Targeting realistic geometry in Tokamak code Gysela *

Nicolas Bouzat, Camilla Bressan, Virginie Grandgirard, Guillaume Latu, Michel Mehrenberger

Targeting realistic geometry in Tokamak code GYSELA*

Nicolas Bouzat [1] Camilla Bressan [2] Virginie Grandgirard [3]
Guillaume Latu [3] Michel Mehrenberger [4]

November 3, 2017

Abstract

In magnetically confined plasmas used in Tokamak, turbulence is responsible for specific transport that limits the performance of this kind of reactors. Gyrokinetic simulations are able to capture ion and electron turbulence that give rise to heat losses, but require also state-of-the-art HPC techniques to handle computation costs. Such simulations are a major tool to establish good operating regime in Tokamak such as ITER, which is currently being built. Some of the key issues to address more realistic gyrokinetic simulations are: efficient and robust numerical schemes, accurate geometric description, good parallelization algorithms. The framework of this work is the Semi-Lagrangian setting for solving the gyrokinetic Vlasov equation and the GYSELA code. In this paper, a new variant for the interpolation method is proposed that can handle the mesh singularity in the poloidal plane at $r = 0$ (polar system is used for the moment in GYSELA). A non-uniform meshing of the poloidal plane is proposed instead of uniform one in order to save memory and computations. The interpolation method, the gyroaverage operator, and the Poisson solver are revised in order to cope with non-uniform meshes. A mapping that establish a bijection from polar coordinates to more realistic plasma shape is used to improve realism. Convergence studies are provided to establish the validity and robustness of our new approach.

Introduction

Understanding and control of turbulent transport in thermonuclear plasmas in magnetic confinement devices is a major goal. This aspect of first principle physics plays a key role in achieving the level of performance expected in fusion reactors. In the ITER design[6], the latter was estimated by extrapolating an empirical law. The simulation and understanding of the turbulent transport in Fusion plasmas remains therefore an ambitious endeavor.

The Fusion energy community has been engaged in high-performance computing (HPC) for a long time. For example, gyrokinetic simulations are time-hungry (thousands up to millions of CPU-hours) and we then need large amount of computational time that are typically provided by advanced computational facilities. Computer simulation is and will continue to be a key tool for investigating several aspects of Fusion energy technology, because right now there is no

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

[1]Inria, F-54600 Villers-lès-Nancy, France, hosted by CEA/IRFM, F-13108 Saint-Paul-lez-Durance, France
[2]D-85748 Garching, & Technische Universität München, D-85748 Garching, Germany.
[3]CEA/IRFM, F-13108 Saint-Paul-lez-Durance, France
[4]IRMA, Université de Strasbourg, FR-67084 Strasbourg & Inria, F-54600 Villers-lès-Nancy, France

[6]http://www.itercad.org/

burning plasma experiments like ITER. Some of the key issues to address realistic simulations of the Tokamak are: efficient and robust numerical schemes, accurate geometric description, good parallelization algorithms.

The *gyrokinetic* framework considers a computational domain in five dimensions (3D in space describing a torus geometry, 2D in velocity). Time evolution of the system consists in solving Vlasov equation non-linearly coupled to a Poisson equation (electrostatic approximation, quasi-neutrality is assumed). The code has the originality to be based on a semi-Lagrangian scheme [20] and it is parallelized using an hybrid OpenMP/MPI paradigm [3, 16].

Let $\vec{z} = (r, \theta, \varphi, v_\parallel, \mu)$ be a variable describing the 5D phase space. The time evolution of the ionic distribution function of the guiding-center $\bar{f}(\vec{z})$ (main unknown) is governed by the gyrokinetic Vlasov equation (simplified version without right-hand side terms):

$$\partial_t \bar{f} + \frac{1}{B_\parallel^*} \nabla_{\vec{z}} \cdot \left(\frac{\mathrm{d}\vec{z}}{\mathrm{d}t} B_\parallel^* \bar{f} \right) = 0 \tag{1}$$

The guiding-center motion described by the previous Vlasov/transport equation is coupled to a field solver (3D quasi neutral solver which is a Poisson-like solver) that computes the electric potential $\phi(r, \theta, \varphi)$ (adiabatic electron limit):

$$\frac{e}{T_e}(\phi - \langle\phi\rangle) = \frac{1}{n_0} \int J_0(\bar{f} - \bar{f}_{init}) \, \mathrm{d}\mathbf{v} + \rho_i^2 \nabla_\perp^2 \frac{e\phi}{T_i} \tag{2}$$

We will not describe this last equation (details can be found in [8, 17]). This Poisson-like equation gives the electric field ϕ that corresponds to the particle distribution at each time step t. The derivates of $J_0 \, \phi$ along the torus dimensions are computed. Then, these quantities act as a feedback in the Vlasov equation, they appear into the term $\frac{\mathrm{d}\vec{z}}{\mathrm{d}t} B_\parallel^* \bar{f}$. The Vlasov solver represents the critical CPU part, *i.e.* usually more than 90% of computation time. This equation is solved by splitting it into the advection equations ($\mathcal{X}_G = (r, \theta)$):

$$B_\parallel^* \partial_t \bar{f} + \overrightarrow{\nabla} \cdot \left(B_\parallel^* \tfrac{d\mathcal{X}_G}{dt} \bar{f} \right) = 0 \quad (\hat{\mathcal{X}_G} \quad \text{operator}),$$

$$B_\parallel^* \partial_t \bar{f} + \partial_\varphi \left(B_\parallel^* \tfrac{d\varphi}{dt} \bar{f} \right) = 0 \quad (\hat{\varphi} \quad \text{operator}), \qquad B_\parallel^* \partial_t \bar{f} + \partial_{v_\parallel} \left(B_\parallel^* \tfrac{dv_\parallel}{dt} \bar{f} \right) = 0 \quad (\hat{v_\parallel} \quad \text{operator}).$$

Each advection consists in applying a shift operator along one or two dimensions. A Strang splitting procedure is employed to reach second order accuracy in time. The sequence we choose is $(\hat{v_\parallel}/2, \hat{\varphi}/2, \hat{\mathcal{X}_G}, \hat{\varphi}/2, \hat{v_\parallel}/2)$, where the factor $1/2$ is a shift over a reduced time step $dt/2$.

In this work, we will propose solutions to improve the $\hat{\mathcal{X}_G}$ operator in the Vlasov solver, the gyroaverage J_0 that appears in Eq. (2), and the 2D Poisson equation we need to solve that comes from $\rho_i^2 \nabla_\perp^2 \frac{e\phi}{T_i}$ term in Eq. (2). These three operators are tightly coupled to the geometry in the poloidal plane which is perpendicular (transverse) to the magnetic field direction. Conversely, the $\hat{v_\parallel}$ and $\hat{\varphi}$ operators are quite independent from the poloidal geometry because they act in other dimensions than $\hat{\mathcal{X}_G}$.

The paper is organized as follows: in the first section, the original poloidal geometry and meshing is described, the new non-uniform approach focusing the poloidal plane is explained, and the mapping that handles realistic geometry is given. Then, in the second section, the interpolation method on non-uniform polar mesh is investigated, but also advection and gyroaverage operators on such a mesh. Also, we show the numerical method chosen for the 2D Poisson solver. Finally, numerical results and convergence studies are presented in the third section.

1 New geometry and mapping

Changing the mesh of the poloidal plane while keeping a polar coordinate system should allow us first, to loosen the meshing in order to reduce the typical concentration of points near the center $r = 0$ and second, to have the mesh match more closely the magnetic surfaces of the plasma. We should then have an improvement in execution time by reducing the overall number of points as well as an improvement in accuracy thanks to the grid being closer to the typical pattern of simulated phenomena. The non-uniform meshing will also allows us to focus on a specific location of the plane that we want to solve by using more points there and only solving roughly elsewhere.

1.1 Polar mesh

1.1.1 Original polar mesh

We fix N_r, the number of points in the radial direction and N_θ, the number of points in the poloidal direction. The original polar mesh, as it is defined in GYSELA, is such as $r_i = r_{min} + i\Delta r$ with $r_{min} > 0$, $i \in [\![0, N_r - 1]\!]$, $\Delta r = \frac{r_{\max} - r_{\min}}{N_r - 1}$, and also $\theta_j = \frac{j\, 2\pi}{N_\theta}$ with $j \in [\![0, N_\theta - 1]\!]$. It is worth noting that r_{min} and r_{max} act as boundary conditions. For each operator that is applied within the poloidal domain, specific ad-hoc approaches are setup to handle what is happening in the central hole $r \in [0, r_{min}]$. We will not detail the set of ad-hoc boundary conditions that are described in [6].

1.1.2 New non-uniform polar mesh

The new poloidal grid that we want to use is sketched in Figure 1. The idea is to have, for each different circle labeled by r coordinate, a different number of point in the radial direction θ. For instance, in Figure 1 (p. 6), the first layer (inner circle) has four points, the second to fourth layers have eight points and the remaining layers have sixteen points. This allows either to have a density of grid point which is nearly uniform on the plane, or to model finely a subset of the plane which is better solved with more grid points. This meshing or quite similar approaches have already been used in a set of papers [19, 12, 18]. However, in these previous works the setting and the equations solved were quite different from what we investigate here. Therefore, we have mainly only retained the meshing strategy while redesigning the operators and tools that apply on the mesh.

We have $r_{\min} = \frac{\Delta r}{2}$ and $r_{\max} = r_{\min} + (N_r - 1)\Delta r$ so that $\Delta r = \frac{r_{\max} - r_{\min}}{N_r - 1} = \frac{r_{\max} - \frac{\Delta r}{2}}{N_r - 1}$, which leads to

$$\Delta r = \frac{r_{\max}}{N_r - \frac{1}{2}},$$

and the radial points are

$$r_i = r_{\min} + i\,\Delta r = (i + \frac{1}{2})\Delta r, \quad i = 0, \ldots, N_r - 1.$$

Now, for each one of the r_i we choose a number of points along θ: $N_{\theta_{[i]}}$, and a grid spacing: $\Delta_{\theta\,[i]}$, according to what we want to do. Either to focus on a specific region of the plane or to reduce the overall number of points used on the plane and keep the same accuracy.

1.2 Mapping

The previous approach can be combined with a general mapping, the polar mapping being only a special case. We focus here on mappings with analytical formula and whose inverse can also

be expressed by a formula (to shorten execution time) which was one of the concluding points of [1]. This is of course the case for the polar mapping, but we can also find other more general cases, that can have relevance for the description of the geometry of a tokamak. We consider here the case of a large aspect ratio Tokamak equilibrium, and the mapping that derives from it, as in [5, 2].
For the polar mapping

$$x = r\cos(\theta),\ y = r\sin(\theta),$$

the inverse mapping is given by

$$r = \sqrt{x^2 + y^2},\ \theta = \text{atan2}(y, x).$$

For the large aspect ratio mapping (see [2]; the formula is similar, only ω is changed into $\pi - \omega$), we have the formula

$$\begin{aligned} x &= R_0 + r\cos(\omega) - \delta(r) - E(r)\cos(\omega) + T(r)\cos(2\omega) - P(r)\cos(\omega) \\ y &= r\sin(\omega) + E(r)\sin(\omega) - T(r)\sin(2\omega) - P(r)\sin(\omega), \end{aligned}$$

where $\delta,\ E,\ T$ stand for Shafranov shift, elongation and triangularity. The P notation corresponds to a relabeling of the surfaces. We refer to [2], for the physical interest of such mapping in the tokamaks plasma community. We take here $P = 0,\ T = 0,\ \omega = \theta$, together with $E(r) = E_0 r$ and $\delta(r) = \delta_0 r^2$; this clearly restrict the range of geometries, but enables to get an explicit formula for the inverse. We get

$$\begin{aligned} x &= R_0 - \delta_0 r^2 + (1 - E_0) r\cos(\theta) \\ y &= (1 + E_0) r\sin(\theta). \end{aligned} \tag{3}$$

The inverse mapping can be explicitly given. Putting

$$\tilde{y} = \frac{y}{1 + E_0},\ \tilde{x} = \frac{x - R_0}{1 - E_0},\ \tilde{\delta}_0 = \frac{2\delta_0}{1 - E_0},$$

we are lead to solve

$$\left(x + \frac{\tilde{\delta}_0}{2} r^2\right)^2 + y^2 = r^2.$$

We find

$$r = \left(\frac{2\left(\tilde{x}^2 + \tilde{y}^2\right)}{1 - \tilde{\delta}_0\tilde{x} + \sqrt{(1 - \tilde{\delta}_0\tilde{x})^2 - \tilde{\delta}_0^2(\tilde{x}^2 + \tilde{y}^2)}}\right)^{1/2},\ \theta = \text{atan2}(\tilde{y}, \tilde{x} + \frac{\tilde{\delta}_0}{2} r^2).$$

Note that r is well defined as soon as $\tilde{\delta}_0\left(\tilde{x} + \sqrt{\tilde{x}^2 + \tilde{y}^2}\right) \leq 1$, and when $\delta_0 = E_0 = R_0 = 0$, we recover the polar mapping[1]. We refer to [1] (there, the inverse mapping is also needed) and [11, 10] for some works concerning the semi-Lagrangian method combined with a mapping. In the following, we will take

$$E_0 = 0.3, R_0 = 0.08,\ \delta_0 = 0.2. \tag{4}$$

Figure 3 (page 6) shows a non-uniform grid combined with this specific mapping.

[1] This is not the case for the other solution of the polynomial of degree 2 in r^2: $r = \left(\frac{2(\tilde{x}^2+\tilde{y}^2)}{1-\tilde{\delta}_0\tilde{x}-\sqrt{(1-\tilde{\delta}_0\tilde{x})^2-\tilde{\delta}_0^2(\tilde{x}^2+\tilde{y}^2)}}\right)^{1/2}$

2 Operators in complex geometry

2.1 Lagrange interpolation in 2D

Let first consider a uniform mesh to introduce the notations, *i.e* $\forall i \in [\![0, N_r - 1]\!], N_{\theta_{[i]}} = N_\theta$. Let suppose that $r \in [\frac{\Delta r}{2}, r_{\max}[$, $\theta \in [0, 2\pi[$, but also let us define h and k in $[\![0, N_r - 1]\!]$ and $[\![0, N_\theta - 1]\!]$ such as $r_h \leq r < r_{h+1}$ and $\theta_k \leq \theta < \theta_{k+1}$ where $r_h = (h + \frac{1}{2})\Delta r$ and $\theta_k = k\Delta\theta$. Given an order of interpolation $p - 1$, the Lagrange interpolation polynomial equals

$$L^{(p)}(r, \theta) = \sum_{m=l}^{u} \sum_{n=l}^{u} f(r_{h+m}, \theta_{k+n}) L^{(p)}_{m,n}(r, \theta) \tag{5}$$

where $l = -\lfloor \frac{p-1}{2} \rfloor$, $u = \lfloor \frac{p}{2} \rfloor$ and $L^{(p)}_{m,n}$ are the Lagrange basis polynomial. Then the basis polynomial $L^{(p)}_{m,n}(r, \theta)$ associated to the point (r_{h+m}, θ_{k+n}) reads

$$L^{(p)}_{m,n}(r, \theta) = \prod_{\substack{i=l \\ i \neq m}}^{u} \left(\frac{r - r_{h+i}}{r_{h+m} - r_{h+i}}\right) \times \prod_{\substack{j=l \\ j \neq n}}^{u} \left(\frac{\theta - \theta_{k+j}}{\theta_{k+n} - \theta_{k+j}}\right). \tag{6}$$

We can defined a unique set of (β, γ) such as $r = r_h + \beta\Delta r$ and $\theta = \theta_k + \gamma\Delta\theta$, with $(\beta, \gamma) \in [0, 1[^2$. Then Eq. (6) can be simplified to

$$L^{(p)}_{m,n}(r, \theta) = \prod_{\substack{i=l \\ i \neq m}}^{u} \left(\frac{\beta - i}{m - i}\right) \times \prod_{\substack{j=l \\ j \neq n}}^{u} \left(\frac{\gamma - j}{n - j}\right).$$

If the radial position r goes above r_{max} then the coefficient is computed the same way but a Dirichlet condition is used and $f(r, \theta)$ is cast to $f(r_{max}, \theta)$. If the radial position is located in the interval $r \in [0, \frac{\Delta r}{2}[$, the interpolation scheme has to be adapted because we are crossing the most inner radius of the grid. Let suppose, we have $h + m < 0$ in Eq. (5), we have to recast r and θ coordinates at the same time to cross the center at $r = 0$. The new coordinates of a mesh point with $h + m < 0$ located at (r_{h+m}, θ_s) are set to $(r_{-h-m-1}, \theta_{s+\frac{N_\theta}{2}[N_\theta]})$. We basically continue the stencil on the radially opposite side of the grid by performing a π rotation.

Considering a non-uniform mesh as described in Section 1, we need to take into account the cases where the interpolation stencil covers several radii as shown in Fig. 2. In that case, the number of points along θ for each radius may be different and so the index of the nearest point in θ direction may be different. For instance on this Figure, the indexes of the interpolation points on radius r_1 at $(1.5\,\Delta r)$ are 2, 3, 4, 5 ($2\Delta\theta_{[1]}$, $3\Delta\theta_{[1]}$, $4\Delta\theta_{[1]}$, ...) and on radius r_2 $(2.5\,\Delta r)$ these indexes are 5, 6, 7, 8 ($5\Delta\theta_{[2]}$,$6\Delta\theta_{[2]}$,...). In this way, we always use the closest known points, leading to a good accuracy. To adapt the interpolation calculation to non-uniform meshes, with notation of Eq. (5), one only need to first perform the interpolation in θ before the one along r. It allows to easily take into account that the number $N_{\theta_{[i]}}$ depends on radius r_i. Indeed, if the first interpolation was along r, points along the θ direction would possibly not be available (there is possibly not the same number of points in θ for each radius) and it would require extra 1D interpolations along θ to fix this problem. Algorithm 1 summarizes how the 2D Lagrangian interpolation is performed for non-uniform meshes.

2.2 Gyroaverage operator

The gyroaverage operator is a key element in solving the Vlasov-Poisson system of equations, since it allows for the transformation of the guiding center distribution into the actual particle

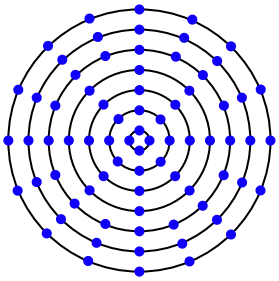


Figure 1: New poloidal grid. The number of points in θ direction (angle) depends on the radial position (distance to the center).

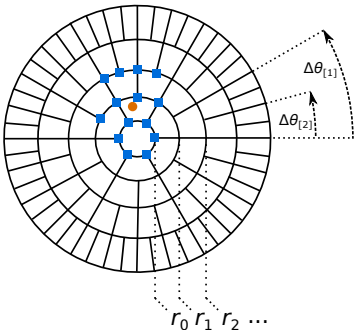


Figure 2: Interpolation of a point ● of the poloidal plane with a stencil of 16 points ■ (Lagrange of order 3).

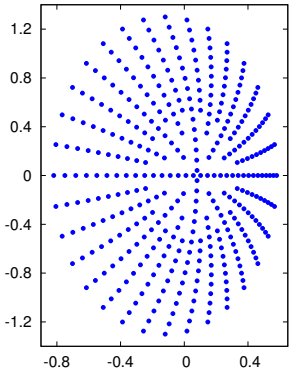


Figure 3: New mapped grid using large aspect ratio equilibrium, with non-uniform meshing along θ.

Algorithm 1: Lagrange interpolation, tensor product in 2D

Data: $f : [\![0, N_r - 1]\!] \times [\![0, N_{\theta_{[i]}} - 1]\!] \to \mathbb{R}$, distribution function
(r, θ), coordinates of the point
p, degree of interpolation.
Result: *value*, interpolation of $f(r, \theta)$.
begin
  $value = 0$
  $h = \lfloor \frac{r - \Delta r/2}{\Delta r} \rfloor$; $\beta = \frac{r - (\Delta r/2 + h\Delta r)}{\Delta r}$ `/* Radial position :` $r = (h + \beta)\Delta r$ `*/`
  $l = -\lfloor \frac{p-1}{2} \rfloor$; $u = \lfloor \frac{p}{2} \rfloor$
  for $m \in [\![l, u]\!]$ **do**
    $c_r = 1$
    for $i \in [\![l, u]\!]$, $i \neq m$ **do** `/* Computation of radial coefficient */`
      $c_r = c_r \times \frac{\beta - i}{m - i}$
    $h' = (h + m)$
    $k = \lfloor \frac{\theta}{\Delta\theta_{[h']}} \rfloor$; $\gamma = \frac{\theta - (k\Delta\theta_{[h']})}{\Delta\theta_{[h']}}$ `/* Poloidal position on radii` m `*/`
    for $n \in [\![l, u]\!]$ **do**
      $c_\theta = 1$
      for $j \in [\![l, u]\!]$, $j \neq n$ **do** `/* Computation of poloidal coefficient */`
        $c_\theta = c_\theta \times \frac{\gamma - j}{n - j}$
      $k' = (k + n)$
      $(h'', k'') =$ `get_plane_indexes`(h', k') `/* Radial boundary conditions */`
      $value = value + f(h'', k'') \times c_r \times c_\theta$
end

distribution, thus reducing the dimensionality of the system of one. The cyclotronic motion of the particles around the magnetic field lines at a distance below the Larmor radius is neglected without loss of accuracy, since this motion is much faster than the turbulence effects usually investigated; moreover, even modern computational power doesn't allow for such highly costly simulations.

A gyroaverage operator has been constructed on the new grid, and we will here briefly describe the numerical implementation which has been adopted in this context.

The gyroaverage operator depending on the spatial coordinates in the polar plane is defined as follows [21]:

$$J_\rho(f)(r,\theta) = \frac{1}{2\,\pi}\int_0^{2\,\pi} g(\mathbf{x}_G + \vec{\rho})d\alpha \tag{7}$$

where $\mathbf{x}_G$ is the guiding center radial coordinate: it is related to $\mathbf{x}$, the position of the particle in the real space, through the Larmor radius ρ, i.e. $\mathbf{x} = \mathbf{x}_G + \vec{\rho}$, which in turn is defined as:

$$\vec{\rho} = \rho(\cos(\alpha)\mathbf{e}_{\perp 1} + \sin(\alpha)\mathbf{e}_{\perp 2})$$

where $\alpha \in [0, 2\pi]$ represents the gyrophase angle and $\mathbf{e}_{\perp 1}$, $\mathbf{e}_{\perp 2}$ the unit vectors of a Cartesian basis in a plane perpendicular to the magnetic field direction $\mathbf{b} = \mathbf{B}/|B|$. The function f and g in equation (7) are defined such that $f : (r,\theta) \in \mathbb{R}^+ \times \mathbb{R} \mapsto f(r,\theta)$ is a polar function and $g : (x_1, x_2) \in \mathbb{R}^2 \mapsto g(x_1, x_2)$ is a Cartesian function such that $g(r\cos\theta, r\sin\theta) = f(r,\theta)$ for any pair (r,θ). The two functions represent an arbitrary field quantity respectively defined on a grid with polar and Cartesian coordinates.

It can be shown [21] that the gyroaverage operator defined in equation (7) can be expressed as a function of the Bessel function of first order, and thus in the Fourier space the gyroaverage is reduced to a multiplication with a Bessel function. In this context though, another approach has been used in order to compute the gyroaverage operator, based on the 2D Lagrangian interpolation. In summary, this method consists in averaging the value of the function over N points equally distributed on a circle of radius ρ: since these points will unlikely correspond to grid points, an interpolation method is used in order to retrieve the value of the function, according to the interpolation procedure described in Section 2.1. This procedure is clarified in Figure 4: the function value for which we want to compute the gyroaverage is marked by an orange circle ●, and the red circumference marks the gyroradius which has been considered. Three triangle green points ▲ are chosen to compute the gyroaverage, and since they do not correspond to any grid point, the value of the function must be retrieved with a preliminary interpolation, using the nearest grid points available, shown in figure as blue squares ■ gathered around the triangles.

We can write the rigorous expression of the operator in the following way [21]:

$$J_\rho(f)_{j,k} \simeq \frac{1}{2\pi}\sum_{\ell=0}^{N-1} P(f)(r_j\cos\theta_k + \rho\cos\alpha_\ell, r_j\sin\theta_k + \rho\sin\alpha_\ell)\Delta\alpha \tag{8}$$

where $\alpha_\ell = \ell\Delta\alpha$, $\Delta\alpha = 2\pi/N$. $P(f)$ is the Lagrange interpolator operator. Radial projection on the border of the domain is used if the points selected for the gyroaverage lie outside the domain for large radius. The requirements on the gyroaverage operator are to be accurate enough in order not to disrupt the data, and to be cheap enough from a computational point of view, since it needs to be applied many times during a simulation. It is expected that the present implementation on the new grid will make the application of the gyroaverage operator cheaper and faster, with a general benefit for the global simulation execution time.

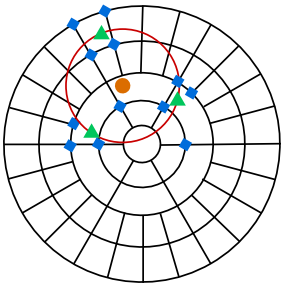


Figure 4: Illustration of the procedure used to compute the gyroaverage: the function value for which we want to compute the gyroaverage is represented by ●, and the red circumference marks the gyroradius which has been considered. The green triangles ▲ are the points chosen to compute the gyroaverage, while the blue squares ■ are those needed for the interpolation of the function on the green points (Lagrange interpolation of order 1).

2.3 Advection operator

Advection consists in the transport of a scalar or vectorial quantity over a vector field. In our case, the transported quantity is the distribution function. The advection is performed backward (Backward Semi-Lagrangian scheme) which means that considering a grid point at time step t^{N+1} we perform the advection with a velocity field in the opposite direction to find where the quantity was at time step t^N (see Figure 14). As the displaced point at time t^N seldom corresponds to another grid point, a Lagrange interpolation is performed.

The general equation solved by the advection operator for the given distribution function f at point (x, y) is:

$$f(x, y, t^{N+1}) = f(x - v_x \Delta t, y - v_y \Delta t, t^N) \tag{9}$$

where v_x and v_y are the velocities along their respective dimensions and Δt is the time step. The right-hand side term is solved as explained above by calling the interpolation operator described in 2.1. In GYSELA, velocities are defined using a Taylor expansion as described in [7, p. 402].

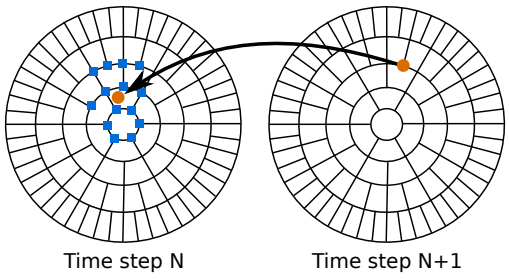


Figure 5: Advection of a point ● of the poloidal plane and interpolation from time step t^N points ■ (Lagrange interpolation of order 3).

2.4 2D finite differences for Poisson solver in polar coordinates

As said in introduction, in a gyrokinetic code the 5D Vlasov equation is coupled to a 3D quasi-neutrality equation. In GYSELA code this last equation is projected in Fourier space in the θ dimension and solved by 1D finite differences in the radial direction. This numerical treatment

is well adapted to concentric circular magnetic configuration but will be no longer applicable to more realistic magnetic configuration. Radial and poloidal directions can indeed no more be split and a 2D treatment of the poloidal (r, θ) cross-section is required. A 2D finite element method is often used in the gyrokinetic codes including D-shape magnetic configurations. For the Poisson solver, we will examine two specific meshes: (i) a non-uniform circular mesh (see Figure 1) and (ii) a uniform mesh based on a large aspect ratio equilibrium and mapping (see Section 1.2). We choose to use finite differences to solve this problem. In this section, we consider the 2D Poisson equation in polar coordinates on a domain Ω,

$$\frac{\partial^2 f}{\partial r^2} + \frac{1}{r}\frac{\partial f}{\partial r} + \frac{1}{r^2}\frac{\partial^2 f}{\partial \theta^2} = R(r, \theta) \tag{10}$$

with Dirichlet boundary conditions $f(r = r_{\max}, \theta) = g(\theta)$ on $\partial\Omega$.

2.4.1 2D finite differences for a non-uniform circular mesh

Let us first consider equation (10) on a disk $\Omega = \{(r, \theta) : 0 < r < r_{\max}$ with $r_{\max} \in \mathbb{R}$ and $0 \leq \theta \leq 2\pi\}$ where Ω is described by a non-uniform circular mesh Ω_k. To overcome the singularity problem at $r = 0$, we use the same centered finite difference method as proposed in Lai's paper [14] [2]. One of the trick consists in solving Equation (10) for $r \in [r_{\min}, r_{\max}]$ with $r_{\min} = \Delta r/2$ and a half-integered grid in radial direction and an integered grid in poloidal direction. In this section, we propose an extension of the method proposed for an uniform circular mesh by Lai to a non-uniform one. The difficulty is to adapt the scheme to allow a different number of poloidal points per radius. This implies the adding of interpolations. The scheme proposed in the following is based on Lagrange interpolation of third order which is a good compromise between accuracy and complexity. Let $N = N_r - 1$ be the number of cells in radial direction and $N_{\theta_{[i]}}$ be the number of cells along θ on the circle of radius r_i. Let us call γ_i the ratio between number of poloidal mesh points for $r = r_i$ and the one for circle of radius $r = r_{i-1}$, namely $\gamma_i = N_{\theta_{[i]}}/N_{\theta_{[i-1]}}$. Let us add the two constraints on Ω_k: (i) $N_{\theta_{[i]}}$ is even and (ii) $\gamma_i \geq 1$. Then, Ω_k is defined as

$$r_i = (i - \frac{1}{2})\Delta r \quad \text{for all } i = 1, 2, \cdots, N+1 \tag{11}$$

$$\theta_{j_{[i]}} = (j_{[i]} - 1)\Delta\theta_{[i]} \quad \text{for all } j_{[i]} = 1, 2, \cdots, N_{\theta_{[i]}} + 1 \tag{12}$$

where $\Delta r = r_{\max}/(N + 1/2)$ and $\Delta\theta_{[i]} = 2\pi/N_{\theta_{[i]}}$. Let us notice that these indexes used are different from the one used in section 1.1.2 (indices starting here at 1 instead of 0 previously). Let the discrete values be denoted by $R_{i,j} = R(r_i, \theta_j)$, $g_j = g(\theta_j)$ and $f_{i,j} = f_{i,j_{[i]}}$ where $f_{i,j_{[k]}} = f(r_i, \theta_{j_{[k]}})$. Then, the discrete version of Eq. (10) becomes, for $i = 1, \cdots, N$ and $j_{[i]} = 1, 2, \cdots, N_{\theta_{[i]}}$:

$$\frac{f(r_{i+1}, \theta_{j_{[i]}}) - 2f_{i,j} + \tilde{f}(r_{i-1}, \theta_{j_{[i]}})}{(\Delta r)^2} + \frac{1}{r_i}\frac{f(r_{i+1}, \theta_{j_{[i]}}) - \tilde{f}(r_{i-1}, \theta_{j_{[i]}})}{2\Delta r} + \frac{1}{r_i^2}\frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta\theta_{[i]})^2} = R_{i,j} \tag{13}$$

where the boundary values are given: (i) radially by the Dirichlet condition $f_{N+1,j} = g_j$ for all $j \in [1, N_{\theta_{[N+1]}}]$ and (ii) poloidally by $f_{i,0} = f_{i,N_{\theta_{[i]}}}$ and $f_{i,1} = f_{i,N_{\theta_{[i]}}+1}$ for all $i \in [1, N+1]$

[2] Note that another trick to cope with the singularity at the origin can be found in [22], p334, see also [23] and references therein for further references. In particular, the idea to use a shifted mesh of one half mesh size seems to date back to [4].

due to 2π periodic boundary conditions. The term $f(r_{i+1},\theta_{j_{[i]}})$ is equal to $f(r_{i+1},\theta_{j_{[i+1]}})$ with $j_{[i+1]} = (j_{[i]}-1)\gamma_{i+1}+1$ where $(r_{i+1},\theta_{j_{[i+1]}})$ is a mesh point, so $f(r_{i+1},\theta_{j_{[i]}}) = f_{i+1,(j_{[i]}-1)\times\gamma_{i+1}+1}$. The term $\tilde{f}(r_{i-1},\theta_{j_{[i]}})$ corresponds to an approximation of f at the point $(r_{i-1},\theta_{j_{[i]}})$ because if $\gamma_i > 1$ then $\theta_{j_{[i]}}$ is not automatically a mesh point (see Figure 6). The value $\tilde{f}(r_{i-1},\theta_{j_{[i]}})$ is defined as

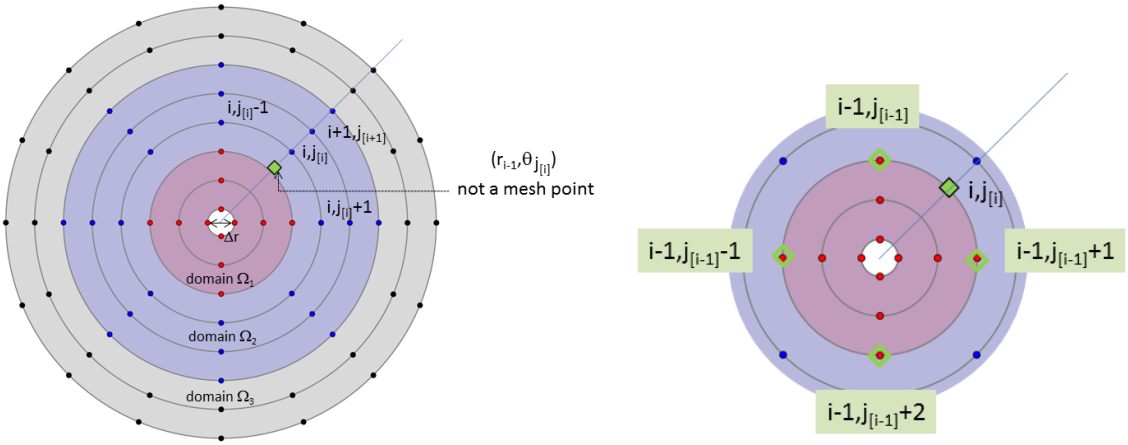
$$\tilde{f}(r_{i-1},\theta_{j_{[i]}}) = \begin{cases} f(r_{i-1},\theta_{j_{[i-1]}}) \text{ if } j_{[i-1]} = (j_{[i]}-1)/\gamma_i + 1 \in \mathbb{N} \\ \text{approximation of } f \text{ at position } (r_{i-1},\theta_{j_{[i]}}) \text{ otherwise} \end{cases} \tag{14}$$

The required approximations are computed by using a Lagrange interpolation of third order. So let us consider $k \in \mathbb{N}$ the integer such that $\theta_k < \theta_{j_{[i-1]}} < \theta_{k+1}$, then using (5)-(6) notations,

$$\tilde{f}(r_{i-1},\theta_{j_{[i]}}) \approx \sum_{n=-1}^{2} L_n^{(3)}(\theta_{j_{[i-1]}}) f(r_{i-1},\theta_{k+n}) \quad \text{with } j_{[i-1]} = \text{int}\left(\frac{j_{[i]}-1}{\gamma_i}\right) + 1 \tag{15}$$

where the Lagrange polynomials $L_n^{(3)}$ are defined by

$$L_n^{(3)}(\theta) = \prod_{\substack{i=-1 \\ i\neq n}}^{2} \frac{(\theta-\theta_{k+i})}{(\theta_{k+n}-\theta_{k+i})} \quad \text{for all } \theta \in [\theta_k,\theta_{k+1}[\quad \text{and } n = -1,\cdots,2 \tag{16}$$



(a) (b)

Figure 6: (a) Example of a non-uniform mesh divided into three parts with 4 points by radius in the first one, 8 in the second one and 16 points for the last one. To compute value at mesh point (r_i,θ_j) with finite differences of second order the points $(r_i,\theta_{j_{[i]}+1})$, $(r_i,\theta_{j_{[i]}-1})$, $(r_{i+1},\theta_{j_{[i]}})$ and $(r_{i-1},\theta_{j_{[i]}})$. The problem is that the last one $(r_{i-1},\theta_{j_{[i]}})$ is not a mesh point. (b) The value at position $(r_{i-1},\theta_{j_{[i]}})$ is computed by Lagrange interpolation of 3rd order by using the mesh points represented by a square.

Let us define, for all $i \in [1, N+1]$,

$$\lambda_i = \frac{\Delta r}{2r_i} = \frac{1}{2(i-1/2)} \quad \text{and} \quad \beta_i = \frac{(\Delta r)^2}{r_i^2(\Delta\theta_{[i]})^2} = \frac{1}{(i-1/2)^2(\Delta\theta_{[i]})^2} \tag{17}$$

Then, equation (13) reads

$$\begin{aligned}(1-\delta_{i,1})(1-\lambda_i)\tilde{f}(r_{i-1},\theta_{j_{[i]}}) + \beta_i\left[(1-\delta_{j,1})f_{i,j-1} + \delta_{j,1}f_{i,N_{\theta_{[i]}}}\right]\\ -(2+2\beta_i)f_{i,j} + \beta_i\left[(1-\delta_{j,N_{\theta_{[i]}}})f_{i,j+1} + \delta_{j,N_{\theta_{[i]}}}f_{i,1}\right]\\ +(1-\delta_{i,N})(1+\lambda_i)f_{i+1,(j_{[i]}-1)\gamma_{i+1}+1} = (\Delta r)^2 R_{i,j} - \delta_{i,N}(1+\lambda_N)g_j \end{aligned} \tag{18}$$

Let us notice that due to the choice of $r_1 = \Delta r/2$, $(1-\lambda_1) = 0$. At the opposite of what is proposed in Lai's paper, let us order the unknowns $f_{i,j}$ radius by radius, such that the unknown vector u of size $N_{\text{tot}} = \sum_1^N N_{\theta_{[i]}}$ is defined as

$$u^t = [f_1\ , f_2\ , \cdots, f_N] \quad \text{with} \quad f_i^t = \left[f_{i,1}\ , f_{i,2}\ , \cdots, f_{i,N_{\theta_{[i]}}}\right] \tag{19}$$

The matrix system associated to the discrete equation system (18) reads $Au = b$ where A is a $N_{\text{tot}} \times N_{\text{tot}}$ sparse matrix, given by

$$A = \begin{bmatrix} A_1 & \Lambda_1^+ & & & & & \\ \tilde{\Lambda}_2^- & A_2 & \Lambda_2^+ & & & 0 & \\ & \ddots & \ddots & \ddots & & & \\ & & \tilde{\Lambda}_i^- & A_i & \Lambda_i^+ & & \\ & & & \ddots & \ddots & \ddots & \\ & 0 & & & \tilde{\Lambda}_{N-1}^- & A_{N-1} & \Lambda_{N-1}^+ \\ & & & & & \tilde{\Lambda}_N^- & A_N \end{bmatrix} \tag{20}$$

The N matrices A_i are $N_{\theta_{[i]}} \times N_{\theta_{[i]}}$ matrix defined as

$$A_i = \begin{bmatrix} -(2+2\beta_i) & \beta_i & 0 & & \beta_i \\ \beta_i & -(2+2\beta_i) & \beta_i & & \\ & & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_i \\ \beta_i & 0 & & \beta_i & -(2+2\beta_i) \end{bmatrix} \tag{21}$$

The matrix Λ_i^+ is a $N_{\theta_{[i]}} \times N_{\theta_{[i+1]}}$ matrix while $\tilde{\Lambda}_i^-$ is a $N_{\theta_{[i]}} \times N_{\theta_{[i-1]}}$ matrix, both defined as

$$\Lambda_i^+ = \begin{bmatrix} D_i^1 \\ \vdots \\ D_i^k \\ \vdots \\ D_i^{N_{\theta_{[i]}}} \end{bmatrix} \quad \text{and} \quad \tilde{\Lambda}_i^- = \begin{bmatrix} C_i^1 \\ \vdots \\ C_i^k \\ \vdots \\ C_i^{N_{\theta_{[i-1]}}} \end{bmatrix} \tag{22}$$

where D_i^k is a line matrix $1 \times N_{\theta_{[i+1]}}$ matrix where all elements are equal to 0 except the $((k-1)\gamma_{i+1}+1)$th term is equal to $(1+\lambda_i)$. Furthermore, C_i^k is the $\gamma_i \times N_{\theta_{[i-1]}}$ matrix given by

$$C_i^k = (1-\lambda_i)\begin{bmatrix} 0 & & \cdots & 0 & 1 & 0 & \cdots & & 0 \\ & \cdots & 0 & L_{-1}(\kappa_i^k+1) & L_0(\kappa_i^k+1) & L_1(\kappa_i^k+1) & L_2(\kappa_i^k+1) & 0 & \cdots \\ & & 0 & \vdots & \vdots & \vdots & \vdots & 0 & \\ 0 & \cdots & 0 & L_{-1}(\kappa_i^k+\gamma_i) & L_0(\kappa_i^k+\gamma_i) & L_1(\kappa_i^k+\gamma_i) & L_2(\kappa_i^k+\gamma_i) & 0 & \cdots & 0 \end{bmatrix}$$

with $\kappa_i^k = (k-1)\gamma_i + 1$ and L_m the Lagrange polynomials defined by Eq.(16) where for more readability, $L_m(l) = L_m(\theta_{j_{[i]=l}})$. Let us notice that the column position of the value 1 of first row of C_i^k is equal to k. Finally, the right hand side vector b can be expressed as

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \quad \text{with} \quad b_i = (\Delta r)^2 \begin{bmatrix} R_{i,1} \\ R_{i,2} \\ \vdots \\ R_{i,N_{\theta_{[i]}}} \end{bmatrix} - \delta_{i,N}(1+\lambda_N) \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_N \end{bmatrix} \tag{23}$$

Let us notice, that Poisson equation (10) on a circular uniform mesh ($N_{\theta_{[i]}} = N_\theta, \forall i$) can be trivially deduced from the previous $Au = b$ matrix system. The $N_{\rm tot} \times N_{\rm tot}$ matrix A is given by Eq. (20) with $N_{\rm tot} = N\,N_\theta$ where $\tilde{\Lambda}_i^-$ and $\tilde{\Lambda}_i^+$ are $N_\theta \times N_\theta$ diagonal matrices with $\mathrm{diag}(\tilde{\Lambda}_i^\pm) = 1 \pm \lambda_i$.

2.4.2 2D finite differences on a mapped uniform mesh

One difficulty was to extend the Poisson solver [14] to a non uniform mesh, as done in the previous subsection. Another one is to deal with a mapping. So, we focus here on this point, starting with a uniform mesh. The combination of both schemes will be the subject of further work and is not tackled here. We refer to [11] for the use of a Mudpack solver, and [10] for the use of a finite element solver based on B-splines. Such solvers might be adapted, but here we consider a specific treatment for the center; so we develop a stand-alone solution with finite differences. Note that some adaptations have to be done with respect to the previous case [14] and we will propose two examples of solvers with 7 and 9 points (we could not get a 5 points solution, here due to the appearance of mixed terms from the mapping as we will see). We consider here the Poisson equation first on a elliptic domain and then for the large aspect ratio mapping (see Eq. (3) and (4) for the latter). We write

$$x(r,\theta) = ar\cos(\theta),\ y(r,\theta) = br\sin(\theta).$$

From Eq.(10), which reads $\Delta U = F$ and writing $u(r,\theta) = U(x(r,\theta), y(r,\theta)),\ f(r,\theta) = F(x(r,\theta), y(r,\theta))$, we have the relations

$$\nabla_{x,y} U = J^{-T} \nabla_{r,\theta} u,\ \nabla_{x,y} \cdot A = \frac{1}{|J|} \nabla_{r,\theta} \cdot (|J| J^{-1} A),\ J = \begin{pmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{pmatrix},\ |J| = \det(J),$$

which lead to

$$\Delta U = \nabla_{x,y} \cdot \nabla_{x,y} U = \frac{1}{|J|} \nabla_{r,\theta} \cdot (|J| G \nabla_{r,\theta} u)\,,\ G = J^{-1} J^{-T}.$$

We have here

$$J^{-T} = \frac{1}{|J|} \begin{pmatrix} \frac{\partial y}{\partial \theta} & -\frac{\partial y}{\partial r} \\ -\frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial r} \end{pmatrix},\ |J|G = \frac{1}{|J|} \begin{pmatrix} (\frac{\partial x}{\partial \theta})^2 + (\frac{\partial y}{\partial \theta})^2 & -\frac{\partial x}{\partial r}\frac{\partial x}{\partial \theta} - \frac{\partial y}{\partial r}\frac{\partial y}{\partial \theta} \\ -\frac{\partial x}{\partial r}\frac{\partial x}{\partial \theta} - \frac{\partial y}{\partial r}\frac{\partial y}{\partial \theta} & (\frac{\partial x}{\partial r})^2 + (\frac{\partial y}{\partial r})^2 \end{pmatrix}.$$

For an ellipse, we have $|J| = abr$ and

$$J^{-T} = \frac{1}{|J|} \begin{pmatrix} br\cos(\theta) & -b\sin(\theta) \\ ar\sin(\theta) & a\cos(\theta) \end{pmatrix},\ |J|G = \frac{1}{|J|} \begin{pmatrix} r^2\left(b^2\cos^2(\theta) + a^2\sin^2(\theta)\right) & (a^2-b^2) r\sin(\theta)\cos(\theta) \\ (a^2-b^2) r\sin(\theta)\cos(\theta) & a^2\cos^2(\theta) + b^2\sin^2(\theta) \end{pmatrix}$$

For the large aspect ratio mapping, we have $|J| = (1-E_0)br - 2\delta_0 br^2\cos(\theta)$,

$$J^{-T} = \frac{1}{|J|} \begin{pmatrix} (1+E_0) r\cos(\theta) & -(1+E_0)\sin(\theta) \\ (1-E_0) r\sin(\theta) & (1-E_0)\cos(\theta) - 2\delta_0 r \end{pmatrix}$$

and

$$|J|G = \frac{1}{|J|}\begin{pmatrix} r^2((1+E_0)^2\cos^2(\theta)+(1-E_0)^2\sin^2(\theta)) & -4E_0 r\cos(\theta)\sin(\theta)-2(1-E_0)\delta_0 r^2\sin(\theta) \\ -4E_0 r\cos(\theta)\sin(\theta)-2(1-E_0)\delta_0 r^2\sin(\theta) & (1+E_0)^2\sin^2(\theta)+((1-E_0)\cos(\theta)-2\delta_0 r)^2 \end{pmatrix}.$$

Writing $|J|G = (a_{ij})$, we get the equation

$$\frac{\partial}{\partial r}\left(a_{11}\frac{\partial u}{\partial r}+a_{12}\frac{\partial u}{\partial \theta}\right)+\frac{\partial}{\partial \theta}\left(a_{21}\frac{\partial u}{\partial r}+a_{22}\frac{\partial u}{\partial \theta}\right)=|J|f,\ 0<r<1,\ 0\le\theta<2\pi$$

and $u(1,\theta)=g(\theta)$.
Let $N, M \in \mathbb{N}^*$. We write

$$r_i=(i-1/2)\Delta r,\ \Delta r=\frac{2}{2N+1},\ i=1,\dots,N+1;\ \theta_j=(j-1)\Delta\theta,\ \Delta\theta=\frac{2\pi}{M},\ j=1,\dots,M+1.$$

We consider the following finite difference scheme with 7 points

$$\begin{aligned}
&\frac{1}{\Delta r^2}\left(a_{11}^{i+1/2,j}(u_{i+1,j}-u_{ij})-a_{11}^{i-1/2,j}(u_{i,j}-u_{i-1,j})\right)\\
&+\frac{1}{2\Delta r\Delta\theta}\left(a_{12}^{i+1/2,j}(u_{i,j+1}-u_{ij})-a_{12}^{i-1/2,j}(u_{i-1,j+1}-u_{i-1,j})\right)\\
&+\frac{1}{2\Delta r\Delta\theta}\left(a_{12}^{i+1/2,j}(u_{i+1,j}-u_{i+1,j-1})-a_{12}^{i-1/2,j}(u_{i,j}-u_{i,j-1})\right)\\
&+\frac{1}{2\Delta r\Delta\theta}\left(a_{21}^{i,j+1/2}(u_{i+1,j}-u_{ij})-a_{21}^{i,j-1/2}(u_{i+1,j-1}-u_{i,j-1})\right)\\
&+\frac{1}{2\Delta r\Delta\theta}\left(a_{21}^{i,j+1/2}(u_{i,j+1}-u_{i-1,j+1})-a_{21}^{i,j-1/2}(u_{i,j}-u_{i-1,j})\right)\\
&+\frac{1}{\Delta\theta^2}\left(a_{22}^{i,j+1/2}(u_{i,j+1}-u_{ij})-a_{22}^{i,j-1/2}(u_{i,j}-u_{i,j-1})\right)=|J|_{ij}f_{ij},
\end{aligned}$$

for $j=1,\dots,M$ and $i=1,\dots,N$. We have here $f_{ij}=f(r_i,\theta_j)$, $|J_{ij}|=J(r_i,\theta_j)$ and $a_{k\ell}^{pq}=a_{k\ell}(r_p,\theta_q)$, $p,q\in\frac{1}{2}\mathbb{Z}$. The system is modified as follows in order to deal with the boundary conditions

- u_{ij} is replaced by $u_{i\tilde{j}}$ where $j=\tilde{j}+kM, k\in\mathbb{Z}$ and $1\le\tilde{j}\le M$.
- $u_{0,j}$ is replaced by $u_{1,M/2+j}$ for $j=1,\dots,M/2$ and by $u_{1,M/2-j}$ for $j=M/2+1,\dots,M$.
- $u_{N+1,j}$ is replaced by $g_j=g(\theta_j)$, for $j=1,\dots,M$.

Note that here a 7 points stencil is needed. We have to take special care on the boundary condition: $u_{0,j}$ does not cancel and it is replaced by $u_{1,N/2-j}$ (we assume here that N is even). For the case of a circle, we get the standard 5 points stencil (the terms $a_{21}, a_{2,2}$ cancel) and $u_{0,j}$ cancels. Next, we give also another scheme with a 9 points stencil, that is using contributions of $u_{i-1,j-1}$ and $u_{i+1,j+1}$. With respect to [14], we underline that the following adaptations have been done:

- a 5 points stencil is (at least seems) no more possible for second order accuracy because of the mixed terms, and several schemes are possible (see [13] p204, [9] p103 and [22] p335 for similar schemes using a stencil with 7 or 9 points)
- due to the mixed terms, the term $u_{0,j}$ does not cancel, we have to use the value $u_{1,N/2-j}$, as done in [15].

A 9 points scheme We now derive another scheme for the mixed terms $a_{k,\ell} = a_{21}^{k,\ell} = a_{12}^{k,\ell}$. For $\frac{\partial}{\partial r}\frac{\partial au}{\partial \theta}$, we can use $\frac{1}{4}a_{i-1/2,j+1/2}(u_{i,j+1}+u_{i,j}+u_{i-1,j+1}+u_{i-1,j})$ and $\frac{1}{4}a_{i-1/2,j-1/2}(u_{i,j}+u_{i,j-1}+u_{i-1,j}+u_{i-1,j-1})$, which gives

$$\frac{1}{4\Delta\theta}a_{i-1/2,j-1/2}(u_{i,j}+u_{i,j-1}+u_{i-1,j}+u_{i-1,j-1})-\frac{1}{4\Delta\theta}a_{i-1/2,j+1/2}(u_{i,j+1}+u_{i,j}+u_{i-1,j+1}+u_{i-1,j}),$$

and

$$\begin{aligned}4\Delta r\Delta\theta\frac{\partial}{\partial r}\frac{\partial au}{\partial\theta} \quad &\simeq -a_{i+1/2,j-1/2}(u_{i+1,j}+u_{i+1,j-1}+u_{i,j}+u_{i,j-1})\\ &+a_{i+1/2,j+1/2}(u_{i+1,j+1}+u_{i+1,j}+u_{i,j+1}+u_{i,j})\\ &+a_{i-1/2,j-1/2}(u_{i,j}+u_{i,j-1}+u_{i-1,j}+u_{i-1,j-1})\\ &-a_{i-1/2,j+1/2}(u_{i,j+1}+u_{i,j}+u_{i-1,j+1}+u_{i-1,j}).\end{aligned}$$

We have also

$$\frac{\partial}{\partial r}a\frac{\partial u}{\partial\theta}+\frac{\partial}{\partial\theta}a\frac{\partial u}{\partial r}=\frac{\partial}{\partial r}\frac{\partial au}{\partial\theta}-\frac{\partial}{\partial r}\frac{\partial a}{\partial\theta}u+\frac{\partial}{\partial r}\frac{\partial u}{\partial\theta}a.$$

We finally get [3]

$$\begin{aligned}&4\Delta r\Delta\theta\left(\frac{\partial}{\partial r}a\frac{\partial u}{\partial\theta}+\frac{\partial}{\partial\theta}a\frac{\partial u}{\partial r}\right)\\ &=(a_{i,j}+a_{i+1/2,j+1/2})u_{i+1,j+1}+(a_{i,j}+a_{i-1/2,j-1/2})u_{i-1,j-1}\\ &-(a_{i,j}+a_{i-1/2,j+1/2})u_{i-1,j+1}-(a_{i,j}+a_{i+1/2,j-1/2})u_{i+1,j-1}\\ &+(a_{i+1/2,j+1/2}-a_{i+1/2,j-1/2})u_{i+1,j}-(a_{i-1/2,j+1/2}-a_{i-1/2,j-1/2})u_{i-1,j}\\ &+(a_{i+1/2,j+1/2}-a_{i-1/2,j+1/2})u_{i,j+1}-(a_{i+1/2,j-1/2}-a_{i-1/2,j-1/2})u_{i,j-1}\\ &-3\left(a_{i+1/2,j+1/2}+a_{i-1/2,j-1/2}-a_{i-1/2,j+1/2}-a_{i+1/2,j-1/2}\right)u_{i,j}.\end{aligned}$$

3 Convergence results

3.1 Interpolation

The interpolation operator is of utmost importance, it is used as a building block by more complex operators. As such, it is essential that this operator remains accurate enough to keep

[3] We have the intermediate steps:

$$\begin{aligned}&4\Delta r\Delta\theta\left(\frac{\partial}{\partial r}a\frac{\partial u}{\partial\theta}+\frac{\partial}{\partial\theta}a\frac{\partial u}{\partial r}\right)\\ &=-a_{i+1/2,j-1/2}(u_{i+1,j}+u_{i+1,j-1}+u_{i,j}+u_{i,j-1})+a_{i+1/2,j+1/2}(u_{i+1,j+1}+u_{i+1,j}+u_{i,j+1}+u_{i,j})\\ &+a_{i-1/2,j-1/2}(u_{i,j}+u_{i,j-1}+u_{i-1,j}+u_{i-1,j-1})-a_{i-1/2,j+1/2}(u_{i,j+1}+u_{i,j}+u_{i-1,j+1}+u_{i-1,j})\\ &+a_{i,j}(u_{i+1,j+1}+u_{i-1,j-1}-u_{i-1,j+1}-u_{i+1,j-1})\\ &-4\left(a_{i+1/2,j+1/2}+a_{i-1/2,j-1/2}-a_{i-1/2,j+1/2}-a_{i+1/2,j-1/2}\right)u_{i,j}\\ &=-a_{i+1/2,j-1/2}(u_{i+1,j}+u_{i+1,j-1}+u_{i,j-1})+a_{i+1/2,j+1/2}(u_{i+1,j+1}+u_{i+1,j}+u_{i,j+1})\\ &+a_{i-1/2,j-1/2}(u_{i,j-1}+u_{i-1,j}+u_{i-1,j-1})-a_{i-1/2,j+1/2}(u_{i,j+1}+u_{i-1,j+1}+u_{i-1,j})\\ &+a_{i,j}(u_{i+1,j+1}+u_{i-1,j-1}-u_{i-1,j+1}-u_{i+1,j-1})\\ &-3\left(a_{i+1/2,j+1/2}+a_{i-1/2,j-1/2}-a_{i-1/2,j+1/2}-a_{i+1/2,j-1/2}\right)u_{i,j}\end{aligned}$$

the simulated physics valid. Performance are not detailed in the paper though it is critical and impact almost every piece of the code. It will be presented in future work where it will be integrated in GYSELA and compared to previous schemes. The accuracy of the Lagrange interpolation depends on three parameters: the degree of the Lagrange polynomial, the mesh discretization in the r direction and in the θ direction.

In order to test our implementation, we perform interpolations from the polar mesh to a uniform Cartesian grid of size $[-r_{max} : r_{max}] \times [-r_{max} : r_{max}]$ with $2\,N_r$ points in each direction. Points outside of the polar mesh are discarded. The mesh is initialized using a sine product $f(x, y) = \sin(5x) \times \cos(4y)$ for the following tests. Solution is thus analytically known everywhere on the plane. The following figures give different norms (L_1, L_2 and L_{inf}) of the error done when performing the interpolation on the whole Cartesian grid. The results are given for the uniform and non-uniform mesh. The base mesh used in the simulation is: $N_r = 256$, $N_\theta = 256$ and Lagrange order is 7 for the uniform mesh. For the non-uniform mesh the Lagrange order is also 7, $N_r = 256$ and the $N_{\theta_{[i]}}$ are given as such $(2 : \underline{32}, 8 : \underline{64}, 64 : \underline{128}, 182 : \underline{256})$, which reads: there are 32 points in θ direction on the 2 inner most *radii* ($N_{\theta_{[0]}} = N_{\theta_{[1]}} = 32$), 64 points on the 8 following *radii*, and so on. This gives 15% less points for the non-uniform mesh than for the uniform one with $N_\theta = 256$.

In Figure 7, the error is presented against the degree of the Lagrange interpolation which ranges from 1 to 15. Both the uniform and the non-uniform meshes have the same behaviour. There is a convergence phase where the error decreases steadily before stopping at a plateau. It either reaches hardware precision or the accuracy allowed by the meshing on both dimensions. In this case it is hardware precision for a double-precision floating-point (10^{-15}) which is achieved by L_{inf} norm (maximum value of the error). The non-uniform mesh proves to be less accurate because the set of values of $N_{\theta_{[i]}}$ we have chosen is good but not optimal.

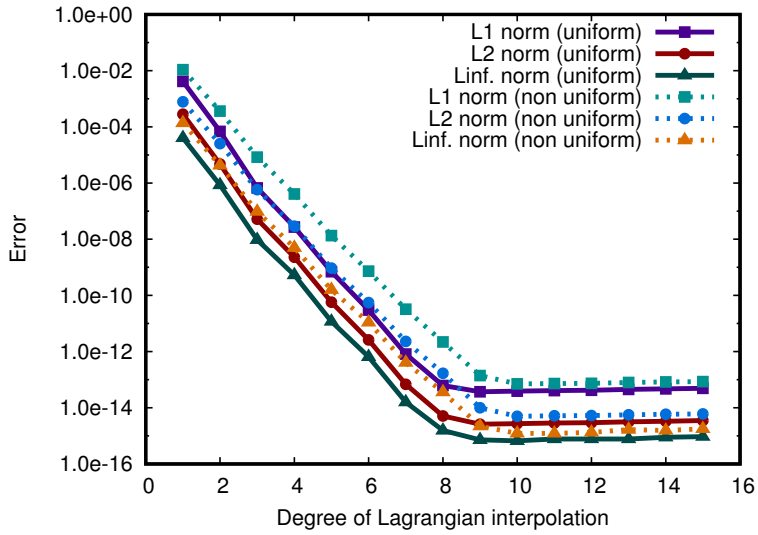
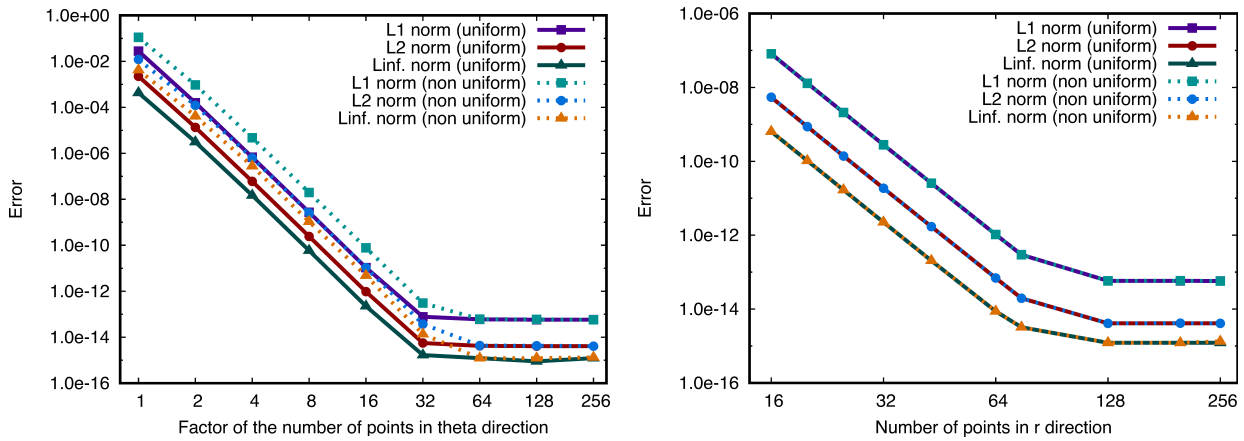


Figure 7: Convergence in degree for the 2D Lagrangian interpolation.

Figure 8a, representing the convergence in θ direction, shows the error against the discretization factor along θ. For the uniform mesh we start at 1 with a mesh of size of $N_r = 256$, $N_\theta = 16$, then 2 with $N_r = 256$, $N_\theta = 32$ and then at n with $N_r = 256$, $N_\theta = 2^{n-1} \times 16$. For the non-uniform mesh at a general discretization factor n we have $N_r = 256$ and $(2 : \underline{2^{n-1} \times 2}, 8 :$



(a) Convergence in θ dimension.

(b) Convergence in r dimension.

Figure 8: Space convergence for the 2D Lagrangian interpolation.

$\underline{2^{n-1} \times 4}, 64 : \underline{2^{n-1} \times 8}, 182 : \underline{2^{n-1} \times 16}$). On the convergence study in θ direction we find the same difference between the two meshes as shown in Figure 7. The convergence rate is the same which assesses the correctness of the interpolation operator. On Figure 8b the curves perfectly match because the number of points in θ direction is chosen high enough not to influence accuracy (2048 on uniform mesh and ranged from 256 at the center to 2048 at the outter edge on non-uniform mesh). As the meshing method along r has not been changed, both mesh types gives the same convergence results.

On these simple test functions (sine products), using a more complex set of $N_{\theta_{[i]}}$ tailored to each specific function allows to reach a reduction of more than a half of the number of points with even fewer accuracy loss compared to a uniform mesh. Whether such $N_{\theta_{[i]}}$ set exists for realistic distribution functions is still unknown.

3.2 Gyroaverage operator

Numerical results concerning the verification of the implementation of the gyroaverage operator, described in Section 2.2, will now be presented.

A certain family of functions has been considered, whose analytical gyroaverage is known. More precisely, their gyroaverage can be obtained simply by a multiplication of a Bessel function [21]. Given the function

$$f(r, \theta) = C_m(zr) \exp(im\theta), \tag{24}$$

where $m \geq 0$ is an integer which defines the index of the Bessel function C_m (the symbols J_m and Y_m are used to identify respectively the Bessel functions of the first and the second kind); $z \in \mathbb{C}$ and $(r,\ \theta)$ represent the usual polar coordinates. The gyroaverage of the function described in Eq. (24) can be written as in [21]:

$$J_\rho(f)(r_0, \theta_0) = J_0(z\rho) C_m(zr_0) \exp(im\theta_0), \tag{25}$$

where ρ is the gyroradius, while (r_0, θ_0) define a specific point in the polar mesh.

We have to consider also the boundary conditions. In our case it corresponds to set an homogeneous Dirichlet condition on $r_{\max}$. We can now list the family of functions we used as

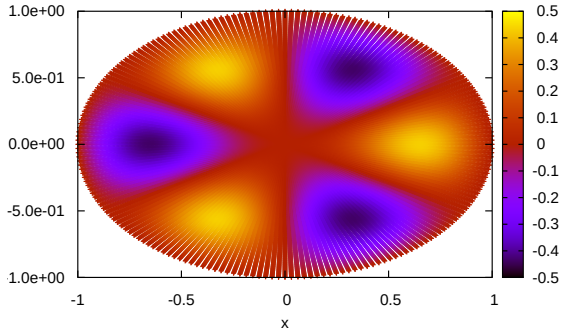


Figure 9: A color map of the test function used for the convergence tests of the gyroaverage operator. The exact expression of the function is $f_1(r,\theta) = J_m\Big(r\frac{j_{m,\ell}}{r_{\max}}\Big)\exp(im\theta)$, where the order of the Bessel function considered was $m = 3$ and its first zero was used.

test cases. It comes directly from the definition (24), and it's written as:

$$f_1(r,\theta) = J_m\Big(r\frac{j_{m,\ell}}{r_{\max}}\Big)\exp(im\theta), \tag{26}$$

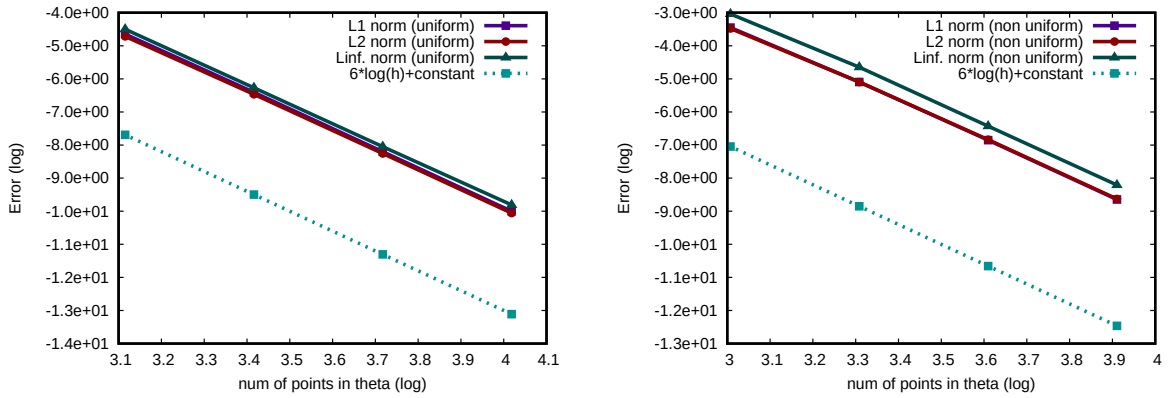
where $j_{m,\ell}$ is the ℓ-th zero of J_m. The function in Eq. 26 is defined on a disk $[0, r_{\max}]\times[0, 2\pi]$ and verifies the Dirichlet boundary condition for which $f_1(r_{\max},\theta) = 0, \quad 0 \le \theta \le 2\pi$. The analytical gyroaverage of (26) evaluated at the point (r_0,θ_0) is [21]:

$$J_\rho(f_1)(r_0,\theta_0) = J_0\Big(\rho\frac{j_{m,\ell}}{r_{\max}}\Big)f_1(r_0,\theta_0) \tag{27}$$

The convergence study results will be shown for the first class of functions presented, described by Eq. (26): the order of the Bessel function has been chosen equal to 3, and the first zero has been considered in the argument. A plot of this function can be seen in Figure 9. The uniform and non uniform grid cases have been addressed, and the convergence tests have been performed in both the r and θ directions as well as in the degree of the underlying Lagrange interpolation. Similar tests have been repeated changing the parameters of the test function, namely the order of the Bessel function and the particular zero of the Bessel function chosen: not all of them are shown here, as the results are very close to the ones presented in this section.

3.2.1 Convergence tests

Figure 10 presents the tests for the convergence in the θ direction: in particular, the logarithm of the L^2 norm, L^1 norm and L^∞ norm are shown with respect to the number of points in the θ direction in a logarithmic scale. The degree of the Lagrange interpolation was fixed to be equal to 5, while the number of points in the r direction and on the gyroaverage circle was respectively equal to $N_r = 256$ and $N_{gp} = 128$, in order to avoid spurious errors related to these parameters; the gyroradius was set equal to $\rho = 0.1$. For the uniform grid (described in Section 1.1.1), the number of points in the θ direction was respectively 32, 64, 128 and 256. For the non uniform case however the grid (described in section 1.1.2) has been built respectively as $(10 : \underline{4}, 30 : \underline{8}, 50 : \underline{16}, 166 : \underline{32})$, $(10 : \underline{8}, 30 : \underline{16}, 50 : \underline{32}, 166 : \underline{64})$, $(10 : \underline{16}, 30 : \underline{32}, 50 : \underline{64}, 166 : \underline{128})$, $(10 : \underline{32}, 30 : \underline{64}, 50 : \underline{128}, 166 : \underline{256})$, with the same reading convention as described in



(a) uniform grid

(b) non uniform grid

Figure 10: Convergence test for the gyroaverage operator in the θ direction for the uniform and non uniform grids

Section 3.1. This means, for example, 4 points in the θ direction have been used for the first 10 radial positions, 8 points in the θ direction have been used for the subsequent 30 radial positions, and so on. For each of these cases, comparing the uniform and non uniform case, we were thus able to reduce the number of points in the θ direction by 22%.

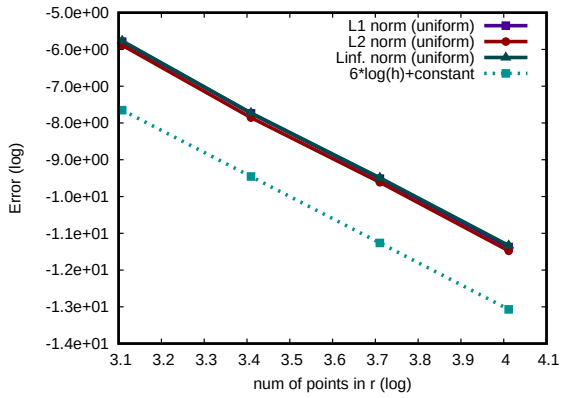
For both the uniform and non uniform polar meshes we obtain satisfactory convergence results, in accordance with the theoretical expectations given by the blue dashed curve in the following figures. The theoretical slope is in fact given by $n \cdot log(h) + c$, where h is the mesh spacing, c an arbitrary constant and the multiplicative factor n in front of the logarithm is given by $p + 1$, with p being the degree of the Lagrange polynomials used for the interpolation.

Focusing on the sole logarithm of the L_2 norm, the same convergence test described for the Figure 10b has been repeated for different values of the degree of the Lagrange interpolation polynomial. All the other parameters, namely N_r, N_θ, ρ and N_{gp}, have been kept as in the setting described for the convergence tests in the non uniform θ-direction. The results of this scan is shown in Figure 11b, where the degree of the interpolation has been changed in the range of $(5, 7, 9)$.

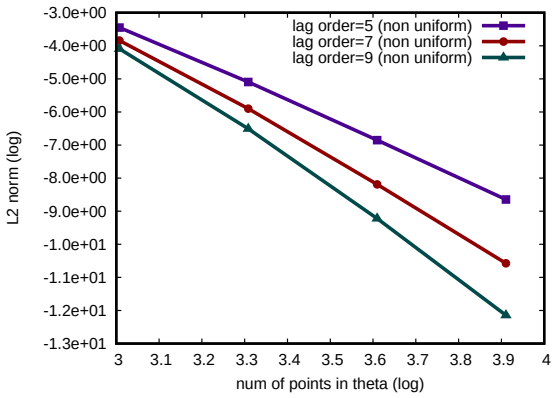
Figure 11a presents the last convergence test which has been performed, namely for the r direction. The logarithm of the L_2 norm, L_1 norm and L_∞ norm are plotted with respect to an increasing number of points in the r direction in a logarithmic scale. Among the parameters kept fixed during the convergence scan in r, the degree of the Lagrange polynomial was set equal to 5, the number of points uniformly distributed in the θ direction was equal to $N_\theta = 504$, while the number of points on the gyroaverage circle was set equal to $N_{gp} = 128$. The gyroradius was still considered to be equal to $\rho = 0.1$. The number of points in the θ direction had to be chosen uniformly and large enough, in order to avoid a contribution of the error due to discretization in the r direction. The expected theoretical slope, *e.g.* $6\,log(h) + c$, due to the degree 5 of the polynomial interpolation used, has been observed in the numerical results.

3.2.2 Uniform and non uniform case comparison

For a conclusive comparison between the uniform and non uniform case, the error in the L^2 norm has been investigated, given the same number of points in the two grids, consequently differently distributed in the space. In particular, the non uniform sequence grid $(10 : \underline{32}, 30 : \underline{64}, 50 :$



(a) in the r direction



(b) on the degree of the Lagrange polynomial

Figure 11: Convergence test of the gyroaverage operator in the r direction and on the degree of the Lagrange polynomial

$\underline{128}, 166 : \underline{256}$) has been compared with a uniform grid of 200 points in each radial position, the sequence $(10 : \underline{16}, 30 : \underline{32}, 50 : \underline{64}, 166 : \underline{128})$ with a uniform grid of 100 points for each radial position, the sequence $(10 : \underline{8}, 30 : \underline{16}, 50 : \underline{32}, 166 : \underline{64})$ with a grid constituted of 50 points for each radial position, and finally the non uniform sequence grid $(10 : \underline{4}, 30 : \underline{8}, 50 : \underline{16}, 166 : \underline{32})$ with a uniform grid with 25 points for each radial direction. The other parameters have been kept fixed during the test, and a sufficient amount of points in the r direction ($N_r = 256$) and for the gyroaverage discretization ($N_{gp} = 128$) have again been used in order to avoid possible spurious contributions related to these parameters. Among the other parameters, the degree of the Lagrange interpolation was equal to 5 and the gyroradius ρ fixed to 0.1.

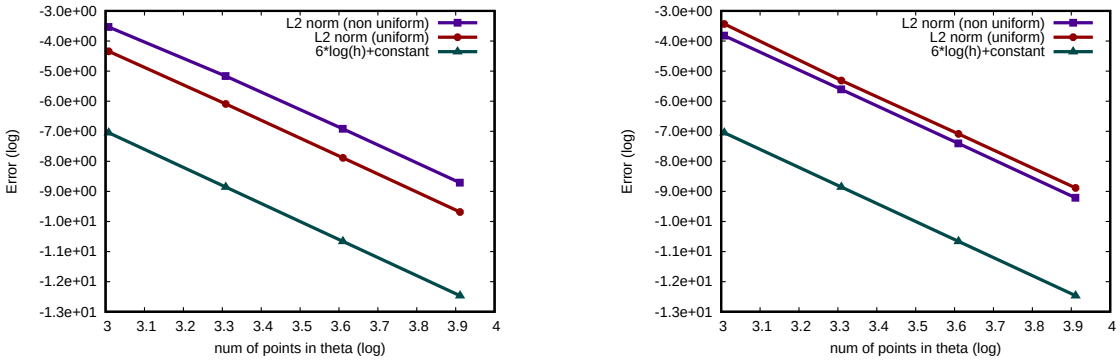
The results are shown in Figure 12. In Figure 12a we can see that the error is smaller in the uniform case, as expected, since we are restricting the domain region to radii smaller than 0.4. But, if on the other hand we investigate the outer region of the domain ($0.4 < r < 0.85$), which usually is the one with more need to be accurately resolved due to the interesting physical structures which develop especially here, we can see in Figure 12b that the error is smaller in the non uniform grid case.

Given the results of the convergence tests shown in Figures 10 and 11, we can assess that the gyroaverage operator works satisfactorily on the new grid, thus providing a workable implementation. Considering also the last results presented in Figure 12, we can conclude that the operator is more accurate on the new non uniform grid in the physically interesting region of the domain, thus proving the usefulness of the method proposed.

3.3 Advection operator

Several test cases have been performed for the advection operator. The most important is the one testing the resilience of structure when advecting through the center as it is where the mesh is loosen the most. The same meshes as in 3.1 are used (*i.e.* 15% less points on non-uniform mesh than on uniform mesh). The function studied is the following:

$$f(r,\theta) = \begin{cases} (\cos(1 - 2\pi \frac{r-r_2}{r_1-r_2}))(1 - \cos(2\pi \frac{\theta-\theta_2}{\theta_1-\theta_2})), & (r,\theta) \in [r_1, r_2] \times [\theta_1, \theta_2]; \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$



(a) on the domain $r < 0.4$

(b) on the domain $0.4 < r < 0.85$

Figure 12: Error comparison between the uniform and non uniform grid in different regions

where $r_{min} \leq r_1 < r_2 \leq r_{max}$ and $0 \leq \theta_1 < \theta_2 \leq 2\pi$.

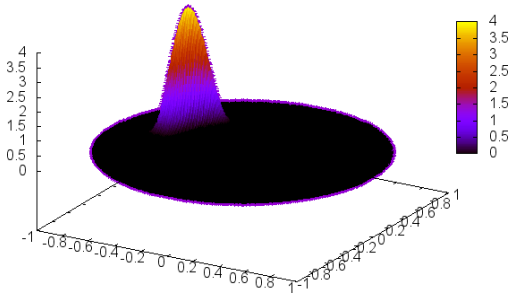
This ensures $\mathcal{C}^1$ continuity for Lagrangian interpolation. The function is then centered on $(r = 0.7, \theta = \frac{6\pi}{7})$ ($r_1 = 0.6$, $r_2 = 0.8$, $\theta_1 = \frac{5\pi}{7}$, $\theta_2 = \pi$), with $r_{max} = 1.0$ and then advected through the center with speed $(v_x = \frac{2}{3}, v_y = -\frac{1}{3})$ until $r = 0.7$ is reached on the other side. The opposite advection is finally performed to bring back the structure to its original position. The overall displacement is performed in 40 time steps. Initial, middle and final snapshots of the advected function are presented on Figure 13. Middle and final snapshots of the error are presented on Figure 14a and 14c for the uniform mesh and on Figure 14b and 14d for the non-uniform mesh. These show the difference between the exact value and the interpolated one for the whole plane at time steps 1, 20 and 40. During the advection, the error done when interpolating quickly grows when approaching and going through the center (from almost flat error at time step 1 to 14a). But it does not evolve on the outer *radii* of the mesh, nor when going back through the center. It is more pronounced for the non-uniform mesh.

This is best seen in Figure 15 which gives the L_1 and L_2 norms of the error for the whole plane at each time step for both meshes. The structure first undergoes accuracy loss when entering the center near time step 12 for the uniform mesh and near time step 8 for the non-uniform mesh. This means that one of the $N_{\theta_{[i]}}$ in that region was not high enough to reach the corresponding accuracy. *Radii* closer to the center did not have a $N_{\theta_{[i]}}$ high enough either as shows the higher overall error for the non-uniform mesh. Once the structure has gone once through the coarsely solved part of the mesh, the accuracy does not undergo any other drastic reduction anymore. It means we have reached the minimum resolution offered by the mesh. This result is satisfactory.

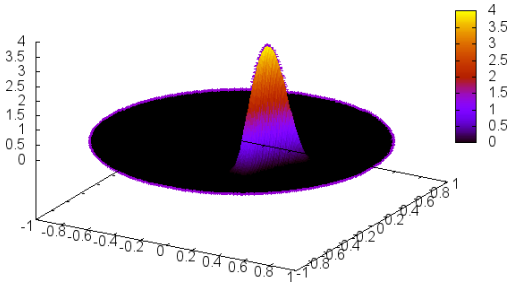
Once again, the operator works as expected. The accuracy highly depends on the choice of the $N_{\theta_{[i]}}$, especially near the center where we wanted the number of points to be scarce. It could be useful to have a tuning tool, which, given the desired accuracy, the number of *radii* and the typical variation of the function (size of the smallest structure to solve), would give the optimal $N_{\theta_{[i]}}$, but we do not have it yet.

Results with large aspect ratio mapping Results on the large aspect ratio mapping, for the advection are reported on Figures 16, 17, 18 and 19a, 19b.

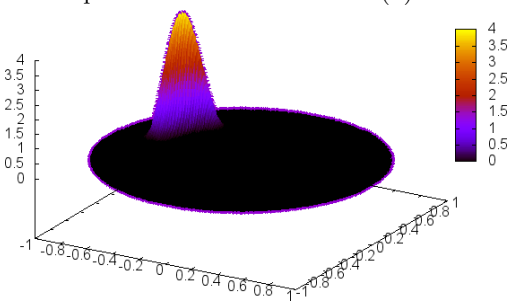
The results are very similar to the previous polar case. We had to change a little the test case, so that the solution does not go outside the domain. The initial condition is now in polar



(a) Function at time step 1.

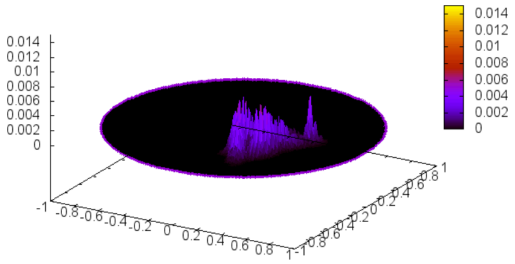


(b) Function at time step 20.

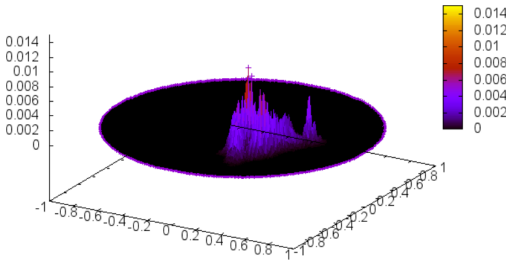


(c) Function at time step 40.

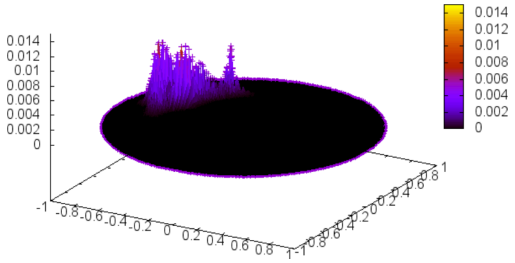
Figure 13: Evolution of the reference function during the advection.



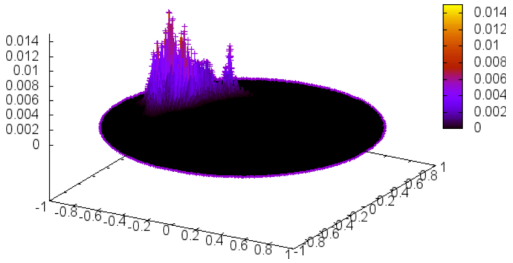
(a) Error at time step 20 on uniform mesh.



(b) Error at time step 20 on non-uniform mesh.



(c) Error at time step 40 on uniform mesh.



(d) Error at time step 40 on non-uniform mesh.

Figure 14: Evolution of the error for an advection on uniform and non-uniform mesh.

coordinates:

$$f(r,\theta) = \left(1 - \cos(2\pi \frac{r - 0.4}{0.2})\right)\left(1 - \cos(2\pi \frac{\theta - 5\pi/7}{2\pi/7})\right),\ 0.4 \leq r \leq 0.6,\ \frac{5\pi}{7} \leq \theta \leq \frac{7\pi}{7}$$

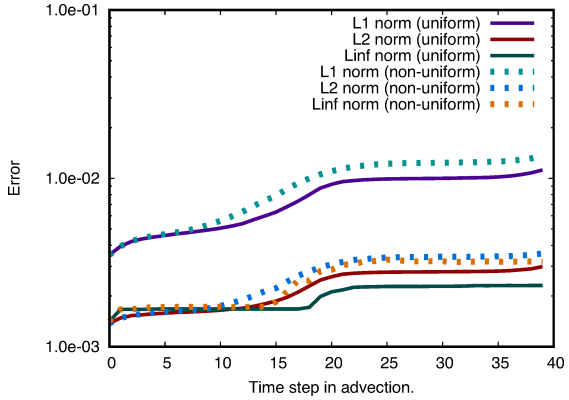
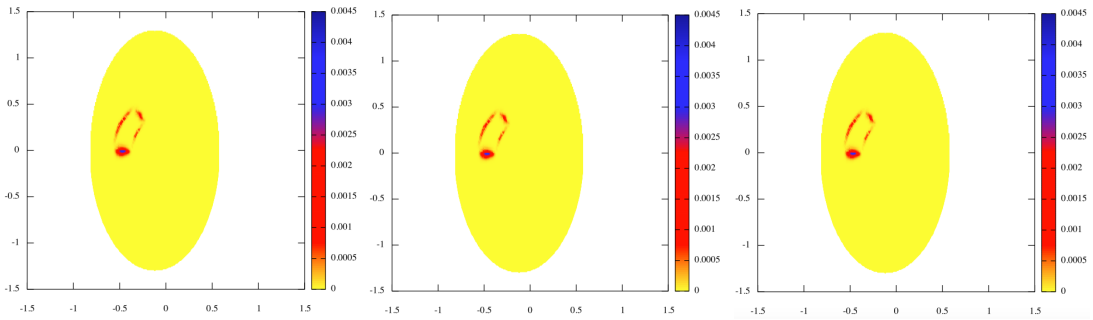


Figure 15: Evolution of mean error when advecting a structure forth and back through the center of the plane.

and $f(r,\theta) = 0$ elsewhere. We do 40 time steps and use here $v_x = \frac{5}{7}\frac{4}{40}$, $v_y = -\frac{5}{7}\frac{2}{40}$ and time step $dt = \frac{1}{2}$ for the 20 first time steps and then $v_x = -\frac{5}{7}\frac{4}{40}$, $v_y = \frac{5}{7}\frac{2}{40}$ and time step $dt = \frac{1}{2}$ for the 20 last time steps.
The L^1 and L^2 norms are computed as in the polar case (we do not include the Jacobian corresponding to the new mapping here).
We have used here $N_r = 256$, $N_\theta = 256$ for the uniform grid and the sequence $(2:\underline{32}, 8:\underline{64}, 64:\underline{128}, 182:\underline{256})$ for the first non uniform grid and the sequence $(10:\underline{32}, 30:\underline{64}, 50:\underline{128}, 166:\underline{256})$ for the second non uniform mesh.



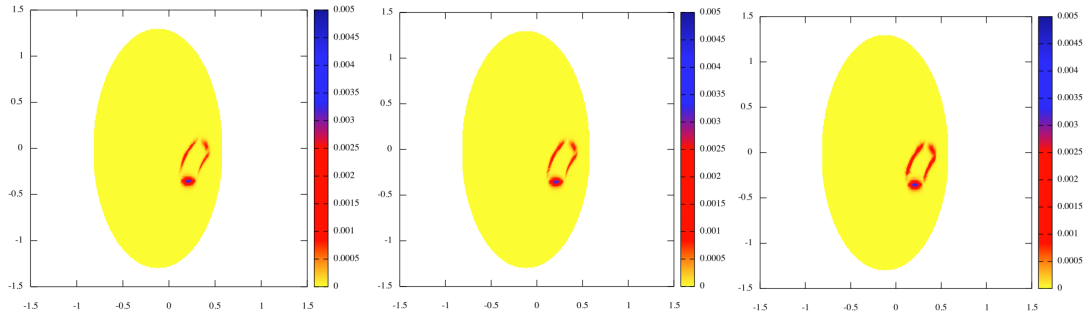
(a) Uniform mesh (b) First non uniform mesh (c) Second non uniform mesh

Figure 16: Error at time step 1, for an advection on different meshes (large aspect ratio mapping)

3.4 Numerical results for Poisson solver

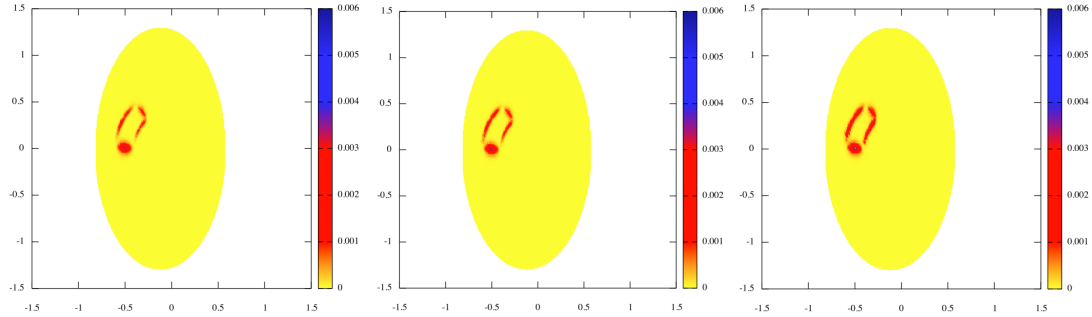
In the following, the previous Poisson Equation (10) is solved for the right hand side $R(r,\theta) = 2 * \exp(r(\cos(\theta) + \sin(\theta)))$ on the polar domain $\Omega = \{(r,\theta) : 0 < r < 1 \text{ and } 0 \leq \theta \leq 2\pi\}$. The linear sparse matrix system (20)-(23) is solved by using the Intel PARDISO solver[4] which is available through the INTEL MKL library. The number of threads (`MKL_NUM_THREADS`) has been chosen equal to 8. All the following simulations have been performed on a SandyBridge machine

[4] `http://pardiso-project.org/`



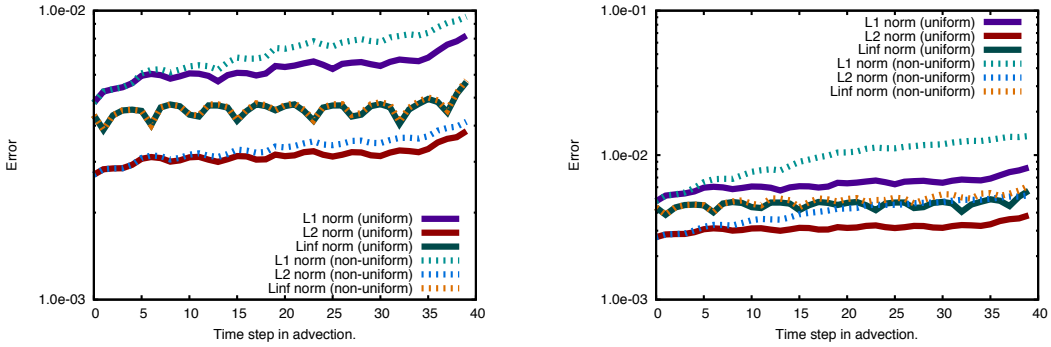
(a) Uniform mesh (b) First non uniform mesh (c) Second non uniform mesh

Figure 17: Error at time step 20, for an advection on different meshes (large aspect ratio mapping)



(a) Uniform mesh (b) First non uniform mesh (c) Second non uniform mesh

Figure 18: Error at time step 40, for an advection on different meshes (large aspect ratio mapping)



(a) Evolution of mean error when advecting a structure forth and back through the center of the plane (large aspect ratio mapping case, with first non-uniform mesh)

(b) Evolution of mean error when advecting a structure forth and back through the center of the plane (large aspect ratio mapping, with second non-uniform mesh)

Figure 19: Evolution of mean error for the advection

(Intel E5-2670 v1, 2.60GHz, 8-cores per socket, 2 sockets per node). Numerical results have been compared with analytic results $f(r,\theta) = \exp(r(\cos(\theta) + \sin(\theta))) = 0.5 * R(r,\theta)$.

3.4.1 Reference case with uniform mesh

First tests have been performed on a uniform mesh where the number of poloidal points N_θ has been fixed equal to $N_\theta = 2N$ (N being the radial cell number). The analytic solution is plotted for the case $N = 512$ on Figure 20 and the corresponding relative error for the numerical solution is shown in Figure 21. The maximum relative error depending on the mesh discretization is summarized in Table 1 for N varying from 8 to 2048. Table 1 also shows that the numerical scheme is as expected of second order. These results act as reference results for the next tests on non-uniform meshes. Let us notice that the results on a uniform mesh are the same than the one obtained by Lai [14]. As said in section 2.4, the matrix system is constructed in the opposite way compared to the one proposed by Lai in order to be able to extend it to meshes with non-uniform number of poloidal points. Both matrix systems have been also compared in terms of CPU time and are equivalent (results for our matrix system are recorded in Table 1).

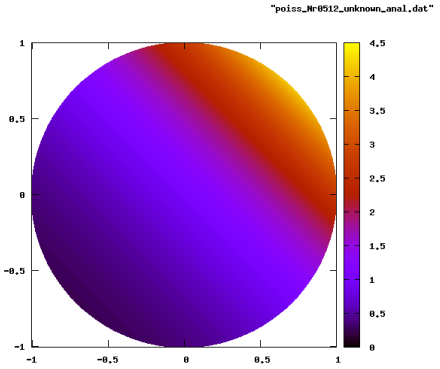


Figure 20: Analytic solution $f(r,\theta) = \exp(r(\cos(\theta)+\sin(\theta)))$ for the case $N = 512$ and $N_\theta = 1024$.

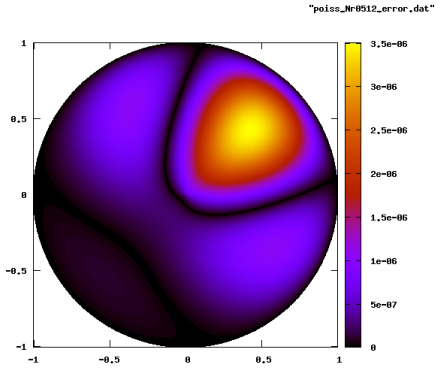


Figure 21: Relative error between numerical results and analytic solution for the case $N = 512$ and $N_\theta = 1024$.

3.4.2 Non-uniform mesh

In this section, tests are now performed on non-uniform meshes defined with N_r points in the radial direction and divided into N_d sub-domains Ω_d depending on the number of points in the poloidal direction $N_{\theta[\Omega_d]}$, the maximum number being equal to $N_{\theta\max} = 2(N_r - 1) = 2N$. Let ν_d denotes the reduction of poloidal points for domain Ω_d compared to the maximum number, namely $N_{\theta[\Omega_d]} = N_{\theta\max}/\nu_d$ and $\nu = (\nu_1, \cdots, \nu_{N_d})$. Each sub-domain Ω_d of $\Omega_{\text{non-uniform}}$ is an annulus defined by

$$\Omega_d = \left\{ (r,\theta) : \left(\frac{(d-1)N}{3} + \frac{1}{2}\right)\Delta r < r < \left(\frac{d\,N}{3} + \frac{1}{2}\right)\Delta r \text{ and } 0 \leq \theta \leq 2\pi \right\}$$

For all the numerical results presented in this section, the circular domain $\Omega_{\text{non-uniform}}$ is divided into three sub-domains with $\nu = [4, 2, 1]$ (see Figure 6 for example with $N_r = 8$). Two strategies have been studied to solve Poisson equation on a non-uniform mesh $\Omega_{\text{non-uniform}}$. The first one where Poisson is solved as in the previous section 3.4.1 on a uniform mesh Ω_{uniform} of size $N \times 2N$ where for all missing points $P_{i,j}$ ($i.e$ $P_{i,j} \in \Omega_{\text{uniform}}$ but $P_{i,j} \notin \Omega_{\text{non-uniform}}$), the RHS $R_{i,j}$ in

N	Maximum Error	Order	CPU time precompute	CPU time solve
4	$5.6814\ 10^{-2}$		$0.006607\,s$	$0.001120\,s$
8	$1.3950\ 10^{-2}$	2.026	$0.001959\,s$	$0.000124\,s$
16	$3.5244\ 10^{-3}$	1.985	$0.005970\,s$	$0.000990\,s$
32	$8.8807\ 10^{-4}$	1.988	$0.010281\,s$	$0.000315\,s$
64	$2.2261\ 10^{-4}$	1.996	$0.037393\,s$	$0.000451\,s$
128	$5.5755\ 10^{-5}$	1.997	$0.167770\,s$	$0.001448\,s$
256	$1.3950\ 10^{-5}$	1.999	$0.749632\,s$	$0.006083\,s$
512	$3.4890\ 10^{-6}$	1.999	$3.413063\,s$	$0.024860\,s$
1024	$8.7244\ 10^{-7}$	1.999	$16.345875\,s$	$0.101600\,s$
2048	$2.1839\ 10^{-7}$	1.998	$82.168015\,s$	$0.469672\,s$

Table 1: For uniform circular mesh, maximum relative error between numerical results and analytic solution for 10 different values of radial cell number N. Poloidal mesh number are defined as $N_\theta = 2N$. CPU time is reported for the two steps of the direct Poisson solver: (i) precomputation (symbolic analysis + factorization) and (ii) solving.

equation (13) are approximated by Lagrange interpolation of third order. The second strategy is the one for which matrix system (20)-(23) has been especially designed, *i.e.* Poisson is directly solved on the non-uniform $\Omega_{\mathrm{non-uniform}}$ domain. The two strategies will be respectively called in the following: (i) case1: non-uniform mesh with uniform Poisson and (ii) case 2: non-uniform mesh with non-uniform Poisson. The number of equations is 1.6 times bigger for case 1 than for case 2. Results for case 1 are summarized in Table 2 while results for case 2 are detailed in Table 3, both show that the numerical scheme is still of second order. As expected, case 1 is simpler to implement but twice more expensive. As seen in Table 2 and Table 1, CPU time for case 1 is of the same order than for the uniform mesh case because here only the CPU time related to the precomputation (symbolic analysis+factorization) and to the solving are recorded. So calculation of the right hand side with approximation (Lagrange interpolation for missing points) should also be taken into account in case 1. The maximum relative errors obtained for case 1 and case 2 are of the order of the one obtained for a uniform mesh for the smallest number of N_θ points.

Therefore, as a summary, we have shown in this section that Poisson equation (10) can be solved successfully on a non-uniform circular mesh with the coupling of 2D finite differences in polar coordinates and Lagrange interpolation of third order. The associated numerical scheme proposed in section 2.4 is of second order. Higher order for Lagrange polynomials has not been tested in this paper but the matrix system (20)-(23) could be easily generalized. In both cases, CPU time for the solve step remains quite reasonable even for the biggest meshes. This is a key point for a future implementation in GYSELA code where the precomputation step will be only performed one times at the beginning.

3.4.3 Mapping

For validation of the code, we consider the following solution given in cartesian coordinates. We take here $a = 1$ and $b = 0.5$

$$u(x, y) = \frac{\exp(x) + \exp(y)}{1 + xy}.$$

We get the results in Tables 4a and 4b using $M = 2N$. It means uniform mesh along

	sub-domain Ω_1 ($N_\theta = 2N/4$)	sub-domain Ω_2 ($N_\theta = 2N/2$)	sub-domain Ω_3 ($N_\theta = 2N$)	global domain $\Omega_{\text{non-uniform}}$		
N	Maximum relative error			Order	CPU time precompute	CPU time solve
32	$0.5453\ 10^{-3}$	$0.8412\ 10^{-3}$	$0.8426\ 10^{-3}$		$0.024073\,s$	$0.000555\,s$
128	$0.3653\ 10^{-4}$	$0.5249\ 10^{-4}$	$0.5219\ 10^{-4}$	2.002	$0.168538\,s$	$0.001374\,s$
512	$0.2336\ 10^{-5}$	$0.3287\ 10^{-5}$	$0.3260\ 10^{-5}$	1.998	$3.415027\,s$	$0.025067\,s$
2048	$0.1473\ 10^{-6}$	$0.2058\ 10^{-6}$	$0.2039\ 10^{-6}$	1.999	$81.058922\,s$	$0.452550\,s$

Table 2: For case 1 (non-uniform mesh with uniform Poisson solver): maximum relative error between numerical results and analytic solution for 4 different values of radial cell number N for the three sub-domains. The order of the numerical scheme is calculated on the global domain $\Omega_{\text{non-uniform}}$ based on the maximum error on the three sub-domains. CPU time is reported for the two steps of the direct Poisson solver: (i) precomputation (symbolic analysis + factorization) and (ii) solving.

	sub-domain Ω_1 ($N_\theta = 2N/4$)	sub-domain Ω_2 ($N_\theta = 2N/2$)	sub-domain Ω_3 ($N_\theta = 2N$)	global domain $\Omega_{\text{non-uniform}}$		
N	Maximum relative error			Order	CPU time precompute	CPU time solve
32	$0.2894\ 10^{-2}$	$0.2512\ 10^{-2}$	$0.1706\ 10^{-2}$		$0.054278\,s$	$0.006446\,s$
128	$0.2132\ 10^{-3}$	$0.2021\ 10^{-3}$	$0.1344\ 10^{-3}$	1.882	$0.090576\,s$	$0.000930\,s$
512	$0.1406\ 10^{-4}$	$0.1358\ 10^{-4}$	$0.8918\ 10^{-5}$	1.961	$1.835005\,s$	$0.013966\,s$
2048	$0.8912\ 10^{-6}$	$0.8646\ 10^{-6}$	$0.5659\ 10^{-6}$	1.989	$41.658237\,s$	$0.244283\,s$

Table 3: For case 2 (non-uniform mesh with non-uniform Poisson solver): maximum relative error between numerical results and analytic solution for 4 different values of radial cell number N for the three sub-domains. The order of the numerical scheme is calculated on the global domain $\Omega_{\text{non-uniform}}$ based on the maximum error on the three sub-domains. CPU time is reported for the two steps of the direct Poisson solver: (i) precomputation (symbolic analysis + factorization) and (ii) solving.

θ direction in this paragraph. On Figures 22, 23 and Table 5, we give numerical results for $f = 2\exp(x+y)$ and large aspect ratio mapping. We get an order two for the error as expected. We notice that the error behaves similarly for the two schemes and for the two geometries (ellipse and large aspect ratio). It seems that the results are better for the 7 points scheme in the case of the ellipse and for the 9 points scheme in the case of the large aspect ratio mapping, so that there is no clear trend on what scheme would be the best in general.

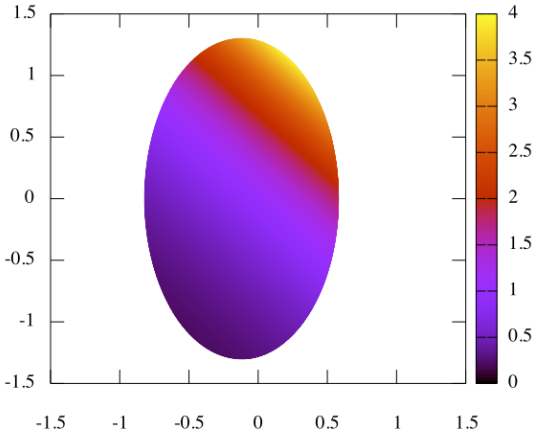


Figure 22: Analytic solution $f = 2\exp(x + y)$ for the case $N = 512$ and $N_\theta = 1024$.

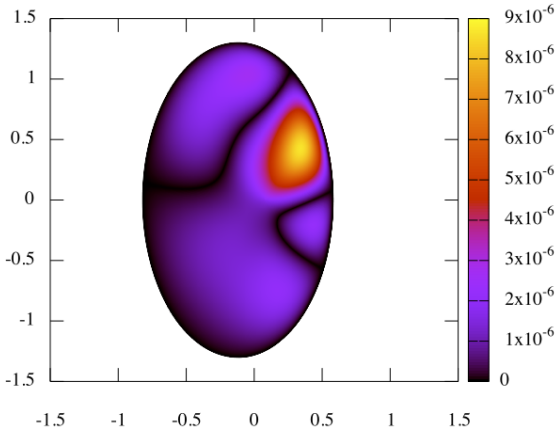


Figure 23: Error between numerical results and analytic solution for the case $N = 512$ and $N_\theta = 1024$ and 9 points scheme.

N	Relative error	Order
4	2.513D-02 (3.518D-02)	
8	4.532D-03 (7.970D-03)	2.47 (2.14)
16	1.352D-03 (2.041D-03)	1.74 (1.97)
32	3.346D-04 (5.185D-04)	2.01 (1.98)
64	8.322D-05 (1.307D-04)	2.01 (1.99)
128	2.079D-05 (3.271D-05)	2.00 (2.00)

(a) Ellipse

N	Relative error	Order
4	1.038D-01 (9.249D-02)	
8	2.941D-02 (2.385D-02)	1.82 (1.96)
16	7.777D-03 (6.089D-03)	1.92 (1.97)
32	2.201D-03 (1.626D-03)	1.82 (1.90)
64	5.592D-04 (4.118D-04)	1.98 (1.98)
128	1.407D-04 (1.037D-04)	1.99 (1.99)

(b) Large aspect ratio mapping

Table 4: Convergence for 7 points (9 points) scheme

Conclusion

In the context of gyrokinetic simulation of turbulence inside a Tokamak plasma, we have developed a strategy that incorporates an adapted non-uniform meshing. Instead of having a circular geometry with a uniform grid along (r, θ) dimensions to describe the poloidal cross-section, we propose a non-uniform spacing along theta direction aiming at computational and memory savings. Additionally, a peculiar mapping coupled with the non-uniform mesh permits to match more complex realistic geometry such as D-shaped plasma, thus exceeding the former limited configuration of circular Tokamak cross-sections. We expect this mapping to rely on analytical formulas in order to keep a relatively low price in term of computations, which is quite crucial for a full-f global code as GYSELA is.

Several features that are typically used in the gyrokinetic code GYSELA have been recast to handle such a new approach. Interpolation, Advection, Gyroaverage and Poisson operators are revisited, upgraded and analyzed in this paper. These operators have been studied separately. All in all, the convergence studies we provide show that this is a workable approach that reach accuracy comparable to uniform meshing. Adaptivity brings really a potential benefit in term of memory savings, but it also permits to refine grid in a specific annulus (small region along radial

N	Error	Order
4	0.1094	
8	2.956D-02	1.89
16	8.718D-03	1.76
32	2.203D-03	1.98
64	5.526D-04	2.00
128	1.383D-04	2.00
256	3.460D-05	2.00
512	8.649D-06	2.00
1024	2.162D-06	2.00

Table 5: Convergence for large aspect ratio mapping and 9 points scheme for $f = 2\exp(x+y)$ (fortran implementation with pardiso)

direction) that will be useful for physics studies incorporating kinetic electrons.

Future works will target the addition of this method in GYSELA. It will require overhauling many data structures and to combine the different operators we have described. As this paper was not focusing on algorithms performance, a subsequent aim will be to write parallel versions of these algorithms and to optimize computation costs in order to compete with the execution time of the former uniform approach. A significant gain in lowering the overall memory footprint is also expected.

References

[1] J. Abiteboul, G. Latu, V. Grandgirard, A. Ratnani, E. Sonnendrücker, and A. Strugarek. Solving the Vlasov equation in complex geometries. *ESAIM: Proceedings*, 32:103–117, 2011.

[2] P. Angelino, X. Garbet, L. Villard, A. Bottino, S. Jolliet, P. Ghendrih, V. Grandgirard, B. F. McMillan, Y. Sarazin, G. Dif-Pradalier, and T. M. Tran. The role of plasma elongation on the linear damping of zonal flows. *Physics of Plasmas*, 15(6), 2008.

[3] N. Crouseilles, G. Latu, and E. Sonnendrücker. A parallel Vlasov solver based on local cubic spline interpolation on patches. *Journal of Computational Physics*, 228:1429–1446, 2009.

[4] D. Eisen. On the numerical solution of $u_t = u_{rr} + \frac{2}{r}u_r$. *Numerische Mathematik*, 10(5):397–409, 1967.

[5] R. Fitzpatrick, C. Gimblett, and R. Hastie. On the 1 1/2 -D evolution of tokamak plasmas in the case of large aspect ratio. *Plasma Physics and Controlled Fusion*, 34(2):161, 1992.

[6] V. Grandgirard, J. Abiteboul, J. Bigot, T. Cartier-Michaud, N. Crouseilles, G. Dif-Pradalier, C. Ehrlacher, D. Esteve, X. Garbet, P. Ghendrih, G. Latu, M. Mehrenberger, C. Norscini, C. Passeron, F. Rozar, Y. Sarazin, E. Sonnendrücker, A. Strugarek, and D. Zarzoso. A 5D gyrokinetic full-f global semi-Lagrangian code for flux-driven ion turbulence simulations. *Computer Physics Communications*, 207:35 – 68, 2016.

[7] 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*, 217(2):395–423, 2006.

[8] V. Grandgirard, M. Brunetti, P. Bertrand, N. Besse, X. Garbet, P. Ghendrih, G. Manfredi, Y. Sarazin, O. Sauter, E. Sonnendrücker, J. Vaclavik, and L. Villard. A drift-kinetic Semi-Lagrangian 4D code for ion turbulence simulation. *Journal of Computational Physics*, 217(2):395 – 423, 2006.

[9] W. Hackbusch. *Elliptic Differential Equations: Theory and Numerical Treatment*, volume 18 of *Springer Series in Computational Mathematics*. Springer, 2017.

[10] A. Hamiaz, M. Mehrenberger, A. Back, and P. Navaro. Guiding center simulations on curvilinear grids*. *ESAIM: Proc.*, 53:99–119, 2016.

[11] A. Hamiaz, M. Mehrenberger, H. Sellama, and E. Sonnendrücker. The semi-lagrangian method on curvilinear grids. *Communications in Applied and Industrial Mathematics.*, 7(3):99–137, 2016.

[12] B. Holman and L. Kunyansky. A second-order finite difference scheme for the wave equation on a reduced polar grid, 2015.

[13] B. S. Jovanović and E. Süli. *Analysis of finite difference schemes: for linear partial differential equations with generalized solutions*, volume 46. Springer Science & Business Media, 2013.

[14] M.-C. Lai. A note on finite difference discretizations for poisson equation on a disk. *Numerical Methods for Partial Differential Equations*, 17(3):199–203, 2001.

[15] M.-C. Lai. A simple compact fourth-order poisson solver on polar geometry. *Journal of Computational Physics*, 182(1):337–345, 2002.

[16] G. Latu, N. Crouseilles, V. Grandgirard, and E. Sonnendrücker. Gyrokinetic semi-Lagrangian parallel simulation using a hybrid OpenMP/MPI programming. In *Recent Advances in PVM and MPI*, volume 4757 of *Lecture Notes in Computer Science*, pages 356–364. Springer, 2007.

[17] G. Latu, V. Grandgirard, N. Crouseilles, and G. Dif-Pradalier. Scalable quasineutral solver for gyrokinetic simulation. In *PPAM (2)*, LNCS 7204, pages 221–231. Springer, 2011.

[18] A. Mock. Subgridding scheme for fdtd in cylindrical coordinates. In *Progress In Electromagnetics Research Symposium Proceeding*, 2011.

[19] K. Mohseni and T. Colonius. Numerical treatment of polar coordinate singularities. *Journal of Computational Physics*, 157(2):787 – 795, 2000.

[20] E. Sonnendrücker, J. Roche, P. Bertrand, and A. Ghizzo. The semi-Lagrangian method for the numerical resolution of the Vlasov equation. *Journal of Computational Physics*, 149(2):201 – 220, 1999.

[21] C. Steiner, M. Mehrenberger, N. Crouseilles, V. Grandgirard, G. Latu, and F. Rozar. Gyroaverage operator for a polar mesh. *The European Physical Journal D*, 69(1):18, 2015.

[22] J. C. Strikwerda. *Finite difference schemes and partial differential equations*. SIAM, 2004.

[23] L. Wright. *A fourth-order finite difference scheme for Poisson's equation in polar coordinates on the unit disc*. PhD thesis, Colorado School of Mines. Arthur Lakes Library, 2013.