Parallel lattice-boltzmann transport solver in complex geometry

Matthieu Boileau, Bérenger Bramas, Emmanuel Franck, Philippe Helluy, Laurent Navoret

PARALLEL LATTICE-BOLTZMANN TRANSPORT SOLVER IN COMPLEX GEOMETRY

MATTHIEU BOILEAU, BÉRENGER BRAMAS, EMMANUEL FRANCK, PHILIPPE HELLUY, LAURENT NAVORET

Abstract. We present an efficient solver for the conservative transport equation with variable coefficients in complex geometries. The solver is based on a kinetic formulation resembling the Lattice-Boltzmann approach. The chosen formalism allows to obtaining an explicit and conservative scheme that requires no matrix inversion and that is unconditionally stable with respect to the time step size. We present the method and its optimized parallel implementation on complex toroidal geometries. This solver will be used as a building block for tokamak plasma physics simulation.

1. Introduction

In many fields of physics one is conducted to solve conservative transport equations of the form

$$\partial_t \sigma + \nabla \cdot (\sigma \boldsymbol{v}) = 0. \tag{1.1}$$

In this equation, the unknown is the function $\sigma(\boldsymbol{x}, t)$ that depends on a space variable $\boldsymbol{x} \in \mathbb{R}^3$ and of the time t. The vector field $\boldsymbol{v}(\boldsymbol{x}, t)$ is given.

Such equation arises for instance in plasma physics, where σ is the charged particles density function.

A tokamak is an experimental device for creating hot plasma. It generally has a toroidal shape and present a cylindrical symmetry around the vertical axis. See Figure 1.1. The geometry of the poloidal plane however can be complicated. It is therefore quite natural to consider numerical methods that are well adapted to unstructured meshes in the poloidal planes, while the mesh is structured in the toroidal direction. See Figure 1.1. This what we do in this paper.

Another aspect of plasma physics in tokamak is that some charged particles can be fast. If the transport equation is solved with a classical explicit scheme, this imposes very small time steps, because of the Courant Friedrichs Lewy or CFL condition. This is annoying because the fast particles generally have a light contribution to the main physical phenomena. In this paper we propose a method that is free of any stability condition on the time step size.

The method reuses ideas coming from the Lattice-Bolzmann Method (LBM) [4] or from the kinetic schemes [3, 5]. The main point is to replace the transport equation (1.1), where the velocity $\boldsymbol{v}$ is not constant, by a few transport equations at constant velocity, coupled by a stiff local source term. The coupled system is solved with splitting algorithm that separate the free transport steps and the stiff source terms. The free transport steps are easier to solve, because of the transport is at constant velocity. They can also be solved by a CFL-less Discontinuous Galerkin (DG) method in the toroidal plane and by an exact characteristic method in the toroidal direction. It is also possible, by an adequate splitting algorithm, to achieve high order accuracy of the whole procedure. In the end we obtain a CFL-less high-order and conservative scheme with interesting properties. The price to pay is that the initial number of unknowns is increased by a factor of four (but this increase is also observed in the implementation of high-order time integration schemes).

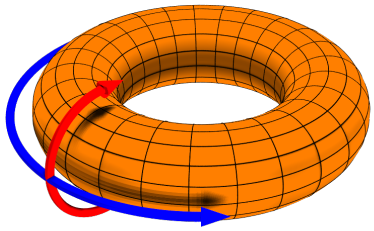

Figure 1.1. Tokamak shape. The toroidal direction is indicated by the blue arrow. The poloidal direction is represented by the red arrow. Source: Wikipedia.

The resulting scheme has also nice parallelization possibilities. In a poloidal plane, the DG method involves block-triangular linear systems that are well solved by an optimized task-based OpenMP implementation. In the toroidal direction, the transport equations are solved by a simple shift operator. It is here implemented by simple MPI send/receive operations.

In this work we describe the whole mathematical, numerical and programming construction. We then verify its accuracy and efficiency with some test-cases.

2. Mathematical formalism

2.1. **Change of coordinates.** In this work, we are particularly interested in solving the transport equation in tokamak geometry. It is useful for this purpose to be able to write the conservative transport equation in an arbitrary system of coordinates. Let us thus consider a change of variable

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{r}).$$

The Jacobian of this change of variables is defined by

$$j(\boldsymbol{r}) = \det \boldsymbol{x}'(\boldsymbol{r}).$$

We consider then the change of unknown

$$\rho(\boldsymbol{r}, t) = j(\boldsymbol{r})\sigma(\boldsymbol{x}(\boldsymbol{r}), t).$$

The velocity field in the $\boldsymbol{r}$ variable is given by

$$\boldsymbol{u}(\boldsymbol{r}, t) = j(\boldsymbol{r})\boldsymbol{x}'(\boldsymbol{r})^{-1}\boldsymbol{v}(\boldsymbol{x}(\boldsymbol{r}), t).$$

In the new variables, the transport equation becomes

$$\partial_t \rho + \nabla \cdot \rho \boldsymbol{u} = 0.$$

For tokamak applications, we are particularly interested in the change form Cartesian to cylindrical coordinates. We denote by

$$\boldsymbol{x} = (x, y, z)^T$$

the Cartesian coordinates and by

$$\boldsymbol{r} = (r, z, \varphi)^T$$

the cylindrical coordinates. The change of variables is given by

$$\begin{aligned} x &= r\cos\varphi, \\ y &= r\sin\varphi, \\ z &= z. \end{aligned}$$

We also define the cylindrical frame

$$\boldsymbol{e}_r = \begin{pmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{pmatrix}, \quad \boldsymbol{e}_z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{e}_\varphi = \frac{1}{r}\begin{pmatrix} -\sin\varphi \\ \cos\varphi \\ 0 \end{pmatrix}.$$

Remark 1. The vector $\boldsymbol{e}_\varphi$ is not a unit vector: its Euclidean norm is equal to $1/r$. This is important for keeping a conservative form to the transport equation in cylindrical coordinates (see below).

We now consider the conservative transport equation (1.1). We assume that the velocity field is expressed in cylindrical coordinates

$$\boldsymbol{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = u_r\boldsymbol{e}_r + u_z\boldsymbol{e}_z + u_\varphi\boldsymbol{e}_\varphi,$$

and we set

$$\boldsymbol{u} = (u_r, u_z, u_\varphi)^T.$$

Then, defining

$$\rho = \sigma r,$$

the conservative transport equation (1.1) keeps a conservative form in cylindrical coordinates

$$\partial_t \rho + \boldsymbol{\nabla}_{(r,z,\varphi)} \cdot (\rho \boldsymbol{u}) = 0. \tag{2.1}$$

This can be rewritten

$$\partial_t \rho + \boldsymbol{\nabla}_{(r,z)} \cdot \left(\rho \begin{pmatrix} u_r \\ u_z \end{pmatrix}\right) + \partial_\varphi(\rho u_\varphi) = 0. \tag{2.2}$$

Remark 2. The conclusion of these considerations is that, even in cylindrical coordinates, it is possible to solve a transport equation of the form (1.1) as if we were in Cartesian coordinates. In the test cases of Section 5 we thus note the three coordinates (x, y, z) even if we are considering cylindrical cases.

2.2. **Lattice-Boltzmann Method (LBM).** We consider a LBM with n_v kinetic velocities $\boldsymbol{\lambda}_k$, $k = 0 \dots n_v - 1$. The kinetic velocities are constant in cylindrical coordinates. The $n_v - 2$ first velocities are used for (r, z) transport. For instance, we can take $n_v = 6$

$$\boldsymbol{\lambda}_k = \lambda \boldsymbol{e}_k,$$

and

$$\boldsymbol{e}_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{e}_1 = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{e}_3 = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}.$$

The last two velocities are used for the φ direction

$$\boldsymbol{e}_{n_v-2} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{e}_{n_v-1} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}.$$

The kinetic approximation is given by

$$\rho = \sum_{k=0}^{n_v-1} f_k,$$

$$\partial_t f_k + \boldsymbol{\lambda}_k \cdot \boldsymbol{\nabla} f_k = \frac{1}{\tau}(f_k^{eq} - f_k), \tag{2.3}$$

with

$$f_k^{eq} = \frac{\rho}{n_v} + \frac{\rho \boldsymbol{u} \cdot \boldsymbol{e}_k}{2\lambda}.$$

This can also be written

$$f_k^{eq} = \frac{\rho}{n_v} + \frac{\rho \boldsymbol{u} \cdot \boldsymbol{\lambda}_k}{2\lambda^2}. \tag{2.4}$$

If we use a splitting method, we have to solve free transport equations

$$\partial_t f_k + \boldsymbol{\lambda}_k \cdot \boldsymbol{\nabla} f_k = 0, \tag{2.5}$$

interlaced with relaxation steps for applying the source term in (2.3). For more details, we refer to [1, 2, 3, 5, 7].

3. Transport solver

For several reasons (parallelism, memory optimizations, tokamak geometries, *etc.*) we consider meshes with a cylindrical symmetry. The starting point is thus a two-dimensional unstructured, but conformal mesh made of second order curved quadrilaterals with eight nodes ("Q8" family in the finite element terminology). In practice, this mesh represents one poloidal plane ($\varphi = \text{Cst}$) of the tokamak. This mesh is extruded in the third direction for obtaining the full tokamak.

In the poloidal plane, the transport equations will be solved with an implicit DG scheme, which we describe below. In the toroidal direction we can solve the transport equations with a simple shift, because of the cylindrical geometry. In practice, a possible implementation is to associate to each poloidal plane one MPI process. Thus, the shift simply consists in an MPI send/receive operation with the neighbor planes. This approach imposes a constraint on the time step. An alternative would be to replace the shift by a semi-Lagrangian solver. This would imply exchanges with more neighbors and thus more MPI communications.

3.1. **DG formulation.** In this section we describe briefly the transport solver in the poloidal plane. For more details we refer (for instance) to [8, 2]. The objective is to solve a two-dimensional transport equation, with constant velocity $\boldsymbol{v}$

$$\partial_t f + \boldsymbol{v} \cdot \boldsymbol{\nabla}_{(x,y)} f = 0.$$

The poloidal solver is based on the Discontinuous Galerkin (DG) method. In order to avoid too constraining CFL conditions, we only envisage implicit solvers. The poloidal mesh is a two-dimensional mesh made of curve quadrilateral cells with eight nodes also none as "Q8" elements in the finite element literature. In each cell L we consider polynomial basis functions ψ_i^L of degree p. For efficiency reason, the basis functions are Lagrange polynomials based on Gauss-Lobatto quadrature points.

The transported function f is approximated in cell L by a linear expansion on the basis functions

$$f(x, n\Delta t) \simeq f_L^n(x) = \sum_j f_{L,j}^n \psi_j^L(x), \quad x \in L.$$

The unknowns are the coefficients $f_{L,j}^n$ of the linear expansion.

We first consider the simplest first order implicit DG approximation scheme. It reads: $\forall L, \forall i$

$$\int_L \frac{f_L^n - f_L^{n-1}}{\Delta t} \psi_i^L - \int_L \boldsymbol{v} \cdot \boldsymbol{\nabla} \psi_i^L f_L^n + \int_{\partial L} \left(\boldsymbol{v} \cdot \boldsymbol{n}^+ f_L^n + \boldsymbol{v} \cdot \boldsymbol{n}^- f_R^n \right) \psi_i^L = 0. \tag{3.1}$$

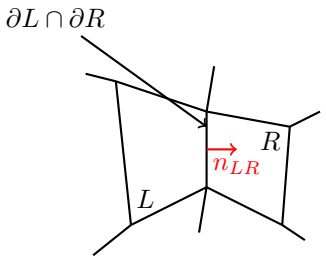

FIGURE 3.1. Normal vector convention.

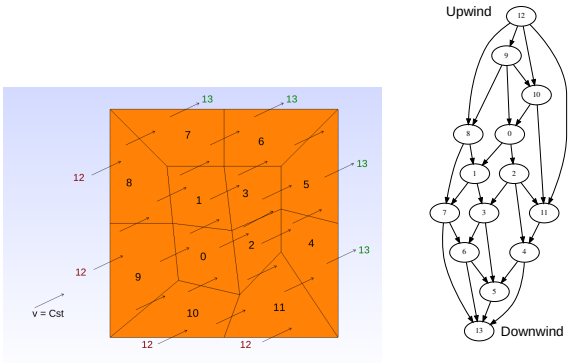

FIGURE 3.2. Dependency graph.

where:

- R denotes the neighbor cells along ∂L.
- $\boldsymbol{v} \cdot \boldsymbol{n}^+ = \max(\boldsymbol{v} \cdot \boldsymbol{n}, 0)$, $\boldsymbol{v} \cdot \boldsymbol{n}^- = \min(\boldsymbol{v} \cdot \boldsymbol{n}, 0)$. We thus use an upwind numerical flux.
- $\boldsymbol{n}_{LR}$ is the unit normal vector on ∂L oriented from L to R. See Figure 3.1.

In the above formalism, we only describe a first order time scheme. In practice, we actually use a second order implicit Crank-Nicolson time stepping [2]. But it is very similar to the above description.

3.2. **Downwind algorithm.** An important feature of the DG method is that because of the upwind nature of the numerical flux, it is rather easy to solve the linear system for computing the distribution function at time t^n, f^n from the distribution function f^{n-1} of the previous time step. It can be solved step by step simply sweeping the mesh in the direction of the velocity vector. More precisely, we say that a cell L is *upwind* with respect to a cell R if $\boldsymbol{v} \cdot \boldsymbol{n}_{LR} > 0$ on $\partial L \cap \partial R$. In a cell L, the solution depends only on the values of f in the upwind cells. For a given velocity v we can build a dependency graph. Vertices are associated to cells and edges to cells interfaces or boundaries. We consider two fictitious additional vertices: the "upwind" vertex and the "downwind" vertex. The dependency graph for simple unstructured mesh and a given constant velocity is represented on Figure 3.2. The construction can be generalized to any unstructured mesh with flat faces (in order to avoid loops in the graph).

For solving one transport equation for a given velocity $\boldsymbol{v}$, the algorithm is the following:

- First we perform a topological ordering of the dependency graph.
- First time step: Assembly, LU decomposition and storage of the local cell matrices. These computations can also be redone during each time-step for saving memory (but it is more CPU demanding).
- For each cell (in topological order):
  - Compute volume terms.
  - Compute upwind fluxes.
  - Solve the local linear system.
  - Extract the results to the downwind cells.

3.3. **Parallelization.** Because of the dependency graph we cannot perform all the computations in parallel. For instance, for the mesh of Figure 3.2. It is necessary to compute the solution in cell 0 first. Then cells 1 and 2 can be done in parallel, *etc.*

The parallelization is done by a task graph approach. Depending on the compilation options, this can be based on OpenMP tasks or on a specialized DAG (Direct Acyclic Graph) clustering algorithm.

4. Kinetic solver

4.1. **Transport.** The main task of the kinetic solver is devoted to the resolution of the transport equations (2.5). If the kinetic velocity λ_k is in the (r, z) direction, then one uses the DG algorithm of Section 3.2. If the kinetic velocity is in direction φ then the shift algorithm is used.

4.2. **Relaxation.** We denote by f_k^n the kinetic fields at time step n, corresponding to time $t_n = n\Delta t$. We denote by f_k^* the value of the fields after the free transport step (2.5).

For obtaining the new value of the field at time step $n+1$, the kinetic fields f_k^* are recombined according to

$$f_k^{n+1} = 2f_k^{*,eq} - f_k^*.$$

This over-relaxation formula ensures second order accuracy in time [5, 6].

4.3. **Boundary conditions.** For the moment, the boundary conditions are based on a known exterior state. More precisely, in the DG scheme (3.1) if R corresponds to a boundary, then the unknown value of f_R is given by

$$f_R = f^{eq}(\rho, u),$$

where ρ and u are imposed boundary data and f^{eq} is given by (2.4).

5. Numerical tests

5.1. **Transport in two-dimensional geometry.** We present now a few test cases for assessing the good behavior of the solver described above.

We consider the two-dimensional rotation of a Gaussian pulse. The pulse is given by

$$g(x, y, z, t) = \exp(-30(((x' - 1)^2 + y'^2 + z'^2),$$

with

$$\begin{aligned} x' &= \cos(\alpha t)x + \sin(\alpha t)y, \\ y' &= \cos(\alpha t)y - \sin(\alpha t)x, \\ z' &= 0, \end{aligned}$$

and (for instance)

$$\alpha = 1/4.$$

This exact solution satisfies the transport equation

$$\partial_t g + \boldsymbol{v} \cdot \boldsymbol{\nabla} g = 0,$$

with

$$\boldsymbol{v} = \alpha \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix}.$$

We check that

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$$

and thus

$$\partial_t g + \boldsymbol{\nabla} \cdot (g\boldsymbol{v}) = 0,$$

The computational domain is the disk

$$\Omega = \left\{(x, y, z), \quad x^2 + y^2 < 4, \quad z = 0\right\}.$$

In practice, this test case validates the solver in a single poloidal plane, where the variables (r, z) are replaced by (x, y) (see Remark 2).

We compute numerically the above solution with two different meshes with refinement levels of 5 and 10 (see Figure 5.2) and with $n_t = 500$ and $n_t = 1000$ time steps. The time step is given by $\Delta t = 2\pi/n_t$. The numerical solution is plotted at time $t_{\max} = 2\pi$ on Figure 5.1. We check that the Gaussian shape is well preserved by the LBM scheme. We also check the order of convergence in the L^2 norm. The measured numerical order is here 2.405.

The time scheme is only second order accurate and is thus the limiting factor for the convergence. Anyway we also check the spatial convergence of the scheme for higher order DG approximation. For this, we use a sufficiently small time step in such a way that the error due to the time approximation is negligible. We perform simulations with two meshes with two different refinements of 10 and 20 (see Figure 5.2). We then obtain the numerical orders of Table 1.

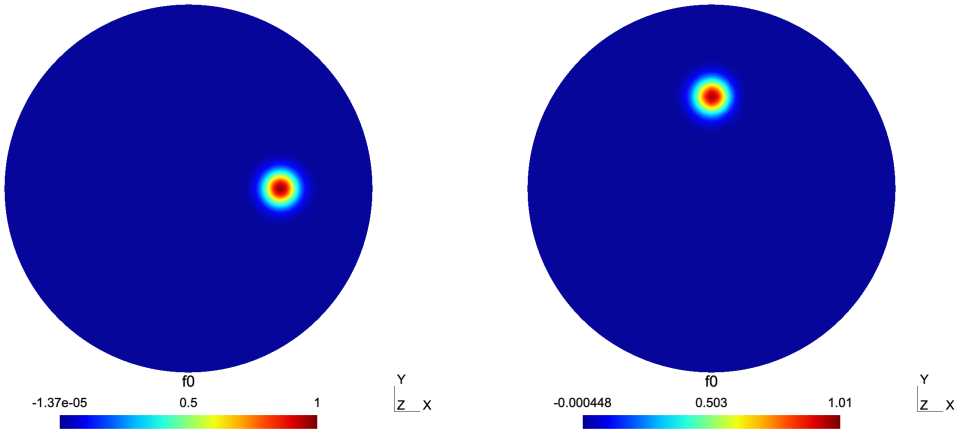

FIGURE 5.1. Numerical solution at times $t = 0$ (left) and $t = 2\pi$ (right) after the Gaussian pulse has done a quarter turn.

p	num. order
1	0.997
2	2.68
3	3.772

TABLE 1. Numerical order of the DG scheme for several values of the polynomial order p. In this test, the time step is chosen in such a way that the time error is negligible.

5.2. **Transport in a 3D periodic cylinder.** Now we activate also transport in the third direction. We consider a three-dimensional helical shift of a Gaussian pulse. The pulse is given by

$$g(x, y, z, t) = \exp(-30((x' - 1)^2 + y'^2 + z'^2),$$

with

$$\begin{aligned} x' &= \cos(2\pi\beta t)x + \sin(2\pi\beta t)y, \\ y' &= \cos(2\pi\beta t)y - \sin(2\pi\beta t)x, \\ z' &= z + \alpha t. \end{aligned}$$

We can also consider a periodic function in the z direction

$$g(x, y, z, t) = \exp(-30((x' - 1)^2 + y'^2)\sin(\pi z').$$

This exact solution satisfies the transport equation

$$\partial_t g + \boldsymbol{v} \cdot \boldsymbol{\nabla} g = 0,$$

with

$$\boldsymbol{v} = \alpha \begin{pmatrix} -2\pi y \\ 2\pi x \\ -1 \end{pmatrix}.$$

We check that

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$$

and thus

$$\partial_t g + \boldsymbol{\nabla} \cdot (g\boldsymbol{v}) = 0,$$

The computational domain is the cylinder

$$\Omega = \left\{(x, y, z), \quad x^2 + y^2 < 4, \quad -1 = z_{\min} < z < 1 = z_{\max}\right\}.$$

We wish a maximal time $t_{\max} = 1$. In this way the Gaussian pulse will move a little bit, without touching the boundaries. Parallelism is managed by OpenMP and MPI. OpenMP is used for optimizing the transport solver in the poloidal planes. MPI communications are used for the parallelism in the toroidal direction. If, for instance, we choose $n_p = 64$ MPI processes, and a kinetic velocity $\lambda = 1$, we have

$$\Delta z = \frac{z_{\max} - z_{\min}}{n_p},$$

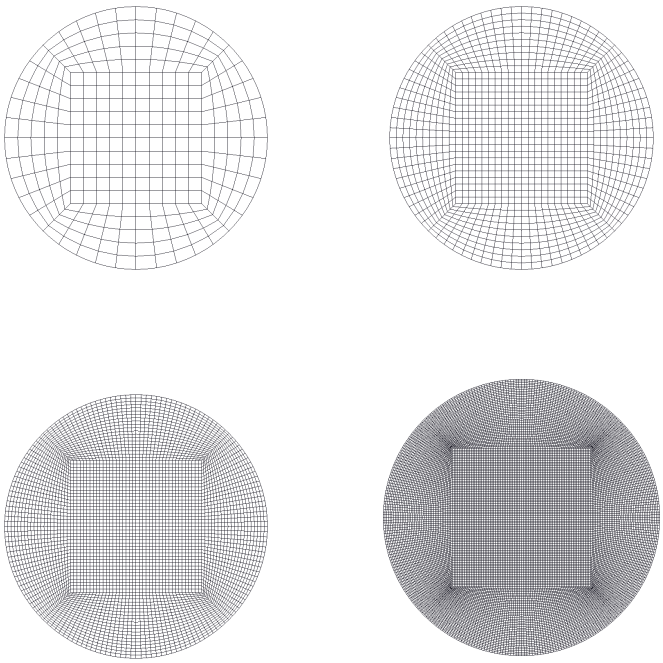

FIGURE 5.2. Poloidal meshes with refinements 5, 10, 20 and 40.

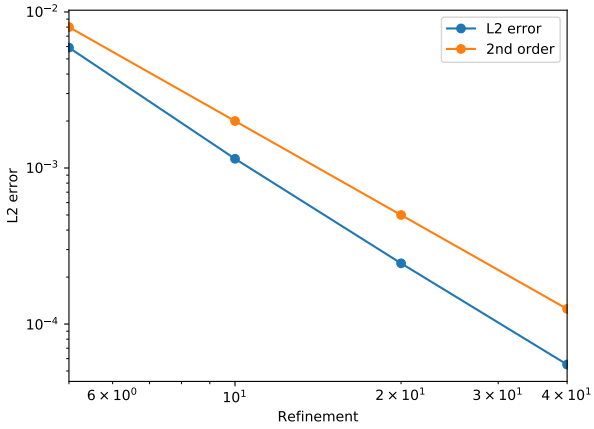

FIGURE 5.3. L^2 error as a function of the refinement level

the time step is given by

$$\Delta t = \frac{\Delta z}{\lambda} = 0.03125.$$

We thus have to perform n_t time iterations

$$n_t = \frac{t_{\max}}{\Delta t} = 32.$$

In order to check the convergence order, we have considered three different meshes of the poloidal plane with refinement of 5,10, 20 and 40. A few meshes are represented on Figure 5.2.

We then perform computations with respectively 32, 64, 128, 256 MPI processes and compute the error in the L^2 norm. We obtain the error curve of Figure 5.3. The order of convergence based on the 64 and 128 refinement levels is 2.226 (2.155 for 128-256 refinements).

6. Conclusion

In this work, we have proposed a new optimized numerical method for solving non-homogeneous conservative transport equations in toroidal geometries. The method is conservative high-order in space and time, has the complexity of a time-explicit scheme and is efficiently parallelized. It is also able to handle unstructured meshes of the poloidal plane, which is very useful for numerical simulations in tokamaks, like ITER.

The method will now be applied to more physical configurations. The transport has first to be coupled with a toroidal Poisson solver for computing the electric potential generated by the charge density motion. It is also envisaged to add to the model the imposed magnetic field, which is essential for confining the plasma. Finally, the model can also be enriched with more realistic toroidal geometries and more complex transport models, with several populations of particles such as in richer gyrokinetic models.

References

[1] Denise Aregba-Driollet and Roberto Natalini. Discrete kinetic schemes for multidimensional systems of conservation laws. *SIAM Journal on Numerical Analysis*, 37(6):1973–2004, 2000.

[2] Jayesh Badwaik, Matthieu Boileau, David Coulette, Emmanuel Franck, Philippe Helluy, Christian Klingenberg, Laura Mendoza, and Herbert Oberlin. Task-based parallelization of an implicit kinetic scheme. *ESAIM: Proceedings and Surveys*, 63:60–77, 2018.

[3] François Bouchut. Construction of BGK models with a family of kinetic entropies for a given system of conservation laws. *Journal of Statistical Physics*, 95(1-2):113–170, 1999.

[4] Shiyi Chen and Gary D Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(1):329–364, 1998.

[5] David Coulette, Emmanuel Franck, Philippe Helluy, Michel Mehrenberger, and Laurent Navoret. High-order implicit palindromic Discontinuous Galerkin method for kinetic-relaxation approximation. *Computers & Fluids*, 2019.

[6] Paul J Dellar. An interpretation and derivation of the lattice Boltzmann method using Strang splitting. *Computers & Mathematics with Applications*, 65(2):129–141, 2013.

[7] Florence Drui, Emmanuel Franck, Philippe Helluy, and Laurent Navoret. An analysis of over-relaxation in a kinetic approximation of systems of conservation laws. *Comptes Rendus Mécanique*, 347(3):259–269, 2019.

[8] Jan S Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.