $\Delta t = 2^{-7}$. This may come from the peaks, whose behaviour seems to be random (a small perturbation of the grid or number of points can eliminate them [17]). To conclude, we are able to simulate also the most distorted case $\alpha = 0.9$, at the price of taking a sufficiently small Δt; interpolation method using cubic splines is not so adequate when the time step becomes too small; a better option would be to use Hermite interpolation with reconstruction of the derivatives of odd order (see [24]).

Diocotron instability in an annular domain with sheared mesh

We refer to [15, 29, 12, 23] for this test case. Initial condition is

$$f(t=0,x,y) = \begin{cases} 0, \ r_{\min} \le r < r_-, \\ 1+\epsilon\cos(\ell\theta), \ r_- \le r < r_+, \\ 0, \ r_+ \le r \le r_{\max}, \end{cases} \tag{17}$$

with $r = \sqrt{x^2+y^2}, \theta = \arctan(y/x)$. We take $\epsilon = 10^{-6}$, $r_{\min} = 1$, $r_{\max} = 10$ and $\ell = 3$. The sheared mesh is given by

$$x(\eta_1,\eta_2) = \eta_1\cos(\eta_2+\alpha\eta_1), \ y(\eta_1,\eta_2) = \eta_1\sin(\eta_2+\alpha\eta_1),$$

for $(\eta_1,\eta_2) \in [r_{\min}, r_{\max}] \times [0, 2\pi]$. The shear parameter α is set to $\alpha = 0.1$ (for $\alpha = 0$, it is the standard polar mesh). We compare the results with a polar code (classical BSL method on polar grid as in [12]), for which slopes of linear phase has been validated according to analytical formula (see [15, 29, 12] for details). We consider two cases called Dirichlet and Neumann-Dirichlet; the first one corresponds to set Dirichlet boundary condition on inner and outer boundary; and the second stands for setting Neumann boundary condition at the inner boundary ($r = r_{\min}$) and Dirichlet boundary condition at the outer boundary ($r = r_{\max}$).

On Figure 7, we represent the shear mesh ($\alpha = 0.1$) on the left and standard polar mesh on the right on a 32×64 grid, in order to see how the mesh is distorted with respect to the polar mesh ($\alpha = 0$). We skip here the $1d$ diagnostics and a study with respect to the shear parameter, due to space limitation and as such study has already been done in the previous case. Also in the Neumann-Dirichlet case, we do not know if the different quantities are preserved (see [12]), when the solution begins to touch the inner boundary (which is observed around $t = 140$). We show here on Figure 8 the distribution function $f(t,x,y)$ at this time $t = 140$ (left, curvilinear code on shear mesh; right polar code, on polar geometry) for Dirichlet case (top) and Neumann-Dirichlet case. We observe that boundary conditions really change the solution, and it is thus a good test for checking if the boundary conditions are good coded. Analytical slopes of the linear phase are 0.3307 for Dirichlet and 0.2267 for Neumann-Dirichlet case; it is thus coherent that the non linear phase is more advanced on the Dirichlet case (more details are present). We observe same results for the two codes, which permits to validate the curvilinear code. We make a study of the efficiency (a rough measure of the number of points treated) of the code using different degrees for the elliptic solver and different grid sizes and compare it to the polar code; results are shown on Table 1. A "x" symbol is put, when we were not able to run the simulation (UMFPACK raised there an error indicated that not enough memory is available; we have used a local cluster for performing the simulations and the code is sequential). In order to overcome this problem, we can use iterative or parallel solvers; we just checked here on a single Poisson

$N_x \times N_y$	shear $d=2$	shear $d=3$	shear $d=4$	shear $d=5$	shear $d=6$	shear $d=7$	polar
$2^5 \times 2^5$	0.20	0.19	0.16	0.14	0.13	0.11	0.61
$2^5 \times 2^6$	0.20	0.17	0.15	0.14	0.12	0.10	0.60
$2^6 \times 2^6$	0.20	0.18	0.13	0.11	0.10	0.090	0.70
$2^6 \times 2^7$	0.18	0.12	0.14	0.087	0.082	0.082	0.53
$2^7 \times 2^7$	0.18	0.17	0.12	0.090	0.083	0.077	0.70
$2^7 \times 2^8$	0.16	0.12	0.12	0.087	0.091	0.069	0.69
$2^8 \times 2^8$	0.19	0.14	0.12	0.079	0.075	0.062	0.67
$2^8 \times 2^9$	0.18	0.12	0.09	0.074	x	x	0.61
$2^9 \times 2^9$	0.16	0.12	x	x	x	x	0.53
$2^9 \times 2^{10}$	0.15	x	x	x	x	x	0.39
$2^{10} \times 2^{10}$	x	x	x	x	x	x	0.43
$2^{10} \times 2^{11}$	x	x	x	x	x	x	0.38

Table 1: Diocotron instability case. Efficiency $= (2 \cdot N_x \cdot N_y \cdot T/\Delta t)/(time \cdot 10^6)$ for different grids of size $N_x \times N_y$; comparison between polar code (last column) and curvilinear code with elliptic solver of degree $d \in \{2, 3, \ldots, 7\}$ in each direction on shear geometry, with $\alpha = 0.1$ (second to penultimate column). Parameters are $\Delta t = 2^{-3}$, $T = 200$; Neumann-Dirichlet boundary condition. *time* is the total time of the simulation, including initialization and diagnostics (time diagnostics every time step and distribution function and potentiel Φ at times $i \cdot 10,\ i = 1, \ldots, 20$).

test (but not on the whole diocotron simulation, as it would last too long) that the conjugate gradient method gives a correct solution, when the UMFPACK solver fails. Note that for the polar code, FFT in angle direction and second order finite differences in radial direction are used. No special effort of optimization is done for both codes; but the results may give an idea of the cost of the new curvilinear code. The curvilinear code is about 3 times slower for $d = 2$ and 5 times slower for $d = 3$; using higher degrees is also possible at higher but not exorbitant cost; note that the degree is only a parameter in the code, which may be optimized depending on the simulation. The new price seems acceptable, as we gain in generality; benefits will be even better for kinetic simulations (Vlasov-Poisson 4D, drift/gyro-kinetic codes), as the main overhead of cost concerns the Poisson solver, whose cost remains small with respect to the advection (the latter including the velocity variable(s)).

5 Conclusion and perspectives

Semi-Lagrangian simulations have been performed on curvilinear grids for the guiding center model, with both Poisson and advection solved on the curvilinear grid. Influence of time step, deformation and grid sizes are studied. Efficiency of the code is shown for different degrees of the elliptic solver and mesh sizes. For not too large deformations of the mesh, we are able to reproduce the right

results and we notice that time step has to be diminished in order to handle more distorted meshes and better conserve mass. On the other hand, the time step has not to be too small to prevent from appearance of oscillations, as we use here cubic splines for the interpolation. The new code is validated on both cartesian and polar geometries with deformed meshes and different boundary conditions. Thanks to UMFPACK, we obtain a rather good efficiency of the code.
Further work will be continued in order to deal with other geometries (like D-shape or polygonal shape) and finer meshes, using parallel and/or iterative solvers for the elliptic solver. The latter code could be enhanced in order to permit repeated knots in the interior of the domain [25], which may be for example more compatible for other interpolations semi-Lagrangian schemes. We are also interested in better computing the characteristics when the mesh is strongly distorted; this could have an impact on the numerical results. Finally, we can also compare with Eulerian approaches which are subject to a CFL condition.

References

[1] J. Abiteboul, G. Latu, V. Grandgirard, A. Ratnani, E. Sonnendrücker, A. Strugarek, *Solving the Vlasov equation in complex geometries*, Esaim. Proc., Oct 2011 (CEMRACS'2010), Vol. 32, p103–117.

[2] V.C. Azevedo, M. M. Oliveira, *Efficient Smoke Simulation on Curvilinear Grids*, Pacific Graphics 2013, Vol 32(7) (2013),

[3] A. Back, A. Crestetto, A. Ratnani, E. Sonnendrücker, *An axisymmetric PIC code based on Isogeometric Analysis,* Esaim. Proc., Oct 2011 (CEMRACS'2010), Vol. 32, p118–133.

[4] A. Back, E. Chacon-Golcher, V. Grandgirard, A. Ratnani, E. Sonnendrücker, *A 4D Vlasov-Poisson solver on an arbitrary curvilinear grid*, preprint

[5] N. Besse, E. Sonnendrücker, *Semi-Lagrangian schemes for the Vlasov equation on an unstructured mesh of phase space,* J. Comput. Phys., 191 (2003), 341-376.

[6] M. Bostan, *The Vlasov-Maxwell system with strong initial magnetic field: guiding-center approximation*, Multiscale Model. Simul., 6(3): 1026–1058, 2007.

[7] C. DeBoor, *A practical guide to splines*, Springer-Verlag, New York, Applied Mathematical Sciences 27, 2001.

[8] J. P. Braeunig, N. Crouseilles, M. Mehrenberger, E. Sonnendrücker, *Guiding-center simulations on curvilinear meshes,* Discrete

and Continuous Dynamical Systems Series S, Volume 5, Number 3, June 2012

[9] C. Caldini-Queiros, *Analyse mathématique et numérique de modèles gyrocinétiques*, PhD, University of Besançon, France, 15 November 2013.

[10] F. Charles, B. Després, M. Mehrenberger, *Enhanced convergence estimates for semi-lagrangian schemes Application to the Vlasov-Poisson equation,* SIAM J. Numer. Anal. 2013, 51(2), 840-863.

[11] P. Colella, M. R. Dorrand, J.A. Hittinger and D.F. Martin, *High-order, finite-volume methods in mapped coordinates*, J. Comput. Phys. **230**(8), 2952–2976 (2011).

[12] N. Crouseilles, P. Glanc, S. A. Hirstoaga, E. Madaule, M. Mehrenberger, J. Pétri, *A new fully two-dimensional conservative semi-Lagrangian method: applications on polar grids, from diocotron instability to ITG turbulence,* hal-00977342, accepted in EPJD, topical issue of Vlasovia 2013.

[13] N. Crouseilles, M. Lemou, F. Méhats, *Asymptotic preserving schemes for highly oscillatory kinetic equations,* J. Comput. Phys. **248**, 287–308 (2013).

[14] N. Crouseilles, M. Mehrenberger, E. Sonnendrücker, *Conservative semi-Lagrangian schemes for Vlasov equations,* Journal of Computational Physics 229 (2010), 1927–1953.

[15] R.C. Davidson, *Physics of non neutral plasmas*, Addison-Wesley, Redwood City, CA,1990.

[16] T. A. Davis, *Algorithm 832*, ACM Transactions on Mathematical Software 30 (2) (2004): 196–199.

[17] E. Faou, *Personal communication.*

[18] F. Filbet, C. Yang, *Conservative and non-conservative methods based on Hermite weighted essentially-non-oscillatory reconstruction for Vlasov equations*, Journal of Computational Physics 279 (2014), 18–36.

[19] F. Filbet, C. Yang, *Mixed semi-Lagrangian/finite difference methods for plasma simulations*, arXiv:1409.8519

[20] E. Frénod, S. Hirstoaga, M. Lutz, E. Sonnendrücker, *Long time behaviour of an exponential integrator for a Vlasov-Poisson system with strong magnetic field*, Communications in Computational Physics, accepted, preprint version at https://hal.archives-ouvertes.fr/hal-00974028 (2014).

[21] N. Fujimatsu, K. Suzuki, *New interpolation technique for the CIP method on curvilinear coordinates,* Journal of Computational Physics, vol. 229, pp. 5573–5596, 2010.

[22] V. Grandgirard, M. Brunetti, P. Bertrand, N. Besse, X. Garbet, P. Ghendrih, G. Manfredi, Y. Sarazin, O. Sauter, E. Sonnendrücker, et al., *A drift-kinetic semi-lagrangian 4d code for ion turbulence simulation,* Journal of Computational Physics, vol. 217, no. 2, pp. 395?423, 2006.

[23] S. Hirstoaga, E. Madaule, M. Mehrenberger, J. Pétri, *Semi-Lagrangian simulations of the diocotron instability*, hal-00841504.

[24] A. Hamiaz, M.Mehrenberger, H. Sellama, E. Sonnendrücker *The semi-Lagrangian method on curvilinear grids*, submitted.

[25] Jorek-Django, a new finite element framework for computational plasma physics, http://ratnani.org/jorek_doc/

[26] M. Mehrenberger, M. Bergot, V. Grandgirard, G. Latu, H. Sellama, E. Sonnendrücker *Conservative Semi-Lagrangian solvers on mapped meshes*, hal-00759823.

[27] M. Mehrenberger, L. Mendoza, C. Prouveur, E. Sonnendrücker, *Solving the guiding-center model on a regular hexagonal mesh*, submitted to proceedings of CEMRACS'2014, preprint version at https://hal.archives-ouvertes.fr/hal-01117196

[28] L. Mendoza, E. Sonnendrücker, V. Grandgirard *Solving the Vlasov equation using the semi-Lagrangian method on multiple patches for the GYSELA code*, HEPP Colloquium at Strausberg, Germany, September 2013, poster.

[29] J. Pétri, *Non-linear evolution of the diocotron instability in a pulsar electrosphere: 2D PIC simulations,* Astronomy & Astrophysics, May 7, 2009.

[30] SELALIB, http://selalib.gforge.inria.fr/

[31] M. Shoucri, *Comment on "Three-dimensional stability of drift vortices [Phys. Plasmas 3, 160 (1996)]"*, Phys. Plasmas, Vol.3, No 11, November 1996, comments.

[32] M. Shoucri, *A two-level implicit scheme for the numerical solution of the linearized vorticity equation*, Int. J. Numer. Meth. Eng. 17, 1525–1538 (1981).

[33] E. Sonnendrücker, *Numerical Methods for the Vlasov-Maxwell equations,* submitted.

[34] E. Sonnendrücker, J. Roche, P. Bertrand and A. Ghizzo, *The Semi-Lagrangian Method for the Numerical Resolution of the Vlasov Equation*, J. Comput. Phys. **149**, 201-220 (1999).

[35] Michael Unser, Akram Aldroubi, and Murray Eden, *Fast b-spline transforms for continuous image representation and interpolation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(3):277–285, 1991.

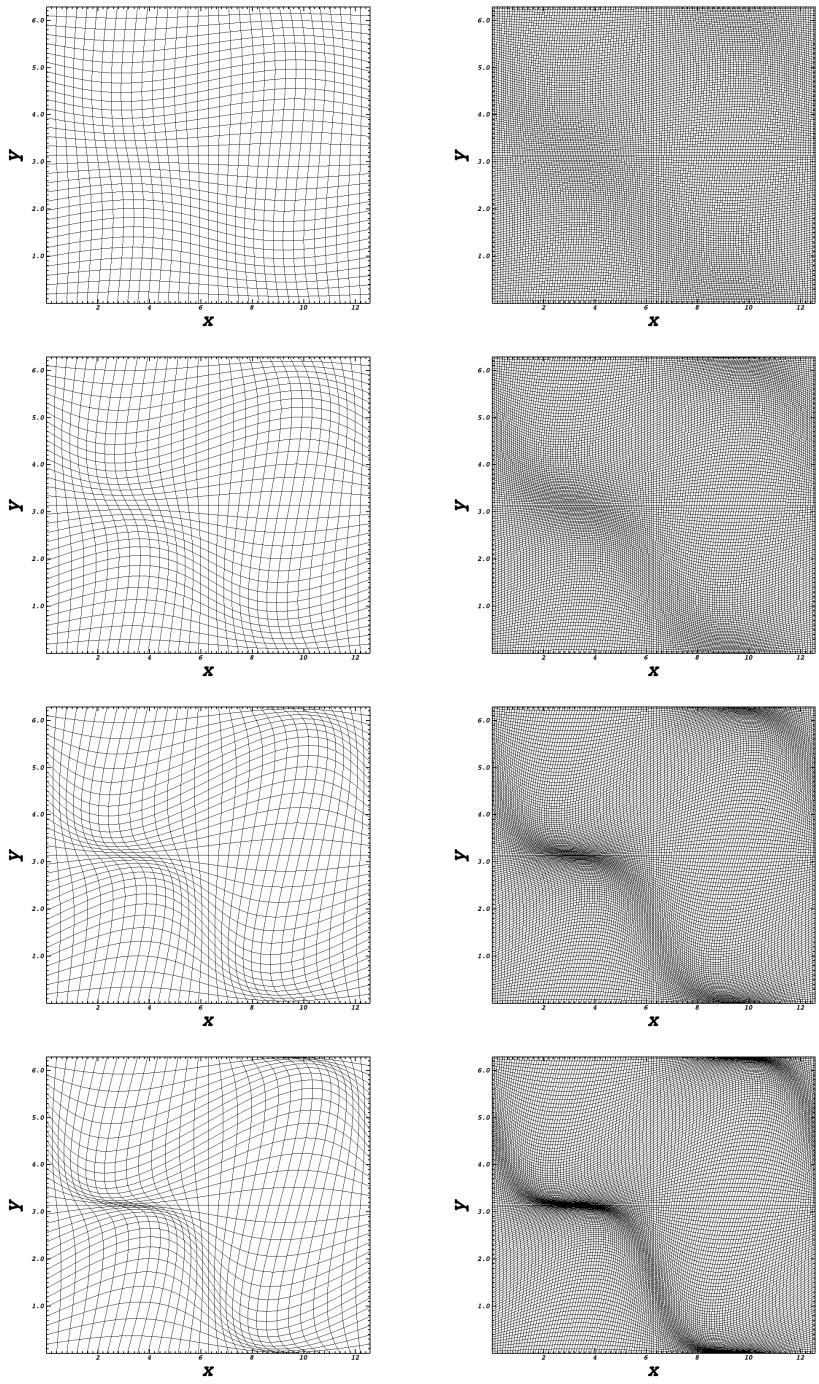

Figure 2: Deformed mesh: grid size 32×32 (left) and 128×128 (right), with different values of α (from top to bottom: 0.25, 0.5, 0.75 and 0.9)

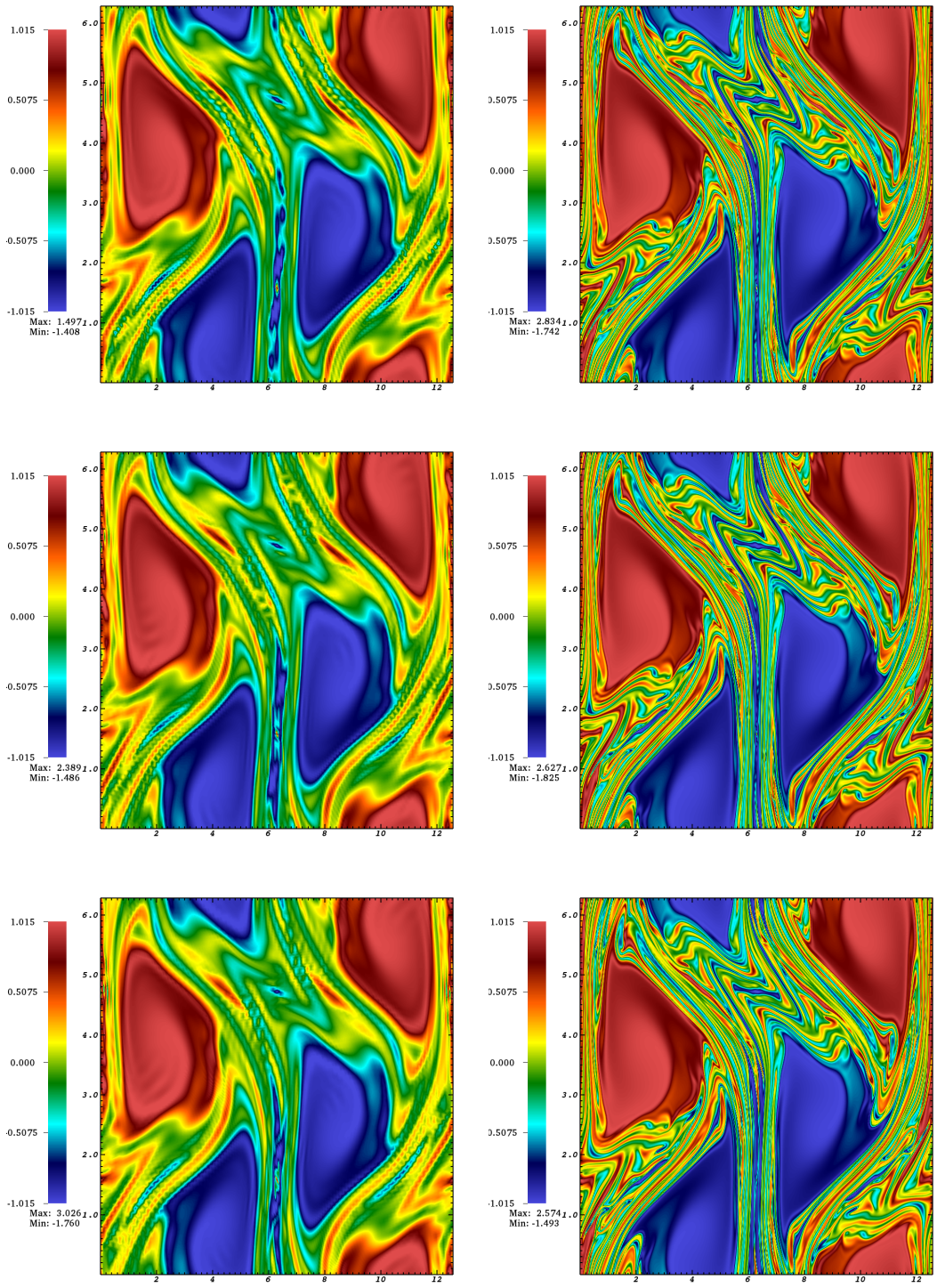

Figure 3: Distribution function $f(t = 50, x, y)$ for different grid sizes (left: 128×128, right: 512×512) and α (top: 0.25, middle: 0.5, bottom: 0.75); $\Delta t = 1/2^3$.

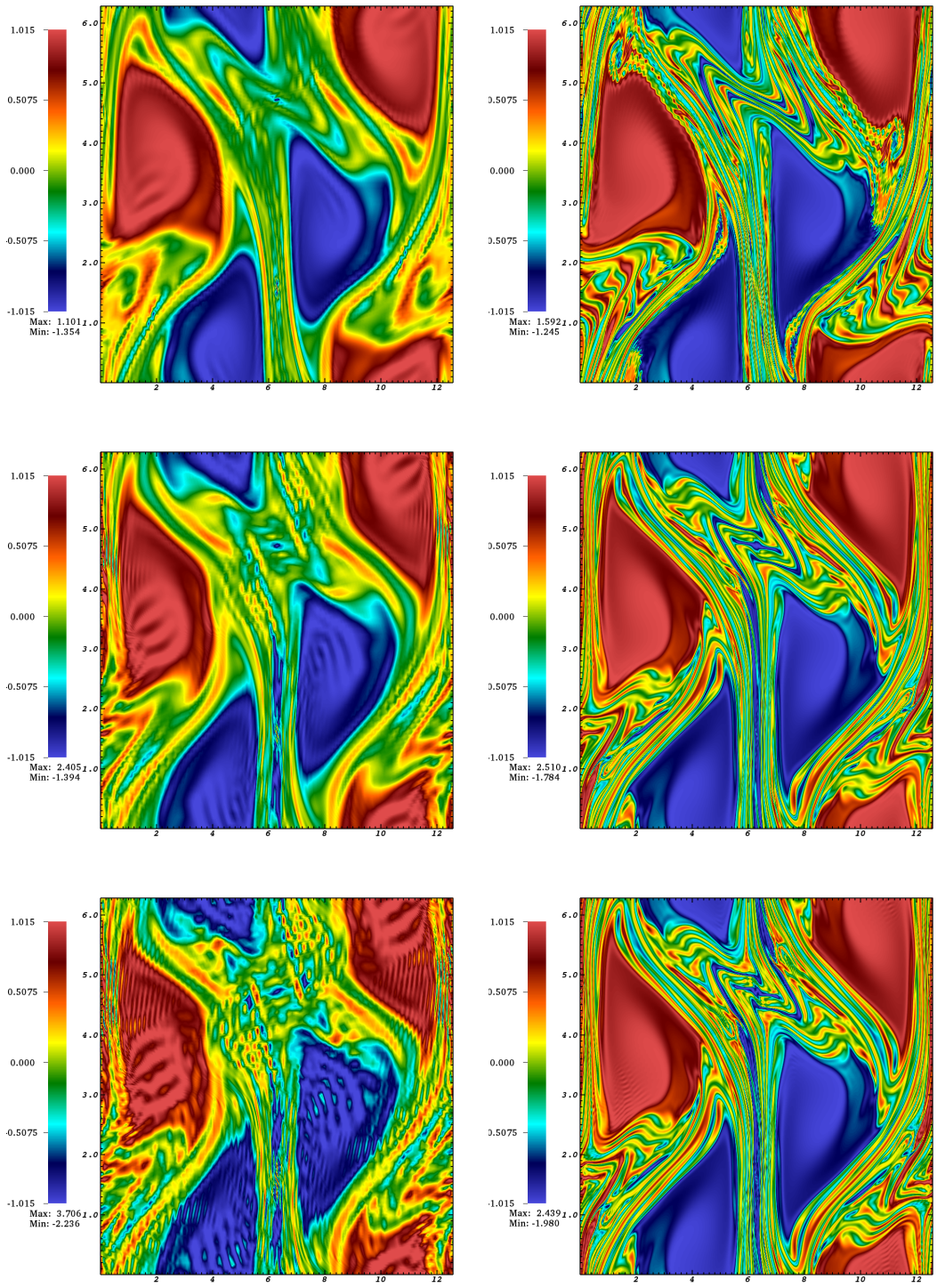

Figure 4: Distribution function $f(t = 50, x, y)$ for different grid sizes (left: 128×128, right: 512×512) and Δt (top: $1/2^3$, middle: $1/2^5$, bottom: $1/2^7$); $\alpha = 0.9$.

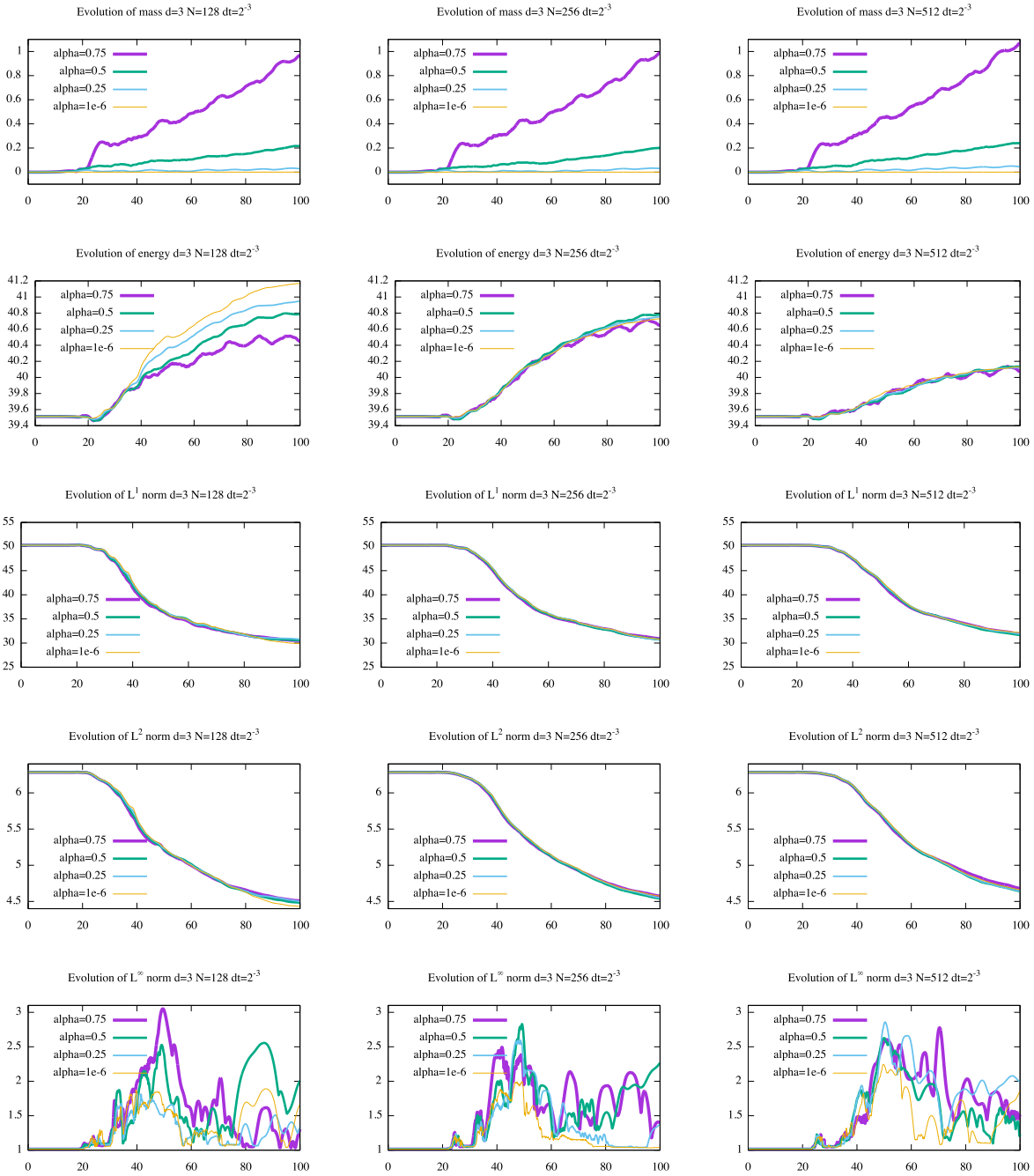

Figure 5: Time evolution of theoretically conserved quantities on $N \times N$ grid (left: $N = 128$, middle, $N = 256$ and right $N = 512$): from top to bottom: mass, electric energy, L^1, L^2 and L^∞ norm; $\Delta t = 2^{-3}$ and $\alpha \in \{0.75,\ 0.5,\ 0.25,\ 10^{-6}\}$.

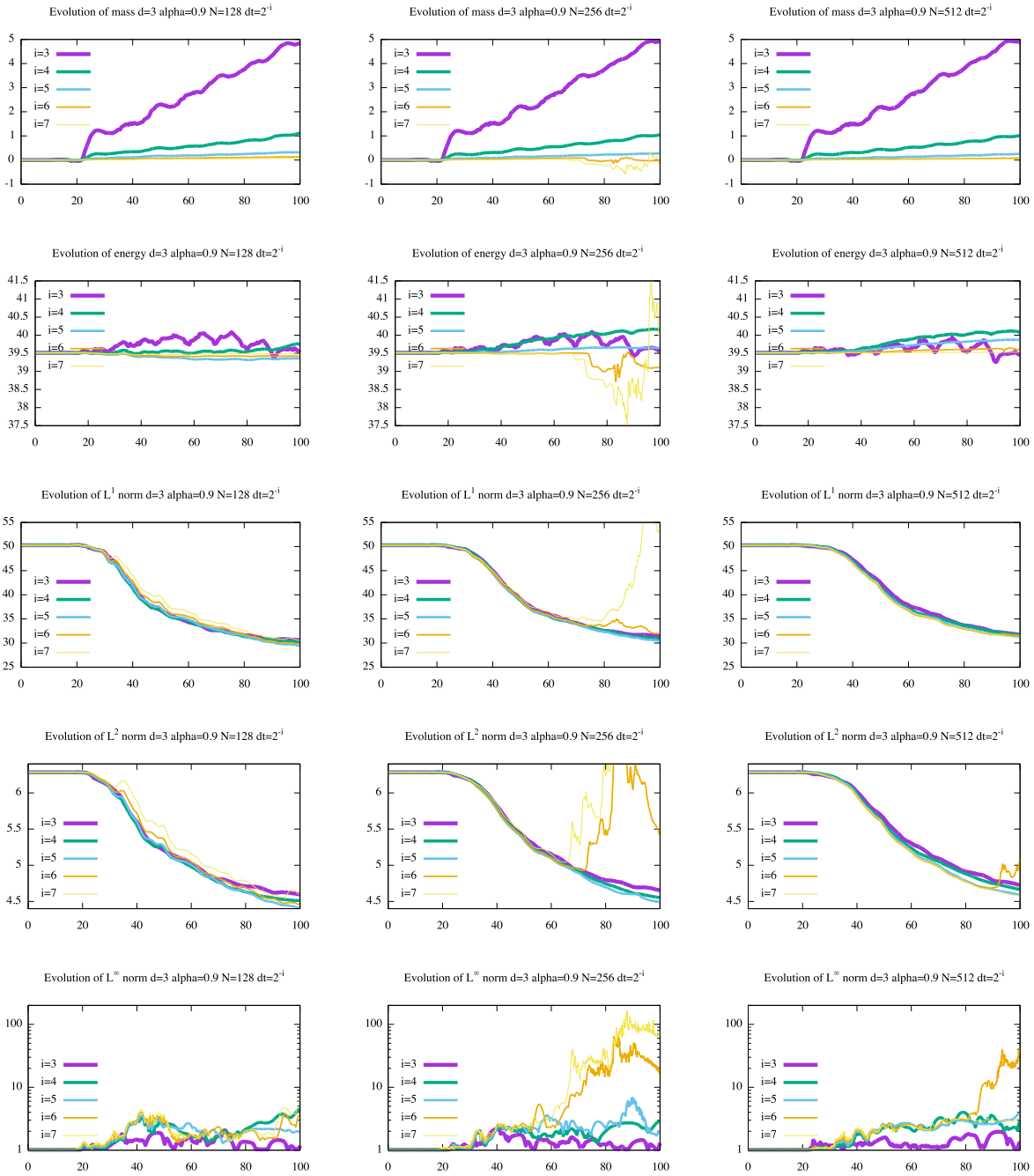

Figure 6: Time evolution of theoretically conserved quantities on $N \times N$ grid (left: $N = 128$, middle, $N = 256$ and right $N = 512$): from top to bottom: mass, electric energy, L^1, L^2 and L^∞ norm; $\alpha = 0.9$ and $\Delta t = 2^{-i}$, $i = 3, \ldots, 7$.

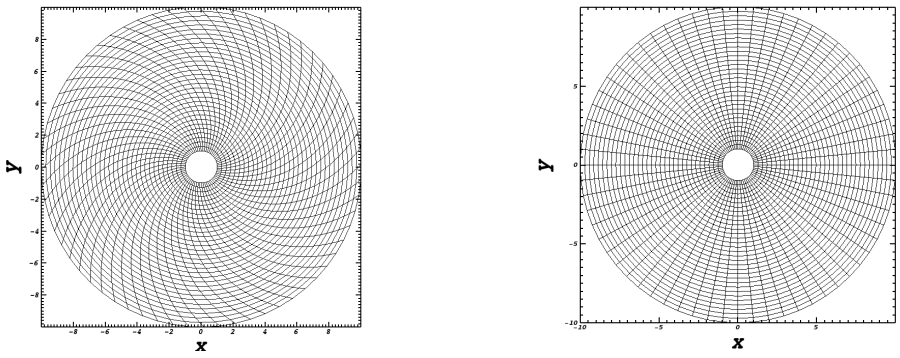

Figure 7: Diocotron test case. Example of mesh for 32×64 grid: shear $\alpha = 0.1$ (left) and polar (right).

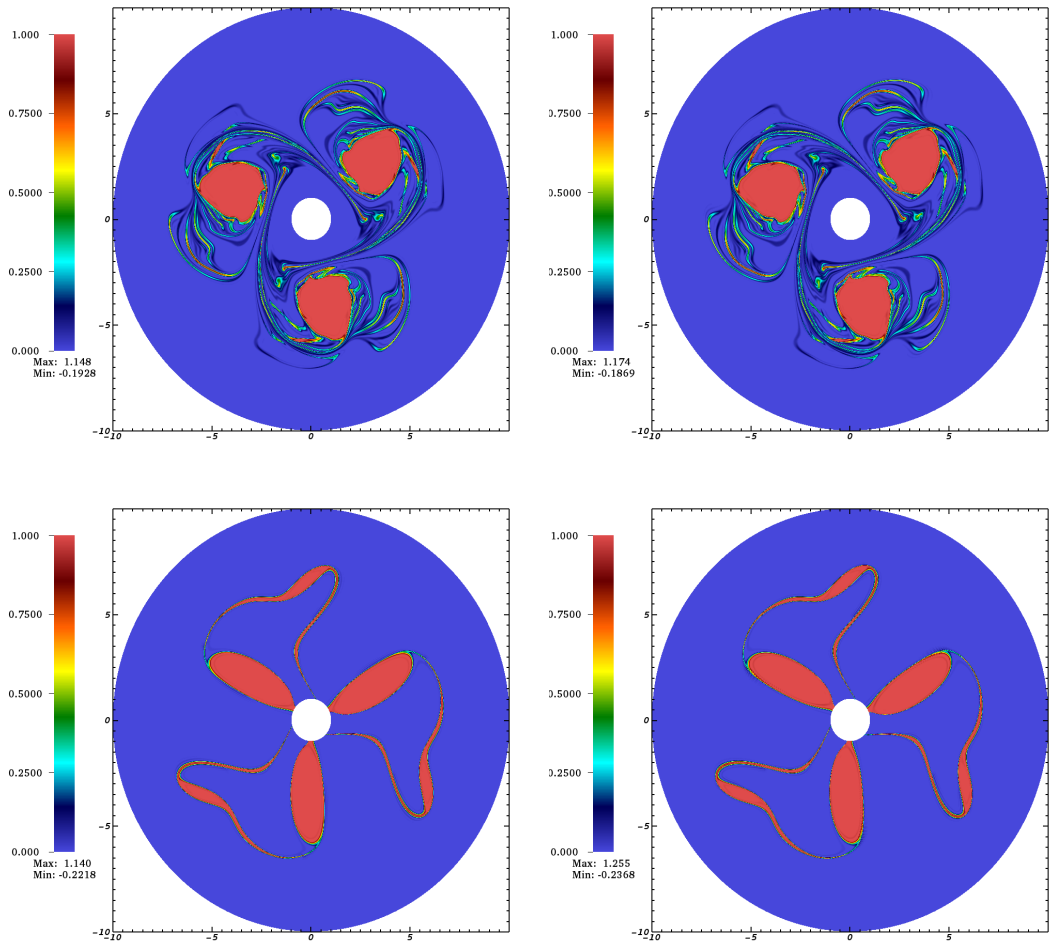

Figure 8: Diocotron instability test case. Distribution function $f(t = 140, x, y)$ with Dirichlet (top) and Neumann-Dirichlet case (bottom). Left: curvilinear code on shear mesh with $\alpha = 0.1$; right: polar code. $\Delta t = 1/2^3$ and 512×1024 grid; $d = 2$ for the elliptic solver.