

HOPE: High Order Program for Kinetic Equations

Documentation

Francis Filbet

October 18, 2026

Abstract

This document describes the four codes of the HOPE folder: a discontinuous Galerkin/Hermite spectral code for the one-dimensional Vlasov–Poisson system (DG-VP1D); an Eulerian code for the two-dimensional guiding-center and Kelvin–Helmholtz models (EULERIAN-METHOD-2D), a particle-in-cell (PIC) code for the two-dimensional Vlasov–Poisson equation with a strong magnetic field (PIC-2D), and a 3D particle-in-cell code in a torus with a strong external magnetic field (PIC-3D). For each code, we present the model, the numerical method, the input and output files, and test cases.

Contents

1	Organization and usage	2
1.1	Content of the folder	2
1.2	Dependencies and compilation	2
1.3	Format of the input files	3
2	Mesh and geometry	3
2.1	The mesh file	4
2.2	Interior, ghost and exterior points	4
2.3	Rectangle	5
2.4	Disk	6
2.5	D-shaped section	7
3	DG-VP1D	9
3.1	Model	9
3.2	Hermite spectral form	9
3.3	Numerical method	10
3.4	Input data and output	10
3.5	Test cases from the papers	11

4	EULERIAN-METHOD-2D	14
4.1	Models	14
4.2	Numerical method	14
4.3	Programs, input data and output	15
4.4	Test cases	15
5	PIC-2D	21
5.1	Model	21
5.2	Numerical method	21
5.3	Input data	22
5.4	Output	23
5.5	Test cases	23
6	PIC-3D	26
6.1	Model	26
6.2	Numerical method	27
6.3	Input data	28
6.4	Output	28
6.5	Results: simulation <code>test_Dshape</code>	29

1 Organization and usage

1.1 Content of the folder

folder	model	method
DG-VP1D	1D Vlasov–Poisson	discontinuous Galerkin in x , Hermite spectral in v
EULERIAN-METHOD-2D	2D guiding center, Kelvin–Helmholtz	Eulerian, RK4 and finite differences
PIC-2D	2D Vlasov–Poisson, $\mathbf{B} = B_z/\varepsilon \mathbf{e}_z$	particles, second-order IMEX
PIC-3D	3D Vlasov–Poisson in a torus, strong $\mathbf{B}$	particles, semi-implicit

The 2D and 3D codes share the same organization:

- a main program `main*.cpp` that reads the data, builds the mesh, the Poisson solver and the unknowns, then runs the time loop;
- classes written in `.hpp` headers: geometry (`Geometry2D`, `Square`, `Disk`, `DShape`), mesh (`Mesh`, `GhostPoint2D`), Poisson solver, unknowns and time scheme (`Variables_*`);
- a small library allocating arrays with arbitrary index ranges (`Array.hpp`, `Array.cpp`);
- input files `*.dat`, read at start-up in the current directory.

The common files `Array`, `Geometry2D`, `Disk`, `DShape`, `GhostPoint2D` and `Parameter` are the same in these three folders. Two files differ between the 2D and 3D versions: the mesh class `Mesh.hpp` of the 3D version also reads the major radius of the torus, and `Config.hpp` defines π exactly in the 2D versions ($4 \arctan 1$, instead of 3.14159265 in the 3D version). The two 2D folders also share the same Poisson solvers (`PoissonDirichlet_2D`, `PoissonPeriodicDirichlet_2D`, `Poisson_nonlinear_2D`), whose files are identical. The one-dimensional code DG-VP1D is independent of the others: it consists of a main program, a single class `H_DG_1D.hpp` and its own version of `Array`, and reads its parameters in the file `geom2d.dat` (Section 3).

Each folder contains the source files and a subfolder `runs`, which holds the input files, the executables and the results of the simulations.

1.2 Dependencies and compilation

The codes are compiled with `mpicxx` (MPICH) and use:

- **MUMPS** (sparse direct solver) for the Poisson equations with Dirichlet boundary conditions;
- **FFTW3** for the Fourier transform in the toroidal direction of the 3D code;
- **BLAS/LAPACK** (OpenBLAS), and **OpenMP** (`libomp`) for the parallel loops of the 2D codes.

The code DG-VP1D does not use MPI: it is compiled with the GCC compiler of MacPorts (`g++-mp-15`), which supports OpenMP, and uses FFTW3 for the diagnostics. On macOS with MacPorts, all these libraries are installed in `/opt/local`, a path set by the variable `MPI_DIR` of the Makefiles. Compile with

```
cd DG-VP1D           ; make           # executable main
cd EULERIAN-METHOD-2D ; make all      # main_kh, main_gc_disk, main_gc_D
cd PIC-2D            ; make           # executable main
cd PIC-3D            ; make           # executable main
```

and clean with `make clean`. The executables read their data in the current directory: the simulations are run in the subfolder `runs` (or in a copy of it), which contains the `.dat` files, for instance with `mpirun -np 8 ./main`.

Reproducibility. The installed version of MUMPS uses several OpenMP threads. The order of floating-point operations may then change from one run to another, which produces relative differences of order 10^{-10} . To obtain bitwise identical results from one run to another, launch with `OMP_NUM_THREADS=1`.

1.3 Format of the input files

Each line of a `.dat` file contains a label followed by `:` and a value. The program reads words up to the first word equal to `:`, then the value: the `:` must therefore be separated from the label by a space. The order of the lines is fixed; lines of stars are only used for layout. The data are printed on screen at start-up.

- `data.dat`: number of time steps per time unit ($1/\Delta t$), number of time units (final time) and period of the diagnostics;
- `Xd.dat`, `Xdisk.dat`, `Xsquare.dat`: mesh and geometry (Section 2);
- `physics.dat` (PIC codes): physical parameters, geometry, initial data and seed of the random number generator.

2 Mesh and geometry

The two-dimensional domain D (the computational domain of the 2D codes, the poloidal cross-section in the 3D code) is not meshed: it is embedded in a rectangle covered by a uniform Cartesian grid, and the boundary is taken into account by ghost points located outside D . This approach, introduced for kinetic equations in [8, 9] and used for the guiding-center model in [10], avoids the singularity of polar coordinates at the origin, separates the treatment of the boundary from the numerical scheme, and applies to any geometry for which one knows how to decide whether a point is inside D and how to project it on the boundary. Three geometries are available: a rectangle, a disk and a D-shaped section.

2.1 The mesh file

The mesh and the geometry are read in a file `X*.dat`, opened by the main program (`Xsquare.dat`, `Xdisk.dat` or `Xd.dat`). Each line contains a label, the character `:` separated from the label by a space, and a value (Section 1.3); the lines of stars are only used for layout. The file contains, in this order:

1. the number of points n_x and the bounds $x_{\min}$, $x_{\max}$ of the grid in x , then the same three values in y and in z . In the 2D codes, z is not used and one takes $n_z = 2$; in the 3D code, z is the toroidal angle ϕ , with $n_z - 1$ cells on $[-\pi, \pi]$;
2. the number of points and the bounds in v_x , v_y and v_z . In the PIC codes, $n_{v_x}n_{v_y}$ (2D) or $n_{v_x}n_{v_y}n_{v_z}$ (3D) is the number of particles per grid cell, the velocities themselves being sampled from the initial distribution; the bounds are not used. The Eulerian code does not use these values;
3. in the 3D code only, the major radius R_0 of the torus (**Toroidal radius**);
4. the parameters of the geometry, described in the following subsections.

The grid points are

$$x_i = x_{\min} + i \Delta x, \quad y_j = y_{\min} + j \Delta y, \quad \Delta x = \frac{x_{\max} - x_{\min}}{n_x - 1}, \quad \Delta y = \frac{y_{\max} - y_{\min}}{n_y - 1},$$

with $-m_d \leq i \leq n_x + m_d - 1$ and $-m_d \leq j \leq n_y + m_d - 1$, where $m_d = 5$ is the number of additional layers of points on each side.

Size of the grid. The rectangle $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ must contain the domain D with a margin of a few cells: the ghost points and the interior points used to extrapolate them (Section 2.2) must belong to the grid. When the program cannot find three interior points on a grid line crossed by the normal, it prints **Error in searching extrapolation points** and stops. This happens when the domain touches the edge of the grid, but also, on fine grids, for some positions of the grid lines with respect to the boundary, because the fallback to lower degree described in Section 2.2 is not implemented in the present version. For the disk of radius 10, for instance, 257 points on $[-10.4, 10.4]^2$ (the default of **EULERIAN-METHOD-2D/runs/Xdisk.dat**) work, but 261 points on the same interval do not; it is then enough to change slightly the bounds or the number of points.

2.2 Interior, ghost and exterior points

Each grid point (x_i, y_j) is given a type, stored in the array **points** of the class **Mesh**:

- 1 for an interior point, that is, a point of D ;
- a negative value for the ghost points, outside D and close to the boundary. The points of type -1 are the neighbors of interior points in the directions x and y : they are the ones where the boundary condition of the Poisson equation is imposed. The next layers $(-2, -3, \dots)$ are needed by the wider stencils of the transport schemes;

- 0 for the other points, called exterior points, which are not used.

The value of an unknown u (the potential ϕ or the density ρ) at a ghost point $\mathbf{x}_g$ is obtained by extrapolation along the normal to the boundary, following [10] (Figure 1). Let $\mathbf{x}_w$ be the point of the boundary associated with $\mathbf{x}_g$ (the foot of the normal), $h = \min(\Delta x, \Delta y)$, and $\mathbf{x}_h$, $\mathbf{x}_{2h}$ the points of the normal at distance h and $2h$ from $\mathbf{x}_w$, towards the interior of D . The ghost value is the quadratic extrapolation

$$u(\mathbf{x}_g) = c_0 u(\mathbf{x}_w) + c_1 u(\mathbf{x}_h) + c_2 u(\mathbf{x}_{2h}),$$

where $u(\mathbf{x}_w)$ is given by the boundary condition, and the weights c_0, c_1, c_2 only depend on the distances between the four points. The values $u(\mathbf{x}_h)$ and $u(\mathbf{x}_{2h})$ are interpolated from interior points. The normal crosses three grid lines, in x or in y depending on its direction, at points P_0^*, P_1^*, P_2^* ; on each line, one takes the three interior points closest to the crossing point. A polynomial $q_2 \in \mathbb{Q}_2$ is built on these nine points and evaluated at $\mathbf{x}_h$ and $\mathbf{x}_{2h}$:

$$u(\mathbf{x}_h) = \sum_{\ell=0}^8 w_{h,\ell} u(\mathbf{x}_\ell), \quad u(\mathbf{x}_{2h}) = \sum_{\ell=0}^8 w_{2h,\ell} u(\mathbf{x}_\ell).$$

When nine interior points cannot be found, for instance near a sharp corner, [10] uses instead a polynomial of degree one on four points (q_1) or of degree zero on one point (q_0). In the present version of the code, this fallback is not implemented: the field `extrapolation_type` of `GhostPoint2D` is declared but not used, and the program stops (Section 2.1). All the weights are computed once, when the mesh is built, by the class `GhostPoint2D`. In the Poisson solvers (see the appendix *Poisson solvers*), the ghost values enter the five-point scheme through these formulas, so that the boundary condition is imposed without changing the stencil.

2.3 Rectangle

The class `Square` describes the rectangle

$$D = [x_0, x_1] \times [y_0, y_1].$$

A grid point is interior when $x_0 \leq x_i \leq x_1$ and $y_0 \leq y_j \leq y_1$. The point of the boundary associated with a ghost point is its orthogonal projection on the closest side: for instance, (x_0, y_j) when $x_i < x_0$. The four parameters are given after the velocity grid; the file `EULERIAN-METHOD-2D/runs/Xsquare.dat`, used for the Kelvin–Helmholtz instability, reads:

```
** Number of points in x      : 129
** Minimum value of x        : 0.00
** Maximum value of x        : 12.
** Number of points in y     : 129
** Minimum value of y        : 0
** Maximum value of y        : 6.28318530717959
** Number of points in z     : 2
** Minimum value of z        : 0
** Maximum value of z        : 1
*****
** Number of points in vx    : 2
** Minimum value of vx      : -1
```

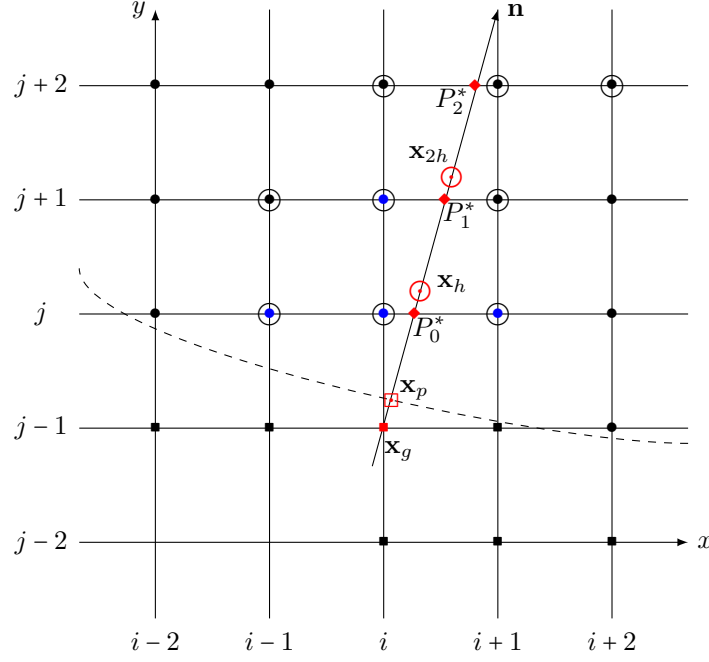


Figure 1: Cartesian mesh near the boundary (from [10]): interior point ($\bullet$), ghost point ($\blacksquare$), point of the boundary, points used for the extrapolation (circles); the dashed line is the boundary. The point $\mathbf{x}_p$ of the figure is the point $\mathbf{x}_w$ of the text.

```

** Maximum value of vx      : +1
** Number of points in vy   : 2
** Minimum value of vy     : -1
** Maximum value of vy     : +1
** Number of points in vz   : 2
** Minimum value of vz     : -1
** Maximum value of vz     : 1
*****
** Parameters for the rectangular domain
*****
** Min x                    : 0
** Max x                    : 12.
** Min y                    : 0
** Max y                    : 6.28318530717959

```

Here the rectangle coincides with the grid, $[0, 12] \times [0, 2\pi]$, since the Kelvin–Helmholtz case is periodic in x and uses its own Poisson solver.

2.4 Disk

The class `Disk` describes the disk of center 0 and radius r ,

$$D = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq r^2\}.$$

A grid point is interior when its distance to the origin is at most r , and the point of the boundary associated with a ghost point $\mathbf{x}_g$ is its radial projection $r \mathbf{x}_g / |\mathbf{x}_g|$, which is the foot

of the normal. The only parameter is the radius, given after the velocity grid, for instance in PIC-2D/runs/Xdisk.dat:

```

** Number of points in x      : 131
** Minimum value of x        : -10.4
** Maximum value of x        : +10.4
** Number of points in y      : 131
** Minimum value of y        : -10.4
** Maximum value of y        : +10.4
** Number of points in z      : 2
** Minimum value of z        : 0
** Maximum value of z        : 1
*****
** Number of points in vx     : 10
** Minimum value of vx       : -5.
** Maximum value of vx       : +5.
** Number of points in vy     : 10
** Minimum value of vy       : -5.
** Maximum value of vy       : +5.
** Number of points in vz     : 2
** Minimum value of vz       : -1
** Maximum value of vz       : 1
*****
** Parameters for the disk domain
*****
** Radius                     : +10.0

```

2.5 D-shaped section

The class `DShape` describes the section bounded by the curve

$$\Gamma(\xi) = \left(r_0 \cos \left(\xi + \arcsin(0.416) \sin \xi \right), 1.66 r_0 \sin \xi \right), \quad \xi \in [0, 2\pi),$$

that is, a section of minor radius r_0 , elongation $\kappa = 1.66$ and triangularity $\delta = 0.416$, centered at the origin. These shape coefficients are fixed in `DShape.hpp`; they are those of the Cerfon–Freidberg equilibrium of the 3D code (see the appendix *Tokamak configuration*, subsection *D-shaped equilibrium: Cerfon–Freidberg solution*). For a grid point $\mathbf{x}$, one computes by Newton’s method the parameter ξ such that $\mathbf{x}$ lies on the half-line from the origin through $\Gamma(\xi)$, that is, the root of

$$F(\xi) = 1.66 x \sin \xi - y \cos \left(\xi + \arcsin(0.416) \sin \xi \right).$$

The point is interior when $|\mathbf{x}| \leq |\Gamma(\xi)|$, and for a ghost point, $\Gamma(\xi)$ is the associated point of the boundary. It is thus the intersection of the boundary with the half-line from the center, which approximates the foot of the normal. The only parameter is the size r_0 (`Size`). In the 2D codes, `Xd.dat` reads:

```

** Number of points in x      : 101
** Minimum value of x        : -10.5
** Maximum value of x        : +10.5
** Number of points in y      : 201
** Minimum value of y        : -17

```

```

** Maximum value of y          : +17
** Number of points in z      : 2
** Minimum value of z         : 0
** Maximum value of z         : 1
*****
** Number of points in vx     : 10
** Minimum value of vx       : -5
** Maximum value of vx       : +5
** Number of points in vy     : 10
** Minimum value of vy       : -5
** Maximum value of vy       : +5
** Number of points in vz     : 10
** Minimum value of vz       : -5
** Maximum value of vz       : +5
*****
** Size                        : 10.

```

In the 3D code, the grid in z describes the toroidal angle and the major radius of the torus comes before the size of the section (file PIC-3D/Xd.dat):

```

** Number of points in x      : 79
** Minimum value of x        : -6.2
** Maximum value of x        : +6.2
** Number of points in y     : 129
** Minimum value of y        : -10.2
** Maximum value of y        : +10.2
** Number of points in z     : 33
** Minimum value of z        : -3.14159265
** Maximum value of z        : +3.14159265
*****
** Number of points in vx    : 2
** Minimum value of vx      : -5
** Maximum value of vx      : +5
** Number of points in vy    : 2
** Minimum value of vy      : -5
** Maximum value of vy      : +5
** Number of points in vz    : 2
** Minimum value of vz      : -5
** Maximum value of vz      : +5
*****
** Toroidal radius           : +20.
*****
** Size                      : 6.

```

With $r_0 = 6$, the section has a half-height $1.66 \times 6 = 9.96$, inside the grid $(-6.2, 6.2) \times (-10.2, 10.2)$.

3 DG-VP1D

3.1 Model

The code solves the one-dimensional Vlasov–Poisson system for electrons, periodic in space, with a neutralizing background of ions:

$$\begin{cases} \partial_t f + v \partial_x f + E \partial_v f = 0, & t \geq 0, \quad x \in (x_{\min}, x_{\max}), \quad v \in \mathbb{R}, \\ \lambda^2 \partial_x E = \rho - \rho_0, \quad E = -\partial_x \phi, \quad \rho = \int_{\mathbb{R}} f dv, \end{cases}$$

where λ is the scaled Debye length (`Quasineutral parameter` in `geom2d.dat`) and the constant ρ_0 ensures that the total charge vanishes. The method is the conservative discontinuous Galerkin/Hermite spectral method of [11], studied in [12].

3.2 Hermite spectral form

The distribution function is expanded in the velocity variable on the asymmetrically weighted Hermite functions

$$\begin{cases} f(t, x, v) = \sum_{n=0}^{N_H-1} C_n(t, x) \Psi_n(v), & \Psi_n(v) = H_n\left(\frac{v}{v_{th}}\right) \frac{e^{-v^2/2v_{th}^2}}{\sqrt{2\pi}}, \\ H_0 = 1, \quad \sqrt{n} H_n(\xi) = \xi H_{n-1}(\xi) - \sqrt{n-1} H_{n-2}(\xi), \quad n \geq 1, \end{cases}$$

where v_{th} is a scaling velocity (`_vs0` in the code). Substituting this expansion in the Vlasov equation gives a hyperbolic system for the coefficients $C_n(t, x)$:

$$\partial_t C_n + v_{th} \left(\sqrt{n+1} \partial_x C_{n+1} + \sqrt{n} \partial_x C_{n-1} \right) - \frac{\sqrt{n}}{v_{th}} E C_{n-1} = 0, \quad 0 \leq n \leq N_H - 1,$$

with $C_{-1} = C_{N_H} = 0$. The density, the momentum and the kinetic energy only involve the first three coefficients:

$$\int_{\mathbb{R}} f dv = v_{th} C_0, \quad \int_{\mathbb{R}} v f dv = v_{th}^2 C_1, \quad \frac{1}{2} \int_{\mathbb{R}} v^2 f dv = \frac{v_{th}^3}{2} (\sqrt{2} C_2 + C_0).$$

Consequently, the truncated system conserves exactly the mass, the momentum and the total energy [11]:

$$\frac{d}{dt} \int C_0 dx = 0, \quad \frac{d}{dt} \int C_1 dx = 0, \quad \frac{d}{dt} \int \left(\frac{v_{th}^3}{2} (\sqrt{2} C_2 + C_0) + \frac{\lambda^2}{2} |E|^2 \right) dx = 0.$$

Filter. To reduce the Gibbs phenomenon of the spectral approximation, the Hou–Li filter is applied to the last third of the modes:

$$\tilde{C}_n = \sigma\left(\frac{n}{N_H}\right) C_n, \quad \sigma(s) = 1 \quad \text{if } s \leq \frac{2}{3}, \quad \sigma(s) = e^{-36 s^{36}} \quad \text{if } s > \frac{2}{3}.$$

The coefficients C_0 , C_1 and C_2 are not modified, so that the conservation properties hold. In the code, the filter is applied at each time step, in the form $\exp(-36 \Delta t (n/N_H)^{36})$.

Weighted L^2 stability. The natural space of the asymmetrically weighted basis is the L^2 space with the weight $e^{v^2/2v_{th}^2}$, in which the Vlasov–Poisson system has no a priori estimate. In [12], the scaling velocity is made time dependent, $v_{th} = 1/\alpha(t)$, with

$$\alpha'(t) = -\frac{\gamma}{2} \|E(t)\|_\infty^2 \alpha(t)^3, \quad \gamma > 0,$$

which gives the stability estimate $\|f(t)\|_{\omega(t)} \leq \|f(0)\|_{\omega(0)} e^{t/4\gamma}$ for the weighted norm with weight $\omega(t, v) = \sqrt{2\pi} e^{(\alpha(t)v)^2/2}$, uniformly in N_H . This adaptation is implemented in `solve_transport` (variables `_alpha` and `dalph`, with $\gamma = \text{CC}$), but is switched off by default ($\gamma = 0$), so that v_{th} stays constant; [12] uses $\gamma = 0.01$. The weighted norm is written in `e.out`.

3.3 Numerical method

- **Space:** discontinuous Galerkin method of degree $k \leq 3$ in each cell for every Hermite coefficient C_n , with the global Lax–Friedrichs flux

$$\hat{g}_n = \frac{1}{2} \left[g_n^- + g_n^+ - a (C_n^+ - C_n^-) \right], \quad g_n = v_{th} (\sqrt{n+1} C_{n+1} + \sqrt{n} C_{n-1}),$$

where $a = v_{th} \sqrt{N_H}$ bounds the spectral radius of the transport operator. In the code, $a = 0$ for $n = 0$, so that the mass is exactly conserved. The modes whose coefficients vanish initially (with their neighbours) are not computed (function `adaptive`).

- **Poisson:** local discontinuous Galerkin method in the same polynomial space, written as the first-order system $E = -\partial_x \phi$, $\lambda^2 \partial_x E = \rho - \rho_0$, with centred fluxes and a penalty on the jumps of ϕ . The block-tridiagonal periodic system is assembled and factorized once (function `getA`). With the flux choices of [11], the semi-discrete scheme conserves mass, momentum and energy.
- **Time:** second-order strong stability preserving Runge–Kutta scheme (Heun), with a Poisson solve at each stage.

3.4 Input data and output

The file `geom2d.dat` contains, in this order: the number of time steps per time unit ($1/\Delta t$), the number of time units, the period of the diagnostics, the number N_x of cells and the bounds of the domain in x , the number N_H of Hermite modes and the bounds in v/v_{th} (used only to reconstruct f on a grid), the degree k of the polynomials, the parameter λ , the amplitude δ of the perturbation of the density, and the number of OpenMP threads (≤ 0 : value of `OMP_NUM_THREADS`). The default file reads:

```
** Number of iterations per unit      : 500
** Number of units                    : 10.
** Frequency of diagnostics           : 0.1
** Number of points in x              : 32
** Minimum value of x                 : -10.
** Maximum value of x                 : +10.
```

```

** Number of points in v          : 64
** Minimum value of v/v_th       : -6.
** Maximum value of v/v_th       : +6.
** Degree of polynomial          : 2
** Quasineutral parameter        : 1.
** Amplitude of perturbation of the density : 0.1
** Number of OpenMP threads (<=0: use OMP_NUM_THREADS) : 1

```

The initial data is chosen in `main.cpp`, by the argument of `initialise` (the code must then be compiled again):

- **0, Landau damping:** $f_0(x, v) = (1 + \delta \cos(kx)) M(v)$, with M the Maxwellian of temperature 1 and $k = 2\pi/(x_{\max} - x_{\min})$;
- **1, two-stream instability:** $f_0(x, v) = \frac{2}{7}(1 + 5v^2) \left(1 + \delta ((\cos 2kx + \cos 3kx)/1.2 + \cos kx)\right) M(v)$;
- **2, bump-on-tail instability:** $f_0(x, v) = f_b(v) (1 + \delta \cos(3kx))$, with a bulk Maxwellian of density 0.9 and temperature 1, and a beam of density 0.1, temperature 0.25 and drift velocity 4.5, and $v_{th} = 1.4$.

The code writes:

- **e.out:** time, variations of the mass and of the momentum, kinetic energy, electric energy, penalty term, variation of the total energy, α , weighted and unweighted L^2 norms of f ;
- **enorm.out:** time, logarithms of $\|E\|_{L^2}$, of $\|E\|_{L^\infty}$ and of the first four Fourier modes of E ;
- **f.<num>:** distribution function on the (x, v) grid; **m.<num>:** ϕ , E , ρ , current and kinetic energy in each cell; **b.<num>:** Hermite spectrum C_n .

3.5 Test cases from the papers

The results below are reproduced from [11] and [12], where they were obtained with versions of this method; they have not been computed again with the present version of the code. The velocity domain is $[-8, 8]$ and the degree is $k = 2$.

Landau damping [11]. Initial data 0 with $\delta = 0.01$, $k = 0.5$ (that is, a domain of length 4π), $N_x = 64$ cells and $N_H = 256$ modes, up to the time 60. Mass, momentum and total energy are conserved up to the machine precision, and the electric field is damped at the rate -0.1533 given by the linear theory (Figure 2). In [11], the convergence orders $k + 1$ in space and 2 in time are also checked on this test.

Two-stream instability [11]. Initial data 1 with $\delta = 0.01$, $k = 0.5$, $N_x = 128$ cells and $N_H = 512$ modes, up to the time 60. The conservations hold up to 10^{-13} , and the growth and saturation of the electric field agree with a semi-Lagrangian reference solution on a 256×512 grid (Figure 3).

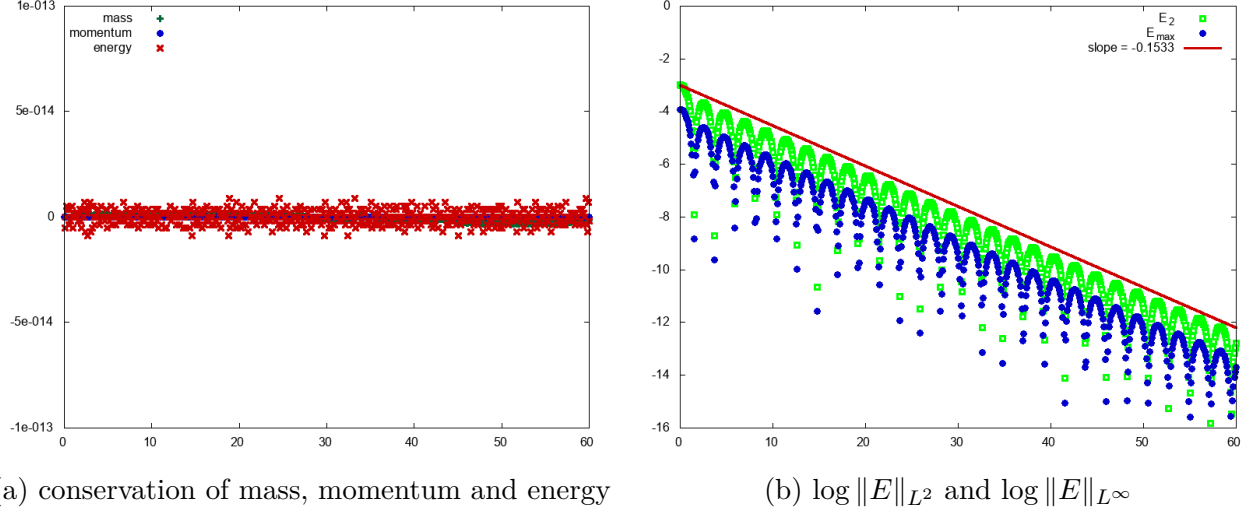


Figure 2: Weak Landau damping ($\delta = 0.01$, $k = 0.5$, $N_x \times N_H = 64 \times 256$), from [11].

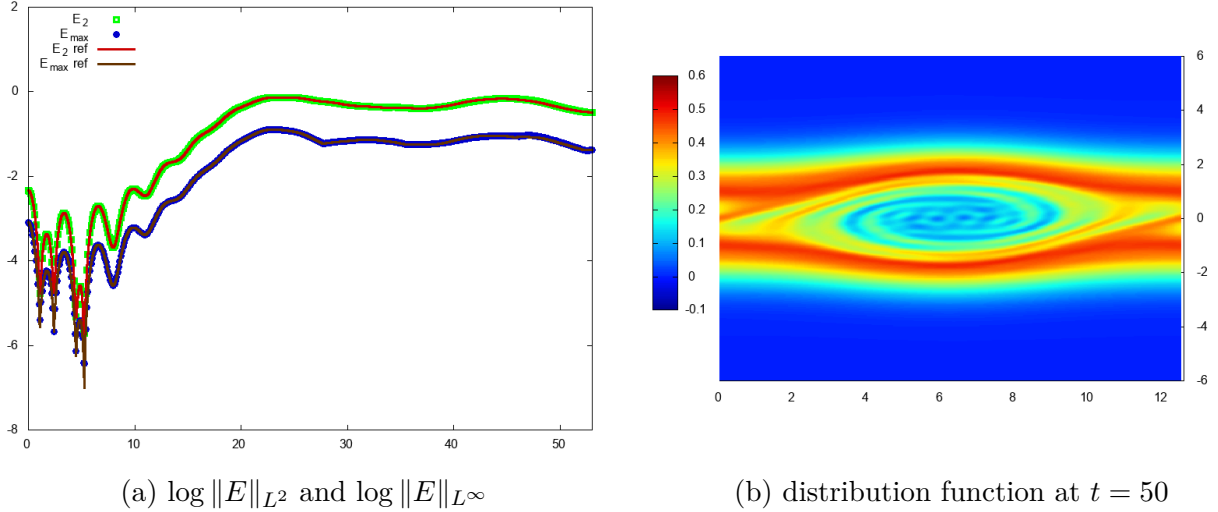
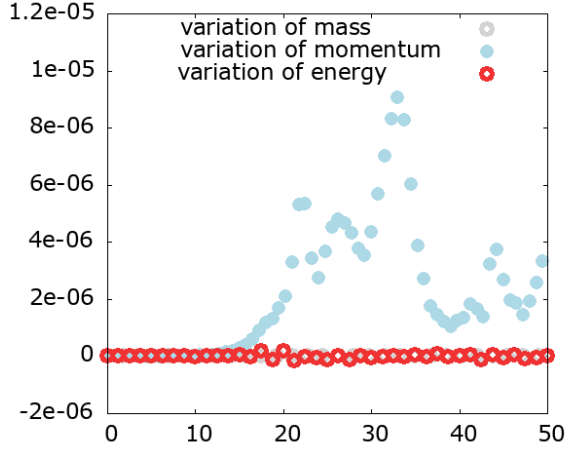


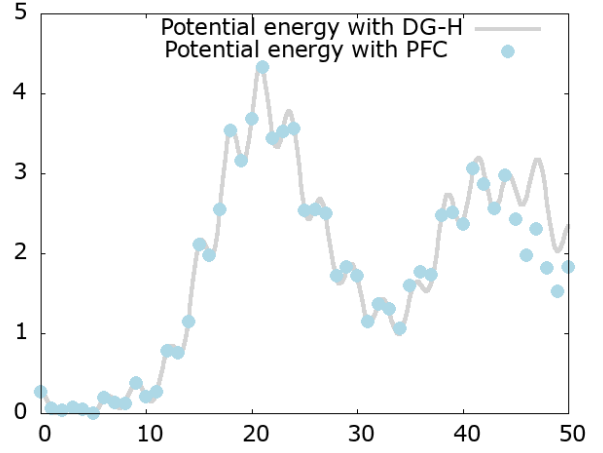
Figure 3: Two-stream instability ($\delta = 0.01$, $k = 0.5$, $N_x \times N_H = 128 \times 512$), from [11].

Bump-on-tail instability [12]. Initial data 2 with $\delta = 0.04$, a domain of length 62, $N_x = 64$ cells, $N_H = 128$ modes, the time-dependent scaling with $\gamma = 0.01$ and $\alpha(0) = 5/7$, up to the time 50; the reference is a positive and flux conservative (PFC) scheme on a 256×1024 grid. Mass and total energy are conserved up to the machine precision, whereas the momentum varies by 10^{-5} . The weighted L^2 norm first grows almost exponentially, then stabilizes, while α slowly decreases (Figure 4).

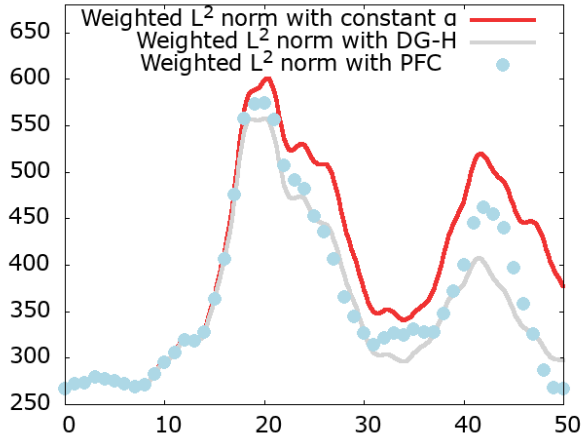
Related work. The Hermite decomposition in velocity is also the basis of the analysis of [13], which studies a finite volume discretization of the linear Vlasov–Fokker–Planck equation written as a hyperbolic system for the Hermite coefficients: the scheme preserves the equilibrium and the weighted L^2 relative entropy, converges exponentially fast to equilibrium (discrete hypocoercivity), and is asymptotic preserving in the diffusive limit.



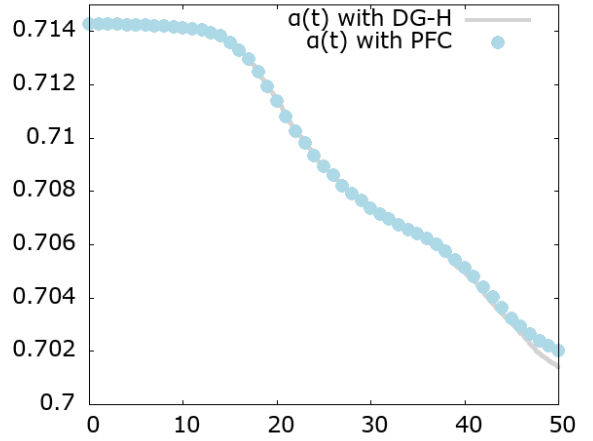
(a) conservation of mass, momentum and energy



(b) $\log \|E\|_{L^2}$



(c) weighted L^2 norm of f



(d) scaling function $\alpha(t)$

Figure 4: Bump-on-tail instability ($N_x \times N_H = 64 \times 128$, $\gamma = 0.01$), from [12].

4 EULERIAN-METHOD-2D

4.1 Models

Guiding center. In the limit $\varepsilon \rightarrow 0$ of the Vlasov–Poisson model of Section 5, the charge density is transported by the electric drift:

$$\begin{cases} \partial_t \rho + \mathbf{U} \cdot \nabla \rho = 0, & \mathbf{U} = \nabla^\perp \phi = (-\partial_y \phi, \partial_x \phi), \\ -\Delta \phi = \rho - \rho_0 \text{ in } D, & \phi = 0 \text{ on } \partial D. \end{cases}$$

Since $\mathbf{U} = -\mathbf{E}^\perp$ with $\mathbf{E} = -\nabla \phi$, this is the limit of the model of Section 5 for a uniform magnetic field $b = 1$. The velocity field is divergence free, and, if one ignores the effect of the boundary, the smooth solutions satisfy [10]

$$\begin{cases} \rho(t, \mathbf{x}) \geq 0 \text{ if } \rho(0, \mathbf{x}) \geq 0, & \frac{d}{dt} \int_D \rho \, d\mathbf{x} = 0, & \frac{d}{dt} \|\rho(t)\|_{L^p(D)} = 0, \quad 1 \leq p \leq \infty, \\ \frac{d}{dt} \int_D |\nabla \phi|^2 \, d\mathbf{x} = 0. \end{cases}$$

These are the quantities written in `Evolution.out`: the mass, the energy $\frac{1}{2} \int \phi \rho = \frac{1}{2} \int |\nabla \phi|^2$, the norms $\|\rho\|_{L^2}$ and $\|\rho\|_{L^4}$, and $\min \rho$. Their variations measure the numerical dissipation: they remain small as long as the solution is resolved, and grow when filaments thinner than the mesh step appear. In [10], the semi-Lagrangian methods are found accurate with large time steps during the linear phase, whereas the conservative finite difference schemes are more appropriate in the nonlinear phase, when filaments appear; the conservative scheme is the default choice of the code.

Kelvin–Helmholtz. The same equations, which are the incompressible Euler equations written with the vorticity and the stream function, are solved on the rectangle $[0, L_x] \times [0, 2\pi]$, periodic in x . The Poisson equation is then solved by `PoissonPeriodicDirichlet_2D` (Appendix).

4.2 Numerical method

- **Transport:** the equation is written in conservative form,

$$\partial_t \rho + \partial_x (U_x \rho) + \partial_y (U_y \rho) = 0.$$

The fluxes are computed by a conservative upwind finite difference scheme of order 1 to 5; order 5 is used (`FD_1D`, `FD_2D`). These schemes are described in [10].

- **Time:** fourth-order Runge–Kutta (`RK4_Trnsprt2D`), with a Poisson solve at each stage.
- **Velocity:** $\mathbf{U}$ is computed from ϕ by fourth-order centered differences.
- **Alternative:** a semi-Lagrangian scheme (`SL_PEC2`) with WENO3, HWENO3 or HWENO5 interpolations (`Itpl2D_*`) is available [10]; it is selected through the commented lines of the main programs.

4.3 Programs, input data and output

program	geometry	mesh	initial data
<code>main_kh</code>	periodic rectangle	<code>Xsquare.dat</code>	shear layer (below)
<code>main_gc_disk</code>	disk	<code>Xdisk.dat</code>	diocotron annulus (below)
<code>main_gc_D</code>	D-shaped section	<code>Xd.dat</code>	close to equilibrium (below)

The initial data are, with θ the polar angle,

$$\left\{ \begin{array}{ll} \text{Kelvin-Helmholtz:} & \rho(0, x, y) = \sin y + 0.05 \sin \frac{y}{2} \cos(k_x x), \quad k_x = \frac{2\pi}{L_x}, \\ \text{diocotron in the disk:} & \rho(0, x, y) = (1 + 10^{-2} \cos 7\theta) e^{-4(r-6.5)^2} \mathbf{1}_{\{5 \leq r \leq 8\}}, \\ \text{D-shaped section:} & \rho(0, x, y) = (1 + 10^{-2} \cos 7\theta) e^{-r^2/8}, \end{array} \right.$$

the last one being a perturbation of a Gaussian profile centered at the origin. For the disk, this is the initial data of the diocotron instability in [10]; `Variables_GC.hpp` also contains, as a comment, a discontinuous annulus $(1 + 10^{-2} \cos 4\theta) \mathbf{1}_{\{4 \leq r \leq 5\}}$, which is much more dissipated by the scheme. The three programs read `data.dat` and their mesh file. They write `Evolution.out`, one line per diagnostic:

- guiding center: time, mass, second moment $\int \rho |\mathbf{x}|^2$, energy $\frac{1}{2} \int \phi \rho$, norms $\|\rho\|_{L^2}$ and $\|\rho\|_{L^4}$, $\|\mathbf{U}\|_{L^2}$, $\max |\mathbf{U}|$, $\min \rho$;
- Kelvin-Helmholtz: time, mass, energy, $\frac{1}{2} \|\mathbf{U}\|_{L^2}^2$, $\|\rho\|_{L^2}$, $\|\rho\|_{L^4}$, $\max \rho$, $\min \rho$, then three control quantities on the velocity;

as well as `rho*.vtk`, `phi*.vtk` and, for the guiding center, `vel*.vtk`.

4.4 Test cases

The default `data.dat` uses $\Delta t = 0.025$ (40 iterations per time unit) and the final time 100. The Kelvin-Helmholtz and D-shaped tests below were run with $\Delta t = 0.05$ and take a few seconds; the diocotron test uses the default files and takes about 90 seconds.

Kelvin-Helmholtz instability. Rectangle $[0, 12] \times [0, 2\pi]$ (`Xsquare.dat`: 129×129 points), up to the time 40. The perturbation grows exponentially between the times 5 and 18, then the shear layer rolls up into two vortices of opposite sign (Figure 5). In the linear phase, the quantity of column 9 of `Evolution.out`, which measures the deviation of the velocity from its mean in x , grows like $e^{0.118t}$. The total charge, which is zero at the initial time, stays below 10^{-3} of $\int |\rho|$; the energy varies by less than $3 \cdot 10^{-3}$, and $\|\rho\|_{L^2}$ and $\|\rho\|_{L^4}$ decrease by 1.4 % and 2 % once the filaments become thinner than the mesh step (Figure 6).

Diocotron instability in the disk. This is the test of [10], run with the default files: `Xdisk.dat` (257×257 points on $[-10.4, 10.4]^2$, that is, $\Delta x = 0.081$) and `data.dat` ($\Delta t = 0.025$, up to the time 100). During the linear phase, the annulus develops the mode 7 of the perturbation; it breaks into seven vortices around the time 30, which then rotate around the

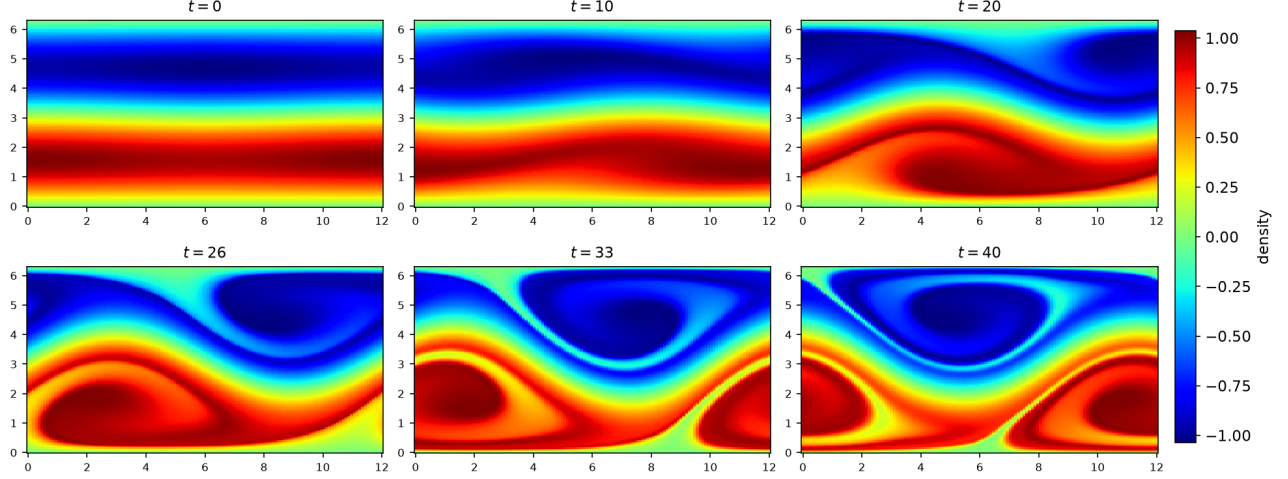


Figure 5: Kelvin-Helmholtz instability: density at several times.

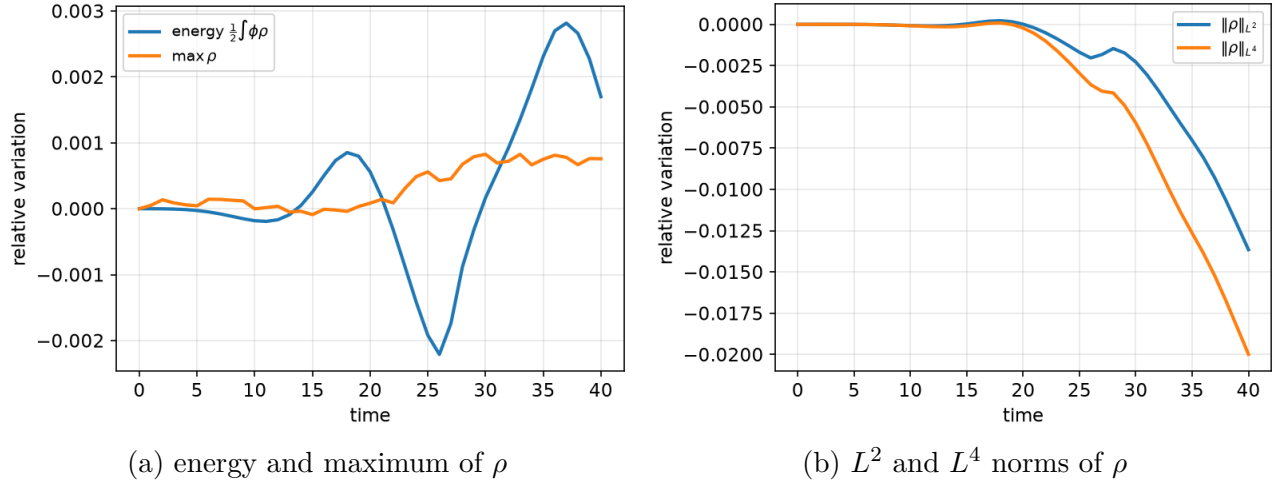


Figure 6: Kelvin-Helmholtz instability: relative variations of the conserved quantities.

center and remain stable up to the time 100, surrounded by thin filaments (Figure 7). The mass is conserved up to $4 \cdot 10^{-8}$. The norms and the energy are conserved during the linear phase ($\|\rho\|_{L^2}$ varies by less than 10^{-3} up to the time 20) and decrease when the filaments become thinner than the mesh step (Figure 8):

$$\begin{cases} \text{at the time 60 : } & \|\rho\|_{L^2} : -8.6\%, \quad \|\rho\|_{L^4} : -7.4\%, \quad \text{energy} : -0.83\%, \\ \text{at the time 100 : } & \|\rho\|_{L^2} : -10.5\%, \quad \|\rho\|_{L^4} : -8.6\%, \quad \text{energy} : -1.3\%. \end{cases}$$

These values agree with those of the mixed semi-Lagrangian/finite difference HWENO5 method of [10] with 256 points ($\|\rho\|_{L^2}$: -4% at the time 40, -8.5% at the time 60 and -10.3% at the time 100; energy: -0.7% at the time 60 and -1% at the time 100). The density takes small negative values, down to $-8 \cdot 10^{-3}$. The resolution matters: with 131×131 points, $\|\rho\|_{L^2}$ decreases by 13% and the energy by 2.2% at the time 60; with the discontinuous initial data, the losses are two to three times larger.

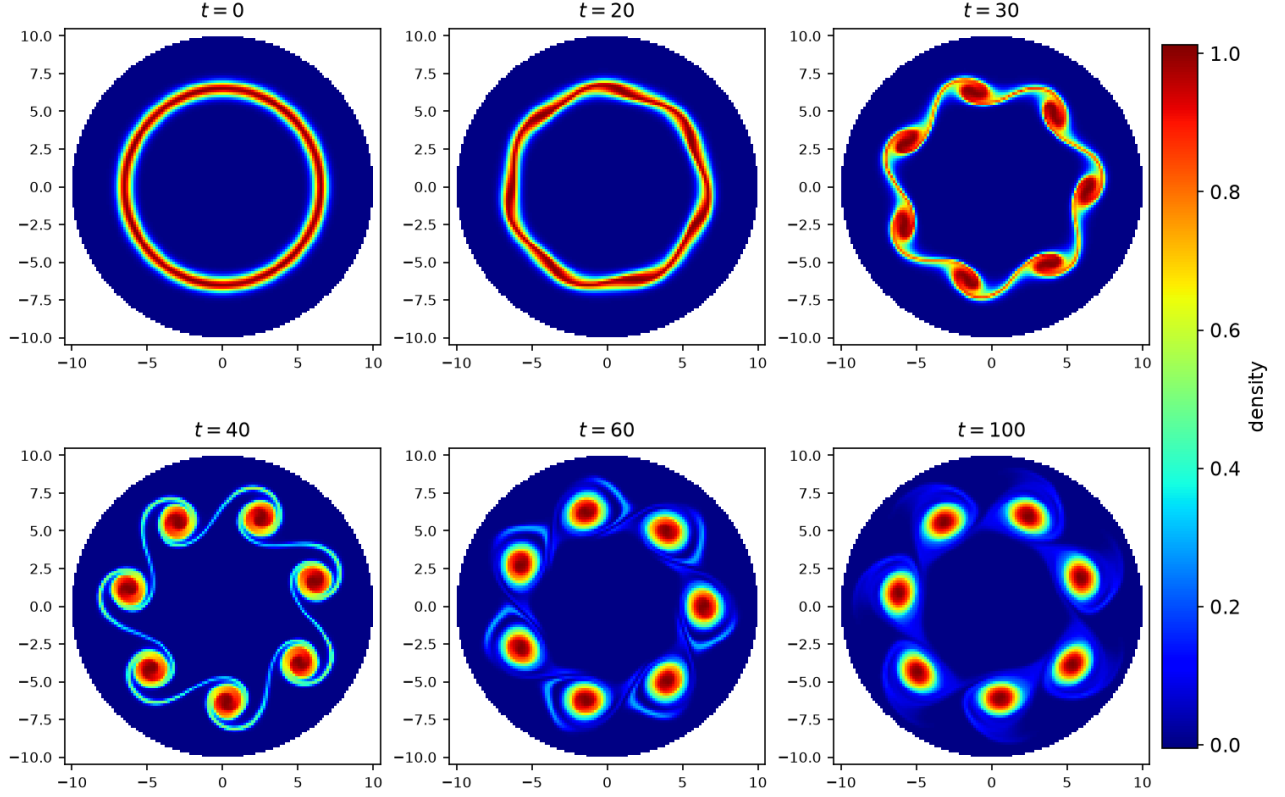
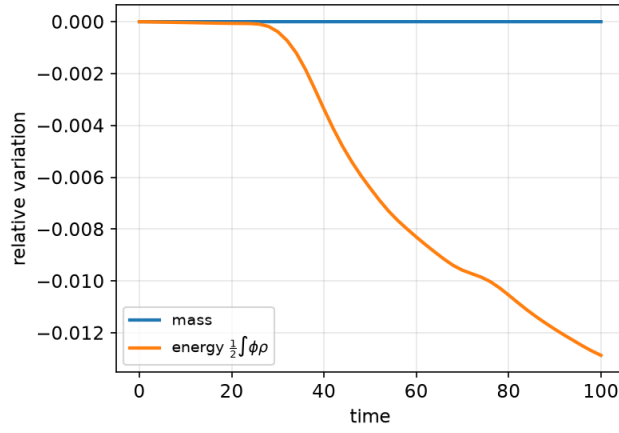
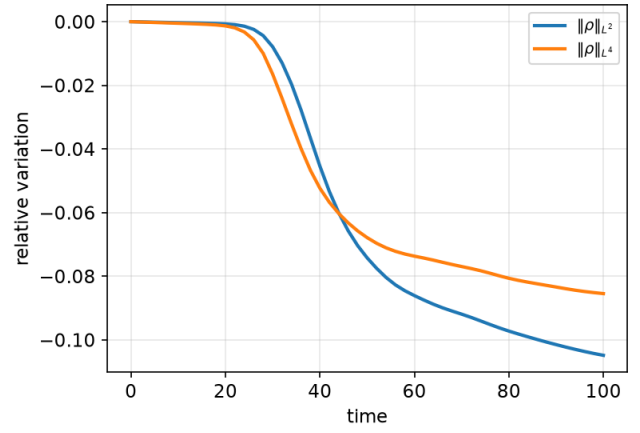


Figure 7: Guiding-center model, diocotron instability in the disk (257×257 points, $\Delta t = 0.025$): density at several times.

D-shaped section. Mesh of `Xd.dat` (101×201 points), up to the time 30. The Gaussian profile remains in place (the density hardly changes), and Figure 9 shows the difference $\rho - \rho_{\text{eq}}$, with $\rho_{\text{eq}} = e^{-r^2/8}$. The mode 7 of the initial perturbation (amplitude 10^{-2}) is quickly damped, and a dipole of amplitude 0.15 appears: since the domain is not symmetric, the Gaussian centered at the origin is not an exact equilibrium, and the vortex moves slowly. The mass is conserved up to $5 \cdot 10^{-7}$, the energy and $\|\rho\|_{L^2}$ vary by less than $4 \cdot 10^{-4}$, and $\|\rho\|_{L^4}$ by less than 10^{-3} (Figure 10).



(a) mass and energy



(b) L^2 and L^4 norms of ρ

Figure 8: Guiding-center model, diocotron instability in the disk: relative variations of the conserved quantities.

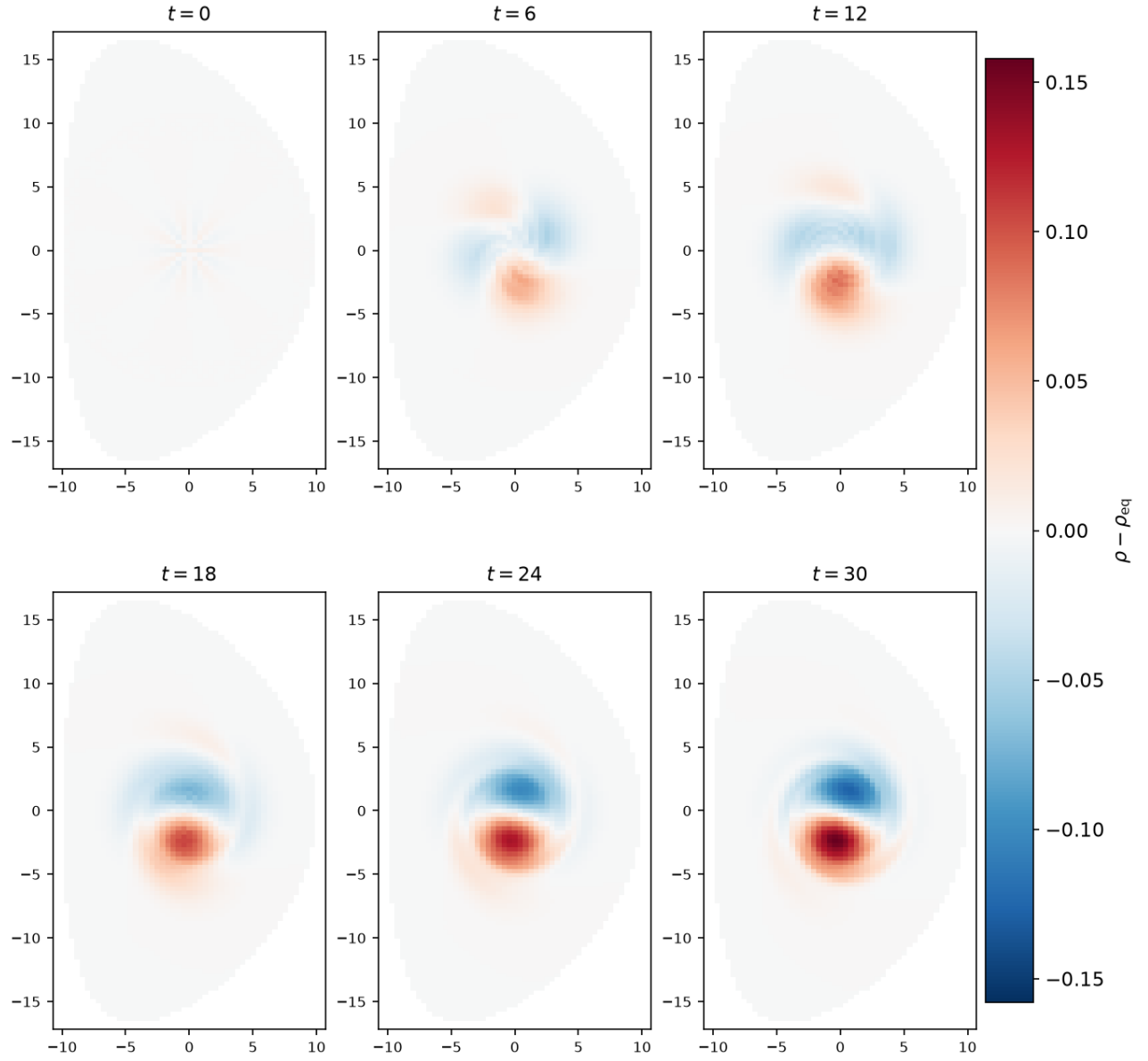
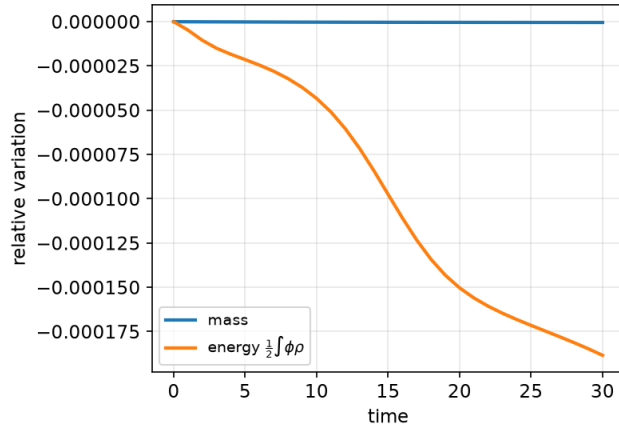
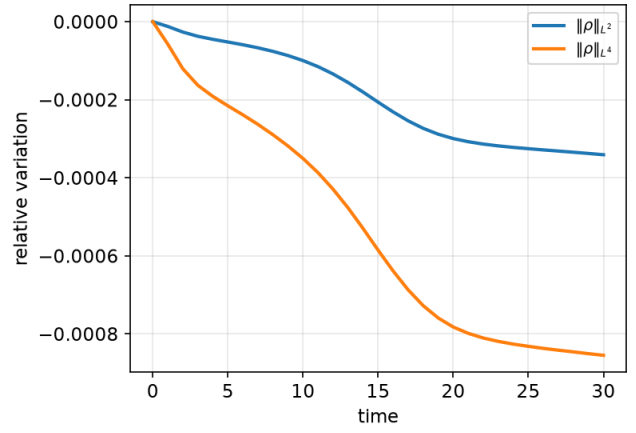


Figure 9: Guiding-center model in the D-shaped section: difference between the density and the Gaussian profile at several times.



(a) mass and energy



(b) L^2 and L^4 norms of ρ

Figure 10: Guiding-center model in the D-shaped section: relative variations of the conserved quantities.

5 PIC-2D

5.1 Model

The code solves the two-dimensional Vlasov–Poisson system with a strong magnetic field orthogonal to the computational plane, in the long-time scaling of [2, 3]:

$$\begin{cases} \varepsilon \partial_t f^\varepsilon + \mathbf{v} \cdot \nabla_{\mathbf{x}} f^\varepsilon + \left(\mathbf{E}^\varepsilon - b(t, \mathbf{x}) \frac{\mathbf{v}^\perp}{\varepsilon} \right) \cdot \nabla_{\mathbf{v}} f^\varepsilon = 0, & \mathbf{v}^\perp = (-v_2, v_1), \\ \mathbf{E}^\varepsilon = -\nabla_{\mathbf{x}} \phi^\varepsilon, & -\Delta_{\mathbf{x}} \phi^\varepsilon = \rho^\varepsilon \text{ in } D, \quad \phi^\varepsilon = 0 \text{ on } \partial D, \quad \rho^\varepsilon = \int_{\mathbb{R}^2} f^\varepsilon d\mathbf{v}, \end{cases}$$

with $\mathbf{x} \in D \subset \mathbb{R}^2$ and $\mathbf{v} \in \mathbb{R}^2$. The factor ε in front of the time derivative means that the solution is followed on times long enough to observe the slow drift of the particles: when $\varepsilon \rightarrow 0$ and $b = 1$, the density converges to the solution of the guiding-center model of Section 4, and an additional drift due to the gradient of b appears when the field is not uniform [2, 3]. In the code, the field b is uniform and equal to 1: it is given by the function `B_z` of the file `Variables_PIC_VP2D.hpp`. The scheme is however written for an arbitrary field $b(t, \mathbf{x})$: it involves $\nabla_{\mathbf{x}} \ln b$, computed by finite differences.

Asymptotic limit. For a general field $b(t, \mathbf{x})$, the formal limit $\varepsilon \rightarrow 0$ of the system is a kinetic equation for the gyro-averaged distribution $f(t, \mathbf{x}, e)$, which depends on the position and on the kinetic energy $e = \|\mathbf{v}\|^2/2$ of the gyration [3]:

$$\begin{cases} \partial_t f + \mathbf{U} \cdot \nabla_{\mathbf{x}} f + u_e \partial_e f = 0, & -\Delta_{\mathbf{x}} \phi = \rho, & \rho = 2\pi \int_{\mathbb{R}^+} f de, \\ \mathbf{U} = \mathbf{F} + e \frac{\nabla_{\mathbf{x}}^\perp b}{b^2}, & u_e = -\operatorname{div}_{\mathbf{x}}(\mathbf{F}) e, & \mathbf{F} = -\frac{\mathbf{E}^\perp}{b}, \quad \mathbf{E} = -\nabla_{\mathbf{x}} \phi. \end{cases}$$

The velocity $\mathbf{U}$ contains the two classical drifts of the guiding center, the $\mathbf{E} \times \mathbf{B}$ drift $\mathbf{F}$ and the drift due to the gradient of b . The field $(\mathbf{U}, u_e)$ is divergence free, so that the L^p norms of f are conserved, as well as the total energy and, when b does not depend on time, the adiabatic invariant:

$$\mathcal{E}(t) = 2\pi \int e f d\mathbf{x} de + \frac{1}{2} \int \|\mathbf{E}\|^2 d\mathbf{x}, \quad \mu(t) = \int \frac{e}{b(\mathbf{x})} f d\mathbf{x} de.$$

These are the quantities written in `e.out`, which should be approximately conserved when ε is small. For $b = 1$, the energy e is only transported, and the density solves the guiding-center model of Section 4.

5.2 Numerical method

The method is the particle-in-cell method with augmented phase space of [3], which extends the method of [2] to non-uniform magnetic fields; its convergence is analysed in [6].

- **Particles:** the density and the current are deposited on the grid with P_1 shape functions; the electric field is computed by centered differences, then interpolated at the particles by cubic Lagrange interpolation.

- **Augmented formulation:** in addition to its position $\mathbf{X}_k$, each particle carries an auxiliary velocity $\mathbf{w}_k$ and an energy e_k , which solve

$$\begin{cases} \varepsilon \frac{d\mathbf{X}_k}{dt} = \mathbf{w}_k, & \varepsilon \frac{de_k}{dt} = \mathbf{E}(t, \mathbf{X}_k) \cdot \mathbf{w}_k, \\ \varepsilon \frac{d\mathbf{w}_k}{dt} = -\frac{b(t, \mathbf{X}_k)}{\varepsilon} \mathbf{w}_k^\perp + \mathbf{E}(t, \mathbf{X}_k) - \chi(e_k, \mathbf{w}_k) \nabla_{\mathbf{x}} \ln b(t, \mathbf{X}_k), \\ \chi(e, \mathbf{w}) = \frac{e}{e + \|\mathbf{w}\|^2/2} \left(e - \frac{\|\mathbf{w}\|^2}{2} \right)^+. \end{cases}$$

They are initialized, and the velocity is recovered, by

$$\mathbf{w}_k^0 = \mathbf{v}_k^0, \quad e_k^0 = \frac{\|\mathbf{v}_k^0\|^2}{2}, \quad \mathbf{v}_k = \sqrt{2|e_k|} \frac{\mathbf{w}_k}{\|\mathbf{w}_k\|}.$$

The exact solution keeps $e_k = \|\mathbf{w}_k\|^2/2$, but the discretization does not: when $\varepsilon \rightarrow 0$, the scheme damps $\mathbf{w}_k$, consistently with the weak convergence of the velocity to zero, while e_k keeps the kinetic energy of the gyration [3].

- **Time scheme:** second-order semi-implicit Runge–Kutta scheme of [3], built on the framework of [1]. The stiff term in $1/\varepsilon$ is treated by an L-stable two-stage SDIRK method with the coefficient

$$\gamma = 1 - \frac{1}{\sqrt{2}},$$

the smallest root of $x^2 - 2x + 1/2$, and the other terms by an explicit Runge–Kutta method. The scheme is uniformly stable with respect to ε , and consistent with the guiding-center limit when $\varepsilon \rightarrow 0$ with Δt fixed (asymptotic preserving property). An explicit fourth-order Runge–Kutta version is kept as a comment in `solve`.

- **Poisson:** solver `PoissonDirichlet_2D` (Appendix).
- **Boundary conditions:** the function `bc` makes the particles periodic on the rectangle of the grid; particles that leave the domain D are not removed.

5.3 Input data

`physics.dat` contains, in this order, as for the 3D code (Section 6):

parameter	meaning	default value
Epsilon	ε	0.01
Geometry	1 = disk (<code>Xdisk.dat</code>), 2 = D-shaped section (<code>Xd.dat</code>)	2
Initial data	1 = diocotron instability, 2 = interaction of two vortices	2
Seed	0 = clock, otherwise fixed seed	0

All MPI processes sample the same particles, because the 2D code does not distribute particles among processes. With a nonzero seed, two successive runs start from the same particles.

Initial data. The velocities follow a Maxwellian with temperature T_0 .

- **Diocotron** (data 1): particles in the annulus $6 \leq r \leq 7$, with an angular perturbation of mode $m_0 = 7$ and amplitude 10^{-3} , $N_0 = 200$, $T_0 = 0.5$.
- **Vortices** (data 2): two Gaussians with variance 1 centered at $(1.5, -1.5)$ and $(-1.5, 1.5)$, each carrying half of the particles, $N_0 = 20$, $T_0 = 1$.

5.4 Output

- **e.out**: time, mass, momentum (m_x, m_y) , kinetic energy, electric energy $\frac{1}{2} \int |\mathbf{E}|^2$, magnetic moment $\sum_k \omega_k e_k / B_z$, total energy (with the electric energy, then with $\frac{1}{2} \int \rho \phi$) and $\int \phi^2$;
- **p.N**: positions and velocities of the particles;
- **rho*.vtk, phi*.vtk**: density and potential.

5.5 Test cases

The two tests below use $\varepsilon = 0.01$ and a fixed seed. The vortex test is run with the input files of the PIC-2D folder ($\Delta t = 0.25$, that is, 4 iterations per time unit in **data.dat**) and takes about one minute on a laptop; the diocotron test uses the default **Xdisk.dat** and a smaller time step, given below, and takes about 1.5 minute up to the time 10.

Interaction of two vortices. D-shaped section (**Xd.dat**: 101×201 points, $100 n_x n_y$ particles), initial data 2, up to the time 10. The two Gaussians rotate around each other and merge into a single vortex surrounded by two spiral arms (Figure 11). The mass is exactly conserved. The kinetic energy is constant, as expected in this strongly magnetized regime, and the magnetic moment varies by less than 10^{-3} . The electric energy, and thus the total energy, decreases by 6 % at the time 10, most of it during the first time steps. This loss comes from the time discretization: over two time units, it is 4.1 %, 1.1 % and 0.3 % for $\Delta t = 1/4$, $1/8$ and $1/16$, that is, it decreases as Δt^2 , in agreement with the order of the scheme.

Diocotron instability. Disk of radius 10, initial data 1, up to the time 10, with the default grid of **Xdisk.dat**, 131×131 points on $[-10.4, 10.4]^2$ (that is, $\Delta x = 0.16$ and $1.7 \cdot 10^6$ particles), and $\Delta t = 0.05$ (20 iterations per time unit in **data.dat**). The annulus develops the mode 7 of the perturbation and breaks into seven vortices around the time 5, which then rotate around the center and remain separated, as in the guiding-center limit (Figure 12). The mass is exactly conserved, and the kinetic energy and the magnetic moment vary by 0.24 %. The electric energy decreases by 0.9 % at the time 10.

These parameters matter. With a grid of 64×64 points on $[-10, 10]^2$ (the former default of **Xdisk.dat**, so that the annulus is only three cells wide) and $\Delta t = 0.25$, the electric energy decreases by 13 % at the time 10. On the grid of 131×131 points, this loss is 3.7 % for $\Delta t = 0.1$ and 0.9 % for $\Delta t = 0.05$: it comes from the time discretization, and is divided by about 4 when the time step is halved.

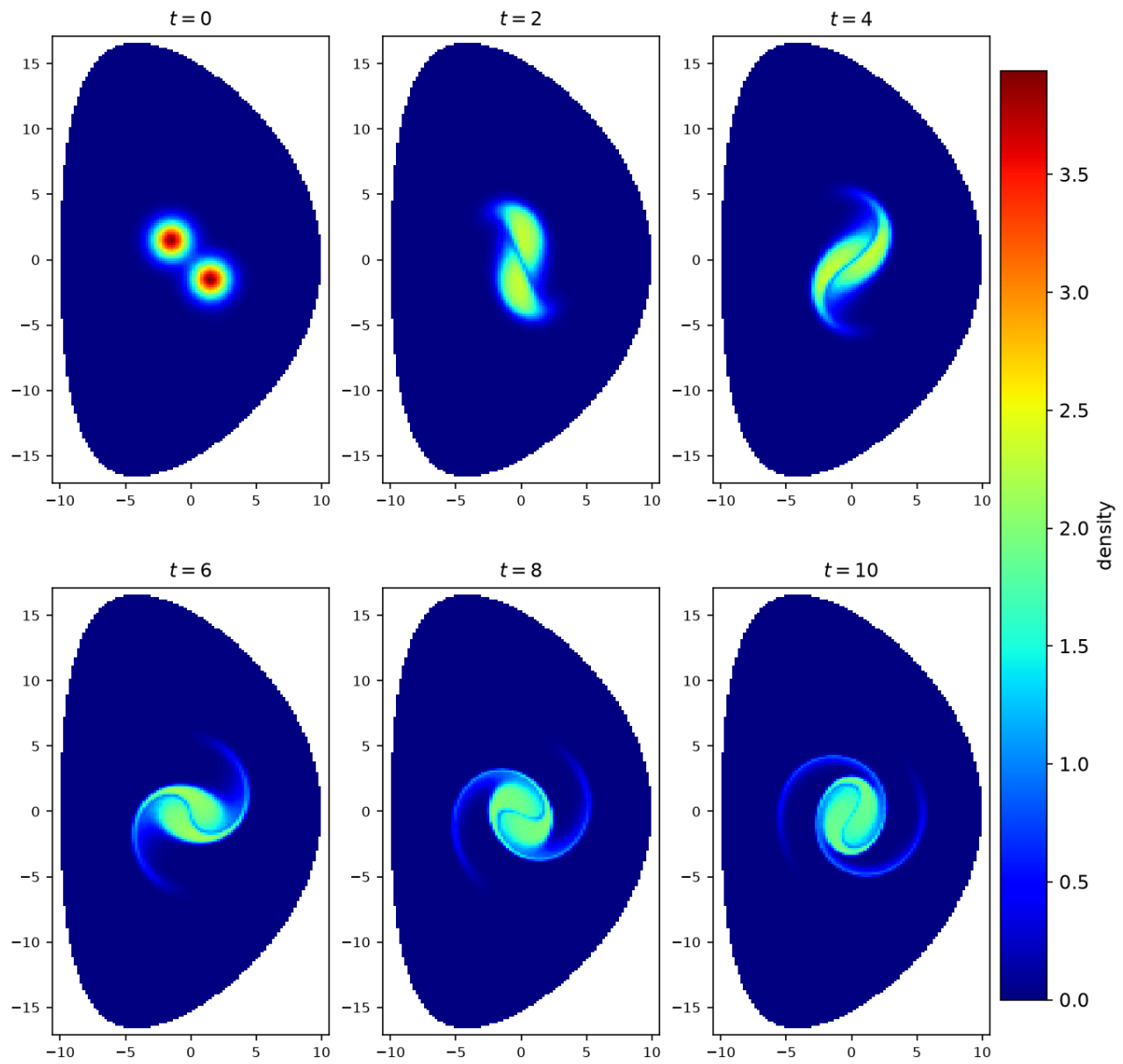


Figure 11: PIC-2D, interaction of two vortices in the D-shaped section ($\varepsilon = 0.01$, $\Delta t = 0.25$): density at several times.

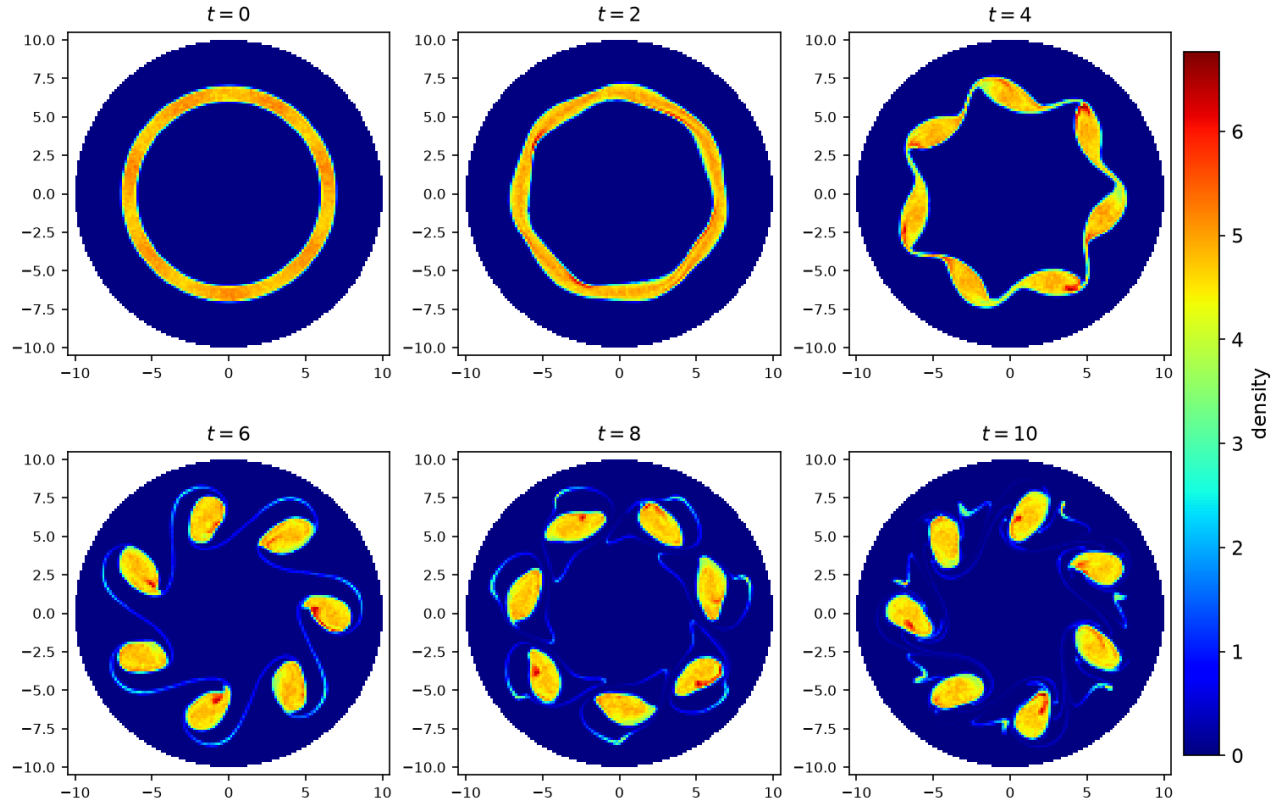


Figure 12: PIC-2D, diocotron instability in the disk ($\varepsilon = 0.01$, 131×131 points, $\Delta t = 0.05$): density at several times.

6 PIC-3D

6.1 Model

We consider the dimensionless Vlasov equation for the density $f^\varepsilon(t, \mathbf{x}, \mathbf{v})$ of one species of charged particles in a torus $\Omega \subset \mathbb{R}^3$ [7]:

$$\begin{cases} \partial_t f^\varepsilon + \mathbf{v} \cdot \nabla_{\mathbf{x}} f^\varepsilon + \left(\mathbf{E} + \frac{\mathbf{v} \wedge \mathbf{B}}{\varepsilon} \right) \cdot \nabla_{\mathbf{v}} f^\varepsilon = 0, \\ \mathbf{E} = -\nabla_{\mathbf{x}} \phi, \quad -\Delta_{\mathbf{x}} \phi = \rho - \rho_0, \quad \rho = \int_{\mathbb{R}^3} f^\varepsilon d\mathbf{v}, \end{cases} \quad (1)$$

where ρ_0 is a neutralizing background and $\mathbf{B}$ is an external magnetic field whose intensity is measured by the small parameter ε ($1/\varepsilon$ is proportional to the cyclotron frequency). We use the toroidal coordinates (a, y, ϕ) ,

$$x_1 = (R_0 + a) \cos \phi, \quad x_2 = y, \quad x_3 = (R_0 + a) \sin \phi,$$

where R_0 is the major radius of the torus and (a, y) describes the poloidal cross-section D , a disk or a D-shaped section. The asymptotic limit $\varepsilon \rightarrow 0$ of this system, which involves the drifts due to the curvature and to the variations of the magnetic field, is derived formally in [4] and rigorously in [5]; the particle method of the code is designed to be consistent with this limit [7], and extends to the torus the methods developed in two dimensions [2, 3] (Section 5).

Asymptotic limit. When $\varepsilon \rightarrow 0$, the fast gyration around the field lines is averaged out, and the slow variables of a particle are the position of its guiding center $\mathbf{X}$, its parallel velocity $v_{\parallel} = \mathbf{v} \cdot \mathbf{e}_{\parallel}$, where $\mathbf{e}_{\parallel} = \mathbf{B}/B$, and its magnetic moment $\mu = \|\mathbf{v}_{\perp}\|^2/(2B)$. At first order, the guiding center moves along the field lines, the parallel velocity is driven by the parallel electric field and by the mirror force, and the magnetic moment is an adiabatic invariant [4, 5, 7]:

$$\begin{cases} \frac{d\mathbf{X}}{dt} = v_{\parallel} \mathbf{e}_{\parallel}(\mathbf{X}), & \frac{dv_{\parallel}}{dt} = \mathbf{E} \cdot \mathbf{e}_{\parallel}(\mathbf{X}) - \mu \mathbf{e}_{\parallel} \cdot \nabla B(\mathbf{X}), & \frac{d\mu}{dt} = 0. \end{cases}$$

The motion across the field lines (the $\mathbf{E} \times \mathbf{B}$ drift, and the drifts due to the curvature and to the gradient of the field) appears at order ε . In [7], the schemes are shown to be consistent with this first-order dynamics, and to be ε^2 -close to a discretization of an effective second-order model that contains these drifts and conserves the energy

$$\frac{v_{\parallel}^2}{2} + B\mu + \phi$$

when the potential does not depend on time. The total energy is thus conserved by the schemes up to $\mathcal{O}(\varepsilon^2 + \Delta t^m)$, for a scheme of order m , even with coarse time steps. The paper writes the equations in coordinates adapted to the field lines; the code uses the same ideas in Cartesian coordinates: Crank–Nicolson treatment of the rotation, energy variable e_k , and force $-\mu \nabla B/B$ (Section 6.2).

Magnetic field. Three fields are available (parameter `Field` of `physics.dat`):

- `Field` = 0: former field of the article [7];
- `Field` = 1: field with a prescribed safety factor profile q ,

$$\begin{cases} \varepsilon \mathbf{B} = \frac{B_0 R_0}{R} \hat{\mathbf{e}}_\phi + \frac{B_0}{q(r) R} (y \hat{\mathbf{e}}_a - a \hat{\mathbf{e}}_y), \\ q(r) = q_0 + (q_a - q_0) \frac{r^2}{r_{\max}^2}, \quad R = R_0 + a, \quad r = \sqrt{a^2 + y^2}; \end{cases}$$

- `Field` = 2: analytic Grad–Shafranov equilibrium of Cerfon and Freidberg [14], used for the D-shaped section:

$$\begin{cases} \varepsilon \mathbf{B} = \frac{F(\psi)}{R} \hat{\mathbf{e}}_\phi + \nabla \psi \wedge \nabla \phi, \\ F(\psi)^2 = B_0^2 R_0^2 - \frac{2A\psi_0^2}{R_0^2} \min(\hat{\psi}, 0), \quad \psi = \psi_0 \hat{\psi} \left(\frac{R}{R_0}, \frac{y}{R_0} \right), \end{cases}$$

where $\hat{\psi}$ is a combination of the Solov’ev particular solution and seven homogeneous solutions, with coefficients $c_1, \dots, c_7$. The plasma is the region $\{\hat{\psi} < 0\}$.

The construction of these fields, the choice of the parameters and the computation of the coefficients are detailed in the appendix *Tokamak configuration: magnetic field and initial data*. The script `cerfon_freidberg.py` (Appendix) recomputes the coefficients for another shape of the cross-section. The simulations use $\varepsilon_{\text{CF}} = 0.3$, $\kappa = 1.66$, $\delta = 0.416$, $A = -0.155$, $\psi_0 = 405.544766$ and a safety factor $q = 1.85$ on the axis. The corresponding coefficients are given in `physics.dat`.

6.2 Numerical method

Particles. The density is approximated by N particles with positions $\mathbf{x}_k$, velocities $\mathbf{v}_k$ and weights ω_k . The weight includes the Jacobian $R_0 + a_k$ of the toroidal coordinates. The density is deposited on the (a, y, ϕ) grid, and the electric field is interpolated at the particles, with P_1 (trilinear) shape functions.

Poisson equation in the torus. The potential is 2π -periodic in ϕ : after a discrete Fourier transform in ϕ , each Fourier mode satisfies a two-dimensional elliptic problem in the poloidal cross-section, solved by MUMPS with a matrix factorized once per mode (see the appendix *Poisson solvers*, subsection *Poisson equation in the torus*).

Time scheme. We use the semi-implicit scheme of [7], with the energy variable e_k of the augmented formulation. A time step proceeds as follows:

1. prediction: from $(\mathbf{x}_k^n, \mathbf{v}_k^n)$, we compute $\mathbf{E}^n$, then the velocity at time $n+1$ with a Crank–Nicolson scheme for the rotation $\mathbf{v} \wedge \mathbf{B}/\varepsilon$, written in explicit form with the matrix $(I - \frac{\Delta t}{2\varepsilon} J\mathbf{B})^{-1}$. The position is advanced with the trapezoidal rule;

2. fixed-point correction: we compute the field again at the midpoint, then update the energy and add the force due to the variations of B ,

$$e_k^{n+1} = e_k^n + \phi(\mathbf{x}_k^n) - \phi(\mathbf{x}_k^{n+1}), \quad \mathbf{F}_k = -\mu_k \frac{\nabla B}{B},$$

where μ_k is deduced from the difference between the energy and the kinetic energy at the midpoint. We iterate until, for all particles, positions, velocities and energy change by less than 10^{-10} from one iteration to the next, with at most 20 iterations. A warning is printed if the fixed point has not converged.

This scheme remains stable and consistent with the guiding-center model when $B \Delta t / \varepsilon \gg 1$, that is, without resolving the gyration.

Parallelism. Each MPI process follows its own particles, sampled with the seed $s(1+p)$, where s is the seed of the input file and p the rank of the process. The densities are summed over all processes, and the Poisson equation is solved by MUMPS over all processes.

6.3 Input data

`physics.dat` contains, in this order:

parameter	meaning
Epsilon	ε
Geometry	1 = disk (<code>Xdisk.dat</code>), 2 = D-shaped section (<code>Xd.dat</code>)
Initial data	1 = uniform disk, 2 = two Gaussians, 3 = profile $n(r)$, 4 = profile $n(\psi)$
rp	radius of the density profile (data 3)
Seed	0 = clock, otherwise fixed seed
Field	0, 1 or 2 (see above)
B0	toroidal field on the axis
q0, qa, rmax	safety factor profile (Field = 1)
A, psi0, c1, ..., c7	Cerfon–Freidberg equilibrium (Field = 2)

The initial data are Maxwellians with temperature $T = 1$, multiplied by the perturbation $1 + 0.05 \cos(2\phi + \theta)$. For initial data 2, the density consists of two Gaussians with variance 1 and amplitude 10, centered at $(a, y) = (0, \pm 2)$. For initial data 4, the density is

$$n = \left(\frac{\hat{\psi}}{\hat{\psi}_{\text{axis}}} \right)^2 \text{ in the plasma } \{\hat{\psi} < 0\}, \quad n = 0 \text{ elsewhere;}$$

it is constant on magnetic surfaces and equal to 1 on the magnetic axis.

6.4 Output

- `e.out`, one line per diagnostic: time, mass, three components of the momentum, kinetic energy K_ε , potential energy U_ε , magnetic moment μ_ε , total energy E_ε , $\max |\mathbf{E}|$ and $\frac{1}{2} \int \rho \phi$, where

$$\mu_\varepsilon = \frac{1}{2} \sum_k \omega_k \frac{\|\mathbf{v}_{k,\perp}\|^2}{B(\mathbf{x}_k)}, \quad E_\varepsilon = K_\varepsilon + U_\varepsilon;$$

- `p.out`: time, then (x, y, z, a, ϕ) for three particles;
- `rho*.vtk`, `phi*.vtk`: density and potential, in VTK format, once per time unit.

Each run directory contains `gnuplot` scripts (`energy.gp`, `mu.gp`) that produce the energy and magnetic moment plots.

6.5 Results: simulation `test_Dshape`

The simulation of the folder `PIC-3D/runs/test_Dshape` uses the D-shaped section, the Cerfon–Freidberg field (`Field = 2`), $R_0 = 20$ and a mesh of 79×129 points on $(-6.2, 6.2) \times (-10.2, 10.2)$, i.e. a step of 0.16, with 33 points in ϕ (32 cells). There are 8 particles per cell and per process; with 8 processes, this gives 20 871 168 particles. The seed is fixed (`Seed = 1`). The parameters are $\varepsilon = 0.01$ and $\Delta t = 0.1$, up to the time 100; the field $|\mathbf{B}|$ ranges between 0.80 and 1.45 in the plasma, so that $B \Delta t / \varepsilon$ lies between 8 and 14.5. The initial data is the initial data 2: two Gaussians centered at $(a, y) = (0, \pm 2)$, which are not constant on magnetic surfaces.

The two Gaussians wind around the magnetic axis under the $\mathbf{E} \wedge \mathbf{B}$ drift (Figures 13 and 14), and part of the potential energy is converted into kinetic energy (Figure 15): at the time 100, the kinetic energy has increased by 16 % of the initial total energy. The mass is conserved up to 10^{-7} and the total energy varies by 0.073 %. The magnetic moment, an adiabatic invariant of the limit dynamics, varies by less than 1 %.

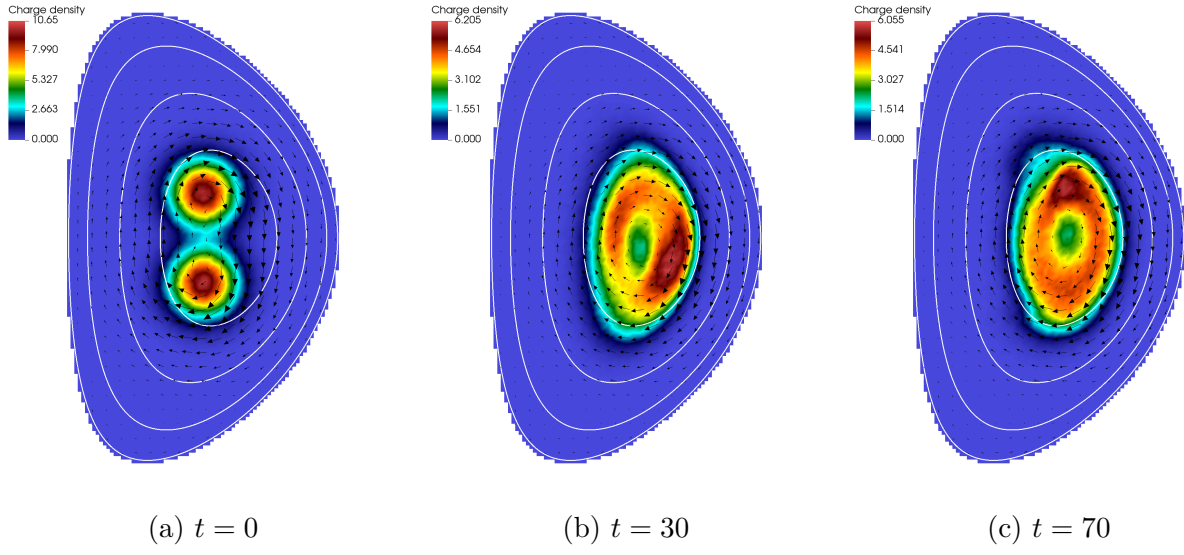


Figure 13: `test_Dshape` ($\varepsilon = 0.01$): charge density in a poloidal cross-section, magnetic surfaces (white) and $\mathbf{E} \wedge \mathbf{B}$ drift velocity (arrows).

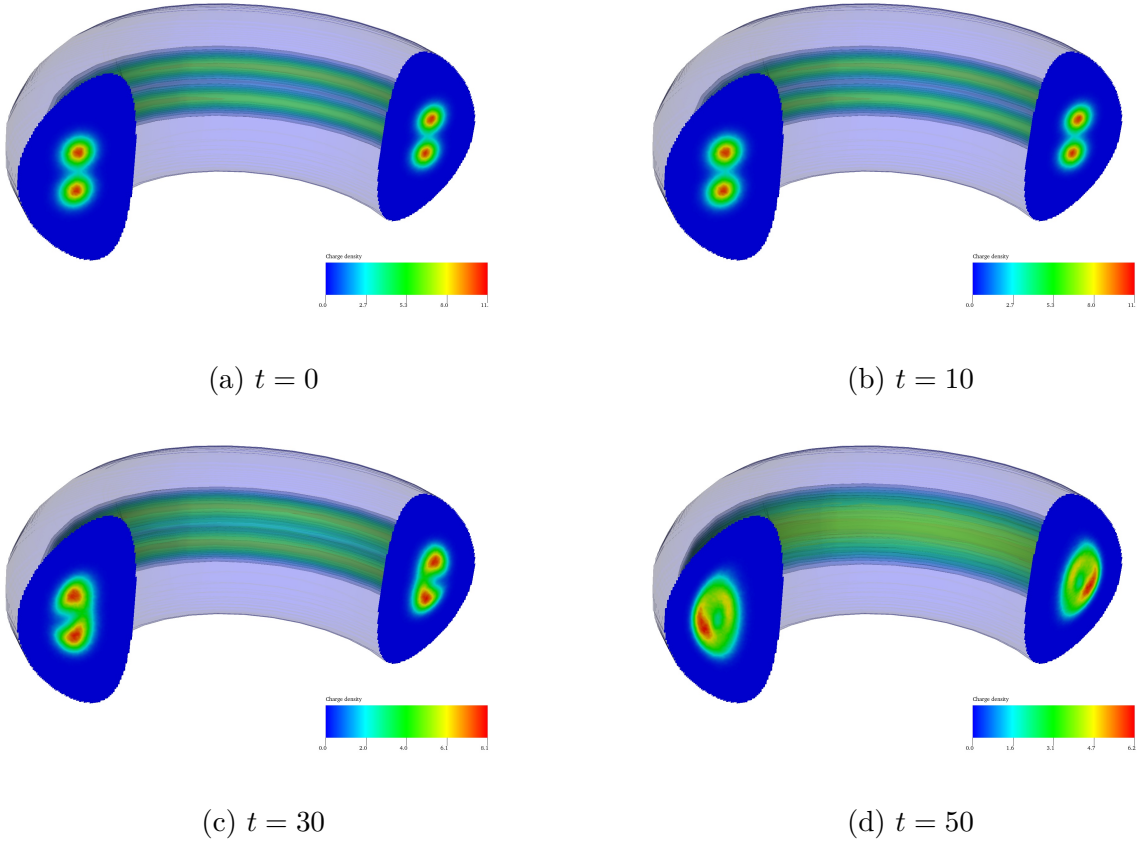


Figure 14: `test_Dshape` ($\varepsilon = 0.01$): density in the torus at several times.

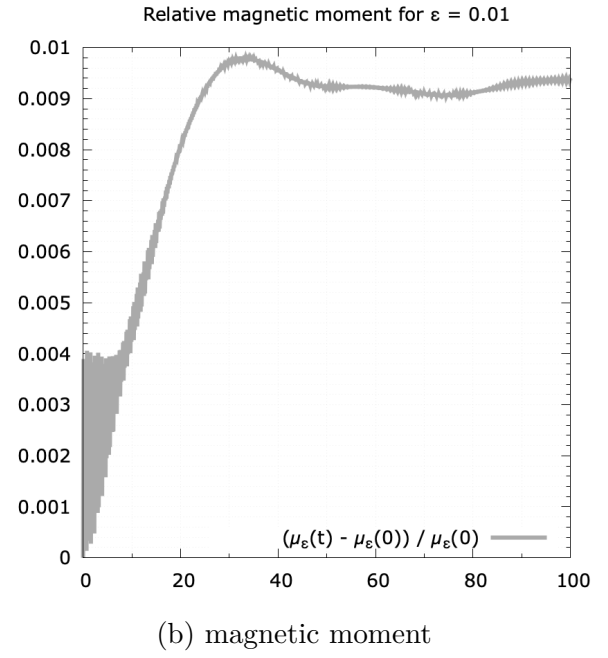
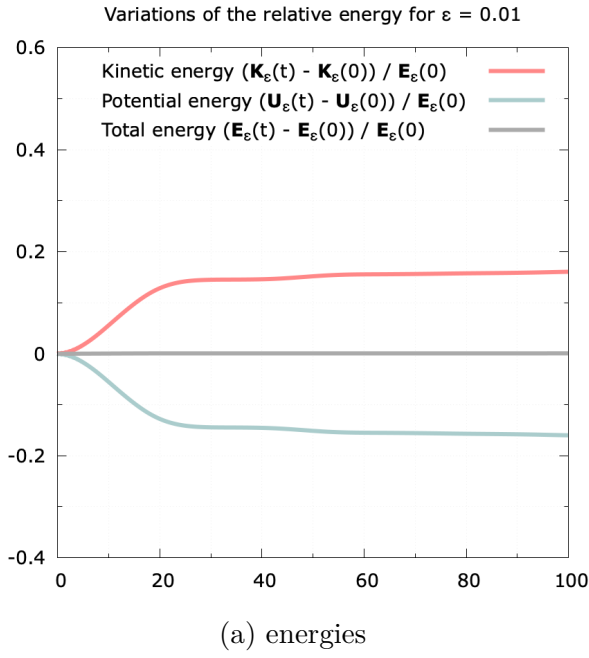


Figure 15: `test_Dshape` ($\varepsilon = 0.01$): relative variations of the kinetic, potential and total energies, and of the magnetic moment.

Poisson solvers

solver	used by	method
PoissonDirichlet_2D	PIC-2D, guiding center	five-point scheme, MUMPS
PoissonDirichlet_3D_torus	PIC-3D	FFT in ϕ , MUMPS for each mode
PoissonPeriodicDirichlet_2D	Kelvin–Helmholtz	FFT in x , sine transform in y
Poisson_nonlinear_2D	not used	Newton’s method, MUMPS

Dirichlet problem in a two-dimensional domain

The solver `PoissonDirichlet_2D` solves

$$\begin{cases} -\Delta\phi = \rho & \text{in } D, \\ \phi = 0 & \text{on } \partial D, \end{cases}$$

with the five-point scheme. When a neighbor of an interior point is a ghost point, its value is replaced by the extrapolation of Section 2: the matrix remains sparse but is no longer symmetric. It is stored in CSR format, converted to the coordinate format of MUMPS, analyzed and factorized (LU) only once, when the solver is built; each solve then only costs a forward and a backward substitution. After the solve, the ghost points are filled by extrapolation, and ϕ is set to zero at exterior points.

The structure of the code is that of the 3D solver (subsection *Poisson equation in the torus* below), of which it is the restriction to a single mode. MUMPS is called on `MPI_COMM_WORLD` and returns the solution on process 0; the 2D solver then broadcasts it to all processes.

Assembly. The unknowns are the values of ϕ at the N_{xy} interior points, numbered with the index j (direction y) varying fastest. For an interior point (i, j) , the row of the matrix is

$$\frac{2\phi_{i,j} - \phi_{i-1,j} - \phi_{i+1,j}}{\Delta x^2} + \frac{2\phi_{i,j} - \phi_{i,j-1} - \phi_{i,j+1}}{\Delta y^2} = \rho_{i,j},$$

where a neighbor that is a ghost point is replaced by $c_1 \phi(\mathbf{x}_h) + c_2 \phi(\mathbf{x}_{2h})$, which involves up to 18 interior points, and the term $c_0 \phi_w$ (zero here) is moved to the right-hand side. The number of nonzero entries is computed first (`Precompute_Matrix_Nnot0`), then the matrix is filled (`Sparse_Matrix`).

Use of MUMPS. The calls follow the sequence of the 3D solver: initialization (`JOB_INIT`) on `MPI_COMM_WORLD`, analysis of the pattern (`job = 1`) and numerical factorization (`job = 2`) in the constructor, then one solve (`job = 3`) per call to `poisson_2D`. The instance is released (`JOB_END`) in the destructor, unless MPI has already been finalized.

Poisson equation in the torus

The potential is 2π -periodic in ϕ . We perform a discrete Fourier transform in ϕ (FFTW), with one mode per cell in ϕ ($n_z - 1$ modes). For each mode k , the real and imaginary parts

of the Fourier coefficient $\hat{\phi}_k(a, y)$ satisfy the two-dimensional problem

$$\begin{cases} -\partial_a^2 \hat{\phi}_k - \frac{1}{R} \partial_a \hat{\phi}_k - \partial_y^2 \hat{\phi}_k + \frac{\omega_k^2}{R^2} \hat{\phi}_k = \hat{\rho}_k & \text{in } D, \\ \hat{\phi}_k = 0 & \text{on } \partial D, \end{cases}$$

where ω_k is the wave number of mode k . The derivatives in a and y are discretized by centered differences, and the boundary condition is imposed with ghost points, as in the subsection *Dirichlet problem in a two-dimensional domain* above. The matrix depends on the mode through the term ω_k^2/R^2 , but not on the right-hand side: the code builds one matrix per mode, with its own MUMPS instance, analyzed and factorized when the solver is built. At each time step, the real and imaginary parts of $\hat{\rho}_k$ are solved one after the other with the same factorization, and the potential is obtained by inverse Fourier transform. The solutions are computed on process 0, which then distributes the potential to the other processes.

Rectangle, periodic in one direction

The solver `PoissonPeriodicDirichlet_2D` treats a rectangle, periodic in x , with Dirichlet conditions in y . It uses a Fourier transform in x and a sine transform in y , both programmed in the file, without external library. This is the solver of the Kelvin–Helmholtz case.

Nonlinear Poisson equation

The solver `Poisson_nonlinear_2D` solves

$$\begin{cases} -\Delta \phi = G(\phi) - \rho & \text{in } D, \\ \phi = 0 & \text{on } \partial D, \end{cases} \quad G(\phi) = e^{-\phi},$$

by Newton’s method, stopped when the residual is below 10^{-6} . The pattern of the matrix is analyzed by MUMPS at the first iteration, then the matrix is factorized again at each iteration. This solver is not used by any main program; it serves to build steady states (commented lines in `Variables_GC.hpp`). At iteration m , the Newton step solves the linear problem

$$-\Delta \phi^{m+1} - G'(\phi^m) \phi^{m+1} = G(\phi^m) - G'(\phi^m) \phi^m - \rho.$$

Tokamak configuration: magnetic field and initial data

This appendix describes the choice of the external magnetic field, of the parameters and of the initial data for the tokamak-like simulations of [7], so that the tests are performed in a tokamak-like configuration and in the asymptotic regime $B \Delta t / \varepsilon \gg 1$ targeted by the scheme. All quantities are dimensionless: the mass and the charge of the particles and the vacuum permittivity are equal to 1.

Shortcomings of the former field

The former field (`Field = 0`) is

$$\begin{cases} \mathbf{B}_{\text{ext}} = \frac{1}{\varepsilon} \left(\frac{B_0}{R} \hat{\mathbf{e}}_\phi + \frac{B_1(1-r)}{R} (y \hat{\mathbf{e}}_a - a \hat{\mathbf{e}}_y) \right), \\ B_0 = 2, \quad B_1 = 0.5, \quad R_0 = 20, \quad \varepsilon = 10^{-2}. \end{cases}$$

With this field:

- the poloidal field dominates the toroidal field ($B_{\text{pol}} \approx 0.75$ against $B_{\text{tor}} \approx 0.1$ at $r = 6$), whereas their ratio is about 0.1 in a tokamak;
- the poloidal field vanishes and changes sign at $r = 1$;
- the safety factor tends to 0 at the edge, instead of increasing from about 1 at the center to 3 or 4 at the edge;
- $B \Delta t / \varepsilon \approx 1$ in the core of the plasma: the gyration is not fast compared with the time step, and the tests are not in the asymptotic regime.

The following sections correct these four points.

Geometry and coordinates

We use the toroidal coordinates (a, y, ϕ) of [7], together with the major radius R , the minor radius r and the orthonormal basis $(\hat{\mathbf{e}}_a, \hat{\mathbf{e}}_y, \hat{\mathbf{e}}_\phi)$:

$$\begin{cases} x_1 = (R_0 + a) \cos \phi, & x_2 = y, & x_3 = (R_0 + a) \sin \phi, \\ R = R_0 + a = \sqrt{x_1^2 + x_3^2}, & r = \sqrt{a^2 + y^2}, & \phi = \text{atan2}(x_3, x_1), \\ \hat{\mathbf{e}}_a = (\cos \phi, 0, \sin \phi), & \hat{\mathbf{e}}_y = (0, 1, 0), & \hat{\mathbf{e}}_\phi = (-\sin \phi, 0, \cos \phi). \end{cases}$$

The cross-section is the disk D of radius r_{max} . We propose $R_0 = 30$ and $r_{\text{max}} = 10$, that is, an aspect ratio of 3, close to that of ITER (3.1). One may keep $R_0 = 20$ (aspect ratio 2) to keep the current mesh.

Magnetic field of the circular case

Definition. We build an axisymmetric field from a constant toroidal function F and a flux function ψ depending only on r :

$$\begin{cases} \varepsilon \mathbf{B}_{\text{ext}} = F \nabla \phi + \nabla \psi \wedge \nabla \phi, \\ F = B_0 R_0, \quad \psi = \psi(r). \end{cases}$$

Such a field is divergence free, and the circles of constant r in the cross-section generate the magnetic surfaces. Since

$$\nabla \phi = \frac{\hat{\mathbf{e}}_\phi}{R}, \quad \nabla \psi = \frac{\psi'(r)}{r} (a \hat{\mathbf{e}}_a + y \hat{\mathbf{e}}_y),$$

we get

$$\varepsilon \mathbf{B}_{\text{ext}} = \frac{B_0 R_0}{R} \hat{\mathbf{e}}_\phi + \frac{\psi'(r)}{r R} (y \hat{\mathbf{e}}_a - a \hat{\mathbf{e}}_y).$$

We choose ψ from a safety factor profile q , which gives the field

$$\begin{cases} \varepsilon \mathbf{B}_{\text{ext}} = \frac{B_0 R_0}{R} \hat{\mathbf{e}}_\phi + \frac{B_0}{q(r) R} (y \hat{\mathbf{e}}_a - a \hat{\mathbf{e}}_y), \\ \psi'(r) = \frac{B_0 r}{q(r)}, \quad q(r) = q_0 + (q_a - q_0) \frac{r^2}{r_{\text{max}}^2}. \end{cases} \quad (2)$$

Compared with the former field, it suffices to replace B_0/R by $B_0 R_0/R$ and the factor $B_1(1-r)$ by $B_0/q(r)$. This is the field `Field = 1` of the code.

Check. The toroidal and poloidal components are

$$B_{\text{tor}} = \frac{B_0 R_0}{R}, \quad B_{\text{pol}} = \frac{B_0 r}{q(r) R},$$

so that the cylindrical safety factor and the ratio of the two components are

$$q_{\text{cyl}}(r) = \frac{r B_{\text{tor}}}{R_0 B_{\text{pol}}} = q(r), \quad \frac{B_{\text{pol}}}{B_{\text{tor}}} = \frac{r}{q(r) R_0}.$$

This ratio is about 0.11 for $q_0 = 1$, $q_a = 3$, $R_0 = 30$ and r between 5 and 10. The field equals B_0 on the magnetic axis $r = 0$.

Cartesian components. We denote by $\mathbf{B}$ the field $\varepsilon \mathbf{B}_{\text{ext}}$. Using

$$\cos \phi = \frac{x_1}{R}, \quad \sin \phi = \frac{x_3}{R}, \quad a = R - R_0, \quad y = x_2,$$

the field (2) reads

$$\begin{cases} B_1(\mathbf{x}) = \frac{1}{R} \left(-B_0 R_0 \frac{x_3}{R} + \frac{B_0 y}{q(r)} \frac{x_1}{R} \right), \\ B_2(\mathbf{x}) = -\frac{B_0 a}{q(r) R}, \\ B_3(\mathbf{x}) = \frac{1}{R} \left(B_0 R_0 \frac{x_1}{R} + \frac{B_0 y}{q(r)} \frac{x_3}{R} \right). \end{cases}$$

The scheme of [7] uses the magnitude B , the direction $\mathbf{e}_\parallel$ and the relative gradient of the field,

$$B = |\mathbf{B}|, \quad \mathbf{e}_\parallel = \frac{\mathbf{B}}{B}, \quad \frac{\nabla B}{B}.$$

Magnitude and gradient. We have

$$B(\mathbf{x}) = \frac{B_0}{R} \sqrt{R_0^2 + S(r)}, \quad S(r) := \frac{r^2}{q(r)^2},$$

and, since the field is axisymmetric,

$$\begin{cases} \nabla B = \partial_a B \hat{\mathbf{e}}_a + \partial_y B \hat{\mathbf{e}}_y, \\ \partial_a B = \frac{B_0}{R} \left(\frac{S'(r)}{2\sqrt{R_0^2 + S(r)}} \frac{a}{r} - \frac{\sqrt{R_0^2 + S(r)}}{R} \right), \\ \partial_y B = \frac{B_0}{R} \frac{S'(r)}{2\sqrt{R_0^2 + S(r)}} \frac{y}{r}, \\ S'(r) = \frac{2r}{q(r)^2} - \frac{2r^2 q'(r)}{q(r)^3}, \quad q'(r) = \frac{2(q_a - q_0)r}{r_{\max}^2}. \end{cases}$$

The terms in a/r and y/r are regular at $r = 0$, since

$$\lim_{r \rightarrow 0} \frac{S'(r)}{r} = \frac{2}{q_0^2}.$$

These formulas, as well as the fact that $\mathbf{B}$ is divergence free, should be checked by finite differences before running simulations.

Choice of the parameters

parameter	value	comment
$R_0, r_{\max}$	30, 10	aspect ratio 3
B_0	1	magnitude on the axis
q_0, q_a	1, 3	safety factor profile
ε	10^{-3} (and 10^{-2} for comparison)	
Δt	0.1, 0.05, 0.025	convergence study
temperature T	1	thermal velocity $v_{th} = 1$
density at the center n_0	1	
radius of the density profile r_p	6	plasma far from the boundary
mesh	$\Delta a = \Delta y = 0.16, N_\phi = 32$	as in [7]

With these parameters, B ranges between 0.75 (outer edge) and 1.5 (inner edge), and we get the following scales:

quantity	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-2}$
cyclotron frequency B/ε	750 – 1500	75 – 150
$B \Delta t/\varepsilon$ for $\Delta t = 0.1$	75 – 150	7.5 – 15
Larmor period $2\pi\varepsilon/B$	$4 \cdot 10^{-3} - 8 \cdot 10^{-3}$	0.04 – 0.08
Larmor radius $\varepsilon v_{th}/B$	$\approx 10^{-3}$	$\approx 10^{-2}$

For $\varepsilon = 10^{-3}$, the gyration is much faster than the time step, which corresponds to the asymptotic regime, and the Larmor radius is small compared with the mesh step (0.16) and with the minor radius (ratio of order 10^{-4}). The other scales remain resolved: with $n_0 = 1$ and $T = 1$, the plasma frequency is about 1, so that $\omega_p \Delta t$ is about 0.1, and the Debye length is about 1, larger than the mesh step.

Initial data of the circular case

Distribution. We take a Maxwellian whose density only depends on r , hence is constant on magnetic surfaces, plus a small perturbation:

$$\begin{cases} f_0(\mathbf{x}, \mathbf{v}) = n(r) \left(1 + \alpha \cos(2\phi + \theta)\right) \frac{1}{(2\pi T)^{3/2}} \exp\left(-\frac{\|\mathbf{v}\|^2}{2T}\right), & \theta = \text{atan2}(y, a), \\ n(r) = n_0 \left(1 - \frac{r^2}{r_p^2}\right)^2 \quad \text{for } r \leq r_p, & n(r) = 0 \quad \text{otherwise,} \end{cases}$$

with $\alpha = 0.05$. Without the perturbation, f_0 only depends on r and $\|\mathbf{v}\|$; it is almost an equilibrium of the guiding-center model, which should reduce the initial transient (in particular the variation of μ observed in the D-shaped case with initial data 2). The perturbation is that of [7]. This is initial data 3 of the code.

Neutralizing background. The total charge must vanish. Taking into account the Jacobian R , the volume of the torus and the total mass are

$$|\Omega| = 2\pi^2 R_0 r_{\max}^2, \quad \int_{\Omega} \int_{\mathbb{R}^3} f_0 = 2\pi R_0 \int_D n(r) da dy = \frac{2\pi^2}{3} R_0 n_0 r_p^2,$$

since the terms in a and in $\cos(2\phi + \theta)$ have zero integral. The constant neutralizing background is thus

$$\rho_0 = \frac{n_0 r_p^2}{3 r_{\max}^2} = 0.12 \quad \text{for } n_0 = 1, r_p = 6, r_{\max} = 10.$$

Sampling of the particles.

1. Sample (a_k, y_k) uniformly in the disk of radius r_p , and ϕ_k uniformly in $(-\pi, \pi)$.
2. Give each particle the weight

$$\omega_k = \frac{2\pi \pi r_p^2}{N} n(r_k) \left(1 + \alpha \cos(2\phi_k + \theta_k)\right) (R_0 + a_k),$$

where N is the total number of particles; the factor $R_0 + a_k$ is the Jacobian of the toroidal coordinates.

3. Sample the velocities with the normal law of covariance $T I_3$, that is, each component with a normal law of variance T .

4. Go back to Cartesian coordinates,

$$\mathbf{x}_k = \left((R_0 + a_k) \cos \phi_k, y_k, (R_0 + a_k) \sin \phi_k \right).$$

5. Initialize the scheme with

$$\mathbf{w}_k^0 = \mathbf{v}_k, \quad e_k^0 = \frac{1}{2} \|\mathbf{v}_k\|^2.$$

With about 60 particles per cell on each cross-section, as in [7], about 4400 occupied cells per cross-section (the area of the disk of radius r_p divided by Δa^2) and $N_\phi = 32$, we get $N \approx 8.5 \cdot 10^6$. One may also sample the positions according to the density $n(r)(R_0 + a)$ and give all particles the same weight, which reduces the noise at the edge of the profile.

D-shaped equilibrium: Cerfon–Freidberg solution

For the D-shaped cross-section, we use a true Grad–Shafranov equilibrium, the analytic solution of Cerfon and Freidberg [14], in its up-down symmetric version. The boundary of the domain is then a magnetic surface.

Flux function. We write Z for the vertical coordinate y of [7], and introduce the normalized variables

$$X = \frac{R}{R_0} = \frac{R_0 + a}{R_0}, \quad Y = \frac{Z}{R_0}.$$

The flux function is

$$\begin{cases} \psi(R, Z) = \psi_0 \hat{\psi}(X, Y), \\ \hat{\psi} = \frac{X^4}{8} + A \left(\frac{X^2 \ln X}{2} - \frac{X^4}{8} \right) + \sum_{i=1}^7 c_i \psi_i(X, Y), \end{cases}$$

where the ψ_i are the homogeneous solutions, even in Y , of the operator Δ^* :

$$\begin{cases} \Delta^* = X \partial_X (X^{-1} \partial_X) + \partial_Y^2, \\ \psi_1 = 1, \quad \psi_2 = X^2, \quad \psi_3 = Y^2 - X^2 \ln X, \quad \psi_4 = X^4 - 4X^2 Y^2, \\ \psi_5 = 2Y^4 - 9Y^2 X^2 + 3X^4 \ln X - 12X^2 Y^2 \ln X, \\ \psi_6 = X^6 - 12X^4 Y^2 + 8X^2 Y^4, \\ \psi_7 = 8Y^6 - 140Y^4 X^2 + 75Y^2 X^4 - 15X^6 \ln X + 180X^4 Y^2 \ln X - 120X^2 Y^4 \ln X. \end{cases}$$

One checks (symbolic computation) that

$$\Delta^* \psi_i = 0, \quad 1 \leq i \leq 7, \quad \Delta^* \hat{\psi} = (1 - A)X^2 + A,$$

so that ψ is a solution of the Grad–Shafranov equation with Solov’ev profiles.

Shape of the cross-section and coefficients. The prescribed boundary is

$$X = 1 + \epsilon \cos(\tau + \alpha \sin \tau), \quad Y = \epsilon \kappa \sin \tau, \quad \alpha = \arcsin \delta, \quad \tau \in [0, 2\pi).$$

We choose $\epsilon = 0.3$, $\kappa = 1.66$, $\delta = 0.416$ and $A = -0.155$, values close to those of ITER. With $R_0 = 20$, the cross-section has a minor radius 6 and a half-height 9.96, which corresponds to the mesh $(-6.2, 6.2) \times (-10.2, 10.2)$ of [7], and to the geometry **DShape** of the code (Section 2).

The coefficients $c_1, \dots, c_7$ are given by the seven conditions of Cerfon and Freidberg, written at the outer point $(1 + \epsilon, 0)$, the inner point $(1 - \epsilon, 0)$ and the top point $(1 - \delta\epsilon, \kappa\epsilon)$:

$$\begin{cases} \hat{\psi} = 0 & \text{at the outer, inner and top points,} \\ \partial_X \hat{\psi} = 0 & \text{at the top point,} \\ \partial_Y^2 \hat{\psi} + N_1 \partial_X \hat{\psi} = 0 & \text{at the outer point,} \quad N_1 = -\frac{(1 + \alpha)^2}{\epsilon \kappa^2}, \\ \partial_Y^2 \hat{\psi} + N_2 \partial_X \hat{\psi} = 0 & \text{at the inner point,} \quad N_2 = \frac{(1 - \alpha)^2}{\epsilon \kappa^2}, \\ \partial_X^2 \hat{\psi} + N_3 \partial_Y \hat{\psi} = 0 & \text{at the top point,} \quad N_3 = -\frac{\kappa}{\epsilon \cos^2 \alpha}. \end{cases}$$

We obtain

coefficient	value
c_1	$7.4791241775 \cdot 10^{-2}$
c_2	$-1.6876892164 \cdot 10^{-1}$
c_3	$-1.0264441674 \cdot 10^{-1}$
c_4	$-6.7308509360 \cdot 10^{-2}$
c_5	$1.6769591989 \cdot 10^{-2}$
c_6	$-1.3814277585 \cdot 10^{-2}$
c_7	$-4.2905850805 \cdot 10^{-4}$

With these values, $|\hat{\psi}|$ is below $3.2 \cdot 10^{-6}$ on the prescribed boundary, and the magnetic axis X_a and the value $\hat{\psi}_a$ of $\hat{\psi}$ on the axis are

$$X_a = 1.042358, \quad \hat{\psi}_a = \hat{\psi}(X_a, 0) = -3.138 \cdot 10^{-2}.$$

The plasma is the region $\{\hat{\psi} < 0\}$. These coefficients depend on $(\epsilon, \kappa, \delta, A)$: for another shape of the cross-section, it suffices to run the script `cerfon_freidberg.py` again (see the appendix *Python code*); it computes the coefficients and checks the above identities. The values of the table are those of `physics.dat`.

Magnetic field. The field and the toroidal function are

$$\begin{cases} \varepsilon \mathbf{B}_{\text{ext}} = \frac{F(\psi)}{R} \hat{\mathbf{e}}_\phi + \nabla \psi \wedge \nabla \phi = \frac{F(\psi)}{R} \hat{\mathbf{e}}_\phi + \frac{1}{R} (\partial_Z \psi \hat{\mathbf{e}}_a - \partial_a \psi \hat{\mathbf{e}}_y), \\ \partial_a \psi = \frac{\psi_0}{R_0} \partial_X \hat{\psi}, \quad \partial_Z \psi = \frac{\psi_0}{R_0} \partial_Y \hat{\psi}, \\ F(\psi)^2 = B_0^2 R_0^2 - \frac{2A \psi_0^2}{R_0^2} \min(\hat{\psi}, 0), \end{cases}$$

where the last relation ensures that ψ is an equilibrium; F equals $B_0 R_0$ outside the plasma. This field is divergence free (checked numerically, the divergence is of order 10^{-12}). This is the field `Field = 2` of the code.

Normalization and safety factor. The constant ψ_0 sets the magnitude of the poloidal field, hence the safety factor. The safety factor on the axis and on a magnetic surface are given by

$$q_{\text{axis}} = \frac{F}{R_a \sqrt{\partial_R^2 \psi \partial_Z^2 \psi}}, \quad R_a = R_0 X_a, \quad q = \frac{1}{2\pi} \oint \frac{F}{R |\nabla \psi|} dl.$$

For $B_0 = 1$, $R_0 = 20$ and a safety factor 1.85 on the axis, we get

$$\psi_0 = 405.544766,$$

and the safety factor profile is

normalized flux $1 - \hat{\psi}/\hat{\psi}_a$	0.1	0.5	0.8	0.95
q	1.93	2.35	2.83	3.17

that is, $q_{95} \approx 3.2$, close to ITER. In the plasma, B ranges between 0.80 and 1.45, and the ratio of the poloidal and toroidal components is at most 0.27. With $\varepsilon = 10^{-3}$ and $\Delta t = 0.1$, $B \Delta t / \varepsilon$ is thus at least 80: we are in the asymptotic regime.

Initial data for the D-shaped case. We either keep the initial data of [7] (two Gaussians centered at $y = \pm 2$, initial data 2 of the code), or take a density depending on $\hat{\psi}$, constant on magnetic surfaces and thus close to an equilibrium (initial data 4):

$$n = n_0 \left(\frac{\hat{\psi}}{\hat{\psi}_a} \right)^2 \quad \text{in the plasma,} \quad n = 0 \quad \text{elsewhere.}$$

The neutralizing background ρ_0 is computed as in the circular case, by integrating the density numerically with the Jacobian R .

Simulations and diagnostics

1. **Circular case, asymptotic regime:** field (2), $\varepsilon = 10^{-3}$, $\Delta t = 0.1$, up to the time 100.
2. **Resolved reference:** same case with a time step resolving the gyration ($\Delta t \leq 10^{-3}$) on a short time (for instance up to the time 5), to compare the density, the energies and μ .
3. **Convergence:** $\Delta t = 0.1, 0.05, 0.025$, and two numbers of particles per cell (for instance 60 and 240).
4. **Cost:** CPU time per time step and in total, compared with that of the resolved reference.
5. **Comparison:** same case with $\varepsilon = 10^{-2}$.

6. **D-shaped case:** Cerfon–Freidberg equilibrium (see the subsection on the D-shaped equilibrium above).

The diagnostics are those of [7]: kinetic, potential and total energies (with the Jacobian R_0+a), magnetic moment

$$\mu(t) = \frac{1}{2} \sum_k \omega_k \frac{\|\mathbf{v}_{k,\perp}\|^2}{B(\mathbf{x}_k)},$$

and snapshots of the density and of the potential. One should also check that no particle leaves the domain.

Python code

Cerfon–Freidberg equilibrium

The script `PIC-3D/cerfon_freidberg.py` computes the coefficients $c_1, \dots, c_7$ of the D-shaped equilibrium (see the appendix *Tokamak configuration: magnetic field and initial data*), checks symbolically that $\hat{\psi}$ solves the Grad–Shafranov equation, computes the magnetic axis, the normalization ψ_0 , the safety factor profile and the range of $|\mathbf{B}|$, and prints the C expressions of $\hat{\psi}$ and of its derivatives used in `Variables_PIC_VP3D_torus.hpp`. It requires `numpy`, `scipy` and `sympy`, and runs in a few seconds with `python3 cerfon_freidberg.py`. The listing below is read directly from the file.

```
"""
Up-down symmetric Cerfon-Freidberg (Solov'ev) equilibrium for the D-shaped test.

Normalized coordinates  X = R/R0,  Y = Z/R0  (Z = y in the paper),
    psi(R,Z) = psi0 * psih(X,Y),
    psih = X^4/8 + A (X^2 ln X / 2 - X^4/8) + sum_{i=1}^7 c_i psi_i(X,Y),
which solves  X d/dX (1/X dpsih/dX) + d2psih/dY2 = (1-A) X^2 + A.

The coefficients c_i are obtained from the 7 boundary conditions of
Cerfon & Freidberg, Phys. Plasmas 17 (2010) 032502, for the boundary
    X = 1 + e cos(t + alpha sin t),  Y = e k sin t,  alpha = arcsin(d).
psi0 is then chosen to prescribe the safety factor on the magnetic axis.
"""

import numpy as np
import sympy as sp
from scipy.optimize import minimize_scalar

# ----- parameters
e_, k_, d_ = 0.3, 1.66, 0.416          # inverse aspect ratio, elongation,
    triangularity
A = sp.Rational(-155, 1000)            # Solov'ev parameter (exact rational for the
    symbolic check)
R0, B0, q_axis = 20.0, 1.0, 1.85      # major radius, field on axis, q on axis

# ----- basis
x, y = sp.symbols('x y', positive=True)
L = sp.log(x)
basis = [sp.Integer(1), x**2, y**2 - x**2*L, x**4 - 4*x**2*y**2,
    2*y**4 - 9*y**2*x**2 + 3*x**4*L - 12*x**2*y**2*L,
    x**6 - 12*x**4*y**2 + 8*x**2*y**4,
    8*y**6 - 140*y**4*x**2 + 75*y**2*x**4 - 15*x**6*L + 180*x**4*y**2*L - 120*x**2*y
    **4*L]
psi_p = x**4/8 + A*(x**2*L/2 - x**4/8)
Dstar = lambda f: sp.simplify(x*sp.diff(sp.diff(f, x)/x, x) + sp.diff(f, y, 2))
assert all(Dstar(f) == 0 for f in basis)
assert sp.simplify(Dstar(psi_p) - ((1 - A)*x**2 + A)) == 0

# ----- boundary conditions
al = np.arcsin(d_)
N1 = -(1 + al)**2/(e_*k**2)
N2 = (1 - al)**2/(e_*k**2)
```

```

N3 = -k_/(e_*np.cos(al)**2)
c = sp.symbols('c1:8')
psi = psi_p + sum(ci*f for ci, f in zip(c, basis))
px, py, pxx, pyy = sp.diff(psi, x), sp.diff(psi, y), sp.diff(psi, x, 2), sp.diff(psi, y,
2)
out, inn, hig = {x: 1 + e_, y: 0}, {x: 1 - e_, y: 0}, {x: 1 - d_*e_, y: k_*e_}
eqs = [psi.subs(out), psi.subs(inn), psi.subs(hig), px.subs(hig),
        (pyy + N1*px).subs(out), (pyy + N2*px).subs(inn), (pxx + N3*py).subs(hig)]
sol = sp.solve([sp.N(eq) for eq in eqs], c)
coef = [float(sol[ci]) for ci in c]
psih = sp.lambdify((x, y), psi.subs(sol), 'numpy')
dpsih = [sp.lambdify((x, y), sp.diff(psi.subs(sol), v), 'numpy') for v in (x, y)]

# ----- axis and normalization
xa = minimize_scalar(lambda s: psih(s, 0.0), bounds=(1 - e_, 1 + e_), method='bounded').x
psia = psih(xa, 0.0)
hxx = float(pxx.subs(sol).subs({x: xa, y: 0})); hyy = float(pyy.subs(sol).subs({x: xa, y:
0}))
psi0 = B0*R0/(xa*q_axis*np.sqrt(hxx*hyy))
for _ in range(50):      #  $F^2 = B0^2 R0^2 - 2 A \psi0^2 \psi h / R0^2$ 
    Fax = np.sqrt(B0**2*R0**2 - 2*float(A)*psi0**2*psia/R0**2)
    psi0 = Fax*R0/(xa*q_axis*np.sqrt(hxx*hyy))

def field(a, z):
    """eps * B_ext in the basis (e_a, e_y, e_phi), at R = R0 + a, Z = z."""
    X, Y = (R0 + a)/R0, z/R0; R = R0 + a
    ph = psih(X, Y)
    F = np.sqrt(B0**2*R0**2 - 2*float(A)*psi0**2*min(ph, 0.0)/R0**2)    #  $F = B0 R0$  outside
    the plasma
    dpa = psi0/R0*dpsih[0](X, Y); dpz = psi0/R0*dpsih[1](X, Y)    #  $d \psi / da, d \psi / dZ$ 
    return np.array([dpz/R, -dpa/R, F/R])    # ( $grad \psi \wedge grad \phi$ )
    + F grad phi

if __name__ == "__main__":
    print("coefficients c1..c7:")
    for i, v in enumerate(coef, 1):
        print(f"  c{i} = {v: .10e}")
    t = np.linspace(0, 2*np.pi, 400)
    bx, by = 1 + e_*np.cos(t + al*np.sin(t)), e_*k_*np.sin(t)
    print("max |psih| on the prescribed boundary:", np.abs(psih(bx, by)).max())
    print(f"magnetic axis X_a = {xa:.6f}, psih(axis) = {psia:.6e}")
    print(f"psi0 = {psi0:.6f} (B0 = {B0}, R0 = {R0}, q_axis = {q_axis})")
    # safety factor profile  $q = (1/2\pi) \oint F/(R |grad \psi|) dl$  on the flux surfaces
    from scipy.optimize import brentq
    th = np.linspace(0, 2*np.pi, 721)
    for s in (0.1, 0.5, 0.8, 0.95):
        lev = psia*(1 - s)
        pts = []
        for t_ in th:
            g = lambda rho: psih(xa + rho*np.cos(t_), rho*np.sin(t_)) - lev
            rho = brentq(g, 1e-8, 0.6)
            pts.append((xa + rho*np.cos(t_), rho*np.sin(t_)))
        pts = np.array(pts)*R0
        mid = 0.5*(pts[1:] + pts[:-1]); dl = np.linalg.norm(pts[1:] - pts[:-1], axis=1)

```

```

    Rm = mid[:, 0]
    gp = psi0/R0*np.hypot(dpsih[0](mid[:, 0]/R0, mid[:, 1]/R0), dpsih[1](mid[:, 0]/R0,
mid[:, 1]/R0))
    F = np.sqrt(B0**2*R0**2 - 2*float(A)*psi0**2*lev/R0**2)
    print(f"  q({s}) = {np.sum(F*d1/(Rm*gp))/(2*np.pi):.3f}")
# range of |B| in the plasma and divergence check
bmin, bmax, rat = 1e9, 0, 0
for a_ in np.linspace(-6, 6, 121):
    for z_ in np.linspace(-10, 10, 201):
        if psih((R0 + a_)/R0, z_/R0) < 0:
            b = field(a_, z_); nb = np.linalg.norm(b)
            bmin, bmax = min(bmin, nb), max(bmax, nb); rat = max(rat, np.hypot(b[0], b
[1])/abs(b[2]))
print(f"|B| in [{bmin:.3f}, {bmax:.3f}],  max Bpol/Btor = {rat:.3f}")
# C code for psih and its derivatives (used in Variables_PIC_VP3D_torus.hpp)
cs = sp.symbols('c[1] c[2] c[3] c[4] c[5] c[6] c[7]'.replace('[', '_').replace(']', ''))
Asym = sp.Symbol('A')
psig = x**4/8 + Asym*(x**2*L/2 - x**4/8) + sum(ci*f for ci, f in zip(cs, basis))
for name, ex in (("psi", psig), ("dpsidX", sp.diff(psig, x)), ("dpsidY", sp.diff(psig,
y))):
    print(name, "=", sp.ccode(sp.collect(sp.expand(ex), cs)))

```

Field of the circular case

The following function returns $\varepsilon \mathbf{B}_{\text{ext}}$, given by (2), at a point $\mathbf{x} = (x_1, x_2, x_3)$.

```

import numpy as np
R0, rmax, B0, q0, qa = 30.0, 10.0, 1.0, 1.0, 3.0

def q(r):
    return q0 + (qa - q0) * r**2 / rmax**2

def Bfield(x):
    # returns eps * B_ext at x = (x1, x2, x3)
    R = np.hypot(x[0], x[2]); a = R - R0; y = x[1]; r = np.hypot(a, y)
    c, s = x[0] / R, x[2] / R
    e_a = np.array([c, 0.0, s])
    e_y = np.array([0.0, 1.0, 0.0])
    e_phi = np.array([-s, 0.0, c])
    return B0 * R0 / R * e_phi + B0 / (q(r) * R) * (y * e_a - a * e_y)

```

References

- [1] S. Boscarino, F. Filbet and G. Russo, *High order semi-implicit schemes for time dependent partial differential equations*, J. Sci. Comput. **68** (2016), 975–1001.
- [2] F. Filbet and L. M. Rodrigues, *Asymptotically stable particle-in-cell methods for the Vlasov–Poisson system with a strong external magnetic field*, SIAM J. Numer. Anal. **54** (2016), 1120–1146.

- [3] F. Filbet and L. M. Rodrigues, *Asymptotically preserving particle-in-cell methods for inhomogeneous strongly magnetized plasmas*, SIAM J. Numer. Anal. **55** (2017), 2416–2443.
- [4] P. Degond and F. Filbet, *On the asymptotic limit of the three dimensional Vlasov–Poisson system for large magnetic field: formal derivation*, J. Stat. Phys. **165** (2016), 765–784.
- [5] F. Filbet and L. M. Rodrigues, *Asymptotics of the three-dimensional Vlasov equation in the large magnetic field limit*, J. Éc. polytech. Math. **7** (2020), 1009–1067.
- [6] F. Filbet, L. M. Rodrigues and H. Zakerzadeh, *Convergence analysis of asymptotic preserving schemes for strongly magnetized plasmas*, Numer. Math. **149** (2021), 549–593.
- [7] F. Filbet and L. M. Rodrigues, *Asymptotically preserving particle methods for strongly magnetized plasmas in a torus*, 2023 (file PIC-3D/Annexe/Bibliographie/FF-LMR-23.pdf).
- [8] F. Filbet and C. Yang, *An inverse Lax–Wendroff method for boundary conditions applied to Boltzmann type models*, J. Comput. Phys. **245** (2013), 43–61.
- [9] F. Filbet and C. Yang, *Numerical simulations of kinetic models for chemotaxis*, SIAM J. Sci. Comput. **36** (2014), B348–B366.
- [10] C. Yang and F. Filbet, *Conservative and non-conservative methods based on Hermite weighted essentially-non-oscillatory reconstruction for Vlasov equations*, J. Comput. Phys. **279** (2014), 18–36.
- [11] F. Filbet and T. Xiong, *Conservative discontinuous Galerkin/Hermite spectral method for the Vlasov–Poisson system*, Commun. Appl. Math. Comput. (2020).
- [12] M. Bessemoulin-Chatard and F. Filbet, *On the stability of conservative discontinuous Galerkin/Hermite spectral methods for the Vlasov–Poisson system*, J. Comput. Phys. **451** (2022), 110881.
- [13] A. Blaustein and F. Filbet, *On a discrete framework of hypocoercivity for kinetic equations*, Math. Comp. **93** (2024), 163–202.
- [14] A. J. Cerfon and J. P. Freidberg, *“One size fits all” analytic solutions to the Grad–Shafranov equation*, Phys. Plasmas **17**, 032502, 2010.